POWERSHELL: Estructura WMI

1. Estructura WMI

WMI (Windows Management Instrumentation), es la implementación realizada por Microsoft de los estándares WBEM[1] y CIM[2]. Dichos estándares pretenden recolectar de forma agnóstica toda información gestionada en un sistema, como puede ser una clave de registro, fichero, servicio o proceso y las acciones asociadas a los mismos.

WMI Representa la mayoría de la información y acciones relacionadas en el sistema operativo en forma de objetos WMI, donde estructuralmente, WMI se organiza (esquema abajo-arriba) en:

- **Propiedad:** Unidad de información formada por tres datos, nombre de propiedad, tipo de datos de la propiedad y valor de la misma. Sería un sinónimo de variable.
- **Método:** Función que realiza una tarea concreta sobre un tipo de información dados unos datos de entrada y con un tipo de respuesta conocido.
- **Clase:** Conjunto de propiedades y métodos que hacen referencia a un tipo de datos concreto, como puede ser un proceso o un fichero.
- **Espacio de nombres:** Estructura jerárquica creada para catalogar las diferentes clases existentes según tipo de información que gestiona, como puede ser hardware, software, BIOS, standard CIM, etc ...

Una vez tenemos la estructura de la información, cuando esta contiene datos, hablamos de:

- **Instancia:** Objeto de clase que hace referencia a un dato concreto, como puede ser el objeto que contiene los datos del proceso notepad.exe cuando este está en ejecución, ruta, PID, memoria utilizada, etc...

En el Windows 10 Pro utilizado para las pruebas, tenemos 62 espacios de nombres y un total de 8611 clases WMI, por lo que, dada la gran cantidad de información a consultar, para la primera toma de contacto vamos a utilizar WMI Explorer 2.0[3].

> **Importante:**
> Los **espacios de nombres** son utilizados para etiquetar **Clases**, que están formadas por **propiedades y métodos**.
>
> La representación de un elemento del sistema se realiza con una **instancia de Clase**, conteniendo esta las propiedades y métodos referentes al elemento concreto referenciado.

2. Búsqueda Genérica

Como primer objetivo vamos a intentar asimilar la estructura de la información en espacios de nombres con posibilidad de anidamiento (estructura de árbol), donde cada rama cataloga un conjunto de clases, que pueden estar instanciadas por objetos WMI haciendo referencia a datos del sistema en ejecución.

Para ello definamos funciones básicas para listar dichas entidades:

- **Obtención de espacios de nombres:**

```
#region EspaciosDeNombres
function Get-WmiNamespace {
        Param (
        # Espacio de nombres donde iniciar la búsqueda
        [Parameter(Mandatory = $False)] [string] $Namespace = "root\cimv2",
        # Listado recursivo
        [Parameter(Mandatory = $False)] [switch] $Recursive
    )
    # Objetos WMI englobados en la clse __NAMESPACE
    Get-WmiObject -ErrorAction SilentlyContinue -Namespace $Namespace -Class __NAMESPACE | ForEach-Object{
        # Se descartan espacios de nombres locales
        if (-not (($_.Name).ToLower()).StartsWith('ms_')) {
            ($ns = '{0}\{1}' -f $_.__NAMESPACE, $_.Name)
            # Si listado recursivo, se continua
            if ($Recursive) {
                Get-WmiNamespace -Recursive -Namespace $ns
            }
        }
    }
}
#endregion EspaciosDeNombres
```

- **Obtención de clases asociadas a un espacio de nombres:**

```
#region Clases
function Get-WmiClass {
    Param(
        # Espacio de nombres donde iniciar la búsqueda
        [Parameter(Mandatory = $False)] [string] $Namespace = "root\cimv2",
        # Filtro para acotar búsquedas
        [Parameter(Mandatory = $False)] [string] $Filter
    )
    # Obtención de todas las clases englobadas en el espacio de nombres especificado
    Get-WmiObject -List -Namespace $Namespace |
        # Filtrado de resultados
                Where-Object{ $_.Name -match $Filter }
}
#endregion Clases
```

- **Listado propiedades de una clase dada:**

```
#region Propiedades
function Get-WmiClassProperty {
    Param(
        # Espacio de nombres a la que pertenece la clase sobre la que buscar.
        [Parameter(Mandatory = $False)] [string] $Namespace = "root\cimv2",
        # Clase sobre la que buscar
        [Parameter(Mandatory = $True)]  [string] $Class,
        # Posible filtro según nombre de propiedad
        [string] $Filter
    )
    # Obtención de Clase deseada
    Get-WmiObject -ErrorAction SilentlyContinue -List -NameSpace $Namespace -Class $Class | ?{
        # Aplicado de filtro en nombre de propiedad
        $_.Properties.Name -match $Filter
    }
}
#endregion Propiedades
```

- **Instancias de clases presentes:**

```
#region Instancias
function Get-WmiClassInstance {
    Param(
        # Espacio de nombres a la que pertenece la clase sobre la que buscar.
        [Parameter(Mandatory = $False)] [string] $Namespace = "root\cimv2",
        # Clase sobre la que buscar
        [Parameter(Mandatory = $True)]  [string] $Class
    )
    # Obtención de objetos de clase según especificado en el parámetro $Class
    Get-WmiObject -ErrorAction SilentlyContinue -Namespace $Namespace -Class $Class
}
#endregion Instancias
```

3. Búsqueda Genérica

Y ahora veamos unos ejemplos básicos de uso de estas funcionalidades que podrían aplicarse en ciertos contextos de la vida real.

> - **Obtención recursiva de los espacios de nombres bajo la rama Root\Microsoft:**
>   - Get-WmiNamespace -Namespace root\Microsoft -Recursive

> - **Obtención de clases bajo la rama Root\Subscription que en su nombre contengan la cadena "Filter":**
>   - Get-WmiClasses -Namespace root\subscription -Filter Event

> - **Listado de propiedades de la clase MSFT_ScheduledTask que se encuentra bajo la rama Root\Microsoft\Windows\TaskScheduler:**
>   - Get-WmiClassProperties -Namespace Root\Microsoft\Windows\TaskScheduler -Class MSFT_ScheduledTask

> - **Obtención de instancias de la clase MSFT_ScheduledTask:**
>   - Get-WmiClassInstances -Namespace Root\Microsoft\Windows\TaskScheduler -Class MSFT_ScheduledTask

4. Referencias

[1] http://www.dmtf.org/standards/wbem
[2] http://www.dmtf.org/standards/cim
[3] https://wmie.codeplex.com/