HOW TO GET REMOTE SYSTEM INFORMATION Â€“ PART 4

27/08/2018 by STEPHANOS — 14 COMMENTS

How To Get Remote System Information â€“ Part 4

How To Get Remote System Information â€“ Part 4

Scenario

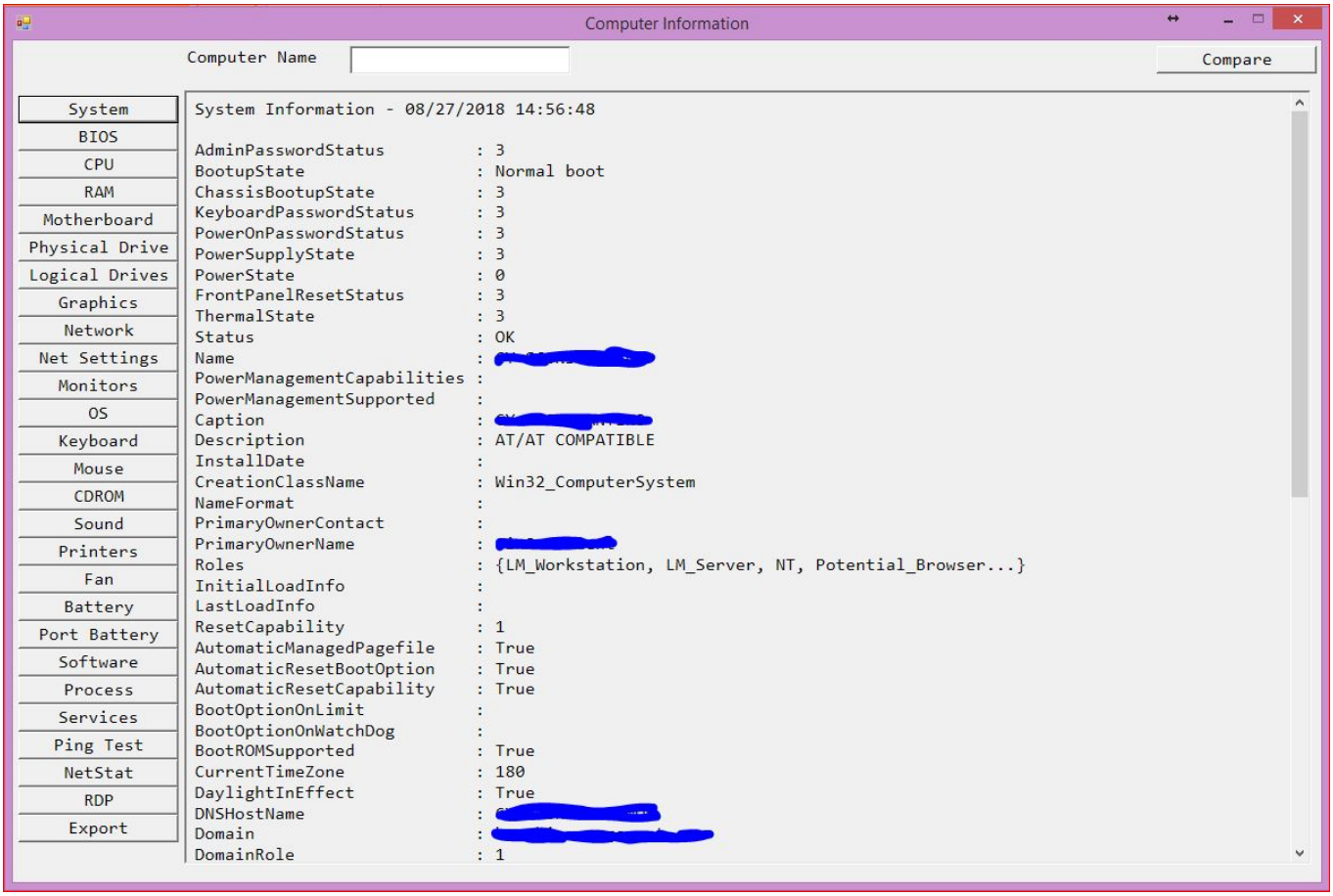
This is another part of this series. I hope you liked the previous parts. In this part we will see also some differences in the script to improve it and also the extra functionality added. As always the script code is provided below and you can also download it through TechNet Library from here.

In this post we will see the below:

- Retrieve information from two remote systems at the same time.
- Compare results of the two remote systems
- Basic error controlling
- UI and code Improvements
- All information is shown in results.

Information to be displayed

In this version of the script, the filtering of information has been removed. Now all the information that each remote system can provide will be displayed based on the class that is used. The reason I chose to remove the filtering of information is that some system administrators might need to see more information that what I have chose to display in GUI. Now the script will provide you with the full information of the class according to what the remote system provides.



[adinserter name="In Article"]

Two remote systems at the same time

After a request that it was done in TechNet, the script now is able to get the information from two remote systems at the same time. A button has been added to convert the GUI from "single" mode to "compare" mode. In "compare" mode you are allowed enter two different systems at the same time and get the information from them. The space for the results is divided and it shows you both results, one next to the other.

[adinserter name="In Article"]

Comparison of the results

As you can see in the above screenshot, at the bottom of the interface, there is a space for differences. The request in TechNet was not only to get the information from two computers at the same time but also highlighting the differences. Instead of highlighting the differences, I am comparing the

Below is the code I used to compare the results:

Code:

```
Function Compare-Computer {
    Param(
        [PSObject]$Computer1,
        [PSObject]$Computer2
```

```
    $InfoProperties = $InfoProperties | Sort-Object | Select-O
    $Differences = @()
    foreach ($InfoProperty in $InfoProperties) {
        $Difference = Compare-Object $Computer1 $Computer2 -Pr

        if ($Difference) {
            $DifferencesProperties = @{
                Property=$InfoProperty
                Computer1=($Difference | Where-Object {$_.Side
                Computer2=($Difference | Where-Object {$_.Side
            }
            $Differences += New-Object PSObject -Property $Dif
        }
    }
    if ($Differences) {
        #return ($Differences | Select-Object Property,Compute
        $lbl_compareinfo.Text = $Differences |
            Select-Object Property,Computer1,Computer2 |
            Out-String}
    else {$lbl_compareinfo.Text = "There is no difference"}
}
```

If there are no differences in the results, It will just show “There is no difference”.

