

# Utilisation de WMI



# Contenu

- **WMI, c'est quoi?**
- **Principes généraux**
- **Utilisation à distance**

# WMI, c'est quoi?

# WMI, c'est quoi?

## Selon Wikipédia :

« Windows Management Instrumentation (WMI) est un système de gestion interne de Windows qui prend en charge la surveillance et le contrôle de ressources systèmes via un ensemble d'interfaces. Il fournit un modèle cohérent et organisé logiquement des états de Windows. »

# WMI, c'est quoi?

- **Permet à des scripts (notamment PowerShell) de**
  - Gérer Windows localement et à distance
  - Obtenir des informations sur le système
- **Installé sur toutes les versions de Windows depuis Windows 2000**
- **Basé sur le standard *Common Information Model (CIM)***

# Extensions

- **WMI est extensible**
- **Il peut donc être utilisé pour contrôler certains logiciels ou périphériques tiers**

# Principes généraux

# Les classes

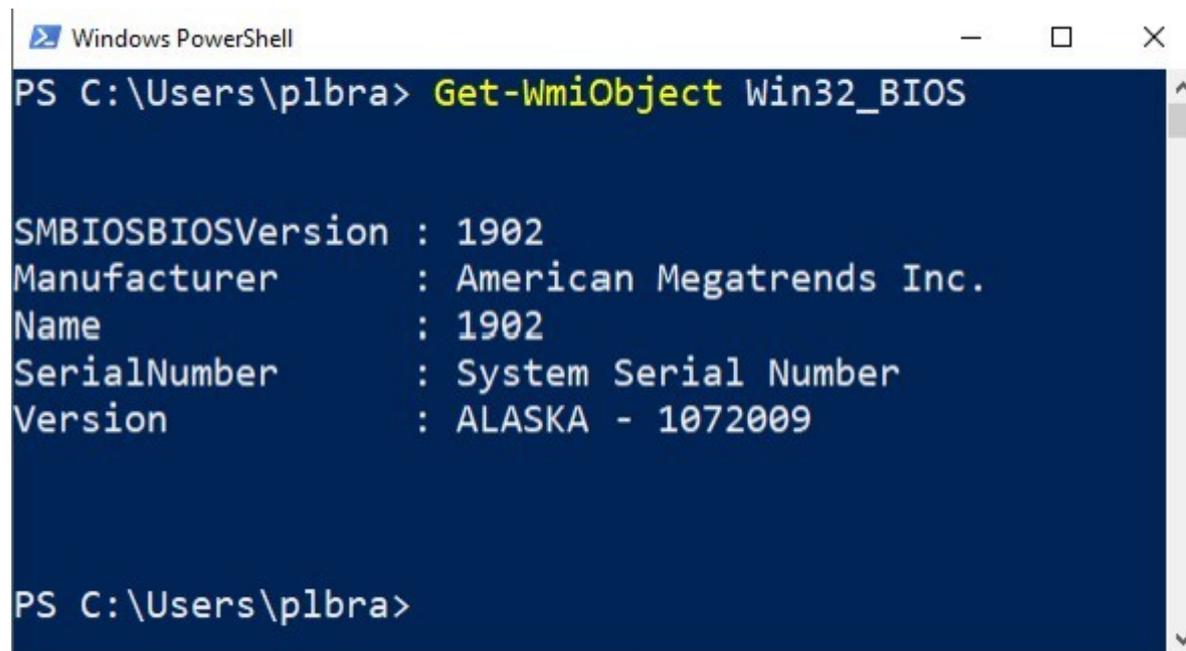
- **WMI représente les caractéristiques de l'ordinateur et du système d'exploitation sous forme de classes**
- **Il y a des classes pour à peu près tout**
  - Processeur
  - BIOS
  - Mémoire vive
  - Comptes utilisateurs
  - Services Windows
  - Etc

