

Windows PowerShell v4.0 For the IT Professional - Part 1

Module 5: Scripts

Student Lab Manual

Version 2.0

Conditions and Terms of Use

Microsoft Confidential - For Internal Use Only

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2014 Microsoft Corporation. All rights reserved.

Copyright and Trademarks

© 2014 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see Use of Microsoft Copyrighted Content at <http://www.microsoft.com/about/legal/permissions/>

Microsoft®, Internet Explorer®, and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

Contents

LAB 5: SCRIPTS 5

 EXERCISE 5.1: WRITING SCRIPTS 6

 Task 5.1.1: Script parameters 6

 Task 5.1.2: Adding script comments & help..... 9

 EXERCISE 5.2: RUNNING SCRIPTS 11

 Task 5.2.1: Executing scripts from outside of PowerShell 11

 Task 5.2.2: Unblocking scripts..... 12

 Task 5.2.3: Signing Scripts..... 12

Lab 5: Scripts

Introduction

Scripts are essential for packaging commands into a reusable form and for scheduling their execution. A script can contain any valid Windows PowerShell commands, including single commands, commands that use the pipeline, functions, and control structures such as If statements and For loops. To write a script, start a text editor (such as Notepad) or a script editor (such as the Windows PowerShell ISE). Type the commands and save them in a file with a valid file name and the .ps1 file name extension.

Objectives

After completing this lab, you will be able to:

- Write a simple script
- Add parameters, help and comments to a script
- Execute scripts from within and outside of Windows PowerShell
- Digitally sign a script & modify script execution behavior

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

30 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 5.1: Writing scripts

Task 5.1.1: Script parameters

1. Scripts accept named and unnamed parameters in exactly the same way as script blocks and functions. Scripts are essentially script blocks without the bounding curly braces '{ }'.
2. Create a new script in the Windows PowerShell ISE Script pane. The command below executes the Best Practice Analyzer (BPA) for Directory Services remotely on the 2012R2-DC (Domain Controller). Add it to the new script in the script pane.

```
Invoke-Command -ScriptBlock {
    Invoke-BPAModel -ModelId "Microsoft/Windows/DirectoryServices"
} -ComputerName 2012R2-DC
```

3. Add another block of code into the script pane. This gets the results of the Directory Services BPA, converts it to Html format, and saves it in an .htm file.

```
Invoke-Command -ScriptBlock {
    Get-BpaResult -ModelId "Microsoft/Windows/DirectoryServices"
} -ComputerName 2012R2-DC | ConvertTo-Html |
Out-File -FilePath C:\PShell\Labs\Lab_5\DS-BpaReport.htm
```

4. Click "File" -> "Save As" to save the script in C:\PShell\Labs\Lab_5\Create-BpaReport.ps1
5. Execute the script by pressing the 'Play' button (F5) in the ISE's menu bar. What happens? Write the error message below.

6. A fresh install of Windows PowerShell will **not** allow scripts to run by default. View the current script execution policy by typing the Cmdlet Get-ExecutionPolicy the ISE command pane.

```
Get-ExecutionPolicy
```

7. Ensure you are running Windows PowerShell elevated (with Administrator privileges). Type the command below to relax the default execution policy and allow the script to run. Click "Yes" when prompted.

"... cannot be loaded because running scripts is disabled on this system."

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
```

8. Run the script again by pressing the 'Play' button. Once the command successfully completes, open the output html file C:\PShell\Labs\Lab_5\DS-BpaReport.htm by double-clicking it from Windows Explorer.
9. From the taskbar, open the Windows PowerShell console (not the ISE) and change your current working directory to C:\PShell\Labs\Lab_5

```
Set-Location C:\PShell\Labs\Lab_5
```

10. Execute the script again by typing its name "Create-BpaReport.ps1" in the Windows PowerShell console. Does this successfully execute the script? If not, what error is returned?

NO. "... is not recognized as the name of a cmdlet, function, script file, or operable program"

11. Retype the script name by either adding the relative path prefix (.\), or specifying the fully qualified path to the script. The script should now run successfully.
12. The Create-BpaReport.ps1 script successfully collects BPA results, but it is hard coded to a particular server name and BPA model Id. To make the script more flexible, we can add parameters to it.
13. Modify the script to add a Param() statement containing three named parameters - \$serverName, \$modelId and \$filePath.

NOTE: Since the script block for each Invoke-Command Cmdlet is executed on a remote machine, we need to prefix each value passed to the script block with the 'Using:' modifier. This will allow the local parameter value to be passed to the script block, running on the remote machine.

