**Services**

**VM / Remote Machine : Manage**

**Install .msi (executable (patch )**

**Windows Update**

**File/Folder – Backup/Delete/Move**

**Email Notification**

**Content Management**

**Missing Application**

**Check Running Service on Remote Machine**

Pulled the servers from the text file by using **Get-Content**, and I placed them into a variable that I called $Servers. Here is the command I used.

**$Servers = Get-Content "C:\Temp\MyServers.txt"**

Now, using the **Foreach** command, I started checking for the service on the remote servers.

The **Foreach** command will loop through each server in the $Servers variable. Each loop will use the variable $Server which will contain one item from the variable $Servers list of servers.

By using the variable $Server (the computer name in this case) and **Get-WMIobject** to connect to remote servers, the script will query for the service OASVC\_Local, and populate the variable $Service with a "Service" object that will contain the properties for each object. Here is the command I arrived at using.

$a= @(1111,2222,3333,433333,44444)

Foreach($n in $a)

{

$n

}

**Foreach ($Server in $Servers)**

**{**

**$Service = Get-WMIobject win32\_service -computername $Server -Filter**

**Foreach(**

If ($Service.state -eq "running")

**{**

**}**

**Else**

**{**

**}**

}

I also want to determine if the service is running or not. I query if the service is running by using the **If** statement. If the *state* equals “running,” then write "RSA is Running for $Server" to a file called RSA.txt, and also write it to the console screen. I wanted to have the results appear on the console screen for quick viewing, in addition to a text file that I could use as a simple search for the key word “NOT” to identify servers that needed attention. (Note: NOT running -BackgroundColor red.) Here is that portion of the script.

If ($Service.state -eq "running")

 {

Write-Output "RSA is Running for $Server" | out-file -append C:\Temp\RSA.txt

Write-Host "RSA is Running for $Server”

} else

 {

Write-Output "RSA is NOT Running for $Server" | out-file -append C:\Temp\RSA.txt

Write-Host "RSA is NOT Running for $Server" -BackgroundColor red

}

}

|  |
| --- |
| Here is the script in its entirety.  $Servers = Get-Content "C:\Temp\MyServers.txt"  Foreach ($Server in $Servers)  {$Service = Get-WMIobject win32\_service -computername $Server -Filter "Name = 'OASVC\_Local'"  If ($Service.state -eq "running")   {  Write-Output "RSA is Running for $Server" | out-file -append C:\Temp\RSA.txt  Write-Host "RSA is Running for $Server"  } else   {  Write-Output "RSA is NOT Running for $Server" | out-file -append C:\Temp\RSA.txt  Write-Host "RSA is NOT Running for $Server" -BackgroundColor red  }  } |

**Powershell script to remote start/reboot/logoff servers**

***Parameter for function Win32Shutdown(1)***

0 = Log off  
1 = Shutdown  
2 = Reboot  
8 = Power off

4 = Forced log off  
5 = Forced shutdown  
6 = Forced reboot  
12 = Forced power off

*$RebootList = Get-Content RebootServers.txt*  
*foreach( $Rsrv in $RebootList )*  
*{*  
*Write-host “Issuing remote reboot command to $Rsrv”*  
*# Command to force reboot the remote server*  
*(*

*gwmi Win32\_OperatingSystem -ComputerName $Rsrv).Win32Shutdown****(6)***  
*}*

**Powershell Script to Copy Files from One Server to Another in the same Domain.**

**Purpose:**  
Simple script to copy files from one computer or server to another in the same domain. Example can  
also email a simple report of files copied. Created to move some files hourly and only email when  
something was moved.

**Notes:**  
Will not work across different domains. Account used to run task from task manager must be a domain  
user with enough rights on both systems to read and write.

|  |  |
| --- | --- |
| 22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  87 | # Email Settings  $fromAddr = "From@domain.com" # Enter the FROM address for the e-mail alert  $toAddr = "To@domain.com" # Enter the TO address for the e-mail alert  $smtpsrv = "mail.domain.com" # Enter the FQDN or IP of a SMTP relay    # Files to Copy and where they are going  # Local Files should be in the formate of C:\Folder\\* for Files only or C:\Folder to copy the folder as well  $Localfiles = "C:\Folder\\*"  # Remote Path should be UNC path with admin share example \\hostname\c$\pathtocopyto  $Remotefiles = "\\hostname\c$\pathtocopyto"    #Enable or Disable Script functions  $EmailAlerts = $TRUE # Turn e-mail alerts on or off. $FALSE or 0 = off  $TestfromPrompt = $FALSE # Turn on output from command line.  $FALSE or 0 = no output  $RemoveFilesafterCopy = $TRUE # Remove files after files have been copied over?    #===========================#  #Main Script                #  #===========================#    #Get the names of the files we want to transfer  $files = Get-childitem "$Localfiles" |foreach { $\_.Name}    if ($files -ne $null)  {       #Generates output to command line if Value = True     if ($TestfromPrompt -eq $TRUE)     {      Write-Host "These are the files being copied"      $files     }    #Copy files to remote computer  copy-item -path "$Localfiles" -Destination "$Remotefiles" -Recurse       #Remove Files if Value = True     if ($RemoveFilesafterCopy -eq $TRUE)     {      #Remove Files after they have been moved      Remove-Item "$Localfiles"     }       #Send Email Alert if Value = True     if ($EmailAlerts -eq $TRUE)     {      $date = Get-Date -Format g      Send-MailMessage -To $toAddr -From $fromAddr -Subject "$date Files Copied" -Body "The Following Files were moved: " + $files -SmtpServer $smtpsrv     }  } |