[adinserter name="In Article"]

Basic error control

In this version of the script, I have added also a basic error control. I have set the default error action to "Stop" in case there is an error. In order to have a basic error control I have used Try Catch statements in the script. When there is an error, it will provide you with some information that might help you

```
$ErrorActionPreference = "Stop"

$ErrorMessage = @"
There was an error while trying to retrieve the information.

It might be one of the below cases:
 - Computer/Server is not reachable
 - Computer/Server turned off
 - Computer/Server name is not correct
 - You do not have permissions
"@

function Get-Info1 {
    $ComputerName = $txt_ComputerName1.Text
    try{
        $Info = Get-CimInstance -Class $Class -ComputerName $C
        $lbl_sysinfo1.ForeColor = "Black"
        $lbl_sysinfo1.Text = $InfoTitle
        $lbl_sysinfo1.Text += $Info |
            Select-Object -Property * |
            Out-String}
    catch{
        $lbl_sysinfo1.ForeColor = "Red"
        $lbl_sysinfo1.Text = $ErrorMessage}}
```

[adinserter name="In Article"]

Requirements

Once again I will provide you with the requirements in order to run the script.

First of all as we are using CIM commands you will need to have at least

will need to run WINRM to be able to connect to it. WMI commands are using DCOM connection to remote system. CIM commands are using WSMAN to connect to the remote system. If you are not running WINRM on them then you will not be able to get any information. Note that script will work if you run it locally as CIM uses DCOM to connect to the local system and if no computer name will be specified the script will provide information of the local system. Currently, the script uses Office 365 to send emails and port 587. In case you are using a different configuration then you will need to change the SMTP server and port to the correct one.

Do you like to include some other functions in the script?

If there are some functions that you might like to add, please let me know. I will check if I am able to add them in the script.

You can download the script here or copy it from below.

Hope you like it. If you have any questions or anything else please let me know in the comments below.

Stay tuned for the next part of this series.

[adinserter name="In Article"]

Related Links

- How to get remote system information â€“ Part 1
- How to get remote system information â€“ Part 2
- How to get remote system information â€“ Part 3
- PowerShell Try Catch Finally
- PowerShell Split Operator
- PowerShell Assignment Operators

- New-Object – Microsoft Docs
- Test-Connection – Microsoft Docs
- Out-File – Microsoft Docs
- Get-Credential – Microsoft Docs
- Send-MailMessage – Microsoft Docs
- Get-NetTCPConnection – Microsoft Docs
- Invoke-Command – Microsoft Docs
- Add-Type – Microsoft Docs
- about_Try_Catch_Finally | Microsoft Docs
- about_Split | Microsoft Docs
- about_Assignment_Operators | Microsoft Docs
- about_Quoting_Rules | Microsoft Docs
- Get-Date – Microsoft Docs
- about_Comparison_Operators | Microsoft Docs
- Out-String – Microsoft Docs
- ForEach-Object – Microsoft Docs
- Select-Object – Microsoft Docs
- Where-Object – Microsoft Docs
- Compare-Object – Microsoft Docs
- Get-Member – Microsoft Docs
- Sort-Object – Microsoft Docs
- about_Switch | Microsoft Docs
- about_If | Microsoft Docs

[adinserter name="In Article"]

