2

Labs

Windows PowerShell 3

Day 2

Chapter 9 Lab

The Pipeline, Deeper

Lab Time: 30

Consider the Get-ADComputer command. This command is installed on any Windows Server 2008 R2 or later domain controller – but you don't need one! You only need to know two things:

- The Get-ADComputer command has a –filter parameter; running Get-ADComputer –filter * will retrieve all computer objects in the domain.
- Domain computer objects have a Name property, which contains the computer's host name.
- Domain computer objects have the TypeName ADComputer. So, Get-ADComputer produces objects of the type ADComputer.

That's all you should need to know. With that in mind, complete these tasks:

NOTE: You're not being asked to run these commands. Instead, you're being asked if these commands will function or not, and why. You've been told how Get-ADComputer works, and what it produces; you can read the help to discover what other commands expect and accept.

1. Would the following command work to retrieve a list of installed hotfixes from all domain controllers in the specified domain? Why or why not? Write out an explanation, similar to the ones we provided earlier in this chapter.

```
Get-Hotfix -computerName (get-adcomputer -filter * |
Select-Object -expand name)
```

2. Would this alternative command work to retrieve the list of hotfixes from the same computers? Why or why not? Write out an explanation, similar to the ones we provided earlier in this chapter.

```
get-adcomputer -filter * |Get-HotFix
```

3. Would this third version of the command work to retrieve the list of hotfixes from the domain controllers? Why or why not? Write out an explanation, similar to the ones I provided earlier in this chapter.

```
get-adcomputer -filter * |
Select-Object @{l='computername';e={$_.name}} |
Get-Hotfix
```

4. Write a command that uses pipeline parameter binding to retrieve a list of running processes from every computer in an AD domain. Don't use parentheses.

5. Write a command that retrieves a list of installed services from every computer in an AD domain. Don't use pipeline input; instead use a parenthetical command (a command in parentheses).

6. Sometimes Microsoft forgets to add pipeline parameter binding to a cmdlet. For example, would the following command work to retrieve information from every domain controller in the domain? Write out an explanation, similar to the ones we provided earlier in this chapter.

```
get-adcomputer -filter * |
   Select-Object @{l='computername';e={$_.name}} |
Get-WmiObject -class Win32_BIOS
```

Chapter 10 Lab

Formatting – and Why It's Done on the Right

Lab Time: 30

See if you can complete the following tasks:

1. Display a table of processes that includes only the process names, IDs, and whether or not they're responding to Windows (the Responding property has that information). Have the table take up as little horizontal room as possible, but don't allow any information to be truncated.

2. Display a table of processes that includes the process names and IDs. Also include columns for virtual and physical memory usage, expressing those values in megabytes (MB).

3. Use Get-EventLog to display a list of available event logs. (Hint: you'll need to read the help to learn the correct parameter to accomplish that.) Format the output as a table that includes, in this order, the log display name and the retention period. The column headers must be "LogName" and "RetDays."

4. Display a list of services so that a separate table is displayed for services that are started and services that are stopped. Services that are started should be displayed first. (Hint: you'll use a -groupBy parameter).

Chapter 11 Lab

Filtering and Comparisons

Lab Time: 30

Following the principle of filter left, try to accomplish the following:

1. Import the NetAdapter module (available in the latest version of Windows, both client and server). Using the Get-NetAdapter cmdlet, display a list of non-virtual network adapters (that is, adapters whose Virtual property is False, which PowerShell represents with the special $False constant).

2. Import the DnsClient module (available in the latest version of Windows, both client and server). Using the Get-DnsClientCache cmdlet, display a list of A and AAAA records from the cache. Hint: If your cache comes up empty, try visiting a few Web pages first to force some items into the cache.

3. Display a list of hotfixes that are security updates.

4. Using Get-Service, is it possible to display a list of services that have a start type of Automatic, but that aren't currently started?

5. Display a list of hotfixes that were installed by the Administrator, and which are updates. Note that some hotfixes won't have an "installed by" value – that's okay.

6. Display a list of all processes running as either Conhost or Svchost.

Chapter 12 Lab

A Practical Interlude

Lab Time: 20

Create a directory called LABS on your computer and share it. For the sake of this exercise you can assume the folder and share don't already exist. Don't worry about NTFS permissions but you need to make sure share permissions are set so that everyone has Read/Write access and Administrators have full control. Since the share will be primarily for files you want to configure the share's caching mode for documents. Your script should show the new share and its permissions.

Chapter 13

Lab

Remote Control: One to One, One to Many

Lab Time: 20

It's time to start combining some of what you've learned about remoting with what you've learned in previous chapters. See if you can accomplish these tasks:

1. Make a one-to-one connection with a remote computer. Launch Notepad.exe. What happens?

2. Using Invoke-Command, retrieve a list of services that aren't started from one or two remote computers. Format the results as a wide list. (Hint: it's okay to retrieve results and have the formatting occur on your computer—don't include the Format- cmdlet in the commands that are invoked remotely).

