**File Handling:**

|  |  |  |
| --- | --- | --- |
| **Example** | **Cmdlet** | **Description** |
| 1 | Get-Content | Read in a typical Transact SQL script. |
| 2 | Set-Content | Write out a typical Transact SQL script. |
| 3 | Add-Content | Add a comment to end of the script. |
| 4 | Clear-Content | Wipe out the contents of the file. |

#

#  1 - Read text file

#

# Path to file

$path = "c:\stocks\"

$srcfile = $path + "create-database-1.sql"

# Read input

$content = Get-Content $srcfile

# Show the contents

Clear-Host

$content

Write-Host " "

# Confirm it is a string array

$content.GetType()

#

#  2 - Write text file

#

# Path to file

$path = "c:\stocks\"

$dstfile = $path + "create-database-2.sql"

# Write output

$content | Set-Content -Path $dstfile

#

#  3 - Append to text file

#

# Carriage return & line feed

$crlf = [char]13 + [char]10

# New content 2 add

$addition = $crlf + "-- Comment at bottom of file"

# Append content 2 bottom

Add-Content -Path $dstfile -Value $addition

#

# 4 - Clear text file

#

# Zero byte resulting file

Clear-Content -Path $dstfile

#

# 5 - Use delimiter parameter

#

# Path to file

$path = "c:\stocks\"

$srcfile = $path + "create-database-1.sql"

# Grab the script

$content = Get-Content $srcfile

# Cast to string

[string]$result = $content

# Lost my crlf?

$result

# Grab the script

$content = Get-Content -Delimiter "`n" $srcfile

# Cast to string

[string]$result = $content

# Saved my crlf?

$result

ForEach ($system in Get-Content "systems.txt")

{

Write-Host $system

}

[io.file]::ReadAllText(".\desktop\git-python\test.ps1").split("`n").count

foreach($line in Get-Content .\file.txt) {

if($line -match $regex){

# Work here

}

}

Get-Content .\file.txt | ForEach-Object {

if($\_ -match $regex){

# Work here

}

}

Read CSV

PS C:\> Get-Process | Export-Csv Processes.csv

PS C:\> $P = Import-Csv Processes.csv

PS C:\> $P | Get-Member

Specify the delimiter

PS C:\> Get-Process | Export-Csv Processes.csv -Delimiter :

PS C:\> $P = Import-Csv Processes.csv -Delimiter :

Data:

Example CSV:

Name | Phone Number | Email

Elvis | 867.5309 | Elvis@Geocities.com

Sammy | 555.1234 | SamSosa@Hotmail.com

$Name = @()

$Phone = @()

Import-Csv H:\Programs\scripts\SomeText.csv |`

ForEach-Object {

$Name += $\_.Name

$Phone += $\_."Phone Number"

}

$inputNumber = Read-Host -Prompt "Phone Number"

if ($Phone -contains $inputNumber)

{

Write-Host "Customer Exists!"

$Where = [array]::IndexOf($Phone, $inputNumber)

Write-Host "Customer Name: " $Name[$Where]

}

**Using Get-childitem to get a list of files modified in the last 3 days**

(Get-ChildItem -Path c:\pstbak\\*.\* -Filter \*.pst | ? {

$\_.LastWriteTime -gt (Get-Date).AddDays(-3)

}).Count

Get-ChildItem -Path . -Recurse| ? {$\_.LastWriteTime -gt (Get-Date).AddDays(-4)}

Get-ChildItem C:\Folder -File -Recurse |

Where-Object { $\_.LastWriteTime -ge "05/01/2013" -and $\_.LastWriteTime -le "05/31/2013" } |

select-Object FullName, LastWriteTime

Sorting

PS D:\Vivid\2008R2\_Solution\Vivid-SCP\vividreports> get-childitem -filter "\*.xls" | sort LastWriteTime –Descending

# Running installers remotely

If you already have the file on the remote system, we can run it with Invoke-Command.

Invoke-Command -ComputerName server01 -ScriptBlock {

c:\software\installer.exe /silent

}

There are two important details to be aware of right away.