Solution / Script

```
<#
.SYNOPSIS
  Name: Get-SysInfo.ps1
  The purpose of this script is to retrieve information of rem
```

```
  It will gather hardware specifications, peripherals, install
  and Operating System through a very simple and functioning G
  Remote Desktop and export the resutls in a text file or emai
  information from two remote systems at the same time and com

.RELATED LINKS

  "Home



.NOTES
  Version:      1.5

  Updated:      23-08-2018          - Added ability to retrieve
                                    - Added comparison of result
                                    - Code improvements and opti
                                    - Added basic error control
                                    - Remove filtering of inform
                                    - UI Improvements

  Updated:      13-03-2018          - Added ability to email res
                                    - Added date and time for th
                                    - Updated description

  Updated:      02-03-2018          - Added ability to export re
                                    - Added TCP Connection infor
                                    - Added Title for each infor

  Updated:      25-02-2018          - Added Fan Information
```

```
                                    - Added ping connection test
                                    - Added Remote Desktop conne
                                    - Warning for the use of Win
                                    - Added option for Win32Reg_
                                    - Added visibility to Taskba
                                    - Added Help information

  Release Date: 22-02-2018

  Author: Stephanos Constantinou

.EXAMPLE
  Run the Get-SysInfo script to retrieve the information.
  Get-SysInfo.ps1
#>

$ErrorActionPreference = "Stop"

$ErrorMessage = @"
There was an error while trying to retrieve the information.

It might be one of the below cases:
 - Computer/Server is not reachable
 - Computer/Server turned off
 - Computer/Server name is not correct
 - You do not have permissions
"@

function Get-NetStat {
    $CompareText = $btn_Compare.Text
    $lbl_compareinfo.Text = ""
```

```
    $CompareText = $btn_Compare.Text
    $lbl_compareinfo.Text = ""

    If ($CompareText -eq "Compare"){Test-Ping1}
    elseif ($CompareText -eq "Single"){Test-Ping2}}

function Get-Info {
    $CompareText = $btn_Compare.Text
    $lbl_compareinfo.Text = ""

    If ($CompareText -eq "Compare"){Get-Info1}
    elseif ($CompareText -eq "Single"){Get-Info2}}

function Get-Info1 {
    $ComputerName = $txt_ComputerName1.Text
    try{
        $Info = Get-CimInstance -Class $Class -ComputerName $C
        $lbl_sysinfo1.ForeColor = "Black"
        $lbl_sysinfo1.Text = $InfoTitle
        $lbl_sysinfo1.Text += $Info |
            Select-Object -Property * |
            Out-String}
    catch{
        $lbl_sysinfo1.ForeColor = "Red"
        $lbl_sysinfo1.Text = $ErrorMessage}}

function Get-Info2 {
    $ComputerName1 = $txt_ComputerName1.Text
    $ComputerName2 = $txt_ComputerName2.Text
    try{
        $Info1 = Get-CimInstance -Class $Class -ComputerName "
        $Info2 = Get-CimInstance -Class $Class -ComputerName "
```

```
            Out-String
        $lbl_sysinfo3.ForeColor = "Black"
        $lbl_sysinfo3.Text = $InfoTitle
        $lbl_sysinfo3.Text += $Info2 |
            Select-Object -Property * |
            Out-String

         Compare-Computer $Info1 $Info2}
    catch{
        $lbl_sysinfo2.ForeColor = "Red"
        $lbl_sysinfo2.Text = $ErrorMessage
        $lbl_sysinfo3.ForeColor = "Red"
        $lbl_sysinfo3.Text = $ErrorMessage}}

Function Compare-Computer {
    Param(
        [PSObject]$Computer1,
        [PSObject]$Computer2
    )
    $InfoProperties = $Computer1 | Get-Member -MemberType Prop
    $InfoProperties += $Computer2 | Get-Member -MemberType Pro
    $InfoProperties = $InfoProperties | Sort-Object | Select-O
    $Differences = @()
    foreach ($InfoProperty in $InfoProperties) {
        $Difference = Compare-Object $Computer1 $Computer2 -Pr
        if ($Difference) {
            $DifferencesProperties = @{
                Property=$InfoProperty
                Computer1=($Difference | Where-Object {$_.Side
                Computer2=($Difference | Where-Object {$_.Side
            }
            $Differences += New-Object PSObject -Property $Dif
```

```
            Select-Object Property,Computer1,Computer2 |
            Out-String}
    else {$lbl_compareinfo.Text = "There is no difference"}
}

function Test-Ping1 {
    $ComputerName1 = $txt_ComputerName1.Text
    $lbl_compareinfo.Text = ""

    If ($ComputerName1 -eq ""){
        $lbl_sysinfo1.ForeColor = "Red"
        $lbl_sysinfo1.Text = "Please provide a computer name t
    else {
        try{
            $Ping_Test = Test-Connection $ComputerName1
            $lbl_sysinfo1.ForeColor = "Black"
            $lbl_sysinfo1.Text = "Ping Test Information - $(Ge
            $lbl_sysinfo1.Text += $Ping_Test |
                Out-String}
        catch{$lbl_sysinfo1.Text = $ErrorMessage}}}

function Test-Ping2 {
    $ComputerName1 = $txt_ComputerName1.Text
    $ComputerName2 = $txt_ComputerName2.Text
    $lbl_compareinfo.Text = ""

    switch -Regex ($ComputerName1){
        {($ComputerName1 -eq "") -and ($ComputerName2 -ne "")}
            $lbl_sysinfo2.ForeColor = "Red"
            $lbl_sysinfo2.Text = "Please provide a computer na
            try{
                $Ping_Test2 = Test-Connection $ComputerName2
```

```
    catch{$lbl_sysinfo3.Text = $ErrorMessage}}
{($ComputerName1 -ne "") -and ($ComputerName2 -eq "")}
    $lbl_sysinfo3.ForeColor = "Red"
    $lbl_sysinfo3.Text = "Please provide a computer nar
    try{
        $Ping_Test1 = Test-Connection $ComputerName1
        $lbl_sysinfo2.ForeColor = "Black"
        $lbl_sysinfo2.Text = "Ping Test Information - 
        $lbl_sysinfo2.Text += $Ping_Test1 |
            Out-String}
    catch{$lbl_sysinfo2.Text = $ErrorMessage}}
{($ComputerName1 -eq "") -and ($ComputerName2 -eq "")}
    $lbl_sysinfo2.ForeColor = "Red"
    $lbl_sysinfo2.Text = "Please provide a computer nar
    $lbl_sysinfo3.ForeColor = "Red"
    $lbl_sysinfo3.Text = "Please provide a computer nar
{($ComputerName1 -ne "") -and ($ComputerName2 -ne "")}
    $Ping_Test1 = Test-Connection $ComputerName1
    $Ping_Test2 = Test-Connection $ComputerName2
    try{
        $Ping_Test2 = Test-Connection $ComputerName2
        $lbl_sysinfo3.ForeColor = "Black"
        $lbl_sysinfo3.Text = "Ping Test Information - 
        $lbl_sysinfo3.Text += $Ping_Test2 |
            Out-String}
    catch{$lbl_sysinfo3.Text = $ErrorMessage}
    try{
        $Ping_Test1 = Test-Connection $ComputerName1
        $lbl_sysinfo2.ForeColor = "Black"
        $lbl_sysinfo2.Text = "Ping Test Information - 
        $lbl_sysinfo2.Text += $Ping_Test1 |
            Out-String}
```

```
    $lbl_compareinfo.Text = ""

    if ($ComputerName1 -eq ""){
        try{
            $LocalNetStat = Get-NetTCPConnection
            $lbl_sysinfo1.Text = "NetStat Information - $(Get-
            $lbl_sysinfo1.Text += $LocalNetStat |
                Format-Table |
                Out-String}
        catch{$lbl_sysinfo1.Text = $ErrorMessage}}
    else{
        try{
            $RemoteNetStat = Invoke-Command -ComputerName $Com
            $lbl_sysinfo1.Text = "NetStat Information - $(Get-
            $lbl_sysinfo1.Text += $RemoteNetStat |
                Format-Table |
                Out-String }
        catch{$lbl_sysinfo1.Text = $ErrorMessage}}}

function Get-NetStat2 {
    $ComputerName1 = $txt_ComputerName1.Text
    $ComputerName2 = $txt_ComputerName2.Text
    $lbl_compareinfo.Text = ""

    switch -Regex ($ComputerName1){
        {($ComputerName1 -eq "") -and ($ComputerName2 -ne "")}
            try{
