Windows PowerShell 4.0
For the IT Professional - Part 1

Module 10: Providers 1

Student Lab Manual

Version 2.0

Conditions and Terms of Use

Contents

Lab 10: Providers

Introduction

Windows PowerShell providers expose data from a specialized data store so that you can view and manage it. The data that a provider exposes appears in one or more drives. You can access the data in a path as if you would on a hard disk drive. You can use any of the built-in cmdlets that the provider supports to manage the data in the provider drive. You can also use custom cmdlets designed especially for the data.

Objectives

After completing this lab, you will be able to:

- Discover the built-in Windows PowerShell providers
- Navigate the built-in Windows PowerShell providers
- Create new Windows PowerShell provider drives

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 10.1: Providers and Drives

Introduction

Windows PowerShell has eight built-in Providers. In this exercise you will explore, navigate and create providers and drives using Windows PowerShell provider Cmdlets.

Objectives

After completing this exercise, you will be able to:

- List Providers
- Navigate within and between providers
- Create new provider drive instances

Task 10.1.1: Providers and Drives

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.
2. Use the Get-PSProvider Cmdlet to view all of the currently installed providers.

```
Get-PSProvider
```

3. Display the provider instances. These are referred to as Drives.

```
Get-PSDrive
```

4. Drives can be accessed using the –Path parameter on an *-*Item Cmdlet.

```
Get-ChildItem –Path Function:
```

5. List the contents of each provider drive using Get-ChildItem. You must append a colon (:) to the end of each drive name.

____

6. *-*Item cmdlets have a –Filter parameter. The parameter limits items returned from the provider. The –Filter parameter will only work with a provider that supports the filtering.
   List the providers again. In the “Capabilities” column, which provider has the “Filter” capability?

____

Get-PSProvider |
Where-Object {$_.Capabilities -Like "Filter*"}

FileSystem
ActiveDirectory

7. Filter file names from C:\Windows\System32 starting with “a” and ending in “.exe”. Type the command below.

```
Get-ChildItem –Path C:\Windows\System32 –Filter a*.exe
```

8. List the providers again. What “Registry” provider drive names can you find in the output? Write them below.

   1. ______________________________

   2. ______________________________

{HKLM, HKCU}

9. List the child items in one of the registry drives.

```
Get-ChildItem –Path HKLM:
```

10. Are you able to use the –Filter parameter? Explain why this is the case below.

NO. This provider does not support -Filter

____

____

____

11. Add a new Registry provider drive, called “HKCR” that references the “HKEY_CLASSES_ROOT” registry hive.

```
New-PSDrive -Name HKCR -PSProvider Registry -Root HKEY_CLASSES_ROOT
```

12. List the providers. The new “HKCR” drive should be visible.
13. List the top-level items in this drive using the Get-ChildItem Cmdlet. This make take a while to complete!

```
Get-ChildItem –Path HKCR:\
```

14. Create a new FileSystem provider drive called “Scripts”. Set the root path to C:\PShell\Labs.
15. Use the New-PSDrive Cmdlet to create a FileSystem provider drive called “N” whose root path is \\contoso.com\netlogon. This time, add the –Persist switch parameter.

-persist parameter can only be used for the remote filesystem location

16. Close your current Windows PowerShell session (console or ISE where you executed the above steps).
17. Reopen the Windows PowerShell ISE & list the provider drives. Of the drives created in steps 14 and 15, which drive(s) have persisted into the new Windows PowerShell session? Why do you think this is?

Those created using the –Persist switch

____

____

18. Find a Cmdlet to remove a provider drive. Use it to delete the drive that persisted. Write the command you used below.

