Managing Windows Networks Using Scripts - Part 1: Basic Concepts

This is the first part in a series of articles about Windows scripting networks. This section covers the most basic concepts of scripting techniques (script writing techniques) in networks running Windows operating systems. The rest of the series will go to you

Mitch Tulloch

This is the first part in a series of articles about Windows scripting networks. This section covers the most basic concepts of scripting techniques (script writing techniques) in networks running Windows operating systems. The rest of the series will give you the details of many different aspects of this topic .

There is a saying like this, unknown of any great man or commoner, but I find it very logical: ' *Give a man who is starving to a fish, you can feed him one day; But if you teach him how to fish, you feed him for life* .

What's more right, and even more true in the busy world of IT professionals (which we still know is IT people) when working with scripting techniques: ' *Give an admin a script , you help him solve a problem; but if you teach him how to write a script, you can help him do the work associated with his life* . '

The price that automates daily administrative tasks with scripts, the admin's life will be much more comfortable and gentle. Why need to know and use scripts? There aren't already hundreds of scripts written floating around in the network world that you can download easily, such as from the central source Script Center Script Repository of Microsoft. Why? Hundreds of scripts are available, please say yes. Download it easily, please say that it is true. But that's true but not enough. They are very helpful and helpful, but many of them require specific configuration in your environment to make them difficult. Sometimes in the hundreds of thousands of download scripts you can only select a suitable script and still have to adjust a bit. Simply because the author writes it is not in your organization, does not follow your configuration and their interest is directed towards something else. Then the admin must become skilled repairmen, change this place a bit, change the place a bit, paired together to turn some small scrip into a larger unified script or use data This script's output makes the input for another script, or turns it into an active tool for a remote machine . That's a lot of work!

The mechanic wants to repair the machine, he must understand its structure, which is something no one will reject. So that he 'admin' wants to change, adjust the script, he must understand it, must know how to build and write it, turn new things or what is available into his own, best suited to me . And then he called him 'skilled worker'. Want to be like that, everyone must start with the most basic things, here is Windows scripting. Speaking of scripts, many people seem to be very difficult, really difficult because first . the script is difficult to translate into Vietnamese! Script means' script ', but our tech people are not filmmakers, so the scenario of the IT world is full of crazy code but only experts understand, there are many people' often middle 'like . technology

students do! That's why today we will start from the most basic, then gradually improve the ability to understand the deeper aspects of writing and using scripts on Windows networks. The ultimate goal we are aiming at is even beginners like you, like I can automatically automate the work, to make the admin's life safer. We will do this on both the scripts you write yourself or download from a variety of sources. We will also learn some related resources worth exploring to get a better insight into Windows scripting, as well as some help tools that may be useful in the future.

TCP / IP scripting settings

Most administrators use Visual Basic Script (VBScript) to write a Windows admin script (Windows admin script). VBScript is not only a strong language but its syntax is quite simple to learn and do. VBScript can be used in conjunction with Windows Management Instrumentation (WMI) and Active Directory Services Interfaces (ADSI) to write scripts for any aspect of a Windows operating system or an Active Directory network. We will start learning about Windows scripting by using VBScript with WMI to do one thing that will be very useful: change the IP address of a network adapter.

Why should this be done? That's because we will have to use a virtual server and a virtual PC to set up a test environment. We will need to transfer a virtual machine (VM) running Windows Server 2003 operating system from one virtual network to another to use the server (server) for some other purpose. This means that we will need to change the IP address on the server (also the default gateway). You can do this by opening **Network Connections** in **Control Panel** and right-clicking **Local Area Connections** , selecting **Properties** > Internet **Protocol (TCP / IP)** on the **General** tab and clicking **Properties** , entering a new IP address, then clicking **OK** twice. This is a common way to do it, but it sounds quite long and exhausting. For professionals, they prefer to use Command Promt, the command used here is ***Netsh*** . However, when using this command you need to be careful because it has many different context, commands and parameters that are difficult to remember. Doing the wrong thing can also lead to serious consequences. If you are not really sure, ask for help from the Help section or go back to the first way.

But our goal here is to learn about scripts. Therefore, we will look at how to change the machine's IP address using VBScript and WMI, first of all, to know some basic concepts such as object, method, and properties (property), namespace .

To get started, run the script on a local machine:

strComputer = "."