                $NetStat1 = Get-NetTCPConnection
                $lbl_sysinfo2.Text = "NetStat Information - $(
                $lbl_sysinfo2.Text += $NetStat1 |
                    Format-Table |
                    Out-String}
```

```
            $lbl_sysinfo3.Text += $NetStat2 |
                Format-Table |
                Out-String }
        catch{$lbl_sysinfo3.Text = $ErrorMessage}}
    {($ComputerName1 -ne "") -and ($ComputerName2 -eq "")}
        try{
            $NetStat1 = Invoke-Command -ComputerName $Compu
            $lbl_sysinfo2.Text = "NetStat Information - $(
            $lbl_sysinfo2.Text += $NetStat1 |
                Format-Table |
                Out-String }
        catch{$lbl_sysinfo2.Text = $ErrorMessage}
        try{
            $NetStat2 = Get-NetTCPConnection
            $lbl_sysinfo3.Text = "NetStat Information - $(
            $lbl_sysinfo3.Text += $NetStat2 |
                Format-Table |
                Out-String}
        catch{$lbl_sysinfo3.Text = $ErrorMessage}
        }
    {($ComputerName1 -eq "") -and ($ComputerName2 -eq "")}
        try{
            $NetStat1 = Get-NetTCPConnection
            $lbl_sysinfo2.Text = "NetStat Information - $(
            $lbl_sysinfo2.Text += $NetStat1 |
                Format-Table |
                Out-String}
        catch{$lbl_sysinfo2.Text = $ErrorMessage}
        try{
            $NetStat2 = Get-NetTCPConnection
            $lbl_sysinfo3.Text = "NetStat Information - $(
            $lbl_sysinfo3.Text += $NetStat2 |
```

```
            try{
                $NetStat1 = Invoke-Command -ComputerName $Compu
                $lbl_sysinfo2.Text = "NetStat Information - $(G
                $lbl_sysinfo2.Text += $NetStat1 |
                    Format-Table |
                    Out-String }
            catch{$lbl_sysinfo2.Text = $ErrorMessage}
            try{
                $NetStat2 = Invoke-Command -ComputerName $Compu
                $lbl_sysinfo3.Text = "NetStat Information - $(G
                $lbl_sysinfo3.Text += $NetStat2 |
                    Format-Table |
                    Out-String }
            catch{$lbl_sysinfo3.Text = $ErrorMessage}}}
    Compare-Computer $NetStat1 $NetStat2}

$Compare = {
    $CompareText = $btn_Compare.Text

    if ($CompareText -eq "Compare"){
        $btn_Compare.Text = "Single"
        $txt_ComputerName1.Text = ""
        $txt_ComputerName2.Text = ""
        $lbl_sysinfo1.Text = ""
        $lbl_sysinfo2.Text = ""
        $lbl_sysinfo3.Text = ""
        $lbl_compareinfo.Text = ""
        $lbl_ComputerName1.Text = "Computer Name 1"
        $pnl_sysinfo1.Controls.Remove($lbl_sysinfo1)
        $MainForm.Controls.Remove($pnl_sysinfo1)
        $MainForm.Controls.Remove($btn_RDP)
        $MainForm.Controls.Remove($btn_Export)
```

```
        $MainForm.Controls.Add($pnl_compareinfo)
        $MainForm.Controls.Add($lbl_differences)
        $pnl_sysinfo2.Controls.Add($lbl_sysinfo2)
        $pnl_sysinfo3.Controls.Add($lbl_sysinfo3)
        $pnl_compareinfo.Controls.Add($lbl_compareinfo)}
    elseif($CompareText -eq "Single"){
        $btn_Compare.Text = "Compare"
        $txt_ComputerName1.Text = ""
        $lbl_sysinfo1.Text = ""
        $lbl_sysinfo2.Text = ""
        $lbl_sysinfo3.Text = ""
        $lbl_compareinfo.Text = ""
        $lbl_ComputerName1.Text = "Computer Name"
        $pnl_sysinfo2.Controls.Remove($lbl_sysinfo2)
        $pnl_sysinfo3.Controls.Remove($lbl_sysinfo3)
        $pnl_compareinfo.Controls.Remove($lbl_compareinfo)
        $MainForm.Controls.Add($btn_RDP)
        $MainForm.Controls.Add($btn_Export)
        $MainForm.Controls.Remove($pnl_sysinfo2)
        $MainForm.Controls.Remove($pnl_sysinfo3)
        $MainForm.Controls.Remove($lbl_ComputerName2)
        $MainForm.Controls.Remove($txt_ComputerName2)
        $MainForm.Controls.Remove($lbl_differences)
        $MainForm.Controls.Add($pnl_sysinfo1)
        $MainForm.Controls.Remove($pnl_compareinfo)
        $pnl_sysinfo1.Controls.Add($lbl_sysinfo1)}
        $MainForm.Refresh()}

$System_info = {
    $Class = "Win32_ComputerSystem"
    $InfoTitle = "System Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}
```

```
    Get-Info $Class $InfoTitle}

$CPU_info = {
    $Class = "Win32_Processor"
    $InfoTitle = "CPU Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$RAM_info = {
    $Class = "Win32_PhysicalMemory"
    $InfoTitle = "RAM Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$MB_info = {
    $Class = "Win32_BaseBoard"
    $InfoTitle = "MotherBoard Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$PhysicalDrives_info = {
    $Class = "Win32_DiskDrive"
    $InfoTitle = "Physical Drives Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$LogicalDrives_info = {
    $Class = "Win32_LogicalDisk"
    $InfoTitle = "Logical Drives Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$GPU_info = {
    $Class = "Win32_VideoController"
    $InfoTitle = "GPU Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}
```

```
$NetSettings_info = {
    $Class = "Win32_NetworkAdapterConfiguration"
    $InfoTitle = "Network Configuration Information - $(Get-Da
    Get-Info $Class $InfoTitle}

$Monitor_info = {
    $Class = "Win32_DesktopMonitor"
    $InfoTitle = "Monitors Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$OS_info = {
    $Class = "Win32_OperatingSystem"
    $InfoTitle = "Operating System Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$Keyboard_info = {
    $Class = "Win32_Keyboard"
    $InfoTitle = "Keyboard Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$Mouse_info = {
    $Class = "Win32_PointingDevice"
    $InfoTitle = "Pointing Device Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$CDROM_info = {
    $Class = "Win32_CDROMDrive"
    $InfoTitle = "CD-ROM Drives Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$Sound_info = {
```

```
$Printers_info = {
    $Class = "Win32_Printer"
    $InfoTitle = "Printers Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$Fan_info = {
    $Class = "Win32_Fan"
    $InfoTitle = "Fans Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$Battery_info = {
    $Class = "Win32_Battery"
    $InfoTitle = "Battery Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$PortBattery_info = {
    $Class = "Win32_PortableBattery"
    $InfoTitle = "Portable Battery Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$Software_info = {
    $Product = {
        $Warning = [System.Windows.MessageBox]::Show('Are you

        switch ($Warning){
            Yes{
                $SoftwareOption.Close()
                $Class = "Win32Reg_Product"
                $InfoTitle = "Software Information - $(Get-Dat
                Get-Info $Class $InfoTitle}
            No{Break}
        }
```

```
    $Class = "Win32Reg_AddRemovePrograms"
    $InfoTitle = "Software Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$SoftwareOption = New-Object system.Windows.Forms.Form
$SoftwareOption.Text = "Class Option"
$SoftwareOption.Size = New-Object System.Drawing.Size(500,
$SoftwareOption.AutoSize = $False
$SoftwareOption.AutoScroll = $False
$SoftwareOption.MinimizeBox = $False
$SoftwareOption.MaximizeBox = $False
$SoftwareOption.WindowState = "Normal"
$SoftwareOption.SizeGripStyle = "Hide"
$SoftwareOption.ShowInTaskbar = $True
$SoftwareOption.Opacity = 1
$SoftwareOption.FormBorderStyle = "Fixed3D"
$SoftwareOption.StartPosition = "CenterScreen"

$lbl_SoftwareOption = New-Object System.Windows.Forms.Labe
$lbl_SoftwareOption.Location = New-Object System.Drawing.P
$lbl_SoftwareOption.Size = New-Object System.Drawing.Size(
$lbl_SoftwareOption.Text = "Please select the class that y
$lbl_SoftwareOption.Font = $Font
$SoftwareOption.Controls.Add($lbl_SoftwareOption)

$btn_Product = New-Object System.Windows.Forms.Button
$btn_Product.Location = New-Object System.Drawing.Point(10
