# Abstracta WMI Monitor

## Introducción

Abstracta WMI Monitor es una herramienta que busca facilitar el acceso a la información y funcionalidades expuestas mediante WMI. Está desarrollada en .NET v4.5, lenguaje C#. Tiene una interfaz gráfica WPF (GUI: Graphic User Interface), una interfaz por línea de comandos (CL: Command Line) y provee además una simple API programática.

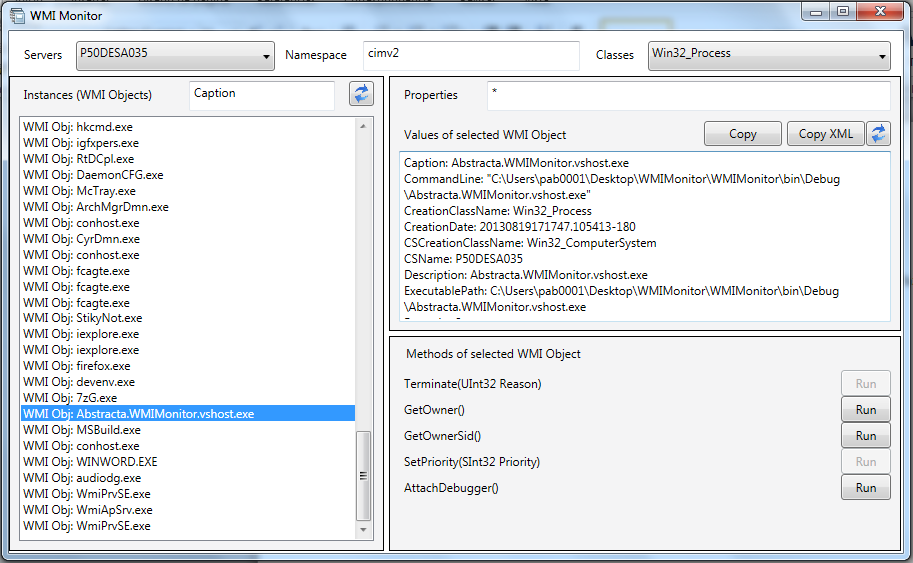


Ilustración 1 - GUI

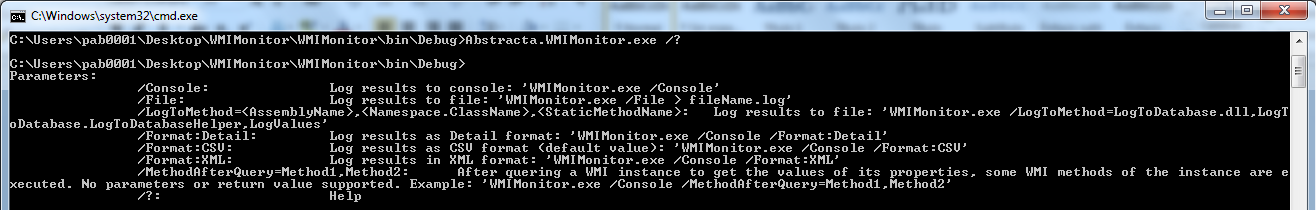


Ilustración 2 - CL

## Inicialización de la aplicación

La aplicación se inicializa desde dos archivos XML: *Abstracta.WMIMonitor.exe.config*, y *Providers.xml*.



Ilustración 3 - Abstracta.WMIMonitor.exe.config.xml

Si bien **en modo GUI no es necesario definir estas propiedades**, en modo CL se requiere que los parámetros *WMINamespace, WMIClassName, WMIKeyProperty y WMIFilterProperties* estén correctamente definidos.



Ilustración 4 - Providers.xml

Abstracta WMI Monitor permite acceder a objetos WMI expuestos en servidores de la LAN. A tales fines se deben crear usuarios con permisos adecuados.

## Interfaces de acceso

Las interfaces de acceso (GUI, CL, API) tienen diferentes propósitos, por lo que proveen diferentes funcionalidades. La interfaz GUI pretende ser una interfaz de “debug”, donde se puede navegar y buscar entre los elementos WMI publicados, mientras que la interfaz CM y la API programática permiten automatizar el acceso a los objetos WMI, ya sea para ejecutar métodos como para consultar los valores expuestos.

