|  |  |  |  |
| --- | --- | --- | --- |
|  | 420-536 Windows Networking |  | Worth 1.5% |
| Winter 2017 |  |  |

Assignment #1a: Intro to PowerShell, part #1

In this lab you will configure and test Windows PowerShell. If you would prefer to use Windows 8 as the client system, please note that PowerShell 3 is already installed.

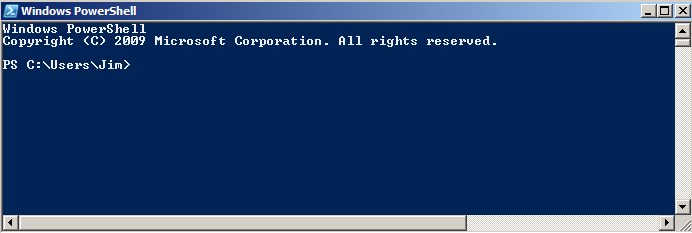
Note that for the purposes of this course, PowerShell 2 is sufficient, so if you have Windows 7 at home, you do not need to upgrade the shell to complete assignment #1.

**Starting PowerShell for the First Time**

So now you have PowerShell installed on your system. Let’s look at what is available. In the “Accessories” folder there should be two application for you to choose from, **‘Windows PowerShell’** and **‘Windows PowerShell ISE’.** The first is the text based interface and the second in the GUI interface. Let’s start with the first one.

**Opening the Command Shell**

Go to ‘**Start – All Programs – Accessories** and click on the selection **‘Windows PowerShell’**. You should see the **‘PowerShell’** command line interface, as illustrated below. Notice that it looks much like the regular cmd.exe program we have seen in Windows systems.



The command-line interface acts just like cmd.exe does, in that you can enter regular command line instructions.

**Try This:**

At the command prompt, type in

**ipconfig**

You should see the result of the ipconfig command.

Now Type:

**Get-command**

(you should see a listing of all PowerShell built-in commands)

This shows that the PowerShell acts just like the traditional command shell, but it can do a whole lot more. With PowerShell, you have a whole set of new commands and also have the ability to execute scripts in the form of **cmdlets.**

**For now, close the command shell.**

**The PowerShell GUI**

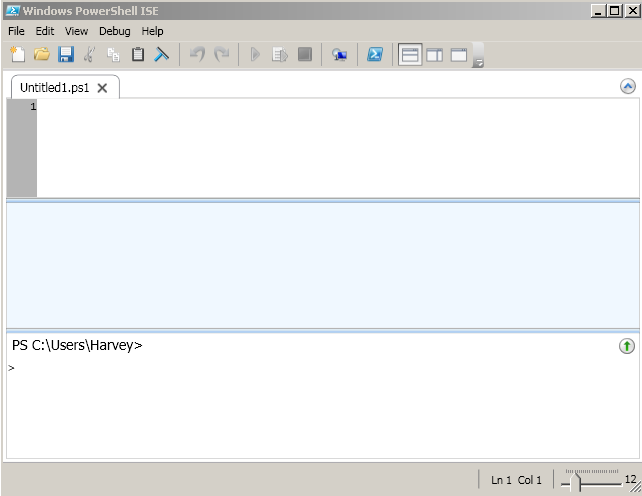
If you think the regular command shell is boring, you may find the GUI version more interesting. It provides a rich environment for creating, editing and executing PowerShell commands. To open it do the following;

**Opening PowerShell ISE** (‘Integrated Scripting Environment’)

Open the **‘Windows PowerShell ISE’** application. You should see the GUI command shell as illustrated below (note: you may need to configure the view settings if not all the panes are present in your environment).

**NOTE**: Windows Server does not have an execution pane while Windows 7/8 do.

**Script/Editor pane**, where you can create/edit PowerShell scripts



**Command pane** that acts just like the regular command shell.

**Execution pane**, where results of commands/scripts are displayed.

Just to test this GUI, let’s try a couple of things.

**Do This:**

In the **‘Command’** pane, type the **ipconfig** command at the prompt and press ‘Enter’.

# **Questions**

What happened when the command was executed? \_the command retrieved all network info from all network adapters.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Is this what you would expect to see from the Windows regular command shell? \_\_\_\_\_\_\_\_\_\_\_\_\_\_

In the **‘Script/Editor’** pane, type the following (do not type line numbers):

Clear (or you can use ‘cls’)

ipconfig



Now you need to execute the script. Look for the ‘Run Script’ Icon or press ‘F5” to execute the script.

**Question**

Was the output the same as the output created at the command prompt? no\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Type in the following commands/scripts to see what they do. We will be going into more detail about the command language in future labs.

Type this in the Editor and execute it (without the line numbers)

**1 cls**

**2 Get-Service**

Type this in the Editor and execute it.

**get-Service \* |where {$\_.Status -eq "Stopped"}**

What do the above scripts show? \_\_gets all processes that are stopped at the time of script execution

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

This is the ‘back-tick’ character. It is a line continuation character in PowerShell. It is used to denote that the following line is part of this one.

Type this in the Editor and execute it:

**get-Service \* |Sort-Object -property ServiceType `  
 | format-Table name, ServiceType, status, CanStop, -auto**

**DO THIS**

Type this in the Editor and execute it.