$btn_Product.Size = New-Object System.Drawing.Size(230,25)
$btn_Product.Text = "Win32_Product"
$btn_Product.Font = $Font
$btn_Product.Add_Click($Product)
$SoftwareOption.Controls.Add($btn_Product)
```

```
    $btn_AddRemove.Text = "Win32_AddRemovePrograms"
    $btn_AddRemove.Font = $Font
    $btn_AddRemove.Add_Click($AddRemove)
    $SoftwareOption.Controls.Add($btn_AddRemove)

    $SoftwareOption.ShowDialog()
}

$Process_info = {
    $Class = "Win32_Process"
    $InfoTitle = "Processes Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$Services_info = {
    $Class = "Win32_Service"
    $InfoTitle = "Services Information - $(Get-Date)"
    Get-Info $Class $InfoTitle}

$RDP_Connection = {
    $ComputerName1 = $txt_ComputerName1.Text
    mstsc /v:$ComputerName1}

$Export = {
    $ComputerName1 = $txt_ComputerName1.Text

    $TextFile = {
        $ExportOption.Close()

        if ($ComputerName1 -eq ""){
            try{
                $ComputerName1 = (Get-CimInstance -Class Win32_
                $lbl_sysinfo1.Text |
```

```
    if ($ComputerName1 -eq ""){
        try{
            $ComputerName1 = (Get-CimInstance -Class Win32_
            $lbl_sysinfo1.Text |
                Out-File C:\Scripts\$ComputerName1.txt}
        catch{$lbl_sysinfo1.Text = $ErrorMessage}}

    $To  = @(($txt_Recipients.Text) -split ',')
    $Attachement = "C:\Scripts\$ComputerName1.txt"
    $Recipients.Close()
    $EmailCredentials = Get-Credential
    $From = $EmailCredentials.UserName
    $EmailParameters = @{
        To = $To
        Subject = "System Information - $ComputerName1"
        Body = "Please find attached the information that
        Attachments = $Attachement
            UseSsl = $True
            Port = "587"
            SmtpServer = "smtp.office365.com"
            Credential = $EmailCredentials
            From = $From}

    Send-MailMessage @EmailParameters}

$RecipientsDetails = {
    $ExportOption.Close()

    $Recipients = New-Object system.Windows.Forms.Form
    $Recipients.Text = "Recipients"
    $Recipients.Size = New-Object System.Drawing.Size(500,
    $Recipients.AutoSize = $False
```

```
        $Recipients.SizeGripStyle = "Hide"
        $Recipients.ShowInTaskbar = $True
        $Recipients.Opacity = 1
        $Recipients.FormBorderStyle = "Fixed3D"
        $Recipients.StartPosition = "CenterScreen"

        $RecipientsInfo = @"
Please enter the recipient.

If there are multiple recipients, separate recipients with com
"@

        $lbl_Recipients = New-Object System.Windows.Forms.Labe
        $lbl_Recipients.Location = New-Object System.Drawing.P
        $lbl_Recipients.Size = New-Object System.Drawing.Size(
        $lbl_Recipients.Text = $RecipientsInfo
        $lbl_Recipients.Font = $Font
        $Recipients.Controls.Add($lbl_Recipients)

        $txt_Recipients = New-Object System.Windows.Forms.Textl
        $txt_Recipients.Location = New-Object System.Drawing.P
        $txt_Recipients.Size = New-Object System.Drawing.Size(
        $txt_Recipients.Font = $Font
        $Recipients.Controls.Add($txt_Recipients)

        $btn_Recipients = New-Object System.Windows.Forms.Butt
        $btn_Recipients.Location = New-Object System.Drawing.P
        $btn_Recipients.Size = New-Object System.Drawing.Size(
        $btn_Recipients.Text = "OK"
        $btn_Recipients.Font = $Font
        $btn_Recipients.Add_Click($Email)
        $Recipients.Controls.Add($btn_Recipients)
```

```
$ExportOption.Text = "Export Method"
$ExportOption.Size = New-Object System.Drawing.Size(500,13
$ExportOption.AutoSize = $False
$ExportOption.AutoScroll = $False
$ExportOption.MinimizeBox = $False
$ExportOption.MaximizeBox = $False
$ExportOption.WindowState = "Normal"
$ExportOption.SizeGripStyle = "Hide"
$ExportOption.ShowInTaskbar = $True
$ExportOption.Opacity = 1
$ExportOption.FormBorderStyle = "Fixed3D"
$ExportOption.StartPosition = "CenterScreen"

$lbl_ExportOption = New-Object System.Windows.Forms.Label
$lbl_ExportOption.Location = New-Object System.Drawing.Poi
$lbl_ExportOption.Size = New-Object System.Drawing.Size(50
$lbl_ExportOption.Text = "Please select how you want to ex
$lbl_ExportOption.Font = $Font
$ExportOption.Controls.Add($lbl_ExportOption)

$btn_TextFile = New-Object System.Windows.Forms.Button
$btn_TextFile.Location = New-Object System.Drawing.Point(1
$btn_TextFile.Size = New-Object System.Drawing.Size(230,25
$btn_TextFile.Text = "Text File"
$btn_TextFile.Font = $Font
$btn_TextFile.Add_Click($TextFile)
$ExportOption.Controls.Add($btn_TextFile)

$btn_Email = New-Object System.Windows.Forms.Button
$btn_Email.Location = New-Object System.Drawing.Point(250,
$btn_Email.Size = New-Object System.Drawing.Size(230,25)
$btn_Email.Text = "Email"
```

```
    $ExportOption.ShowDialog()}

Add-Type -AssemblyName System.Windows.Forms

$Font = New-Object System.Drawing.Font("Consolas",12,[System.D

$MainForm = New-Object system.Windows.Forms.Form
$MainForm.Text = "Computer Information"
$MainForm.Size = New-Object System.Drawing.Size(1200,800)
$MainForm.AutoScroll = $False
$MainForm.AutoSize = $False
$MainForm.FormBorderStyle = "FixedSingle"
$MainForm.MinimizeBox = $True
$MainForm.MaximizeBox = $False
$MainForm.WindowState = "Normal"
$MainForm.SizeGripStyle = "Hide"
$MainForm.ShowInTaskbar = $True
$MainForm.Opacity = 1
$MainForm.StartPosition = "CenterScreen"
$MainForm.ShowInTaskbar = $True
$MainForm.Font = $Font

$btn_Compare = New-Object System.Windows.Forms.Button
$btn_Compare.Location = New-Object System.Drawing.Point(1035,5
$btn_Compare.Size = New-Object System.Drawing.Size (145,25)
$btn_Compare.Font = $Font
$btn_Compare.Text = "Compare"
$btn_Compare.Add_Click($Compare)
$MainForm.Controls.Add($btn_Compare)

$lbl_ComputerName1 = New-Object System.Windows.Forms.Label
$lbl_ComputerName1.Location = New-Object System.Drawing.Point(
```

```
$txt_ComputerName1 = New-Object System.Windows.Forms.TextBox
$txt_ComputerName1.Location = New-Object System.Drawing.Point(
$txt_ComputerName1.Size = New-Object System.Drawing.Size(200,2
$txt_ComputerName1.Font = $Font
$MainForm.Controls.Add($txt_ComputerName1)

$lbl_ComputerName2 = New-Object System.Windows.Forms.Label
$lbl_ComputerName2.Location = New-Object System.Drawing.Point(
$lbl_ComputerName2.Size = New-Object System.Drawing.Size(150,2
$lbl_ComputerName2.Font = $Font
$lbl_ComputerName2.Text = "Computer Name 2"

$txt_ComputerName2 = New-Object System.Windows.Forms.TextBox
$txt_ComputerName2.Location = New-Object System.Drawing.Point(
$txt_ComputerName2.Size = New-Object System.Drawing.Size(200,2
$txt_ComputerName2.Font = $Font

$pnl_sysinfo1 = New-Object System.Windows.Forms.Panel
$pnl_sysinfo1.Location = New-Object System.Drawing.Point(155,4
$pnl_sysinfo1.Size = New-Object System.Drawing.Size(1020,700)
$pnl_sysinfo1.BorderStyle = "Fixed3D"
$pnl_sysinfo1.AutoSize = $False
$pnl_sysinfo1.AutoScroll = $True
$pnl_sysinfo1.Font = $Font
$pnl_sysinfo1.Text = ""
$MainForm.Controls.Add($pnl_sysinfo1)

$lbl_sysinfo1 = New-Object System.Windows.Forms.Label
$lbl_sysinfo1.Location = New-Object System.Drawing.Point(5,5)
$lbl_sysinfo1.Size = New-Object System.Drawing.Size(490,490)
$lbl_sysinfo1.AutoSize = $True
```

```
$pnl_sysinfo2 = New-Object System.Windows.Forms.Panel
$pnl_sysinfo2.Location = New-Object System.Drawing.Point(155,4
$pnl_sysinfo2.Size = New-Object System.Drawing.Size(510,400)
$pnl_sysinfo2.BorderStyle = "Fixed3D"
$pnl_sysinfo2.AutoSize = $False
$pnl_sysinfo2.AutoScroll = $True
$pnl_sysinfo2.Font = $Font
$pnl_sysinfo2.Text = ""

$lbl_sysinfo2 = New-Object System.Windows.Forms.Label
$lbl_sysinfo2.Location = New-Object System.Drawing.Point(5,5)
$lbl_sysinfo2.Size = New-Object System.Drawing.Size(490,490)
$lbl_sysinfo2.AutoSize = $True

$lbl_sysinfo2.Font = $Font
$lbl_sysinfo2.Text = ""

$pnl_sysinfo3 = New-Object System.Windows.Forms.Panel
$pnl_sysinfo3.Location = New-Object System.Drawing.Point(665,4