Here, the *str-* prefixed object is used to refer to *strComputer* as a variable containing the string, and the dot is the reference symbol to the local machine and is used as a starting point of the WMI namespace. So what is the namespace WMI? In fact, it is a set of hierarchies of different object classes, which can be used to manage various aspects of Windows computers. For example, there is an original namespace and below it are dozens of other child namespaces such as SECURITY, CIMV2, perfmon . Most useful WMI classes are in the *rootcimv2* namespace and before working with any of them. , we need to interpret them into objects. Then consider the properties of these objects and call the method to manipulate them.

Classes, objects, properties, methods - what are they? Here's a simple analysis that can help you understand them: consider the MicrowaveOven class, which is the abstract set of all microwaves (no one real furnace is included in it). This class can have attributes: color (*Color*), cube size (*CubicInches*), round face (*HasTurntable*) . Perhaps you understand the attribute is the characteristics, characteristics specific to one Class.

In other words, these microwaves will have a certain color, have a certain internal size and they can spin or not.

The MicrowaveOven class also has methods. Method, ie a calculation function or defined by a certain rule so that the class can manipulate or you can manipulate the class. With this specific class, some methods can be used as *SetCookingTime* (set cooking time), *SetPowerLevel* (set the power level to use), *Reset* (reset) . Usually, to call a method you have to give parameters for it. For example, to call the *SetCookingTime* method, we can define the *CookingTime* variable in a given number of seconds and then put this variable into the *SetCookingTime* method set for a field. Specific combination of this class (a real case, not a microwave in the abstract class). With WMI VBScript, we can do the following:

```
intCookingTime = 120
errSetCookingTime = objMicrowave.SetCookingTime (intCookingTime)
```

But where is the microwave object (*objMicrowave*)? We have not created it yet, so create it using the Set command and the CreateObject method:

```
Set objMicrowave = CreateObject ("MicrowaveOven")
```

In fact, if you look closely, *objMicrowave is* n't the *MicrowaveOven* class object. More precisely, it is an object that references an instance of the *MicrowaveOven* class. But now we only start with the most basic things so these deeper aspects will be explored later.

Next, create more strColor variables to set the color properties for our microwave. Set the variable to Green (green is the tree), the script will look like the following (with some comments next to it):

```
strColor = "Green" to color the microwave
intCookingTime = 120 'specifies the cooking time (in seconds)
Set objMicrowave = CreateObject ("MicrowaveOven") 'creates an instance of the object
errSetCookingTime = objMicrowave.SetCookingTime (intCookingTime) 'calls a method to
'Set the cooking time and record the resulting error code
objMicrowave.Color = strColor 'set the Color attribute value (color)
```

It's not too hard, isn't it?

Return to the script

To access the machine's TCP / IP configuration settings using WMI, you need to write the code:

```
Set objWMIService = GetObject ("winmgmts:" & strComputer & "rootcimv2")
```

This command will connect you to the namespace *rootcimv2* on the local machine by defining an object named *objWMIService* and setting it equal to the return value of the GetObject method. After connecting to this namespace, you can gather information like below:

```
Set colNetAdapters = objWMIService.ExecQuery ("Select * from Win32_NetworkAdapterConfiguration where
IPEnabled = TRUE")
```

How does this command run? First, you can see the object named *objWMIService* that we just described a

minute ago in the above line. After this object is ExecQuery , it can be an attribute that can also be a method (the structure of the command is always **doituong.thuoctinh** or **doituong.phuongthuc**). We can easily guess that it is a method because behind it is a query. The ExecQuery method is invoked by adding a parameter to it. The parameter here is an SQL (SELECT) command, which returns the set (marked by the prefix ' col- ') of all (asterisks) configuring the network adapter on a TCP / IP envelope. and allowed on the adapter. The returned set after executing this method will be assigned to the variable *colNetAdapters* .

What can we do with this collection? When there is a set in hand, you must loop it, use a repeat command like For Each . The next loop will look like this:

```
For Each objNetAdapter in colNetAdapters
'do something to each network adapter's configuration
next
```

You always have to loop around collections, even if that collection has only one object.

Now, what we really want is to change the IP address for our adapter. Therefore, define some additional variables:

```
arrIPAddress = Array ("172.16.11.99")
arrSubnetMask = Array ("255.255.255.0")
```

Note that the variables defining the new IP address and subnet mask must be array variables. Why is that? The first reason is that Windows computers sometimes don't have only one IP address, one default gateway . So why not use array variables for all IP settings that are consistent. And the second reason, if you search for the *Win32_NetworkAdapterConfiguration* class in the WMI Reference on MSDN, you will see an array variable. We will delve deeper into the WMI Reference in the future, but for now it is temporarily accepted at a somewhat unclear level.