Remove-PSdrive

____

Task 10.1.2: Navigating providers

1. Navigating between and within providers can be achieved easily using the familiar "cd" command.
   Set your current working directory to C:\Windows.

```
cd C:\Windows
```

2. Find your current working directory using the "pwd" command alias.

```
pwd
```

3. List the actual Cmdlet names executed when each of these aliases are used. Write their names below.

```
Get-Alias –Name cd, pwd
```

cd :

pwd :

Set-Location

Get-Location

4. Providers present data in a consistent format that resembles a file system drive.
   Use "Set-Location", or the alias "cd" to navigate to the following path:
   HKLM:\Software\Microsoft\PowerShell\3.
5. List the items in the current working directory.

```
Get-ChildItem
Get-Alias –Definition Get-ChildItem
dir
ls
```

6. List all of the current working directory's path property values.

```
Get-Location | Format-List –Property *
```

   Notice that the "Path" and "ProviderPath" property values are slightly different. The "ProviderPath" value is the path used by the native data store.
7. Use the reg.exe external command to query the registry using the "ProviderPath" string. Type the following command.

```
Reg.exe query HKEY_LOCAL_MACHINE\Software\Microsoft\powershell\3
```

8. Any Cmdlet with a –Path parameter can accept a provider path, so we do not have to change our current working directory in order to manipulate items in other provider data stores.
9. Display the current working directory path.
10. We can easily list certificates in the LocalMachine\My certificate store, using the Get-ChildItem Cmdlet, while our current working directory is located in another provider path.

```
Get-ChildItem –Path Cert:\localmachine\my
```

NOTE: When navigating between providers or accessing them through a provider path, the provider drive names must have a colon (:) character appended to them.

11. Forward and backward slashes can be used interchangeably with any provider path

```
Get-ChildItem -Path C:\Windows/System32\spool
```

12. Providers accept relative paths. A dot character (.) followed by a forward or backslash indicates a relative path.
13. Type the following commands.

```
Set-Location C:\Windows\system32
Set-Location .\LogFiles
```

To move up a level in the data store hierarchy, use the same double-dot (..) syntax as in the Windows Command shell. Moving to the top-level folder in a provider also uses the same cmd.exe style "\" syntax to represent the root container. Experiment typing the commands below and notice the command prompt path change automatically.

```
Set-Location C:\Windows\System32\Drivers
Set-Location ..
Get-Location
Get-ChildItem
Set-Location \
```

14. Note that if a path includes spaces the entire string must be enclosed in single or double quotes.

```
Get-ChildItem 'C:\Program Files'
```

NOTE: If you use Tab completion for provider paths, Windows PowerShell will automatically enclose the path in quotes if it contains spaces.

15. Using the correct provider and the Measure-Object and Where-Object Cmdlets, construct pipeline commands to solve each of the tasks below. Remember to first use Get-Member to discover the type of objects you are working with.

a) Measure the total number of child items in the "function:" PSdrive
b) Return the environment variable where the "Name" of the variable is "COMPUTERNAME".
c) Return the environment variable where the "Name" of the variable is "PROCESSOR_IDENTIFIER"
d) Using the –Recurse parameter, find all certificates in the certificate store where the "ThumbPrint" property equals the following string "CDD4EEAE6000AC7F40C3802C171E30148030C072".

Get-ChildItem -Path function: | Measure-Object

Get-ChildItem -Path Env: | Where Name -eq "COMPUTERNAME"

Get-ChildItem -Path Env: | Where Name -eq "PROCESSOR_IDENTIFIER"

Get-ChildItem -Path Cert: -Recurse | Where thumbprint -eq "CDD4EEAE6000AC7F40C3802C171E30148030C072"

Exercise 10.2: *-Item Cmdlets

Introduction

The *-Item Cmdlets can be used to create, enumerate, modify and remove any object in any provider, using the same syntax and methodology.

Objectives

After completing this exercise, you will be able to:

- Manipulate objects in the FileSystem provider with the *-Item Cmdlets.
- Manipulate objects in the Registry provider with the *-Item Cmdlets.

Scenario

The IT team is required to create a script that will copy all documents more than 30 days old from each user's Documents folder to a local archive folder. The file extensions of the identified files should be changed. The script should demonstrate using the *-Item Cmdlets and the Windows PowerShell pipeline. Also, ensure the script will only run once per day.