$pnl_sysinfo3.Size = New-Object System.Drawing.Size(510,400)
$pnl_sysinfo3.BorderStyle = "Fixed3D"
$pnl_sysinfo3.AutoSize = $False
$pnl_sysinfo3.AutoScroll = $True
$pnl_sysinfo3.Font = $Font
$pnl_sysinfo3.Text = ""

$lbl_sysinfo3 = New-Object System.Windows.Forms.Label
$lbl_sysinfo3.Location = New-Object System.Drawing.Point(5,5)
$lbl_sysinfo3.Size = New-Object System.Drawing.Size(490,490)
$lbl_sysinfo3.AutoSize = $True
$lbl_sysinfo3.Font = $Font
$lbl_sysinfo3.Text = ""
```

```
$lbl_differences.Font = $Font
$lbl_differences.Text = "Differences"

$pnl_compareinfo = New-Object System.Windows.Forms.Panel
$pnl_compareinfo.Location = New-Object System.Drawing.Point(15
$pnl_compareinfo.Size = New-Object System.Drawing.Size(1020,28
$pnl_compareinfo.BorderStyle = "Fixed3D"
$pnl_compareinfo.AutoSize = $False
$pnl_compareinfo.AutoScroll = $True
$pnl_compareinfo.Font = $Font
$pnl_compareinfo.Text = ""

$lbl_compareinfo = New-Object System.Windows.Forms.Label
$lbl_compareinfo.Location = New-Object System.Drawing.Point(5,
$lbl_compareinfo.Size = New-Object System.Drawing.Size(100,100
$lbl_compareinfo.AutoSize = $True
$lbl_compareinfo.Font = $Font
$lbl_compareinfo.Text = ""

$btn_System = New-Object System.Windows.Forms.Button
$btn_System.Location = New-Object System.Drawing.Point(5,50)
$btn_System.Size = New-Object System.Drawing.Size(145,25)
$btn_System.Font = $Font
$btn_System.Text = "System"
$btn_System.Add_Click($System_info)
$MainForm.Controls.Add($btn_System)

$btn_BIOS = New-Object System.Windows.Forms.Button
$btn_BIOS.Location = New-Object System.Drawing.Point(5,75)
$btn_BIOS.Size = New-Object System.Drawing.Size(145,25)
$btn_BIOS.Font = $Font
$btn_BIOS.Text = "BIOS"
```

```
$btn_CPU.Location = New-Object System.Drawing.Point(5,100)
$btn_CPU.Size = New-Object System.Drawing.Size(145,25)
$btn_CPU.Font = $Font
$btn_CPU.Text = "CPU"
$btn_CPU.Add_Click($cpu_info)
$MainForm.Controls.Add($btn_CPU)

$btn_RAM = New-Object System.Windows.Forms.Button
$btn_RAM.Location = New-Object System.Drawing.Point(5,125)
$btn_RAM.Size = New-Object System.Drawing.Size(145,25)
$btn_RAM.Font = $Font
$btn_RAM.Text = "RAM"
$btn_RAM.Add_Click($ram_info)
$MainForm.Controls.Add($btn_RAM)

$btn_MB = New-Object System.Windows.Forms.Button
$btn_MB.Location = New-Object System.Drawing.Point(5,150)
$btn_MB.Size = New-Object System.Drawing.Size(145,25)
$btn_MB.Font = $Font
$btn_MB.Text = "Motherboard"
$btn_MB.Add_Click($mb_info)
$MainForm.Controls.Add($btn_MB)

$btn_PhysicalDrives = New-Object System.Windows.Forms.Button
$btn_PhysicalDrives.Location = New-Object System.Drawing.Point
$btn_PhysicalDrives.Size = New-Object System.Drawing.Size(145,
$btn_PhysicalDrives.Font = $Font
$btn_PhysicalDrives.Text = "Physical Drives"
$btn_PhysicalDrives.Add_Click($PhysicalDrives_info)
$MainForm.Controls.Add($btn_PhysicalDrives)

$btn_LogicalDrives = New-Object System.Windows.Forms.Button
```

```
$btn_LogicalDrives.Add_Click($LogicalDrives_info)
$MainForm.Controls.Add($btn_LogicalDrives)

$btn_Graphics = New-Object System.Windows.Forms.Button
$btn_Graphics.Location = New-Object System.Drawing.Point(5,225
$btn_Graphics.Size = New-Object System.Drawing.Size(145,25)
$btn_Graphics.Font = $Font
$btn_Graphics.Text = "Graphics"
$btn_Graphics.Add_Click($GPU_info)
$MainForm.Controls.Add($btn_Graphics)

$btn_Network = New-Object System.Windows.Forms.Button
$btn_Network.Location = New-Object System.Drawing.Point(5,250)
$btn_Network.Size = New-Object System.Drawing.Size(145,25)
$btn_Network.Font = $Font
$btn_Network.Text = "Network"
$btn_Network.Add_Click($Network_info)
$MainForm.Controls.Add($btn_Network)

$btn_NetSettings = New-Object System.Windows.Forms.Button
$btn_NetSettings.Location = New-Object System.Drawing.Point(5,
$btn_NetSettings.Size = New-Object System.Drawing.Size(145,25)
$btn_NetSettings.Font = $Font
$btn_NetSettings.Text = "Net Settings"
$btn_NetSettings.Add_Click($NetSettings_info)
$MainForm.Controls.Add($btn_NetSettings)

$btn_Monitors = New-Object System.Windows.Forms.Button
$btn_Monitors.Location = New-Object System.Drawing.Point(5,300
$btn_Monitors.Size = New-Object System.Drawing.Size(145,25)
$btn_Monitors.Font = $Font
$btn_Monitors.Text = "Monitors"
```

```
$btn_OS.Location = New-Object System.Drawing.Point(5,325)
$btn_OS.Size = New-Object System.Drawing.Size(145,25)
$btn_OS.Font = $Font
$btn_OS.Text = "OS"
$btn_OS.Add_Click($OS_info)
$MainForm.Controls.Add($btn_OS)

$btn_Keyboard = New-Object System.Windows.Forms.Button
$btn_Keyboard.Location = New-Object System.Drawing.Point(5,350
$btn_Keyboard.Size = New-Object System.Drawing.Size(145,25)
$btn_Keyboard.Font = $Font
$btn_Keyboard.Text = "Keyboard"
$btn_Keyboard.Add_Click($Keyboard_info)
$MainForm.Controls.Add($btn_Keyboard)

$btn_Mouse = New-Object System.Windows.Forms.Button
$btn_Mouse.Location = New-Object System.Drawing.Point(5,375)
$btn_Mouse.Size = New-Object System.Drawing.Size(145,25)
$btn_Mouse.Font = $Font
$btn_Mouse.Text = "Mouse"
$btn_Mouse.Add_Click($Mouse_info)
$MainForm.Controls.Add($btn_Mouse)

$btn_CDROM = New-Object System.Windows.Forms.Button
$btn_CDROM.Location = New-Object System.Drawing.Point(5,400)
$btn_CDROM.Size = New-Object System.Drawing.Size(145,25)
$btn_CDROM.Font = $Font
$btn_CDROM.Text = "CDROM"
$btn_CDROM.Add_Click($CDROM_info)
$MainForm.Controls.Add($btn_CDROM)

$btn_Sound = New-Object System.Windows.Forms.Button
```

```
$btn_Sound.Add_Click($Sound_info)
$MainForm.Controls.Add($btn_Sound)

$btn_Printers = New-Object System.Windows.Forms.Button
$btn_Printers.Location = New-Object System.Drawing.Point(5,450
$btn_Printers.Size = New-Object System.Drawing.Size(145,25)
$btn_Printers.Font = $Font
$btn_Printers.Text = "Printers"
$btn_Printers.Add_Click($Printers_info)
$MainForm.Controls.Add($btn_Printers)

$btn_Fan = New-Object System.Windows.Forms.Button
$btn_Fan.Location = New-Object System.Drawing.Point(5,475)
$btn_Fan.Size = New-Object System.Drawing.Size(145,25)
$btn_Fan.Font = $Font
$btn_Fan.Text = "Fan"
$btn_Fan.Add_Click($Fan_info)
$MainForm.Controls.Add($btn_Fan)

$btn_Battery = New-Object System.Windows.Forms.Button
$btn_Battery.Location = New-Object System.Drawing.Point(5,500)
$btn_Battery.Size = New-Object System.Drawing.Size(145,25)
$btn_Battery.Font = $Font
$btn_Battery.Text = "Battery"
$btn_Battery.Add_Click($Battery_info)
$MainForm.Controls.Add($btn_Battery)

$btn_PortBattery = New-Object System.Windows.Forms.Button
$btn_PortBattery.Location = New-Object System.Drawing.Point(5,
$btn_PortBattery.Size = New-Object System.Drawing.Size(145,25)
$btn_PortBattery.Font = $Font
$btn_PortBattery.Text = "Port Battery"
```

```
$btn_Software.Location = New-Object System.Drawing.Point(5,550
$btn_Software.Size = New-Object System.Drawing.Size(145,25)
$btn_Software.Font = $Font
$btn_Software.Text = "Software"
$btn_Software.Add_Click($Software_info)
$MainForm.Controls.Add($btn_Software)

$btn_Process = New-Object System.Windows.Forms.Button
$btn_Process.Location = New-Object System.Drawing.Point(5,575)
$btn_Process.Size = New-Object System.Drawing.Size(145,25)
$btn_Process.Font = $Font
$btn_Process.Text = "Process"
$btn_Process.Add_Click($Process_info)
$MainForm.Controls.Add($btn_Process)

$btn_Services = New-Object System.Windows.Forms.Button
$btn_Services.Location = New-Object System.Drawing.Point(5,600
$btn_Services.Size = New-Object System.Drawing.Size(145,25)
$btn_Services.Font = $Font
$btn_Services.Text = "Services"
$btn_Services.Add_Click($Services_info)
$MainForm.Controls.Add($btn_Services)

$btn_Ping = New-Object System.Windows.Forms.Button
$btn_Ping.Location = New-Object System.Drawing.Point(5,625)
$btn_Ping.Size = New-Object System.Drawing.Size(145,25)
$btn_Ping.Font = $Font
$btn_Ping.Text = "Ping Test"
$btn_Ping.Add_Click({Test-Ping})
$MainForm.Controls.Add($btn_Ping)

$btn_NetStat = New-Object System.Windows.Forms.Button
```

