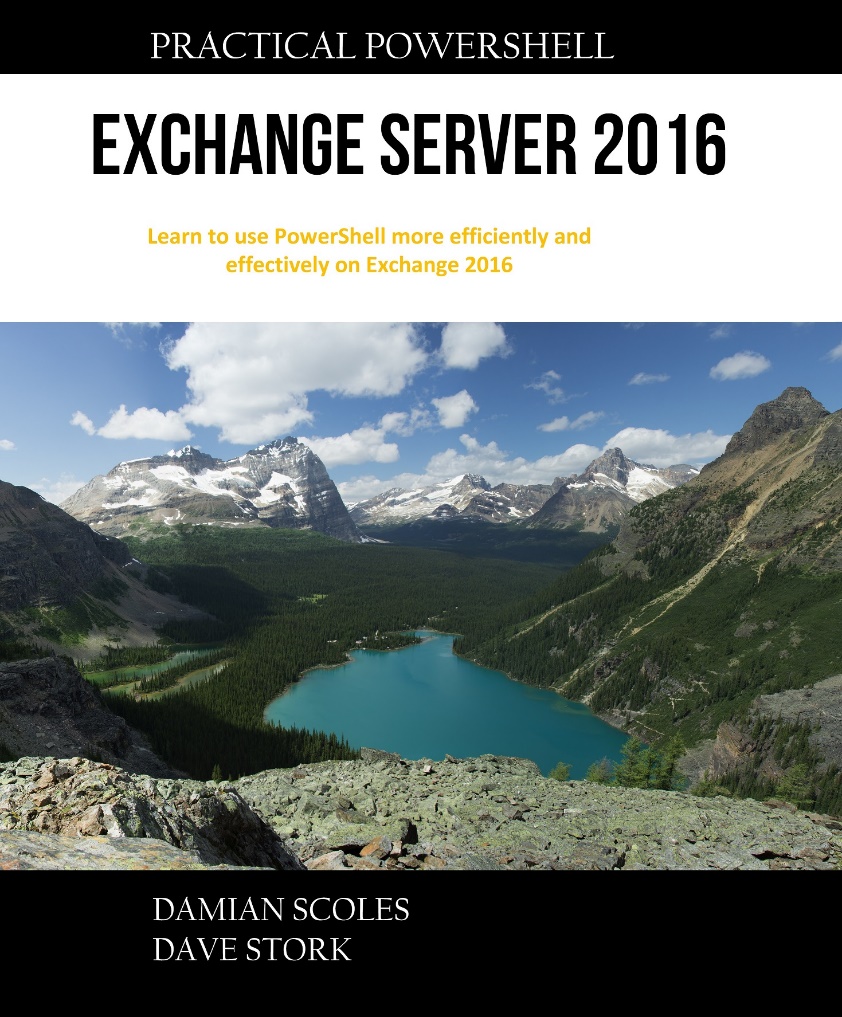
Code Snippets from:

**Exchange Server 2016 Book**



**Introduction**

Get-Mailbox | Get-MailboxStatistics

$Mailboxes = Get-Mailbox

Foreach ($Mailbox in $Mailboxes) {

Get-MailboxStatistics $Mailbox.Alias

}

Get-Mailbox | Get-MailboxDatabase

Get-Mailbox | Remove-Mailbox -WhatIf

Get-Command \*mailbox\*

Get-Command \*database\*

Import-Module ActiveDirectory

Get-Help Get-ExchangeServer

Get-Help Get-ExchangeServer -Full

Get-Database | Dismount-Database

Get-MailboxDatabase | Dismount-Database

Get-MailboxDatabase | Mount-Database

Get-Command \*mobile\*

**Chapter 1**

$Values = 1,2,3,4,5

$Names = “Dave”,“Matt”,“John”,“Michael”

$Hash = @ { }

$Servers = @{Dallas = ‘Exchange01’; Orlando = ‘Exchange02’ ; Chicago = ‘Exchange03’}

$CSVFileData = Import-CSV “C:\temp\MailboxData.csv”

$Mailbox = Get-Mailbox

If ($Mailbox -eq “Damian”) {

Set-Mailbox $Mailbox -ForwardingSMTPAddress DaveStork@PracticalPowershell.Com

}

If ($Quota -lt 2000000) {

Set-Mailbox $Mailbox -IssueWarningQuota 5gb

}

If (“Mouse” -lt “Wolf) {

Write-Host “The Wolf eats the Mouse!”

