Lecture 5
Getting access to remote machines (WMI Basics)

print page

In this lecture, we learn how to get access to remote computers via WMI in order to remotely control a network computer. Nowadays, computer security has become extremely complicated. The way to access a remote computer depends heavily on the security settings and the Internet settings of both computers. Therefore, you should not expect that all the code listed in this lecture can be run straight away on your computer. You have to try your luck and set up your network and computers properly before you test the scripts.

Key words

Windows Management, WMI (Windows Management Instrumentation), WMI class, WMI service, WMI query, WMI Scripting Library, CIM (Common Information Model), CIM Repository, Namespace, WBEM (Web-Based Enterprise Management), Security Settings, Internet/Network Settings

Reference to textbook chapters

This lecture covers Chapters 17 and 29 of the textbook. See also Chapter 3 of the eBook. (Don Jones, VBScript, WMI and ADSI unleashed : using VBScript, WMI, and ADSI to automate Windows administration eBook: Chapter 17. Understanding WMI; Chapter 19. Querying Complex WMI Information).

Shutting down remote computers: an introductory script

```
Dim sMachine
sMachine = InputBox("Shut down which computer?")

Dim sWMI
sWMI = "SELECT * FROM Win32_OperatingSystem WHERE Primary = true"

Dim oOS
Set oOS = GetObject("winmgmts://" & sMachine & "/root/cimv2").ExecQuery(sWMI)

Dim oItem
For Each oItem in oOS
oItem.Shutdown
Next
```

In the above script, the first two statements are used to input the name of a remote computer. Win32_OperatingSystem is a WMI class, containing the information about the Windows operating systems installed on that computer. The string, sWMI, defines a WMI query to get all instances of `Win32_OperatingSystem` that are the primary operating system on the remote machine. `Cimv2` is a namespace under the root of the computer (root\cimv2 namespace) that contains all of the Win32 classes, including the `Win32_OperatingSystem` class. The third group of statements executes the WMI query. Finally, the script shutting down all Windows operating systems that are running on the computer (the existing windows systems up to date can only allow one primary windows operating system running at the same time but Microsoft leaves the possibility for the future that more than one windows operating systems can be run at the same time).

With similar code, we can also reboot, logoff, or power off a remote computer by changing the statement: `oItem.Shutdown`. See more details from Win32Shutdown Method of the Win32_OperatingSystem Class.

What is WMI?

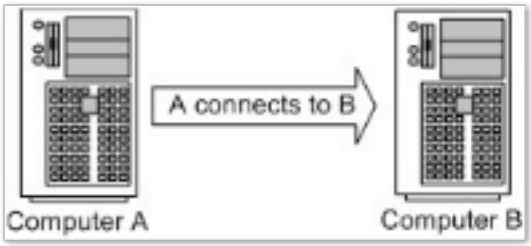
The above script actually uses windows WMI services. Windows Management Instrumentation (WMI) is the primary infrastructure for management data and operations on Windows operating systems. We can utilize the facilities of WMI to write scripts for administrative tasks on local but mainly remote computers. WMI also supplies management data to other parts of the operating system and products, for example, System Center Operations Manager, formerly Microsoft Operations Manager (MOM), or Windows Remote Management. WMI enables consistent and uniform management, control, and monitoring of systems throughout your enterprise. It allows system administrators to query, change, and monitor configuration settings on desktop and server systems, applications, networks, and other enterprise components. System administrators can write scripts that use the WMI Scripting Library to work with WMI and create a wide range of systems management and monitoring scripts.

Connecting to WMI on a Remote Computer

You might consider that remotely controlling a computer is not that hard once we have the above script. But in reality, the remote access is not straight forward. If you don't believe that, download the code shutdown.vbs and run it.

Unless you leave your firewall open you will encounter problems when trying to run the script. A large amount of work on security setting is needed.

Assume that you are working on Computer A and you are going to remotely control Computer B.



There are a few necessary conditions:
1. You have an account on Computer B which is in the Administrator group.
2. The password of your account on Computer A is not blank.
3. Both Computer A and Computer B must be running IPv6 if the computer is running under Vista. Either computer may be running IPv4.

For more details of how to set up the computers, see Connecting to WMI on a Remote Computer.

Understanding WMI

You can check whether the WMI setting works on a computer by running the following program (WbemTest) in the command line:

```
wbemtest
```

You will see a window called Windows Management Instrumentation Tester. The eBook Chapter 3 shows you all the details of this tool. Try first to connect to the remote computer you set. If no luck, you can connect to the local WMI to learn more about WMI. Suppose that you have successfully connected to a WMI (either remotely or locally). WMI contains large groups of classes in different hierarchy, dealing with:

- Computer System Hardware
- Operating System
- Installed Applications
- WMI Service Management
- Performance Counter

The most useful classes are Win32_OperatingSystem, Win32_LogicalDisk, Win32_NetworkAdapterConfiguration and so on. WMI Reference gives a complete list of all WMI classes. There are more than 100 WMI classes and may be more in the upgraded windows versions.

WbemTest can show your all the instances of these classes that are running on the computer to which you have connected (the remote computer or local machine). More importantly, you can retrieve data from these running objects by using a SQL-like query language - WMI Query Language (WQL). For instance, to get information about the Windows operating systems that are running on the machine, you can make the the following query in the query menu (click "query button"):

```
SELECT * FROM Win32_OperatingSystem WHERE Primary = true
```

You will likely see the following outcome or something similar:

```
Win32_OperatingSystem.Name="Microsoft Windows XP
Professional|C:\\WINDOWS|\\Device\\Harddisk0\\Partition1"
```

You may also remember how we get the IP address (See Lecture 3):

```
SELECT IPAddress FROM Win32_NetworkAdapterConfiguration WHERE IPEnabled =
TRUE"
```

Learn more on WQL from http://msdn.microsoft.com/en-us/library/aa392902(VS.85).aspx.

Write WMI Scripts

If the connection is ok, you can write your WMI script for any remote tasks.

First, let's get the other computer's IP address and MAC address with the checkIPnMAC.vbs:

```
Dim computer_B
Dim objWMIService
Dim colItems

'the remote computer's name, here means local
computer_B = "."

Set objWMIService = getObject("winmgmts://" & _
computer_B & "\root\cimv2")

Set colItems = objWMIService.ExecQuery( _
"SELECT * FROM Win32_NetworkAdapterConfiguration", , 48)

For Each objItem in colItems
WScript.Echo "IP Address: " & objItem.IPAddress(0)
WScript.Echo "MAC Adress: " & objItem.MACAddress

Next
```

You can use this code as a template to perform more complicated tasks. Most WMI scripts follow a simple three-step pattern. In general, WMI scripts:

1. Connect to the WMI service.
2. Retrieve a WMI object or collection of objects.
3. Perform some sort of task on the object or objects.

How do you feel - comfortable or had a headache? I suggest you search on the Web using the key words provided here for an in-depth understanding, for instance, WMI Architecture. Of course, have a break first:)