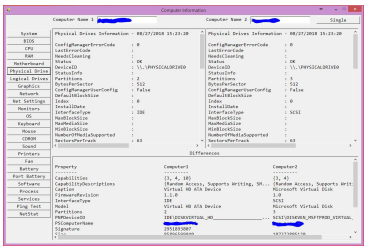
```
$btn_NetStat.Add_Click({Get-NetStat})
$MainForm.Controls.Add($btn_NetStat)

$btn_RDP = New-Object System.Windows.Forms.Button
$btn_RDP.Location = New-Object System.Drawing.Point(5,675)
$btn_RDP.Size = New-Object System.Drawing.Size(145,25)
$btn_RDP.Font = $Font
$btn_RDP.Text = "RDP"
$btn_RDP.Add_Click($RDP_Connection)
$MainForm.Controls.Add($btn_RDP)

$btn_Export = New-Object System.Windows.Forms.Button
$btn_Export.Location = New-Object System.Drawing.Point(5,700)
$btn_Export.Size = New-Object System.Drawing.Size(145,25)
$btn_Export.Font = $Font
$btn_Export.Text = "Export"
$btn_Export.Add_Click($Export)
$MainForm.Controls.Add($btn_Export)

$MainForm.ShowDialog()
```


Summary



Article Name	How To Get Remote System Information â€“ Part 4
Description	Here you will find information about How To Get Remote System Information â€“ Part 4. Stephanos Constantinou Blog
Author	Stephanos
Publisher Name	Stephanos Constantinou Blog
Publisher Logo	