### GUI

Abstracta WMI Monitor provee un modo interfaz gráfica para facilitar la búsqueda y lectura de los objetos WMI expuestos. Desde esta interfaz se puede seleccionar el servidor que expone los objetos WMI, el namespace donde están estos objetos, la clase que implementan, y luego se permite ver las propiedades de los objetos publicados así como sus métodos.

También se permite ejecutar métodos sobre el objeto WMI, aunque con varias limitantes:

1. El método no tiene parámetros de entrada.
2. El método no tiene parámetros de salida.
3. El método está implementado en el objeto (no se soporta la ejecución de un método estático, es decir, implementado en la clase)

Se pretende mejorar este aspecto en futuras implementaciones en la medida que sea útil para algún cliente.

### CL

Abstracta WMI Monitor provee una interfaz por línea de comando, con el objetivo de facilitar la automatización de la consulta a los objetos WMI, así como la ejecución de los métodos expuestos por los mismos.

Para más detalle ejecutar el comando: **Abstracta.WMIMonitor.exe /?**

A modo de ejemplo, en la siguiente ilustración se presenta un .bat que crea una carpeta por día nombrándola con la fecha actual, luego ejecuta WMI Monitor logueando el resultado de la consulta a los objetos WMI a archivo, nombrando el archivo con la hora actual.

Esto sucede cada 5 minutos.

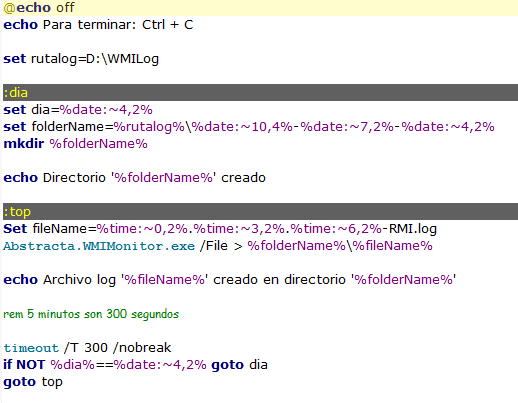


Ilustración 5 - WMI Monitor permanente

El algoritmo interno de WMI Monitor ejecuta de la siguiente forma:

*Para cada <****Servidor****> definido en providers.xml*

*Para cada objeto instancia de la clase <****WMIClassName****> dentro del <****WMINamespace****>*

*Para cada propiedad definida en <****WMIFilterProperties****>*

*Obtener Valor y guardarlo en el archivo*

El formato por defecto es CSV, pero puede guardarse en formato XML o incluso en texto sin formato definido.

### API programática

La API programática ofrece una clase cuyo constructor recibe por parámetro los valores de inicialización ya descriptos { WMINamespace, WMIClassName, WMIKeyProperty, WMIProperties } y dos métodos de consulta.

GetWMIValuesFromLocalMachine()

GetWMIValuesFromServer(string serverName, string userName, string userPassword)

A continuación un código de ejemplo que muestra cómo utilizar estos métodos.

|  |
| --- |
| namespace TestingAssembly  {  using System.Collections.Generic;    public class ExampleClass  {  public void ExampleMethod()  {  var wmiMonitor = new Abstracta.WMIMonitor.API.WMIMonitor("cimv2", "Win32\_Service", "Caption", "\*");  var result = wmiMonitor.GetWMIValuesFromLocalMachine();  foreach (var wmiInstanceName in result.Keys)  {  Dictionary<string, string> propValues;  result.TryGetValue(wmiInstanceName, out propValues);  System.Console.WriteLine("<" + wmiInstanceName + ">");  if (propValues == null) continue;  foreach (var propertyName in propValues.Keys)  {  string propertyValue;  propValues.TryGetValue(propertyName, out propertyValue);  System.Console.WriteLine("\t" + propertyName + ": " + propertyValue);  }  }  }  }  } |