# How to Turn Off Windows Firewall Using PowerShell and Command Prompt

If you want to turn off the Windows Firewall, there are three methods. One is using the GUI which is the method that takes the most time, the other two methods are faster and using PowerShell and command prompt.

Turning Off Firewall Using PowerShell

On the PowerShell, execute the following command. This will turn off your firewall.

Set-NetFirewallProfile -Profile Domain,Public,Private -Enabled False

Turning Off Firewall Using Windows Command Prompt

Open the command prompt and execute the following command:

netsh advfirewall set allprofiles state off

Manage IIS

# Starting, stopping, and restarting IIS

Since you can run command-line programs in PowerShell, the IISResetcommand can be used to start, stop, and restart IIS using the /START, /STOP, and /RESTART switches.

IISReset /start

Install the MIS

psexec.exe \\$computer -s -u Adminuser -p AdminPassword msiexec /i C:\Avamar\AvamarClient-windows-x86\_64-7.0.102-47.msi /qb ADVANCED\_OPTIONS=1 CHANNEL=100

# 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

}

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

}

## 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

}

### PowerCLI Copy-VMGuest

You can use PowerCli to copy files to a vSphere guest with the [Copy-VMGuest](https://www.vmware.com/support/developer/PowerCLI/PowerCLI41U1/html/Copy-VMGuestFile.html) CmdLet.

$VM = Get-VM $computername

Copy-VMGuest -Source $file -Destination 'c:\windows\temp\installer.exe' -VM $VM

1- using new-psdrive to copy and invoke command {msiexec /i } to install

2-

Powershell

$computers = 'pc-1', 'pc-2'

$sourcefile = "\\server\script\CrystalDiskInfo7.0.4.msi"

$jobscript = {

**Param**($computer)

$destinationFolder = "\\$computer\C$\Temp"

**if** (!(Test-Path -path $destinationFolder)) {

New-Item $destinationFolder -Type Directory

}

Copy-Item -Path $sourcefile -Destination $destinationFolder

Invoke-Command -ComputerName $computer -ScriptBlock { Msiexec c:\temp\CrystalDiskInfo7.0.4.msi /i /log C:\MSIInstall.log }

}

$computer |

**ForEach**-Object{

Start-Job -ScriptBlock $jobscript -ArgumentList $\_ -Credential $domaincredentail

}

3-

$computername = 'pc-1', 'pc-2'

$sourcefile = "\\server\script\CrystalDiskInfo7.0.4.msi"

**foreach** ($computer **in** $computername) {

$destinationFolder = "\\$computer\C$\Temp"

**if** (!(Test-Path -path $destinationFolder -Credential $domaincredentials)) {

New-Item $destinationFolder -Type Directory -Credential $domaincredentials

}

Copy-Item -Path $sourcefile -Destination $destinationFolder -Credential $domaincredentials

Invoke-Command -ComputerName $computer -Credential $domaincredentials -ScriptBlock { Msiexec /I \\server\script\CrystalDiskInfo7.0.4.msi /log C:\MSIInstall.log }

# Use PowerShell to install Windows Updates

First thing you need to do is confirm the version of PowerShell you have:

|  |  |
| --- | --- |
| 1 | $PSVersionTable.PSVersion |

If version 5 or above, confirm you are running PowerShell as administrator and continue with:

|  |  |
| --- | --- |
| 1  2 | Install-Module PSWindowsUpdate  Get-Command –module PSWindowsUpdate |

Then you will need to register to use the **Microsoft Update Service** not just the default **Windows Update Service**.

Add-WUServiceManager -ServiceID 7971f918-a847-4430-9279-4a52d1efe18d

Then run:

|  |  |
| --- | --- |
| 1 | Get-WUInstall –MicrosoftUpdate –AcceptAll –AutoReboot |