Task 10.2.1: File System– Creating Files and Folders

1. Open the script C:\PShell\Labs\Lab_10\Create-FilesAndFolders.ps1 in the Windows PowerShell ISE. Press 'play' or press <F5> to execute the script. The script listed below will create new files and folders in the current user's Documents folder.

```
$currentUser = Get-Item Env:\USERNAME
$myDocs = "C:\Users\$($currentUser.value)\Documents"

New-Item -Path $myDocs\Finance -ItemType Directory
New-Item -Path $myDocs\Finance\Public -ItemType Directory
New-Item -Path $myDocs\Finance\Private -ItemType Directory

1..10 |
Foreach-Object {
New-Item –Path $myDocs\Finance\Public –Name Public-$_.txt –ItemType file
New-Item –Path $myDocs\Finance\Public –Name Public-$_.log –ItemType file
New-Item –Path $myDocs\Finance\Private –Name Private-$_.txt –ItemType file
New-Item –Path $myDocs\Finance\Private –Name Private-$_.log –ItemType file
}
```

2. Confirm the file and folder structure exists in your Documents folder on the WIN8-WS virtual machine.

Task 10.2.2: File System – Manipulating Files and Folders

1. Open and review the following script C:\PShell\Labs\Lab_10\Archive-Files.ps1.
2. First, the script creates a new folder on the remote machine 2012R2-MS with the path "C:\DocumentBackup". Note that the –Path parameter accepts UNC paths.

```
New-Item -Path \\2012R2-MS\C$\DocumentBackup -ItemType directory –Force
```

3. Share the new folder on the remote machine 2012R2-MS with the share name "Backup". Grant full access to the "Contoso\Domain Users" group.

```
New-SmbShare -CimSession (New-CimSession -ComputerName 2012R2-MS) `
-Path C:\DocumentBackup -Name Backup -FullAccess "Contoso\Domain Users"
```

4. The next command joins the current user's home directory path with the 'Documents' directory and saves the resulting path in the $docs variable.

```
Join-Path -Path $HOME -ChildPath Documents -OutVariable docs
```

5. Create a new subfolder in the share with the same name as the current user and store the new folder object in the variable $myFolder for later use.

```
New-Item –Path \\2012R2-MS\Backup -Name $env:USERNAME -ItemType directory `
–OutVariable myFolder -Force
```

6. Create a new local sub folder called 'Archive' in C:\Users\<username>\Documents

```
New-Item -Path $docs\Archive -ItemType Directory -Force
```

7. Create a new PSDrive called "MyBackup" using the filesystem provider. This PSDrive will point to the new folder in the remote share.

```
New-PSDrive –Name MyBackup –root $myFolder.FullName –Psprovider filesystem
```

8. Set the LastWriteTime of all files to 60 days before the current date

```
Get-ChildItem -Path $docs\Finance -Recurse |
Set-ItemProperty -Name LastWriteTime -Value (Get-Date).AddDays(-60)
```

9. Write the current user's name to all files with the .txt file extension

```
Get-ChildItem -Path $docs\Finance -Filter *.txt -Recurse |
Set-Content -Path {$_.fullname} -Value $env:USERNAME
```

10. Find all files modified more than a month ago and copy them to C:\Users\<username>\Archive\Finance.

```
Get-Childitem -File -Path $docs\Finance -Recurse |
Where LastWriteTime -lt (Get-Date).addDays(-30) |
Foreach-Object {

New-Item -Path $docs\Archive\Finance\$($_.Directory.Name) -ItemType container `
-ErrorAction SilentlyContinue

Copy-Item -Path $_.FullName -Destination
$docs\Archive\Finance\$($_.Directory.Name)
}
```

11. Rename the copied files file extensions from '.log' to '.old'. Notice that the –NewName parameter's argument is a script block containing the String type's replace() method.

```
Get-ChildItem -Path $docs\Archive\Finance -File -Recurse  |
Rename-Item -NewName { $_.fullname.replace(".log",".old") }
```

12. Finally, the script finds the renamed files and copies them to the remote share on 2012R2-MS using the Background Intelligent Transfer Service (BITS). The Start-BitsTransfer Cmdlet requires the source and destination paths of the files.

```
Get-Childitem $docs\Finance -Recurse |
ForEach-Object { Start-BitsTransfer -Source $_.FullName -Destination
MyBackup:\$_ }
```

13. Run the script C:\PShell\Labs\Lab_10\Archive-Files.ps1 by pressing the 'play' button or pressing <F5> on the keyboard.