The first detail is that you need to maintain a remote session while the installer is running. If the installer does not block execution (it returns control back to the shell while it executes), your script may finish before the installer finishes. This will cancel the install as it closes the remote session.

You will need to call Start-Process -Wait if you are having that issue.

Invoke-Command -ComputerName server01 -ScriptBlock {

Start-Process c:\windows\temp\installer.exe -ArgumentList '/silent' -Wait

}

This brings us to our second important detail. The install needs to be truly silent. Remote sessions are non-interactive. That means that they cannot popup or show windows. This will either cause the program to fail because it cannot show the window or it will cause the installer to hang because it expects someone to click a button that you have no way to click.

# Installing from a remote location

Most of the time if you are running installers on a remote system, you have the installer on a network share (UNC path). At first glance, this looks like it should work:

# Incorrect approach

Invoke-Command -ComputerName server01 -ScriptBlock {

\\fileserver\share\installer.exe /silent

}

This can be the source of a lot of headaches. Ideally you want to run the installer from a UNC path, but you discover that it does not work.

Trying to copy the file inside the remote command give you the same problem.

# Incorrect approach

Invoke-Command -ComputerName server01 -ScriptBlock {

Copy-Item \\fileserver\share\installer.exe c:\windows\temp\

}

# Access denied or file does not exist

Everything tells you that the file either does not exist or you have no permissions to the file. This is kind of a false message because it does exist and you have file access rights. The issue is that your remote session does not have those same rights.

## The double hop problem

This is the double hop problem. The credential used to authenticate with server01cannot be used by server01 to authenticate to fileserver. Or any other network resources for that matter. That second hop is anything that requires authentication that is not on the first remote system.

We can either pre-copy the file or re-authenticate on the remote end.

I will use these place holder variables in the rest of the examples.

$file = '\\fileserver\share\installer.exe'

$computerName = 'server01'

## Pre-copy file using administrator share

The obvious first approach is to use the administrator share of the remote system to push content to a location we can access. Here I place it in the windows temp folder then remotely execute it.

Copy-Item -Path $file -Destination "\\$computername\c$\windows\temp\installer.exe"

Invoke-Command -ComputerName $computerName -ScriptBlock {

c:\windows\temp\installer.exe /silent

}

## Pre-copy using PSSession (PS 5.0)

There is a new feature added in Powershell 5.0 that allows you to copy files using a PSSession. So create a PSSession and copy the file over it using the syntax below. A cool thing about this approach is that with Powershell 5.0, you can create a PSSession to a guest VM over the VM buss (instead of over the network) and you can still copy a file to it.

$session = New-PSSession -ComputerName $computerName

Copy-Item -Path $file -ToSession $session -Destination 'c:\windows\temp\installer.exe'

Invoke-Command -Session $session -ScriptBlock {

c:\windows\temp\installer.exe /silent

}

Remove-PSSession $session

While you can run Invoke-Command on multiple computers at once, be aware that Copy-Item -ToSession only works on a single session.

Install .msi software in silent mode - Powershell

Get-Url http://dl.google.com/chrome/install/375.126/chrome\_installer.exe c:\temp\chrome\_installer.exe

Start-Process -FilePath 'c:\temp\chrome\_installer.exe' -ArgumentList '/silent', '/install' -Wait

Start-Process -FilePath 'C:\temp\DifferentProgram.exe' -ArgumentList '/argument' –Wait

$path=**split-path** $MyInvocation.MyCommand.path

$spath= "$path\setup.x64.msi"

**If**($global:availability -eq $null)

{

    "1. Local Administrator software is not installed in this computer"

**If**(**Test-Path** $spath)

        {

            "2. MSI file is accessible from the directory "

            $status=Start-**Process** -FilePath msiexec.exe -ArgumentList '/i',$spath,'/q' -Wait -PassThru -Verb "RunAs"

**If**($?)

        {

            checksoftware

                    "3.  $($Global:availability.DisplayName)--$($Global:availability.DisplayVersion) has been installed"

        }

**else**{"3. Unable to install the software"}

        }

**Else**

        {

               "2. Unable to access the MSI file form directory"

        }

}

**Else**

{

    "1. Local Administrator software is already existing"

}