# Les classes

- **Le nom d'une classe est constitué du préfixe « Win32\_ » suivi du nom en anglais de ce que la classe représente**
  - Ex: la classe *Win32\_Service* décrit les services
- **Les classes dont les noms commencent par autre chose que « Win32\_ » sont des classes de bas niveau qu'on n'utilise généralement pas**

# Les instances

- **Lorsqu'on connaît déjà le nom d'une classe WMI, on peut obtenir toutes ses instances avec la commande *Get-WmiObject***



```
Windows PowerShell
PS C:\Users\plbra> Get-WmiObject Win32_BIOS

SMBIOSBIOSVersion : 1902
Manufacturer      : American Megatrends Inc.
Name              : 1902
SerialNumber      : System Serial Number
Version           : ALASKA - 1072009

PS C:\Users\plbra>
```

# Obtenir la liste des classes disponibles

- La commande ***Get-WmiObject -List*** permet d'obtenir la liste de toutes les classes disponibles
- Cette liste est très longue et prend du temps à afficher
- La commande ***Where-Object*** permet de filtrer la liste pour avoir seulement les classes dont on a besoin

# Obtenir la liste des classes disponibles

```
Windows PowerShell
PS C:\Users\plbra> Get-WmiObject -List | Where-Object { $_.name -Match "Win32_" }

Namespace : ROOT\cimv2

Name                Methods              Properties
----                -
Win32_DeviceChangeEvent  {}                  {EventType, SECURITY_DESCRI...
Win32_SystemConfigurationChangeE... {}                  {EventType, SECURITY_DESCRI...
Win32_VolumeChangeEvent  {}                  {DriveName, EventType, SECU...
Win32_SystemTrace        {}                  {SECURITY_DESCRIPTOR, TIME...
Win32_ProcessTrace       {}                  {ParentProcessID, ProcessID...
```

# Obtenir la liste des membres d'une instance

- La commande *Get-WmiObject* n'affiche pas toutes les propriétés des instances
- On doit donc utiliser *Get-Member* pour obtenir la liste des autres propriétés, mais aussi des méthodes

```
Windows PowerShell
PS C:\Users\plbra> Get-WmiObject Win32_BIOS | Get-Member

    TypeName : System.Management.ManagementObject#root\cimv2\Win32_BIOS

Name                MemberType      Definition
----                -
PSComputerName      AliasProperty   PSComputerName = __SERVER
BiosCharacteristics Property         uint16[] BiosCharacteri...
BIOSVersion         Property         string[] BIOSVersion {g...
BuildNumber         Property         string BuildNumber {get...
Caption             Property         string Caption {get;set;}
```

# Les filtres

- On veut rarement récupérer la liste de toutes les instances d'une classe
- On peut filtrer le résultat de *Get-WmiObject* avec *Where-Object*:
  - `Get-WmiObject Win32_Process | Where-Object { $_.Name -eq 'powershell.exe' }`
- On peut aussi le filtrer avec un argument *-filter* (plus rapide!)
  - `Get-WmiObject Win32_Process -filter 'name = "powershell.exe"'`
  - `Get-WmiObject Win32_Process -filter 'name like "powershell%"'`

# Accéder directement à une instance

- Chaque instance WMI a un chemin (*path*) unique
- Ce chemin est disponible dans la propriété **\_\_PATH** de l'instance