14. In the Windows PowerShell ISE command pane confirm the file transfer was successful by listing the files transferred to the shared folder on 2012R2-MS.

```
Get-ChildItem -Path MyBackup:\
```

15. The complete C:\PShell\Labs\Lab_10\Archive-Files.ps1 listing is shown below.

```
# Create a new folder on 2012R2-MS
New-Item -Path \\2012R2-MS\C$\DocumentBackup -ItemType directory –Force

# Share the folder as 'Backup'
New-SmbShare -CimSession (New-CimSession -ComputerName 2012R2-MS) `
-Path C:\DocumentBackup -Name Backup -FullAccess "Contoso\Domain Users" `
-ErrorAction SilentlyContinue

# Create a new variable
Join-Path -Path $HOME -ChildPath Documents -OutVariable docs

# Create a directory called 'myFolder' on the 'Backup' remote share
New-Item –Path \\2012R2-MS\Backup -Name $env:USERNAME -ItemType directory `
–OutVariable myFolder -Force

# create a local 'Archive' folder within the current user's 'Documents' directory
New-Item -Path $docs\Archive -ItemType Directory -Force

# Map a new temporary network drive
New-PSDrive –Name MyBackup –root $myFolder.FullName –Psprovider filesystem `
-ErrorAction SilentlyContinue

# Set the LastWriteTime file property to 60 days in the past for local files
Get-ChildItem -Path $docs\Finance -Recurse |
Set-ItemProperty -Name LastWriteTime -Value (Get-Date).AddDays(-60)

# Add the current username value to each .txt file to modify the LastWriteTime
property
Get-ChildItem -Path $docs\Finance -Filter *.txt -Recurse |
Set-Content -Path {$_.fullname} -Value $env:USERNAME
```

```
# Get all files modified more than a month ago
Get-Childitem -File -Path $docs\Finance -Recurse |
Where LastWriteTime -lt (Get-Date).addDays(-30) |
Foreach-Object {

# Preserve folder structure
New-Item -Path $docs\Archive\Finance\$($_.Directory.Name) -ItemType container `
-ErrorAction SilentlyContinue

# copy the files to the 'Finance\Archive' directory
Copy-Item -Path $_.FullName -Destination
$docs\Archive\Finance\$($_.Directory.Name)
}

# Rename file extensions from '.log' to '.old'
Get-ChildItem -Path $docs\Archive\Finance -File -Recurse  |
Rename-Item -NewName { $_.fullname.replace(".log",".old") }

# Use BITS to copy the files & directories to the share on2012MS
Get-Childitem $docs\Archive -Recurse  |
ForEach-Object { Start-BitsTransfer -Source $_.fullname -Destination MyBackup:\$_
}
```

Task 10.2.3: Registry Operations

1. The script is required only to execute once a day.
   To implement this functionality in the C:\PShell\Labs\Lab_10\Archive-Files.ps1 script we can store the last script execution time in the registry.
   Execute the following code to create a new registry key called "FileArchiveScript", in the HKEY_CURRENT_USER\Software key.

```
New-Item –Path HKCU:\Software –Name FileArchiveScript
```

2. Next, create a new registry item property in the new key. The registry value is yesterday's date as a short date string.

```
New-ItemProperty –Path HKCU:\Software\FileArchiveScript –Name LastExecutionDate `
-Value (Get-Date).AddDays(-1).ToShortDateString()
```

4. The first action our script should take is to check whether the script has already executed today. We can read the registry property we just created, using the registry provider. Save the value in a new variable called $lastExecution.

