Scripts

Learnings covered in this Unit

- Write and run script files
- Understand how execution policies can prevent running scripts
- Uses of comments in scripts
- Understand command precedence

Scripts

What are Scripts?

Text file (.ps1) containing one or more PowerShell commands

Simple ‘code packaging’ for distribution purposes and later use

Supports all features a function does:

- Accepts parameters
- Returns values
- Leverages help syntax

Can also be digitally signed for security

Simple Script Example

Windows PowerShell ISE

File Edit View Tools Debug Add-ons Help

Sample.ps1

```
1 Write-Host ‘Start of script’ -BackgroundColor Green
2 Write-Host ‘Display the % CPU Time utilization by the ISE’ -BackgroundColor Green
3 Get-Counter ‘\Process(powershell_ise)\% Processor Time’
4
5
6
7
8
9
10
11
12
```

```
PS C:\scripts> C:\scripts\Sample.ps1
Start of script
Display the % CPU Time utilization by the ISE

Timestamp                 CounterSamples
---------                 --------------
6/25/2018 11:22:39 AM     \\WIN10\process(powershell_ise)\% processor time :
                          14.0038797593275
```

Demonstration

PowerShell Scripts

Launching a script

Running Powershell Scripts

From Command Line:

Full path and file name

```
PS C:\> c:\scripts\script.ps1
```

Relative path

```
PS C:\Scripts> .\script.ps1
```

Spaces in path (use tab completion)

```
PS C:\> & "c:\scripts\my script.ps1"
```

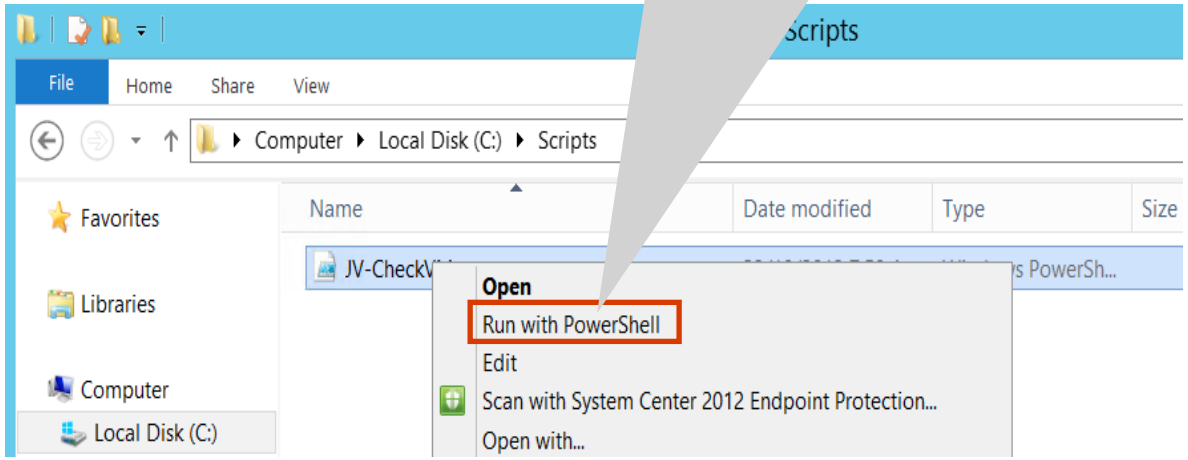
Script is in environment path

```
PS C:\> Script.ps1
```

From GUI:

Script files cannot be run by double-clicking

1. Right-click script
2. Select “Run with Powershell”



Script Param Statement

- Must be first statement in script, except for comments
- Parameter values are available to commands in scripts

Param Statement

ScriptParamExample.ps1

```
1  param ($ComputerName)
2
3  $result = Test-Connection -ComputerName $ComputerName -Quiet -Count 1
4
5  Write-Host $result -ForegroundColor Green
```

Parameter

Parameter Value

```
PS C:\scripts> .\ScriptParamExample.ps1 -ComputerName localhost
True

PS C:\scripts> .\ScriptParamExample.ps1 -ComputerName DoesNotExist
False
```

Demonstration

Running Scripts

Execution Policies

Execution Policy

Determines conditions under which PowerShell will run scripts

Can be set for:

- Local computer
- Current user
- Specific Powershell session
- Group Policy computers and users

Not a full security system:

- Does **NOT** restrict user actions nor typing individual PS commands
- Helps users set basic rules for and prevents unintentional violations of the rules

Execution Policy Levels

Restricted - Default in all Client OS versions

- Scripts cannot be run
- PowerShell interactive-mode only

AllSigned

- Runs a script only if digitally signed with trusted certificate on local machine

RemoteSigned - Default in all Server OS versions (*Recommended Minimum)*

- Runs all local scripts
- Downloaded scripts must be signed by trusted source

Bypass

- Nothing locked, no warnings or prompts
- Used when script is built into larger application that has its own security model

Unrestricted

- All scripts can be run

Execution Policy Scope

AD Group Policy – Computer

- Affects all users on targeted computer

AD Group Policy – User

- Affects users targeted only

Process

- Command-line Parameter (*c:\> powershell.exe –executionpolicy remotesigned)*
- Affects current PowerShell Host session only

