Fiche 04 : Administration via WMI

1. Les objets WMI (Windows Management Instrumentation)

La commande principale est Get-WmiObject, elle permet de lire les ressources des classes WMI. Ces classes étant très nombreuses et spécifiques au système d'exploitation, il est possible de lister les classes d'un poste particulier avec le paramètre –List. Il n'est pas nécessaire que PowerShell soit installé sur le poste cible.

```
Get-WmiObject -List -ComputerName nemo
```

Remarques :
En local, il n'est pas nécessaire de spécifier le paramètre ComputerName, ou le nom peut être remplacé par le point (.).
Le nom du poste distant peut-être remplacé par son adresse IP.
Le compte utilisé doit appartenir au groupe Administrateurs local de l'ordinateur distant pour accéder aux classes WMI.

La liste étant importante, il est possible de ne rechercher que certaines classes avec la commande Select-String.
Exemple, recherche des classes sur le poste local, comportant la chaine "networkadapter":

```
Get-WmiObject -List | Select-String "networkadapter"
…
\\NEMO\ROOT\cimv2:Win32_NetworkAdapterConfiguration
\\NEMO\ROOT\cimv2:Win32_NetworkAdapterSetting
```

La commande Get-Member permet de consulter la liste des propriétés et méthodes de la classe spécifiée.

```
Get-WmiObject win32_networkadapterconfiguration | Get-Member
   TypeName: System.Management.ManagementObject#root\cimv2\Win32_NetworkAdapterConfiguration
Name                        MemberType   Definition
----                        ----------   ----------
DisableIPSec                Method       System.Management.ManagementBaseObject DisableIPSec()
EnableDHCP                  Method       System.Management.ManagementBaseObject EnableDHCP()
…
IPAddress                   Property     System.String[] IPAddress {get;set;}
…
```

2. Lister des propriétés particulières

Il est possible de réduire les propriétés à lister avec la commande Select-Object et le paramètre –Property.
Exemple, lister seulement le lecteur et l'espace disque libre avec la classe WMI win32_logicalDisk :

```
Get-WmiObject Win32_LogicalDisk | Select-Object -Property DeviceID, FreeSpace
```

La même chose en conservant seulement les disques fixes (-Filter "DriverType=3) :

```
Get-WmiObject Win32_LogicalDisk -Filter DriveType=3 | Select-Object -Property DeviceID, FreeSpace
```

Une autre solution consiste à parcourir tous les objets retournés et à spécifier la propriété retenue avec le descripteur $_.

```
Get-WmiObject Win32_LogicalDisk | ForEach-Object{$_.DeviceID+$_.FreeSpace / 1GB}
```

Ici on divise en GB

Exemple pour lister l'adresse MAC, l'adresse IP et les données DHCP avec la classe WMI Win32_NetworkAdapterConfiguration :

```
Get-WmiObject Win32_NetworkAdapterConfiguration -Filter IPEnabled=True | Select-Object -Property
MACAddress, IPAddress, DHCP*
```

Remarque : l'utilisation de l'étoile (*) est autorisée dans la description des paramètres.

3. Modifier la configuration d'un poste

Il faut utiliser une méthode de la classe WMI concernée.
Exemple, activation de DHCP sur la carte réseau d'index N°3 avec le descripteur $_:

```
Get-WmiObject Win32_NetworkAdapterConfiguration -Filter index=3 | ForEach-Object {$_.EnableDHCP()}
```

Exemple d'affectation d'une adresse IP sur la carte réseau d'index N°3 :

```
Get-WmiObject Win32_NetworkAdapterConfiguration -Filter index=3 | ForEach-Object
{$_.EnableStatic('10.0.0.20','255.0.0.0')}
```

Exemple, affecter toutes les cartes réseaux (avec IP activé) au domaine 'ig.laon" :

```
Get-WmiObject Win32_NetworkAdapterConfiguration -Filter IPEnabled=true | ForEach-Object
{$_.SetDNSDomain("ig.laon")}
```

Et pourquoi ne pas faire un test vers le poste d'adresse 10.0.0.20, c'est OK si le StatusCode retourné est égal à 0 :

```
Get-WmiObject Win32_PingStatus -Filter "address='10.0.0.20'" | Select-Object -Property StatusCode
```