Add the code below as the ***first*** line of the C:\PShell\Labs\Lab_10\Archive-Files.ps1 file.

```
Get-ItemProperty –Path HKCU:\Software\FileArchiveScript –Name lastExecutionDate `
-OutVariable lastExecution
```

3. Next, we need to compare the current date with the date stored in the registry. If the current date is equal to the date stored in the registry, the script should not execute. Add the following code to the next line of the script.

```
If ( (Get-Date).ToShortDateString() –eq $lastExecution.LastExecutionDate ) {break}
```

NOTE: We shall explore "if" statements and the "break" flow control keyword in a later module.

4. If the files successfully transferred to the remote share, we need to set the registry value. Add a command to the ***last*** line of the C:\PShell\Labs\Lab_10\Archive-Files.ps1 script that assigns the current date to the "LastExecutionDate" registry item property.

```
Set-ItemProperty –Path HKCU:\Software\FileArchiveScript –Name lastExecutionDate –
Value `
(Get-Date).ToShortDateString()
```

5. Save the script C:\PShell\Labs\Lab_10\Archive-Files.ps1.
6. Run the script and verify it carried out the file operations successfully.

Exercise 10.3: *-Path Cmdlets

Introduction

To make our script more robust we can use the *-Path Cmdlets to check if paths are valid before we use them. We can also manipulate paths as objects and even resolve absolute paths from relative paths. Splitting and joining paths using string manipulation is rather error prone. Windows PowerShell exposes paths as objects and supplies Cmdlets to manipulate them safely.

Objectives

After completing this exercise, you will be able to:

- Test paths exist before manipulating them
- Manipulate paths as objects, rather than strings

Task 10.3.1: Testing paths

1. If not already open, open C:\PShell\Labs\Lab_10\Archive-Files.ps1 in the Windows PowerShell ISE.
2. The first line of the script reads a registry value. The next line determines whether the script has already run today. To make the script more robust, we can use the Test-Path Cmdlet to check whether the registry key exists before trying to read or write its property value.

   Type the command below. What value is returned?

```
Test-Path HKCU:\Software\FileArchiveScript
```

3. The **if** statement can be used to determine whether the Test-Path Cmdlet returned true or false. Type the command below to view the statement's behavior (statements are covered in more detail in a later module).

```
If (Test-Path HKCU:\Software\FileArchiveScript) { “The Registry key exists” }
```

4. The -not operator negates boolean (True/False) data types. If the result of Test-Path is False, the operator reverses this to True, executing the code in the script block and creating the registry key and value.
   Add the following code above the first line of the script.

```
If ( -not (Test-Path  HKCU:\Software\FileArchiveScript) )
{
New-Item HKCU:\Software\FileArchiveScript
New-ItemProperty -Path HKCU:\Software\FileArchiveScript -Name lastExecutionDate `
-Value (Get-Date).ToShortDateString()
}
```

5. Save the script to C:\PShell\Labs\Lab_10\Archive-Files.ps1.

Task 10.3.2: Split-Path, Join-Path & Resolve-Path

1. Paths can be split into their constituent parent paths (-Parent) and file names (-Leaf). Type all of the following commands and review their output.

```
Get-Item –Path C:\Windows\WindowsUpdate.log | Split-path –Parent
Get-Item –Path C:\Windows\WindowsUpdate.log | Split-path -leaf
```

2. Split-Path also has the ability to resolve wildcards.

```
Split-Path -Path c:\windows\sys* -Resolve –leaf
```

3. Split-Path can also detect when relative paths are used.

```
Set-Location –Path C:\PShell\Labs
Split-Path ./Lab_10 –IsAbsolute
```

4. Join-Path allows path manipulation without having to split strings using path separator characters (/ , \ ).

```
Join-Path –Path C:\Windows –ChildPath System32
```

5. Join-Path’s –Resolve switch parameter returns fully qualified paths from wildcard matches.

```
Join-Path –Path C:\Win* -ChildPath Sys*
```

6. The Resolve-Path Cmdlet resolves wildcard characters into provider paths.

```
Resolve-Path –Path HKLM:\software\m*
Resolve-Path –Path ~ # tilde represents the home path of a file system drive
```

7. Convert an absolute path, to a relative path

```
Set-Location –Path C:\Windows
Resolve-Path –Path C:\Windows\System32 -Relative
```