}

$CSVFileData = Import-CSV “C:\Data.csv”

Foreach ($Line in $CSVFileData) {

$DisplayName = $Line.DisplayName

$Size = $Line.MailboxSizeMB

Write-host “The user $displayname has a mailbox that is $Size MB in size.”

}

$Counter = 1

Do {

Write-Host “This is pass # $counter for this loop.”

$Counter++

} While ($Counter -ne 1000)

In the above sample, we use a counter

$Counter = 0

Do {

$Counter++

Write-Host “This is pass # $counter for this loop.”

} While ($Counter -ne 1000)

# Check for Old Disclaimers

Function Check-OldDisclaimers {

$RuleCheck = (Get-TransportRule).ApplyHtmlDisclaimerText

$RuleCheck2 = Get-TransportRule | Where {$\_.ApplyHtmlDisclaimerText -ne $Null}

If ($RuleCheck -eq $Null) {

Write-Host “There are no disclaimers in place now.” -ForegroundColor Green

} Else {

Foreach ($Line in $RuleCheck2) {

Write-Host “There is a transport rule in place called $line that is a disclaimer rule.” -ForegroundColor

Yellow

}

}

} #End of the Check-OldDisclaimers function

Check-OldDisclaimers

**Chapter 2**

Function PagefileSizeCheckInitial {

}

Set-Mailbox –Identity UserA –PrimarySMTPAddress usera@contoso.com # Set Primary SMTP address

# Get-Mailbox DamianScoles | ft DisplayName, Server, Database

Get-Mailbox DamianScoles | ft DisplayName, ServerName, Database

$Mailbox = Get-Mailbox $Name

# Write-host “The current mailbox is $Mailbox.”

$Name = $SourceName.Split([Char]0x0020)

$Name = John, M., Smith

$Name = Michael, G, Larraday

$Alias = $Name[0]+$Name[2]

$Alias = JohnSmith

$Alias = MichaelLarraday

Write-Host "The server $name does not have the hotfix " -ForegroundColor White -NoNewLine;

Write-Host "$Hotfix" -ForegroundColor Red -NoNe

Write-Host "The server $name does not have the hotfix " -ForegroundColor White

Write-Host "$hotfix" -ForegroundColor Red

Write-Host " installed at this time." -ForegroundColor White

Foreach ($Name in $Names) {

Write-Host "The server $name does not have the hotfix " -ForegroundColor White -NoNewLine

Write-Host "$Hotfix" -ForegroundColor Red -NoNewLine

Write-Host " installed at this time." -ForegroundColor White -NoNewLine

}

$Current = "15.00.1178.004"

$CurrentMinus2 = "15.00.1130.007"

$Servers = Get-ExchangeServer

Foreach ($Server in $Servers) {

$Test = $Null

$ExchangeServer = $Server.Name

$Script = { $Ver = Get-Command Exsetup |%{$\_.FileVersionInfo} }

Invoke-Command -ComputerName $ExchangeServer -ScriptBlock $Script

If ($Ver -eq $Current) {

Write-Host "The server $ExchangeServer is up to date [CU12]." -ForegroundColor Green

}

If (($Ver -lt $Current) -or ($Ver -gt $CurrentMinus2)) {

Write-Host "The server $ExchangeServer is within two CUs of being up to date [CU10 or CU11]."