Registry – User

- Affects current user only
- Stored in HKCU registry subkey (Admin access **not** needed)

Registry – Computer

- Affects all users on computer
- Stored in HKLM registry subkey (Admin access **needed** to change)

Highest Priority Wins

Setting / Determining Execution Policy

```
Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy Unrestricted
```

Apply setting to **current** user only

```
PS C:\> Get-ExecutionPolicy -List

Scope            ExecutionPolicy
-----            ---------------
MachinePolicy    Undefined
UserPolicy       Undefined
Process          Undefined
CurrentUser      Unrestricted
LocalMachine     RemoteSigned

PS C:\> Get-ExecutionPolicy
Unrestricted
```

Topmost takes precedence

Effective Policy

Demonstration

Execution Policy

Script Signing

Validates the integrity of the script

Enforced with **Execution Policies**

Certificate used should be of type **Code signing**

Signing a Script

Step 1: Create a certificate variable (2 ways)

Retrieve a code-signing certificate from the certificate provider

```
PS C:\> $cert = Get-ChildItem -Path Cert:\CurrentUser\My -CodeSigningCert
```

-- OR --

Find a code signing certificate

```
PS C:\> $cert = Get-PfxCertificate -Path C:\Test\MySign.pfx
```

Trusted by computer where script will run

Step 2: Sign the script

```
PS C:\> Set-AuthenticodeSignature .\ISECPUTime.ps1 $cert

   Directory: C:\Scripts

SignerCertificate                Status     Path
-----------------                ------     ----
A4..                             Valid      ISECPUTime.ps1
```

Script Before Signing

Sample.ps1 - Notepad

File Edit Format View Help

```
Write-Host ‘Start of script’ -BackgroundColor Green
Write-Host ‘Display the % CPU Time utilization by the ISE’ -BackgroundColor Green
Get-Counter ‘\Process(powershell_ise)\% Processor Time’
```

Ln 1, Col 1

Script After Signing

Sample.ps1 - Notepad

File Edit Format View Help

```
Write-Host ‘Start of script’ -BackgroundColor Green
Write-Host ‘Display the % CPU Time utilization by the ISE’ -BackgroundColor Green
Get-Counter ‘\Process(powershell_ise)\% Processor Time’
# SIG # Begin signature block
# mIIEMwYJKoZIhvcNAQcCoIIEJDCCBCACAQExCzAJBgUrDgMCGgUAMGkGCisGAQQB
# kjcCAQSgWsBZMDQGCisGAQQBgjcCAR4wJgIDAQAABBAfzDtgWUsITrck0sYpfvNR
# agEAAsEAAgEAAgEAAgEAMCEwCQYFKw4DAhoFAAQU6vQAn5sf2qIxQqwWUDwTZnJj
# j5ufgfI9MIICOTCCAaagAwIBAgIQyLeyGZcGA4ZOGqK7VF45GDAJBgUrDgMCHQUA
# agEAAsEAAgEAAgEAAgEAMCEwCQYFKw4DAhoFAAQU6vQAn5sf2qIxQqwWUDwTZnJj
# kjcCAQSgWsBZMDQGCisGAQQBgjcCAR4wJgIDAQAABBAfzDtgWUsITrck0sYpfvNR
# ...........
# SIG # End signature block
```

Ln 1, Col 1

Script signature block

Single-line and Block Comments

Comments

Single line comment - #

```
param ($Computername)
#Testing connectivity to remote computers
$result = Test-Connection -ComputerName $Computername -Quiet -Count 1
Write-Host $result -ForegroundColor Green  # Inline comment
```

Block comment: comment multiple lines **- <# #>**

```
param ($Computername)
<#
    Testing connectivity to remote computers
    Write Boolean output in Green
#>
$result = Test-Connection -ComputerName $Computername -Quiet -Count 1
Write-Host $result -ForegroundColor Green
```

Demonstration

Block Comments

The Requires Statement

Requires Statement

Special comment

Prevents script from running without required elements

Can only be used in scripts (not functions, cmdlets, etc)

Requires Option	Supported Version
#Requires -Version <N>[.<n>]	2.0+
#Requires -PSSnapin <PSSnapin-Name> [-Version <N>[.<n>]]	2.0+
#Requires -ShellId <ShellId>	2.0+
#Requires -Modules { <Module-Name> | <Hashtable> }	3.0+
#Requires -RunAsAdministrator	4.0+

Version Requirement

- Prevents script from running on lower PowerShell versions
- Script errors at start
- Special comment tag: #Requires -Version <N>[.<n>]

```
#requires -Version 3
Get-ChildItem c:\ -Hidden
```

Error when script runs within PowerShell v2

```
.\Test1.ps1 : The script 'Test1.ps1' cannot be run because it contained a
"#requires" statement at line 1 for Windows PowerShell version 3.0. The
version required by the script does not match the currently running version of
Windows PowerShell version 2.0.At line:1 char:12+ .\Test1.ps1 <<<<     +
CategoryInfo          : ResourceUnavailable: (Test1.ps1:String) [],
ScriptRequiresException    + FullyQualifiedErrorId :
ScriptRequiresUnmatchedPSVersion
```

