CSC 360 Lab Assignment #7

Spring 2015 Due: March 27, 2015

The following are the end-of-chapter Labs from Chapter 13 through Chapter 18 of our textbook. Answer each question or describe the PowerShell command(s) necessary to accomplish the task described.

Chapter 13: *Remote control: one to one, and one to many*

NOTE.. For this lab, you'll need any computer running PowerShell v3. Ideally, you'll want to have two computers that are each members of the same Active Directory domain, but if you have only one computer to test with, that's fine.

1. Make a one-to-one connection with a remote computer (or with "localhost" if you only have one 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 (it's OK to use "localhost" twice if you only have one computer). Format the results as a wide list. (Hint: it's OK to retrieve results and have the formatting occur on your computer—don't include the Format-cmdlets 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; if you only have one computer, target "localhost" twice.

4. Create a text file that contains three computer names, with one name per line. It's OK to use the same computer name, or "localhost," 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: *Using Windows Management Instrumentation*

NOTE: For this lab, you'll need any computer running PowerShell v3. 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 (we'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 us to validate our guesses). Keep in mind that PowerShell's help system can't help you find WMI classes.

1. What class can you use to view the current IP address of a network adapter? Does the class have any methods that you could use 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 statuses, their start modes, and the accounts 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: *Multitasking with background job*s

NOTE: For this lab, you'll need a Windows 8 or Windows Server 2012 computer running PowerShell v3.

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's easier to break things down into separate steps.

1. Create a one-time background job to find all the 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 task 1 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 6:00 a.m., in order for it to be ready for you to look at when you come in to work.

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

Chapter 16: *Working with many objects, one at a time*

NOTE: For this lab, you'll need any computer running PowerShell v3.

Try to answer the following questions and complete the specified tasks. This is an especially important lab because it draws on skills that you've learned in many previous chapters and that you should be continuing to use and reinforce as you progress through the remainder of this book.

1. What method of a ServiceController object (produced by Get-Service) will pause the service without stopping it completely?

2. What method of a Process object (produced by Get-Process) would terminate a given process?

3. What method of a WMI Win32_Process object would terminate a given process?

4. Write four different commands that could be used to terminate all processes named “Notepad,” assuming that multiple processes might be running under that same name.

Chapter 17: *Security Alert!*

NOTE: For this lab, you’ll need any computer running PowerShell v3.

Your task in this lab is simple—so simple, in fact, that we won’t even post a sample solution on MoreLunches.com. We want you to configure your shell to allow script execution. Use the Set-ExecutionPolicy cmdlet, and we suggest using the RemoteSigned policy setting. You’re welcome to use AllSigned, but it will be impractical for the purposes of this book’s remaining labs. You could also choose Unrestricted.

That said, if you’re using PowerShell in a production environment, please make sure that whatever execution policy setting you choose is compatible with your organization’s security rules and procedures. I don’t want you getting in trouble for the sake of this book and its labs!

Chapter 18: *Variables: a place to store your stuff*

NOTE: For this lab, you’ll need any computer running PowerShell v3.

Flip back to chapter 15 and refresh your memory on working with background jobs. Then, at the command line, do the following:

1. Create a background job that queries the Win32_BIOS information from two computers (use “localhost” twice if you only have one computer to experiment with).

2. When the job finishes running, receive the results of the job into a variable.

3. Display the contents of that variable.

4. Export the variable’s contents to a CliXML file.