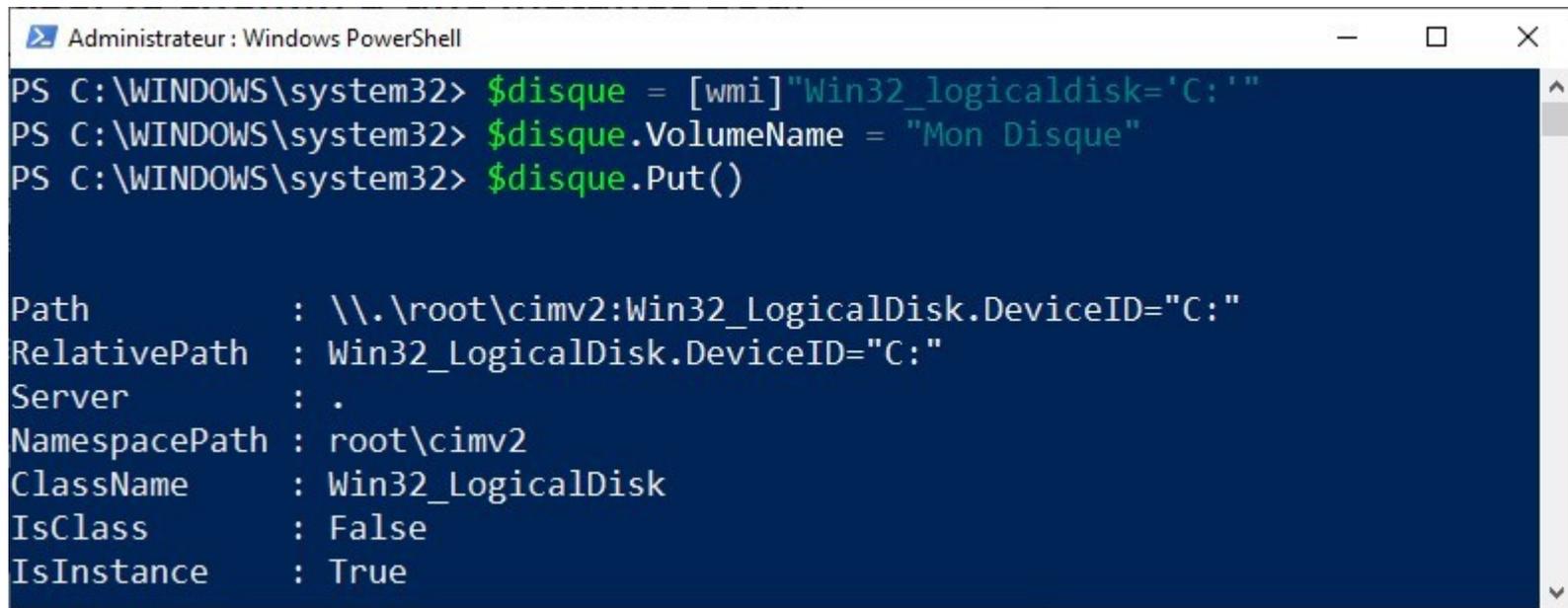
```
Windows PowerShell
PS C:\Users\plbra> Get-WmiObject Win32_Service | ForEach-Object { $_.__PATH }
\\PRINCIPAL\root\cimv2:Win32_Service.Name="AdobeARMservice"
\\PRINCIPAL\root\cimv2:Win32_Service.Name="AdobeFlashPlayerUpdateSvc"
\\PRINCIPAL\root\cimv2:Win32_Service.Name="AJRouter"
\\PRINCIPAL\root\cimv2:Win32_Service.Name="ALG"
\\PRINCIPAL\root\cimv2:Win32_Service.Name="AppIDSvc"
\\PRINCIPAL\root\cimv2:Win32_Service.Name="Appinfo"
\\PRINCIPAL\root\cimv2:Win32_Service.Name="AppMgmt"
\\PRINCIPAL\root\cimv2:Win32_Service.Name="AppReadiness"
\\PRINCIPAL\root\cimv2:Win32_Service.Name="AppVClient"
\\PRINCIPAL\root\cimv2:Win32_Service.Name="AppXSvc"
```

# Accéder directement à une instance

- **On peut accéder directement à l'instance à l'aide de son chemin:**
  - `$monService = [wmi]'Win32_Service.Name="AppInfo"`

# Modifier une propriété d'une instance

- Une instance WMI possède parfois des propriétés modifiables
- Il faut appeler la méthode **Put** de l'instance pour que les changements prennent effet
- Les droits d'administrateur sont souvent requis