Run as Administrator Requirement

- Prevents script from running without elevated permissions
- Script errors at start
- Special comment tag: #Requires –RunAsAdministrator

```
#requires -RunAsAdministrator
Get-ChildItem c:\ -Hidden
```

Error when script runs without elevation

```
.\RunAsAdminTest.ps1 : The script 'RunAsAdminTest.ps1' cannot be run because
it contains a "#requires" statement for
running as Administrator. The current Windows PowerShell session is not
running as Administrator. Start Windows
PowerShell by  using the Run as Administrator option, and then try running the
script again.
```

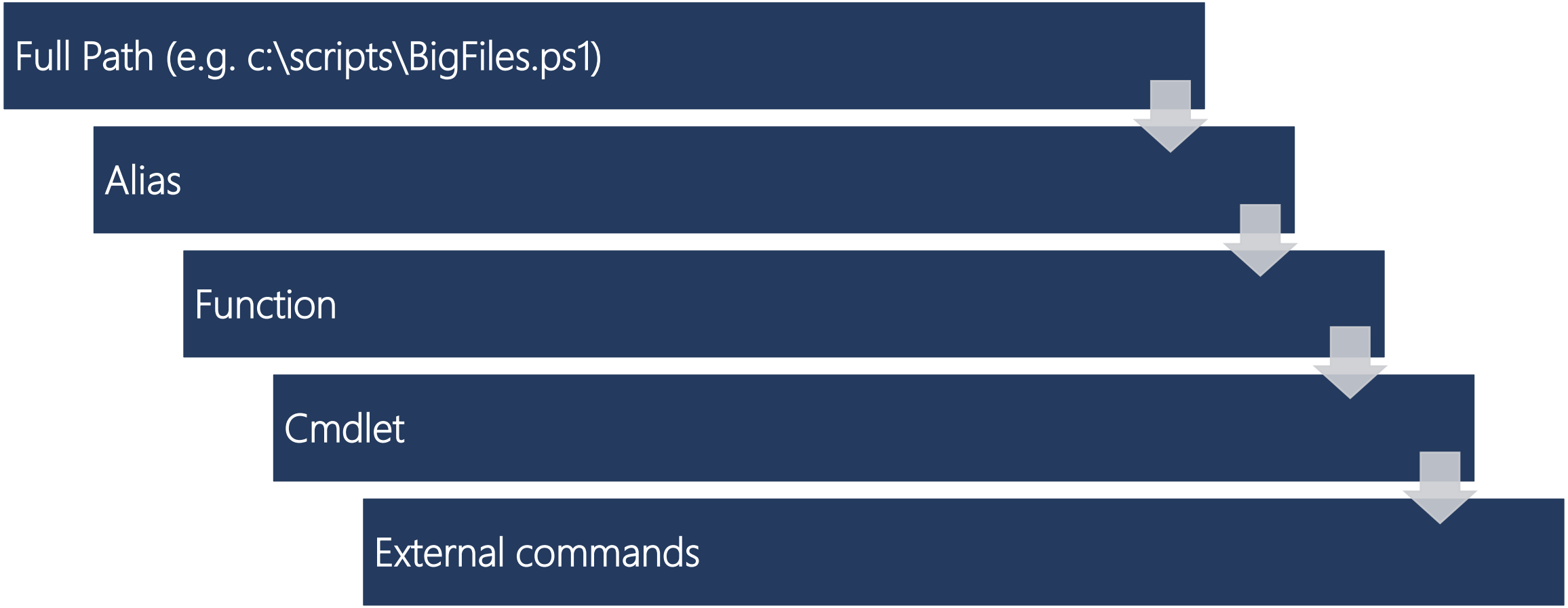
Demonstration

Requires Statement

Command Precedence Rules

Command Lookup Precedence

- Determines which command to run if more than one command have the **same name**
- If the **same type** of command with the same name exists, PowerShell runs the command that was added to the session **most recently**



“Replace” Another Command

```
PS C:\scripts> ping MS

Pinging ms.contoso.local with 32 bytes of data:
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

PS C:\scripts> New-Alias -Name ping -Value Test-Connection
PS C:\scripts> ping MS

Source        Destination     IPV4Address      IPV6Address
------        -----------     -----------      -----------
WIN10         MS           192.168.1.2
WIN10         MS           192.168.1.2
```

“Ping” command now uses cmdlet instead of external command

Module Qualify Command Name

```
#Run normal cmdlet
PS C:\> Get-Process System
Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
   6877       0      304      42336   2,140.66      4   0 System

#Create function with same name
PS C:\> Function Get-Process {“This isn’t the Get-Process cmdlet”}

#Command precedence runs function instead of Cmdlet
PS C:\> Get-Process
This isn’t the Get-Process cmdlet

#Module qualify command name
PS C:\> Microsoft.PowerShell.Management\Get-Process -Name System
Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
   6877       0      304      42312   2,158.09      4   0 System
```

Questions?

Lab 7: Scripts

30 Minutes