```
param($serverName, $modelId, $filePath)

Invoke-Command -ScriptBlock {
    Invoke-BPAModel -ModelId $using:modelId
} -ComputerName $serverName

Invoke-Command -ScriptBlock {
    Get-BpaResult -ModelId $using:modelId
} -ComputerName $serverName |
ConvertTo-Html |
Out-File -FilePath "$filePath\DS-BpaReport.htm"
```

14. Run the script, in the Windows PowerShell console, using the named parameters (this is a single line command).

```
.\Create-BPAReport.ps1 -modelId "Microsoft/Windows/DirectoryServices" `
-serverName "2012R2-DC" -filePath C:\PShell\Labs\Lab_5
```

NOTE: In the ISE command pane, the back tick character will allow line continuation only when followed by the <SHIFT> <ENTER> key combination.

Task 5.1.2: Adding script comments & help

1. Commenting your scripts is a best practice. Formally describing the actions that your script is supposed to be taking makes it far easier for others to understand and maintain your scripts.

Windows PowerShell defines two comment types:

1. Single inline comment
2. Multi-lined, or block comments.

```
# Single line comment character

<#
    block comment tags
#>
```

2. Add comments to the previous script using both methods shown above. Save the resulting script file as C:\PShell\Labs\Lab_5\Create-BpaReport.ps1
3. Windows PowerShell has a very mature help system that your scripts and functions can use to document their functionality and aid their discoverability.

NOTE: To review the conceptual help topic for comment based help, type the command below.

Get-Help about_Comment_Based_Help

Comment-based help employs block comments and a number of dot-prefixed keywords. For comment-based help, block comments must be placed at the beginning of the script file and may only be preceded by blank lines or other comments.

Windows PowerShell will auto generate help content for the following areas:

- Name
- Syntax
- Parameter list
- Common Parameters
- Parameter Attribute table

A number of keywords exist that Windows PowerShell uses to build the help.

```
<#
.Synopsis
    Short description
.DESRIPTION
    Long description
.EXAMPLE
    Example of how to use this cmdlet
.EXAMPLE
    Another example of how to use this cmdlet
.INPUTS
    Inputs to this cmdlet (if any)
.OUTPUTS
    Output from this cmdlet (if any)
```

```
.NOTES
    General notes
.COMPONENT
    The component this cmdlet belongs to
.ROLE
    The role this cmdlet belongs to
.FUNCTIONALITY
    The functionality that best describes this cmdlet
#>
```

4. Add the above section to the top of the script C:\PShell\Labs\Lab_5\Create-BpaReport.ps1.

NOTE: You may find it easier to add the help text using the ISE Snippet feature. Press CTRL+J in the ISE. Select the second item in the list, "Cmdlet (advanced function) – complete". Delete any unneeded template text.

5. Add text underneath each dot-prefixed keyword to further describe the script
6. Save the changes to your script, use Get-Help to view it. If you have updated the script correctly, the help will appear when typing the commands below.

```
Get-Help C:\PShell\Labs\Lab_5\Create-BpaReport.ps1
Get-Help C:\PShell\Labs\Lab_5\Create-BpaReport.ps1 -Detailed
Get-Help C:\PShell\Labs\Lab_5\Create-BpaReport.ps1 -Examples
Get-Help C:\PShell\Labs\Lab_5\Create-BpaReport.ps1 -Full
```

Exercise 5.2: Running scripts

Introduction

Running a script is a lot like running a cmdlet. You type the path and file name of the script and use parameters to submit data and set options. You can run scripts on your computer or in a remote session on a different computer.

Objectives

After completing this exercise, you will be able to:

- Execute scripts from within and outside of Windows PowerShell
- Allow scripts downloaded from a remote source to execute
- Use a code signing certificate to digitally sign a script

Task 5.2.1: Executing scripts from outside of PowerShell

1. Open an elevated (Run as Administrator) Windows command prompt (cmd.exe).
2. Execute the script from the previous exercise, from within cmd.exe.