-ForegroundColor Yellow

}

If ($Ver -lt $CurrentMinus2) {

Write-Host "The server $ExchangeServer is out of date. This means it is running 3 or more CUs

behind [CU9 or less]." -ForegroundColor Red

}

}

$Cert=(dir cert:currentuser\my\ -CodeSigningCert)

Set-AuthenticodeSignature .\MyScript.ps1 $Cert –TimeStampServer “<http://timestamp.globalsign.com/>

scripts/timstamp.dll”

**Chapter 3**

Get-ItemProperty -Path ‘HKLM:\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full’

(Get-ItemProperty -Path “HKLM:\SOFTWARE\Microsoft\NET Framework Setup\NDP\v4\Full”).Release

$DownloadFolder = “c:\install”

$File = “<https://download.microsoft.com/download/E/4/1/E4173890-A24A-4936-9FC9-AF930FE3FA40/>

NDP461-KB3102436-x86-x64-AllOS-ENU.exe”

$DownloadFile = $DownloadFile.Split([char]0x02F)

$DownloadedFile = $DownloadFile[-1]

Start-BitsTransfer -Source “$File” -Destination “$DownloadFolder\$DownloadedFile”

Start-Process ‘.\NDP461-KB3102436-x86-x64-AllOS-ENU.exe’ -ArgumentList ‘/quiet’,’/norestart’ –Wait

$DownloadFolder = “c:\install”

$File = “<http://download.microsoft.com/download/2/C/4/2C47A5C1-A1F3-4843-B9FE-84C0032C61EC/>

UcmaRuntimeSetup.exe”

$DownloadFile = $DownloadFile.Split([char]0x02F)

$DownloadedFile = $DownloadFile[-1]

Start-BitsTransfer -Source “$File” -Destination “$DownloadFolder\$DownloadedFile”

Install-WindowsFeature RSAT-ADDS

Install-WindowsFeature AS-HTTP-Activation, Desktop-Experience, NET-Framework-45-Features, RPCover-

HTTP-proxy, RSAT-Clustering, RSAT-Clustering-CmdInterface, RSAT-Clustering-Mgmt, RSAT55

Clustering-PowerShell, Web-Mgmt-Console, WAS-Process-Model, Web-Asp-Net45, Web-Basic-Auth,

Web-Client-Auth, Web-Digest-Auth, Web-Dir-Browsing, Web-Dyn-Compression, Web-Http-Errors, Web-

Http-Logging, Web-Http-Redirect, Web-Http-Tracing, Web-ISAPI-Ext, Web-ISAPI-Filter, Web-Lgcy-Mgmt-

Console, Web-Metabase, Web-Mgmt-Console, Web-Mgmt-Service, Web-Net-Ext45, Web-Request-

Monitor, Web-Server, Web-Stat-Compression, Web-Static-Content, Web-Windows-Auth, Web-WMI,

Windows-Identity-Foundation

If ((Get-WMIObject Win32\_OperatingSystem).Version -NotMatch ‘6.2’)

 If ((Get-WMIObject Win32\_OperatingSystem).Version -NotMatch ‘6.3’)

 If ((Get-WMIObject Win32\_OperatingSystem).Version -NotMatch ‘10’)

If ((Get-WMIObject Win32\_OperatingSystem).Version -NotMatch ‘6.2’) –and ((Get-WMIObject Win32\_

OperatingSystem).Version -NotMatch ‘6.3’)) {

Write-Host “This server is not Windows Server 2012 or 2012 R2. Exiting.” -ForegroundColor Red

Exit

}

$OSVersion = (Get-WMIObject Win32\_OperatingSystem).Version

If (($OSVersion -NotMatch ‘6.2’) -and ($OSVersion -NotMatch ‘6.3’) -and ($OSVersion -NotMatch ‘10’)) {

Write-Host “This server is not Windows Server 2012, 2012 R2 or Windows 2016. Exiting.”