FILED UNDER: POWERSHELL SCRIPTS

TAGGED WITH: ADD-TYPE, ASSIGNMENT OPERATORS, COMPARE-OBJECT, COMPARISON OPERATORS, FOREACH-OBJECT, FUNCTIONS, GET-CIMINSTANCE, GET-CREDENTIAL, GET-MEMBER, GET-NETTCPCONNECTION, INVOKE-COMMAND, NETSTAT, NEW-OBJECT, OUT-FILE, OUT-STRING, PING, POWERSHELL FOREACH, POWERSHELL IF, POWERSHELL SWITCH, SELECT-OBJECT, SEND-MAILMESSAGE, SORT-OBJECT, TEST-CONNECTION, WHERE-OBJECT, WIN32_BASEBOARD, WIN32_BATTERY, WIN32_BIOS, WIN32_CDROMDRIVE, WIN32_COMPUTERSYSTEM, WIN32_DESKTOPMONITOR, WIN32_DISKDRIVE, WIN32_FAN, WIN32_KEYBOARD, WIN32_LOGICALDISK, WIN32_NETWORKADAPTER, WIN32_NETWORKADAPTERCONFIGURATION, WIN32_OPERATINGSYSTEM, WIN32_PHYSICALMEMORY, WIN32_POINTINGDEVICE, WIN32_PORTABLEBATTERY, WIN32_PRINTER, WIN32_PROCESS, WIN32_PROCESSOR, WIN32_PRODUCT, WIN32_SERVICE, WIN32_SOUNDDEVICE, WIN32_VIDEOCONTROLLER, WIN32REG_ADDREMOVEPROGRAMS

Comments

Hey awesome tool you have created. You have inspired me to learn more about wmi/cim and try and create my own.

Reply

Stephanos says
28/08/2018 at 10:22

Hello Andy,

I'm glad that you got inspired by the script. Creating your own script will help you understand better how it works and build your own custom solution.

Hope my future posts inspire you too.

Thanks
Stephanos

Reply

Francesco Mantovani says
05/09/2018 at 03:26

Awesome tool indeed.
I have 2 feedbacks:
1) When I don't have permissions for the remote computer, can you please prompt a box for entering remote user and password. This way I can connect and compare to the remote PC

Reply

Stephanos says
05/09/2018 at 10:35

Dear Francesco,

Thank you for your feedback.

I have fixed the typo error . I will check your recommendation and for the credentials prompt and I will try to include it in the next update.

Thanks
Stephanos

Reply

HaiNH says
12/10/2018 at 12:55

Thank you for great tool.
I tried to build a similar tool base on your script but it run well with Powershell ISE but not run when right click and choose Run with powershell. What do I wrong?

Reply

Stephanos says
16/10/2018 at 10:26

Have you checked the execution policy?

Thanks
Stephanos

Reply

Johnsons says
25/10/2018 at 23:46

Dear Stephanos,

Good day,

I am unable pull the information and getting below message

it might be one of the below cases:
Computer/Server is not reachable
Computer/Server turned off
Computer/Server name is not correct
You do not have permissions

Reply

Stephanos says
26/10/2018 at 17:30

Hello Johnsons,

Thanks
Stephanos

Reply

Johnson says
28/10/2018 at 21:37

Dear Stephanos,

it’s remote computer.
WinRM default enabled right ?
Thanks,
Johnson.s

Reply

Stephanos says
30/10/2018 at 10:15

Dear Johnson,

From Windows Server 2012 and above, WinRM is enabled by default. You may need to configured listener.

You can run WinRM quickconfig.

Thanks
Stephanos

Johnson says
01/11/2018 at 09:30

Dear Stephanos,

How to configure listener.
How to run WinRM quickconfig?

Regards,
Johnson.s

Reply

Stephanos says
01/11/2018 at 12:37

Dear Johnson,

Just type WinRM quickconfig in PowerShell and follow the instructions.

Thanks
Stephanos

Reply

Guy Firmin says
22/04/2019 at 22:15

Hi Stephanos.

a spreadsheet.

Thank you in advance

Reply

Chathura says
23/04/2019 at 10:57

I am cannot get the information due to getting below message

it might be one of the below cases:
Computer/Server is not reachable
Computer/Server turned off
Computer/Server name is not correct
You do not have permissions

I have already WinRM enabled. Ping Test is working properly but cannot get any other information. kindly please help to fix this.

Reply

Leave a Reply

Your email address will not be published.

Comment

Name

Email

Website

POST COMMENT

This site uses Akismet to reduce spam. Learn how your comment data is processed.

ICS Cube Product Review 26/04/2019
PowerShell Module SysInfo v1.2.0 15/03/2019
PowerShell Module SysInfo v1.1.2 13/11/2018
PowerShell Module SysInfo 24/10/2018
Get-VoltageProbe 24/10/2018
Get-VideoController 24/10/2018
Get-USBController 24/10/2018
Get-TrackPoint 24/10/2018
Get-TrackBall 24/10/2018
Get-TouchScreen 24/10/2018

Modules Cmdlets (57)
PowerShell Modules (5)
PowerShell Scripts (38)
PowerShell Tutorials (35)
Software Reviews (2)

Archives

April 2019 (1)
March 2019 (1)
November 2018 (1)
October 2018 (56)
September 2018 (13)
August 2018 (9)
July 2018 (6)
June 2018 (8)
May 2018 (7)
April 2018 (9)
March 2018 (4)
February 2018 (6)
January 2018 (12)
December 2017 (4)

Planet PowerShell
Reddit – PowerShell
PowerShell Magazine
PowerShell.org
PowerShell Team Blog
Hey, Scripting Guy! Blog
Mike F Robbins
PowerShell Explained with Kevin Marquette
Mike Kanakos – Network Admin
The Lonely Administrator
AskME4Tech