```
C:\> PowerShell.exe -File "C:\PSHell\Labs\Lab_5\Create-BpaReport.ps1" -modelId
"Microsoft/Windows/DirectoryServices" -serverName "2012R2-DC" -filePath
C:\PSHell\Labs\Lab_5
```

3. PowerShell.exe is the text-based Windows PowerShell host application. It has a number of parameters, which can control its behavior. List them by typing the following command in cmd.exe

```
C:\> PowerShell.exe -?
```

4. How would you execute a script block containing the command "Get-Process" from cmd.exe using PowerShell.exe? Write the command below.

```
C:\> PowerShell.exe -Command "& {Get-Process}"
```

```
PowerShell.exe -Command "& {Get-Process}"
```

5. How would you prevent the Windows command prompt from closing once the command has completed?

```
PowerShell.exe -NoExit -Command "& {Get-Process}"
```

Task 5.2.2: Unblocking scripts

In this task, we will investigate the effect of modifying the script execution policy. The “AllSigned” and “RemoteSigned” execution policies prevent Windows PowerShell from running scripts that do not have a digital signature.

1. Launch a new elevated instance of Windows PowerShell ISE
2. Copy the following script from [\\2012R2-MS\\RemoteScripts\\Get-NetAdapters.ps1](#) to C:\PShell\Labs\Lab_5\Get-NetAdapters.ps1
3. Set the execution policy to “RemoteSigned”.
4. Try to execute the script from the Windows PowerShell ISE command pane. What happens?
5. We can allow a downloaded script to run by unblocking it.

Use either of the two methods below to do this:

- a. Locate the script in Windows Explorer and right-click it, then select “Properties”. Click the “Unblock” button.
- b. Using the Unblock-File Cmdlet

```
Unblock-File -Path C:\PShell\Labs\Lab_5\Get-NetAdapters.ps1
```

6. Close and re-open the Windows PowerShell ISE.
 7. Run the script again. Does it now run correctly?
-

Task 5.2.3: Signing Scripts

Scripts can be digitally signed to ensure their integrity and the security of your system. Modifying the script execution policy to “AllSigned” will prevent the execution of unsigned scripts, but will not encrypt their contents or stop commands from being copied into the Windows PowerShell console or ISE and executed.

1. Change the current Windows PowerShell execution policy to “AllSigned”. Ensure the Windows PowerShell console or ISE is running elevated as an administrative user before attempting this. Click “Yes” if prompted.

```
Set-ExecutionPolicy -Executionpolicy AllSigned
```

2. Try to run the script C:\PShell\Labs\Lab_5\Get-NetAdapters.ps1 from the Windows PowerShell ISE command pane and note the error returned.
3. Open the script to confirm there is no Authenticode signature.
4. Sign the script to allow it to run. A code-signing certificate was previously distributed to the WIN8-WS virtual machine. Type the command below to find and display the code-signing certificate.

```
Get-ChildItem -Path Cert:\ -Recurse -CodeSigningCert
```

5. The code-signing certificate is located in Cert:\CurrentUser\My
Get this certificate so that we can use it to sign our script module.

```
$codeSignCert = Get-ChildItem -Path Cert:\CurrentUser\My -CodeSigningCert
```

6. Sign the script with the certificate using the Set-AuthenticodeSignature Cmdlet.

```
Set-AuthenticodeSignature -FilePath C:\PShell\Labs\Lab_5\Get-NetAdapters.ps1 `
-Certificate $codeSignCert
```

7. Examine the newly signed script to find the signature at the bottom of the file.

```
Get-Content C:\PShell\Labs\Lab_5\Get-NetAdapters.ps1
```

8. Try to run the script C:\PShell\Labs\Lab_5\Get-NetAdapters.ps1 from the Windows PowerShell ISE command pane.

Because the code signing certificate has not been added to the local TrustedPublishers store, the first time the script is run you will be prompted. Click the “Always Run” button.

9. Finally, reset the execution policy to “RemoteSigned”.

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned
```