#This script is a bit longer

$strComputer = "."  
$colItems = get-wmiobject -class "Win32\_NetworkAdapterConfiguration" `  
-computername $strComputer | Where{$\_.IpEnabled -Match "True"}  
foreach ($objItem in $colItems) {  
   write-host "MAC Address : " $objItem.MACAddress  
   write-host "IPAddress : " $objItem.IPAddress  
   write-host "IPAddress : " $objItem.IPEnabled

Output lines

Here is the ‘back-tick’ character again..

write-host ""  
}

Properties

**Modify the Script**

Add additional output lines, to the above script, using the following properties

**.DNSServerSearchOrder**

**.IPSubnet**

**Getting Help**

PowerShell has an extensive help/getting started guide system. The easiest way to see these files is to press ‘**F1**’. As we will see, the ‘**Get- <command>,** command will display a help file for any PowerShell imbedded command or cmdlet .

In the PowerShell ISE press ‘F1’ You will open the ‘PowerShell Getting Started Guide’. It would be worth your while to browse through this document and get to know it. It contains valuable information about PowerShell and its’ components.

Also, PowerShell has a ‘get-help’ command. With this you can go directly to any command help file.

At the command prompt type: **Get-Help <enter>**

What was displayed? At line:1 char:10

+ Get-Help <enter>

+ ~

The '<' operator is reserved for future use.

+ CategoryInfo : ParserError: (:) [], ParentContainsErrorRecordException

+ FullyQualifiedErrorId : RedirectionNotSupported

At the Command Prompt Type: **Get-Help Alias <enter>**

To see all of the ‘**alias**’ commands, type the following: **get-alias <enter>**

Notice that many of the original “DOS’ commands have their PowerShell equivalents. Please note that the DOS commands do not always work the same as they would in the old ‘cmd’ shell. For example, while **DIR** works, **DIR /W** does not. The reason for this is that the DIR command is just the name (alias) given to PowerShell’s own cmdlet.

To get help about a command you would use ‘**get-help’** along with the command name.

Type in the following at the command prompt

**Get-help dir**

**Questions:**

Was the output what you expected to see? \_\_all the options for the dir command \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

What did ‘**get-help dir’** actually show you?

That dir is actually just an alias to another powershell command which makes transitioning between systems easier

\_\_all commands and parameters aswell as all aliases

And some remarks \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Type in **Get-alias dir.** This alias is associated with a main PowerShell command (definition).

What is the PowerShell command?

\_\_\_get-childitem

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Using the ‘GET-ALIAS’ command, look through the listing and find all of the Aliases for the answer to the previous question. Write down the names. (DIR is one of them)

\_\_\_gci ls dir\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Exercise**

**Using the PowerShell command (get-command) to list all of the "built in" commands, convert the following Bash/DOS commands to their PowerShell equivalent.**

|  |  |  |
| --- | --- | --- |
|  | **cls** | **Clear** |
|  | **ls** | **Get-childitem** |
|  | **cat myfile.txt** | **Get-Content** |
|  | **echo "This is a line of text"** | **Write-Host “This is a line of text”** |
|  | **ps** | **Get-process** |
|  | **man ls** | **Get-help ls** |
|  | **mv myfile.txt yourfile.txt** | **Move-Item myfile.txt yourfile.txt** |

**More Stuff**

In a typical operating system, files are read and programs executed before you see things on the screen. PowerShell also has the ability to execute script every time you open the shell (interface). This allows you to customize the PowerShell environment and execute script before anything else happens. You will see the importance of this when we look at **Persistent Aliases.**

Each PowerShell interface has its’ own profile. They are as follows;

Text Version - **Microsoft.PowerShell\_profile.ps1**

GUI Version - **Microsoft.PowerShellISE\_profile.ps1**

You can find these files (or where PowerShell thinks they should be, using the **$profile** variable)

**NOTE:** In Windows 7, these files have been created for you. Server 2012R2 typically does not. Therefore, you will have to create the files using notepad and place them in the folders specified by the **$Profile** variable. Make sure the file type is .**ps1** and not .txt.

Open the text based PowerShell Interface

At the command line type

**$profile**

Repeat the same command in the GUI editor

OK, it is assumed that you have the files in their proper locations. Now it’s time to test them out with a simple example.

Open **Microsoft.PowerShell\_profile.ps1**

and type the following

**Write-host “PowerShell rules the world!!”**

Save the file and open the appropriate interface.

Try this with both version of the shell

You may want to clear this line from the file

once you have verified that it works, unless you want

it to come up each time you open the shell.

If you received a big error message (which is good), you can fix it by starting PowerShell “Run as Administrator” and then type in the following;

**Set-executionpolicy unrestricted**

Answer “Yes” when prompted, close and re-open Powershell

When you have completed the above re-set the program by typing

**Set-executionpolicy restricted**

# Security, Commands, Cmdlets and Scripting Basics.

For starters, let’s try something

**DO THIS**

1. In the ‘GUI Editor’ type in and execute the following script

**cls**

**# PowerShell script to list processes and group by company**

**get-Process | sort company | format-Table ProcessName -groupby company**

1. Now open Notepad and type in the same script. Save this file to your drive. Name this file **‘ process.ps1’**
2. Now, back at the PowerShell ISE editor, open the **process.ps1** file and execute it.

What happened?

\_\_\_\_\_\_\_\_\_\_\_\_\_couldnt be run execution policy is not set \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

If you saw an error message, great! If you did not and you got the output, your interface may not be as secure as it should be.

**We will talk about PowerShell security in future labs.**

**Submission:** Submit this file to Lea with answers to the questions filled in.