-ForegroundColor Red

Exit

}

Get-Hotfix | Where {$\_.HotfixID -eq “kb2919355”}

$Hotfix = Get-Hotfix | Where {$\_.HotfixID -eq “kb2919355”}

If ($Hotfix -eq $Null) {

Write-Host “Hotfix kb2919355 is missing.” -Foregroundcolor Red

} Else {

Write-Host “Hotfix kb2919355 is installed.” -Foregroundcolor Green

}

$Hotfix = Get-Hotfix | Where {$\_.HotfixID -eq “kb2919355”}

If ($Hotfix -eq $Null) {

Write-Host “Hotfix kb2919355 is missing.” -Foregroundcolor Red

} else {

DotNet461Install

}

Function Install-KB2919355 {

# KB2919355 Section

$DownloadFolder = “c:\install”

$SourceFile = <https://download.microsoft.com/download/2/5/6/256CCCFB-5341-4A8D-A277>-

8A81B21A1E35/Windows8.1-KB2919355-x64.msu”

$DownloadFile = $DownloadFile.Split([char]0x02F)

$DownloadedFile = $DownloadFile[-1]

Start-BitsTransfer -Source “$File” -Destination “$DownloadFolder\$DownloadedFile”

Start-Process “.\$DownloadedFile” -ArgumentList ‘/quiet’,’/norestart’ –Wait

} # End Install-KB2919355

$Hotfix = Get-Hotfix | Where {$\_.HotfixID -eq “kb2919355”}

If ($Hotfix -eq $Null) {

Install-KB2919355

} Else {

DotNet461Install

}

Function DotNet461Install {

$DownloadFolder = “c:\install”

$File = “<https://download.microsoft.com/download/E/4/1/E4173890-A24A-4936-9FC9-AF930FE3FA40/>

NDP461-KB3102436-x86-x64-AllOS-ENU.exe”

$DownloadFile = $DownloadFile.Split([char]0x02F)

$DownloadedFile = $DownloadFile[-1]

Start-BitsTransfer -Source “$File” -Destination “$DownloadFolder\$DownloadedFile”

Start-Process “.\$DownloadedFile” -ArgumentList ‘/quiet’,’/norestart’ –Wait

} # End of DotNet461Install Function

Function DotNet461Install {

# Set the download folder for the .Net executable

$DownloadFolder = “c:\install”

# Set the file location to be downloaded

$File = “<https://download.microsoft.com/download/E/4/1/E4173890-A24A-4936-9FC9-AF930FE3FA40/>

NDP461-KB3102436-x86-x64-AllOS-ENU.exe”

# Split the file name up to remove the ‘/’ character

$DownloadFile = $DownloadFile.Split([char]0x02F)

$DownloadedFile = $DownloadFile[-1]

# Download the file

Start-BitsTransfer -Source “$File” -Destination “$DownloadFolder\$DownloadedFile”

# Install .Net 4.6.1 with the local file

Start-Process “.\$DownloadedFile” -ArgumentList ‘/quiet’,’/norestart’ –Wait

} # End of DotNet461Install function

<#

.SYNOPSIS

Install Exchange Server 2016 Prerequisites

.DESCRIPTION

Install Exchange Server 2016 Prerequisites using PowerShell

.NOTES

Version : 1.0

Change Log : 1.0 - First iteration

Rights Required : Local admin on server

Exchange Version : 2016

Author : Just A UC Guy [Damian Scoles]

Blog : <http://justaucguy.wordpress.com>

Disclaimer : No support provided by Microsoft

.EXAMPLE

.\Configure-ExchangePrereq-1.1.ps1

#>

$menu = {

write-host “\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*”

write-host “Exchange Server 2016 PreRequisite Installation”

write-host “\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*”

write-host “ ”

write-host “Please select an option from the list below:”

write-host “ ”

write-host “ 1) Prerequisites Part 1 - .Net and Windows Features”

write-host “ 2) Prerequisites Part 2 - UCMA 4.0”

write-host “ ”

write-host “ 98) Restart the Server”

write-host “ 99) Exit”

write-host “ ”

write-host “Select an option.. [1-99]?”

