Windows PowerShell 4.0
For the IT Professional - Part 1

Module 2: Commands 1

Student Lab Manual

Version 2.0

Contents

Lab 2: Commands

Introduction

This exercise will introduce you to PowerShell commands.

Objectives

After completing this lab, you will be able to:

- Launch the PowerShell console and Integrated Scripting Environment (ISE)
- Discover and Execute PowerShell commands
- Find help for PowerShell commands
- Use commands to perform simple administrative tasks

Prerequisites

Start all VMs provided for the workshop labs.

Logon to WIN8-WS as:

Username: **Contoso\Administrator**

Password: **PowerShell4**

Estimated time to complete this lab

45 minutes

NOTE: These exercises use many Windows PowerShell commands. You can type these commands into the Windows PowerShell Integrated Scripting Environment (ISE) or the Windows PowerShell console. For some exercises, you can load pre-typed lab files into the Windows PowerShell ISE, allowing you to select and execute individual commands. Each lab has its own folder under **C:\PShell\Labs\** on **WIN8-WS**.

Some exercises in this workshop may require running the Windows PowerShell console or ISE as an elevated user (Run as Administrator).

We recommend that you connect to the virtual machines (VMs) for these labs through Remote Desktop rather than connecting through the Hyper-V console. This allows you to use copy and paste between VMs and the host machine. If you are using the online hosted labs, then you are already using Remote Desktop.

Exercise 2.1: Cmd.exe vs PowerShell.exe

Introduction

Traditionally Microsoft Windows has included the command prompt (cmd.exe) to execute commands. Windows PowerShell provides backward compatibility to continue to use these legacy commands, or to use Windows PowerShell's native commands (Cmdlets).

Objectives

After completing this exercise, you will be able to:

- Execute Windows PowerShell from within the Windows command prompt
- Launch Windows PowerShell's own command console and ISE
- Use basic Windows PowerShell Cmdlets to get help

Task 2.1.1: Execute PowerShell from cmd.exe

1. Launch cmd.exe. Point to the lower left corner of the screen, and right-click the Start-button. Then select "Command Prompt (Admin)".
2. Type the following commands

```
cd c:\windows
dir
cls
ipconfig
ping 2012R2-MS
net share
```

3. Launch PowerShell.exe from within cmd.exe, by typing the following command

```
PowerShell
```

4. Type the following commands

```
cd c:\windows
dir
cls
ipconfig
ping 2012R2-MS
net share
```

5. Type "exit" to leave Windows PowerShell
6. Type "exit" again, to close the Windows Command prompt console.

Task 2.1.2: Launch Windows PowerShell from Taskbar

1. Click the PowerShell icon on the taskbar. This will launch the blue Windows PowerShell console.
2. Type the following Windows PowerShell commands:

```
Set-Location C:\Windows
Get-ChildItem
Clear-Host
Get-NetIPConfiguration
Get-NetIPConfiguration -Detailed
Test-Connection 2012R2-MS
Get-SMBShare
```

3. Notice the regularity of the "Cmdlet" names. This makes it easy to understand what any given Windows PowerShell Cmdlet will do.
    1. What do you think the Cmdlet name would be to get help? Write the command below then try it in Windows PowerShell.

       Get-Help

       ____

    2. What do you think the Cmdlet would be to get a list of commands? Write it below then try it in PowerShell.

       Get-Command

       ____

4. Type "exit" to close the Windows PowerShell console.

Task 2.1.3: Launch Windows PowerShell ISE from Taskbar

1. Right-click the PowerShell icon on the taskbar and select the "Run ISE as Administrator" option. This will launch the Windows PowerShell Integrated Scripting Environment.

> **NOTE:** The Windows PowerShell Integrated Scripting Environment (ISE) is a host application for Windows PowerShell.
>
> In Windows PowerShell ISE, you can run commands and write, test, and debug scripts in a single Windows-based graphic user interface with multiline editing, tab completion, syntax coloring, selective execution, context-sensitive help, and support for right-to-left languages.
>
> You can use menu items and keyboard shortcuts to perform many of the same tasks that you would perform in the Windows PowerShell console.

2. Type the following PowerShell commands in the blue command pane.

```
Get-
```

3. Wait for the IntelliSense drop down list to appear.
4. Type the letters "pro". Watch as IntelliSense filters the list of Cmdlets
5. Press <Enter> to select the highlighted Cmdlet.
6. Press <Enter> to execute the Cmdlet.
7. Type the following command

```
Get-Process –Name
```

8. Press the <Spacebar>. A list of currently running processes will appear in the drop down list.
9. Select the process "spoolsv" and press <Enter>

Exercise 2.2: Command Discovery

Task 2.2.1: Using Get-Command

Windows PowerShell contains thousands of built-in commands, known as Cmdlets. The ability to search for a relevant command is vital.

1. Type the following in the current Windows PowerShell ISE command pane, then press the <TAB> key to auto complete the Command name.

```
Get-Co
```

2. Press <Enter> to execute the Get-Command Cmdlet.
3. Type the same text 'Get-Co' and press the <TAB> key multiple times. How many Cmdlets appear?

Get-Co <TAB> through the commands
Get-Command, Get-ComputerRestorePoint, Get-Content, Get-ControlPanelItem, Get-Counter

4. Press the "Escape" button on your keyboard to clear the command prompt. Now type the following command and press <Enter>.

```
Get-Command *service
```

5. Write the command names returned, below.

    1.
    2.
    3.
    4.
    5.
    6.
    7.
    8.
    9.