A screenshot of a Windows PowerShell window titled "Administrateur : Windows PowerShell". The window shows three commands being executed in a sequence. The first command is `$disque = [wmi]"Win32_logicaldisk='C:'"`. The second command is `$disque.VolumeName = "Mon Disque"`. The third command is `$disque.Put()`. Below the commands, the properties of the WMI instance are displayed in a list format.

```
PS C:\WINDOWS\system32> $disque = [wmi]"Win32_logicaldisk='C:'"  
PS C:\WINDOWS\system32> $disque.VolumeName = "Mon Disque"  
PS C:\WINDOWS\system32> $disque.Put()  
  
Path           : \\.\root\cimv2:Win32_LogicalDisk.DeviceID="C:"  
RelativePath   : Win32_LogicalDisk.DeviceID="C:"  
Server        : .  
NamespacePath : root\cimv2  
ClassName     : Win32_LogicalDisk  
IsClass       : False  
IsInstance    : True
```

# Les méthodes d'instances

- Les instances WMI possèdent souvent des méthodes qui permettent d'effectuer des opérations sur le système
- Astuce: *Get-Member -MemberType Method* permet d'afficher les méthodes d'une instance

Administrateur : Windows PowerShell

```
PS C:\WINDOWS\system32> $notepadProcesses = Get-WmiObject Win32_Process -filter "name='notepad.exe'"
PS C:\WINDOWS\system32> $notepadProcesses | Get-Member -MemberType Method
```

```
TypeName : System.Management.ManagementObject#root\cimv2\Win32_Process
```

Name	MemberType	Definition
AttachDebugger	Method	System.Management.ManagementBaseObject...
GetAvailableVirtualSize	Method	System.Management.ManagementBaseObject...
GetOwner	Method	System.Management.ManagementBaseObject...
GetOwnerSid	Method	System.Management.ManagementBaseObject...
SetPriority	Method	System.Management.ManagementBaseObject...
Terminate	Method	System.Management.ManagementBaseObject...

```
PS C:\WINDOWS\system32> $notepadProcesses | ForEach-Object { $_.Terminate() }
```

# Les méthodes de classes

- **Dans certains cas, une méthode s'applique à une classe plutôt qu'à une instance**
  - En POO, on dit d'une telle méthode qu'elle est **statique**
- **([wmi]class)"Win32\_NetworkAdapterConfiguration" | Get-Member - MemberType Method**
- **([wmi]class)"Win32\_NetworkAdapterConfiguration".RenewDHCPReleaseAll()**

# Utilisation à distance

# Utilisation à distance

- **Il est possible d'utiliser WMI pour accéder à un autre ordinateur si:**
  - Il y a une connexion réseau entre cet ordinateur et le nôtre
  - Nous possédons des permissions suffisantes sur l'autre ordinateur
  - L'opération n'est pas bloquée par un pare-feu

# Utilisation à distance

- **Pour accéder aux instances de Win32\_Process sur un autre ordinateur:**
  - `Get-WmiObject -ComputerName Nom de l'ordinateur Win32_Process`

# Utilisation à distance

- Avec un autre compte:

```
Administrateur : Windows PowerShell
PS C:\WINDOWS\system32> $credential = Get-Credential

applet de commande Get-Credential à la position 1 du
pipeline de la commande
Fournissez des valeurs pour les paramètres suivants :
Credential
```

Demande d'informations d'identification... ? X



Entrez vos informations d'identification.

Nom d'utilisateur :  ...

Mot de passe :

OK Annuler

```
Administrateur : Windows PowerShell
PS C:\WINDOWS\system32> $credential = Get-Credential

applet de commande Get-Credential à la position 1 du
pipeline de la commande
Fournissez des valeurs pour les paramètres suivants :
Credential
PS C:\WINDOWS\system32> Get-WmiObject -computername pc023
-credential $credential Win32_Process
```

# Fin de la présentation

Des questions?



Photo par Emily Morter sur Unsplash