3. Use Invoke-Command to get a list of the top ten processes for virtual memory (VM) usage. Target one or two remote computers, if you can.

4. Create a text file that contains three computer names, with one name per line. It's okay to use the same computer name three times if you only have access to one remote computer. Then use Invoke-Command to retrieve the 100 newest Application event log entries from the computer names listed in that file.

Chapter 14 Lab

Using Windows Management Instrumentation

Lab Time: 30

Take some time to complete the following hands-on tasks. Much of the difficulty in using WMI is in finding the class that will give you the information you need, so much of the time you'll spend in this lab will be tracking down the right class. Try to think in keywords (I'll provide some hints), and use a WMI explorer to quickly search through classes (the WMI explorer we use lists classes alphabetically, making it easier for me to validate my guesses).

1. What class could be used to view the current IP address of a network adapter? Does the class have any methods that could be used to release a DHCP lease? (Hint: network is a good keyword here.)

2. Create a table that shows a computer name, operating system build number, operating system description (caption), and BIOS serial number. (Hint: you've seen this technique, but you'll need to reverse it a bit and query the OS class first, then query the BIOS second).

3. Query a list of hotfixes using WMI. (Hint: Microsoft formally refers to these as quick fix engineering). Is the list different from that returned by the Get-Hotfix cmdlet?

4. Display a list of services, including their current status, their start mode, and the account they use to log on.

5. Can you find a class that will display a list of installed software products? Do you consider the resulting list to be complete?

Chapter 15 Lab

Multitasking with Background Jobs

Lab Time: 20

The following exercises should help you understand how to work with the different types of jobs and tasks in PowerShell. As you work through these exercises, don't feel you have to write a one-line solution. Sometimes it is easier to break things down into separate steps.

1. Create a one-time background job to find all PowerShell scripts on the C: drive. Any task that might take a long time to complete is a great candidate for a job.

2. You realize it would be helpful to identify all PowerShell scripts on some of your servers. How would you run the same command from the previous exercise on a group of remote computers?

3. Create a background job that will get the latest 25 errors from the system event log on your computer and export them to a CLIxml file. You want this job to run every day, Monday through Friday at 6:00AM so that it is ready for you to look at when you come in to work.

4. What cmdlet do you use to get the results of a job and how would you save the results in the job queue?

Review Lab 2 (Based on Chapters 1-14)

Task 1

Display a list of running processes in a table that includes only the process names and ID numbers. Do not let the table have a large blank area between the two columns.

Task 2

Run this:

```
Get-WmiObject -class Win32_UserAccount
```

Now run that same command again, but format the output into a table that has a Domain and UserName column. The UserName column should show the users' Name property. E.g.,

```
Domain   UserName
=======  ========
COMPANY  DonJ
```

Make sure the second column header says UserName, and not Name.

Task 3

Have two computers (it's OK to use localhost twice) run this command:

```
Get-PSProvider
```

Use Remoting to do this. Ensure that the output includes the computer names.

Task 4

Use Notepad to create a file named C:\Computers.txt. In that file, put the following:

```
      Localhost
      localhost
```

So you should have those two names on their own line in the file – two lines total. Save the file and close Notepad. Then, write a command that will list the running services on the computer names in C:\Computers.txt.

Task 5

Query all instances of Win32_LogicalDisk. Display only those instances which have a DriveType property containing 3, and who have 50% or more free disk space. Hint: To calculate free space percentage, it's freespace/size * 100.

Note that the –Filter parameter of Get-WmiObject cannot contain mathematical expressions.

Task 6

Display a list of all WMI classes in the root\CIMv2 namespace.

Task 7

Display a list of all Win32_Service instances where the StartMode is "Auto" and the State is not "Running."

Task 8

Find a command that can send e-mail messages. What are the mandatory parameters of this command?

Task 9

Run a command that will display the folder permissions on C:\.

Task 10

Run a command that will display the permissions on every subfolder of C:\Users. Just the direct subfolders; you do not need to recurse all files and folders. You will need to pipe one command to another command to achieve this.

Task 11

Find a command that will start Notepad under a credential other than the one you've used to log into the shell.

Task 12

Run a command that makes the shell pause, or idle, for 10 seconds.

Task 13

Can you find a help file (or files) that explains the shell's various operators?

Task 14

Write an informational message to the Application event log. Use a category of 1 and raw data of 100,100.

Task 15

Run this command:

```
Get-WmiObject –Class Win32_Processor
```

Study the default output of this command. Now, modify the command so that it displays in a table. The table should include each processor's number of cores, manufacturer, and Name. Also include a column called "MaxSpeed" that contains the processor's maximum clock speed.

Task 16

Run this command:

```
Get-WmiObject –Class Win32_Process
```

Study the default output of this command, and pipe it to Get-Member if you want. Now, modify the command so that only processes with a peak working set size greater than 5,000 are displayed.