6. Type the following commands.

```
Get-Command –Verb Suspend
Get-Command –Noun Computer
Get-Command –Name Get-Service
```

List all commands with the verb “Get”. Write the command used to achieve this below.

Get-Command –Verb Get

Task 2.2.2: Windows PowerShell Cmdlet Help

1. Type the following command.

```
Get-Help
```

2. After reading the output, how would you display the online help for the Get-Service Cmdlet? Write the command below.

Get-Help Get-Service -Online

NOTE: The VM must be internet-connected for the command to succeed.

3. Type the following commands and examine the outputs.

```
Get-Help Get-Service
Get-Help Get-Service –Examples
Help Get-Service -Examples
Get-Help Get-Service –Full
Get-Help Get-Service -ShowWindow
```

4. The last command opened a new dialog window that displayed help for the Get-Service cmdlet.
   a. Click the ‘Settings’ button
   b. Verify that the ‘Whole Word’ box is checked, then click “OK”
   c. Type the word ‘Include’ in the ‘Find’ text box.
   d. Note the yellow highlights that appear and find the help for the –Include Parameter
   e. Does the –Include parameter accept pipeline input?

No.
```
Accept pipeline input?       false
```

   f. Does this parameter accept wildcard characters?

Yes.
```
Accept wildcard characters?  true
```

5. Close the “Get-Service Help” dialog.

Exercise 2.3: Command Syntax

Task 2.3.1: Exploring Syntax

1. Type the following command in the Windows PowerShell ISE command pane.

```
Get-Service BITS
```

2. Type the following command and review the output

```
Get-Command Get-Service -Syntax
```

   a. What do the square brackets represent in the following help excerpt?

```
<string[]>
```

The parameter accepts multiple values (Array)

____

   b. In the first parameter set, is the –Name parameter *required* or *optional*? Explain why below.

Optional

____

3. Type the following command.

```
Get-Service -DisplayName background*
```

4. Write the service names returned, below.

   1. ______________________ BITS

   2. ______________________ BrokerInfrastructure

5. Type the following command:

```
Get-Service -DisplayName background* -name b*
```

The command fails. Does the –DisplayName parameter exist in the same parameter set as the –Name parameter?

No

____

6. Type the following command.

```
Add-Computer -?
```

Referring to the syntax for the Add-Computer Cmdlet, is it *required* that I enter a -DomainName parameter?

No. You need to enter a value for the -DomainName parameter

____

7. Write the names of some of the *switch parameters* available to the Add-Computer command.

1. ____
2. ____
3. ____
4. ____

Commented [A14]: 1.Force
2.Passthru
3.Restart
4.Unsecure
5.Whatif
6.Confirm

8. Type the following command and press <ENTER>

```
Get-Service –ComputerName
```

Based upon the error output, does the –ComputerName parameter accept multiple values? Explain why below.

Yes.
The parameter accepts multiple string values separated by a comma. System.String[]

____

9. Type the following command.

```
Get-Command Get-Service -Syntax
```

a) How would you describe a “parameter set”?

A subset of parameters with at least one that is unique to the set. Only these parameters may be used together.

____

b) How many parameter sets are shown for the Get-Service Cmdlet?

3

____

c) The syntax for all Cmdlet displays [<CommonParameters>].

   a. Type the following command.

   ```
   Help about_CommonParameters -showwindow
   ```

   b. List the two “Risk Mitigation” *switch parameters* available to all Cmdlets.

1.

2.

-whatif

-confirm

10. Type the following command.

```
Stop-Service –Name Spooler –Whatif
```

11. Did the service stop?

No

12. Type the following command.

```
Start-Process –FilePath notepad
```

13. What command would you use to stop the process and ensure Windows PowerShell prompts you for confirmation? Write the command below.

```
Stop-Process –Name notepad –confirm
```

Exercise 2.4: Command Aliases

Introduction

Aliases save time when typing at the Windows PowerShell console. As a best practice they should be avoided in scripts.

Task 2.4.1: Finding and using aliases

1. List all Cmdlets with the word “alias” by using the “Get-Command” Cmdlet.
2. Which Cmdlet will get all aliases? Execute this command and write the first three aliases below.

   1. __________
   2. __________
   3. __________

3. Type the following command.

```
Get-Help Get-Alias –Full
```

4. Find the parameter name that returns all aliases for a given Cmdlet name. Write the parameter name below.

   ____

5. Type the following commands.

```
Get-Alias -Definition get*
Get-Alias -Definition set*
Get-Alias -Definition new*
Get-Alias -Definition start*
Get-Alias -Definition stop*
```

6. List any aliases from the above output that match commands you have used in other shells.

   ____

   ____

Task 2.4.1: Creating aliases

1. What Cmdlet would you use to create a new alias?

Get-Alias

%, ?, ac

-Definition

dir, ls, pwd, cd, mount, sleep, start, kill

New-Alias

2. Does an alias called “reboot” exist?

No

___

3. Find the Cmdlet that will restart a computer. Write it below. Do not execute this Cmdlet yet!

Restart-Computer

___

4. Create a new alias for this command called “reboot” and write the command used to create the alias below.

New-Alias –Name Reboot –value Restart-Computer

___

5. Type the following command.

```
Reboot -whatif
```

6. Use the alias to reboot the remote computer named “2012R2-MS”