Finally, we need to call the EnableStatic method of the *Win32_NetworkAdapterConfiguration* class to change the IP address and default gateway of the network adapter to the new setting we defined in the array variables. Do as follows:

```
errEnableStatic = objNetAdapter.EnableStatic (arrIPAddress, arrSubnetMask)
```

Here, the *err*- variable is needed, like a place to store an error code that returns when the method runs.

Bring it all together

Now, put all the pieces together and see what we have:

```
strComputer = "."
arrIPAddress = Array ("172.16.11.99")
arrSubnetMask = Array ("255.255.255.0")
Set objWMIService = GetObject ("winmgmts:" & strComputer & "rootcimv2")
Set colNetAdapters = objWMIService.ExecQuery ("Select * from Win32_NetworkAdapterConfiguration")
For Each objNetAdapter in colNetAdapters
errEnableStatic = objNetAdapter.EnableStatic (arrIPAddress, arrSubnetMask)
```

next

You know, this code gives variable definitions, controls errors, uses input data and verifies the return result. We will reuse this code in later sections of the series, but first let's see if it works. Write down the script as ***ChangeIPAddress.vbs*** (remember to turn off Word Wrap in Notepad) and copy it to the desktop of the server with static address 172.16.11.45. Then, open the Command Promp command window as an Administrator user, navigate to the Desktop folder and run the script, using ***Cscript.exe*** . Results returned:

C: Documents and SettingsAdministratorDesktop> **ipconfig**

Windows IP Configuration

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix. :
IP Address. . . . . : 172.16.11.45
Subnet Mask. . . . . : 255.255.255.0
Default Gateway. . . . . : 172.16.11.1

C: Documents and SettingsAdministrator.DC-1Desktop> **cscript ChangeIPAddress.vbs**

Microsoft (R) Windows Script Host Version 5.6
Copyright (C) Microsoft Corporation 1996-2001. All rights reserved.

C: Documents and SettingsAdministratorDesktop> **ipconfig**

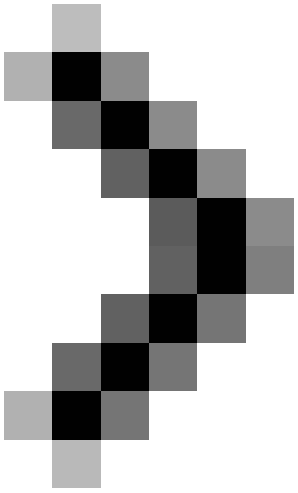
Windows IP Configuration

Ethernet adapter Local Area Connection:

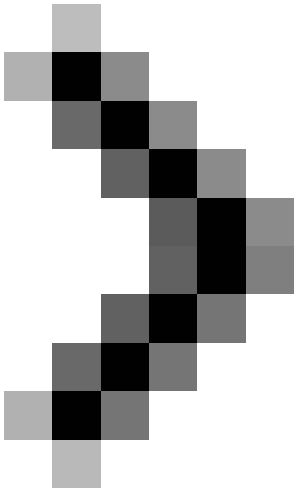
Connection-specific DNS Suffix. :
IP Address. . . . . : 172.16.11.99
Subnet Mask. . . . . : 255.255.255.0
Default Gateway. . . . . : 172.16.11.1

Yes, it worked! The IP address of the device has been successfully changed from .45 to .99 as shown on the second Ipconfig command.

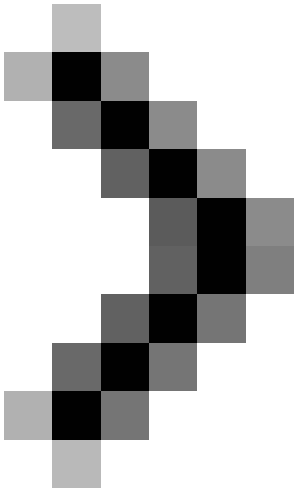
There are many more interesting things about Windows scripting, but please make an appointment at the second part. And now, goodbye!