}

$Name = “16-lg-ex01.16-lg.local”

$Commands = (Get-Command | Where {$\_.ModuleName -eq $Name}).Name

Foreach ($Line in $Commands) {

Get-Help $Line -Full > c:\downloads\command.txt

$Search = Select-String -Path c:\downloads\command.txt -pattern “cmdlet will be removed in a future

version of”

$Search2 = Select-String -Path c:\downloads\command.txt -pattern “cmdlet has been deprecated”

If ($Search -ne $Null) {

Write-Host “$line is going to be deprecated!” -ForegroundColor Yellow

}

If ($Search2 -ne $Null) {

Write-Host “$Line is going to be deprecated!” -ForegroundColor Yellow

}

Remove-Item c:\downloads\command.txt

}

Get-Command –Module A | Export-CliXml CmdletsA.xml

Get-Command –Module B | Export-CliXml CmdletsB.xml

 $CmdA= Import-CliXml .\CmdletsA.xml

$CmdB= Import-CliXml .\CmdletsB.xml

Compare-Object -ReferenceObject $CmdA -DifferenceObject $CmdB -Property Name

**Chapter 4**

$LiveCred = Get-Credential

$Session = New-PSSession -Name ExchangeOnline -ConfigurationName Microsoft.Exchange

-ConnectionUri <https://ps.outlook.com/powershell/> -Credential $LiveCred -Authentication Basic –

AllowRedirection

Import-PSSession $Session

$Cred = Get-Credential

$Session = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri http://<exchange

server FQDN>/PowerShell/ -Credential $Cred -Authentication Kerberos

Import-PSSession $Session

Get-PowerShellVirtualDirectory –Server 16-Tap-Ex01

Set-PowerShellVirtualDirectory “16-tap-ex01\PowerShell (Default Web Site)” -RequireSSL $True

Enter-PSSession -ComputerName LocalHost

Enable-PSRemoting

Get-User <Alias> | ft DisplayName,\*power\*

New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri http://<exchange server

FQDN>/PowerShell/ -Credential (Get-Credential) -Authentication Kerberos

Get-PowerShellVirtualDirectory -Server <Exchange Server> | fl \*auth\*

Get-PowerShellVirtualDirectory -Server <exchange server> | Set-PowerShellVirtualDirectory

-BasicAuthentication $True

# Get list of all Exchange Servers

$Servers = Get-ADComputer -Filter \*

Foreach ($Server in $Servers) {

$MemberOf = (Get-ADComputer $Server | Get-ADObject -Properties MemberOf).MemberOf

If ($MemberOf -Match “Exchange Install Domain Servers”) {

$Name = [string]$Server.Name

$ExchangeServers += ,@($Name)

}

}

$Server = $Args[0]

Write-Host “ “

Write-Host “Analyzing event logs for server $server” -ForegroundColor White

Write-Host “ “

Write-Host “PROCESSING” -ForegroundColor Yellow

Write-Host “ “

# Get a list of Critical Events

# Application Log

$Info = $Null

$App = $Null

$Log = “Application”

$LogCheck = Get-WinEvent -ListLog $Log -ComputerName $Server -ErrorAction SilentlyContinue

If ($LogCheck -ne $Null) {

Write-Host “Application Log Analysis” -ForegroundColor Cyan

$Events = Get-EventLog -ComputerName $Server -LogName $Log | Where {($\_.EntryType -eq “Error”)

-or ($\_.EntryType -eq “Warning”) -or ($\_.EntryType -eq “Critical”)} | Sort-Object EventID | Group-

Object EventID

If ($Events -ne $Null) {

$App = Foreach ($Line2 in $Events) {

$Event = $Line2.Name

$Info2 = Get-WinEvent -ComputerName $Server -FilterHashTable @{LogName=$Log;ID=$Event}

-MaxEvents 1 -Erroraction SilentlyContinue

New-Object PSObject -Property @{

LastOccured = ($Info2.TimeCreated).DateTime

Count = $Line2.Count

Name = $Info2.ProviderName

Event = $Event

} Else {

Write-Host “No events were found.” –ForegroundColor Yellow

}

**Chapter 5**