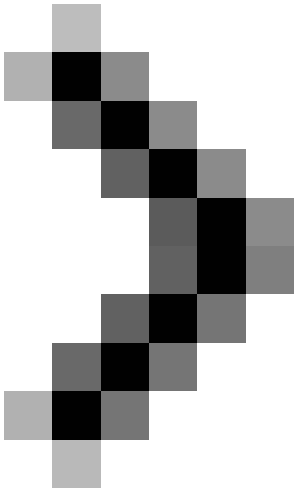
Part 2: Complete the script



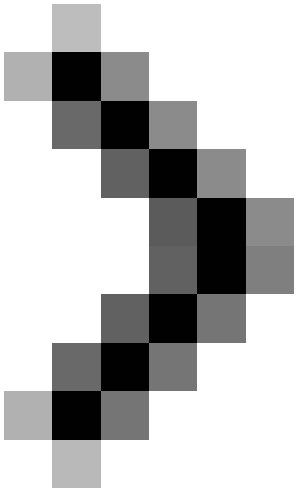
Part 3: Understanding WMI



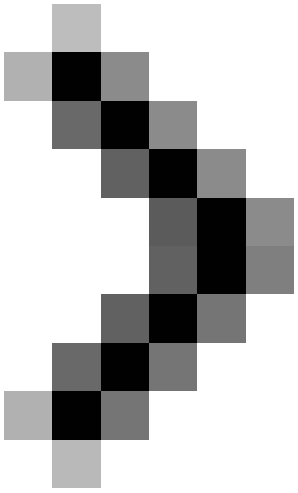
Part 4: Use Win32_NetworkAdapterConfiguration



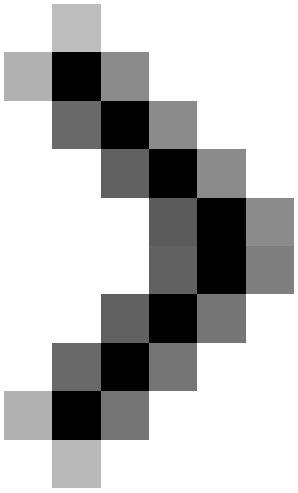
Part 5: Overcoming challenges



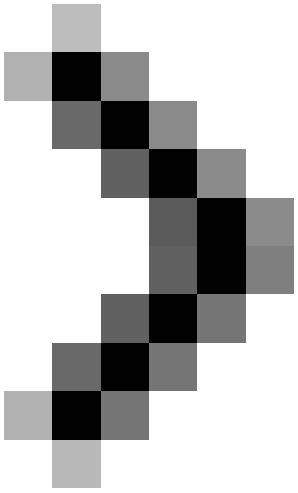
Part 6: The first steps for remote scripting



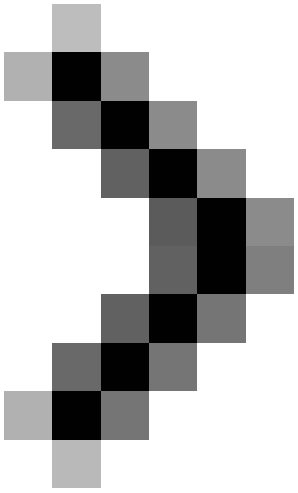
Part 7: Troubleshooting errors



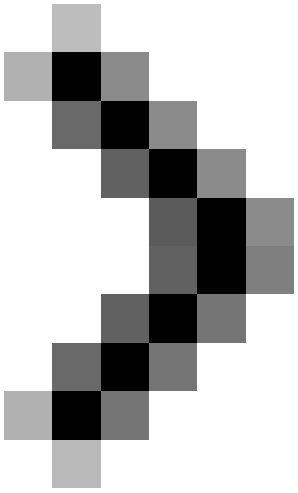
Part 8: Remote script error handling with Network Monitor 3.0



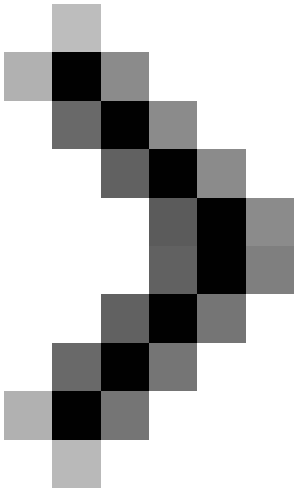
Part 9: Understanding remote control scenarios



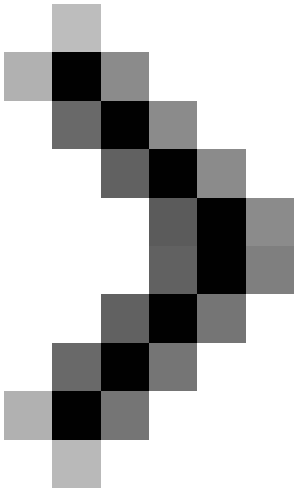
Part 10: Tricks of remote control scenarios



Part 11: Other script tricks



Part 12: Properties of the WMI class



Part 13: The script returns all values

You finished reading the article "**Managing Windows Networks Using Scripts - Part 1: Basic Concepts**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.