*HSS-Windows Team*

Powershell Automation for Quality Checks

|  |  |  |
| --- | --- | --- |
| **Author(s)** | Windows SSP HSS Team |  |
| **Date** | 27-Aug-2020 |
| **Version** | 1.00 |

**VERSION HISTORY**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Update |
| V1.00 | 27-Aug-2020 | Sunil Chaudhari | Document creation |
|  |  |  |  |
|  |  |  |  |

## Purpose

Powershell Automation is the important to reduce the time to perform Quality checks post assignment of Project servers to Windows team and once the required tasks or action items completed from installation and configuration perspective.

In order to reduce the time to manually perform the Quality checks we can automate the tasks using powershell to enhance the productivity of team members.

The powershell script has been created by leveraginh the windows management instrumentation service of windows along with .net framework.

The current infrastructure will be analysed further to find out the best approach to automate the requirement for existing Server environment.

## Audience

Windows HSS SSP team members

## References

|  |  |
| --- | --- |
| Title | Version |
|  |  |
|  |  |

## Terms and abbrevations

| **Acronyms** | **Meaning** |
| --- | --- |
|  |  |
|  |  |

# Powershell Functions

## Routeprint

The **route command** allows you to make manual entries into the network routing tables. The **route command** distinguishes between **routes** to hosts and **routes** to networks by interpreting the network address of the Destination variable, which can be specified either by symbolic name or numeric address.

Roue function used in Powershell script

function routeprint {

$routeprinttemp = route print

$routeprinttemp | out-file "C:\Serverinfo\details\routeprint\_Pre.txt"

}

The above command prints the route to file path C:\Serverinfo\details\routeprint\_Pre.txt

## Get-ipconfig

The ipconfig command is a fast way of determining your computer's **IP address** and other information, such as the address of its default gateway—useful if you want to know the **IP address** of your router's web interface.

function Get-IPconfig {

$Ipconfig = Ipconfig /all

$Ipconfig | out-file "C:\Serverinfo\details\Ipconfig\_Pre.txt"

}

The above command prints the route to file path C:\Serverinfo\details\Ipconfig\_Pre.txt

Once executed using powershell.

## IPdetails

Function IPdetails($servername)

{

$IPAddress1 = ""

$IPAddress = (Get-WmiObject -ComputerName $servername Win32\_NetworkAdapterConfiguration | Where-Object { $\_.IPAddress -ne $null }) | foreach {

$IPAddress2 = $\_.IPAddress

$IPAddress1 = "$IPAddress2 | "

$IPAddress1

The above command finds out the IP address details of the computer using Get-WmiObject.

## Get-dotnetversion

The below powershell script block is used to fetch dotnetversion frameowork version from the windows registry path specifoed in the script.

Function Get-dotnetversion ($servername){

$installdotnet3 = [System.Collections.ArrayList]@()

$installdotnet4 = [System.Collections.ArrayList]@()

$Reg1 = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey('LocalMachine', $server)

if($? -eq $true)

{

$key1 = $Reg1.OpenSubKey("SOFTWARE\Microsoft\NET Framework Setup\NDP").GetSubKeyNames()

foreach($key in $key1)

{

$Reg2 = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey('LocalMachine', $server)

if($? -eq $true)

{

$key2 = $Reg2.OpenSubKey("SOFTWARE\Microsoft\NET Framework Setup\NDP\$key")

$regvalue = $key2.getvalue("Install")

if ($regvalue -eq 1)

{

$installdotnet1 = $key

$add = $installdotnet3.Add($installdotnet1)

}

if($regvalue -eq $null)

{

$key2 = $Reg2.OpenSubKey("SOFTWARE\Microsoft\NET Framework Setup\NDP\$key\client")

$regvalue = $key2.getvalue("Install")

if($? -eq $true)

{

$Versionregvalue = $key2.getvalue("Version")

if($regvalue -eq 1)

{

$installdotnet2 = $regvalue

$add1=$installdotnet4.Add($Versionregvalue)

}

}

}

else

{

Clear-Variable installdotnet1

}

}

}

$dotnetStr1 = ""

foreach($dotnet1 in $installdotnet3)

{

$dotnetStr1 += "$dotnet1|"

}

$dotnetStr2 = ""

foreach($dotnet2 in $installdotnet4)

{

$dotnetStr2 += "$dotnet2|"

}

$dotnet = "$dotnetStr1 $dotnetStr2 "

New-Object -Type PSCustomObject -Property @{

Info = "Installed Dotnet Versions"

Dotnet\_vesrions = $dotnet

}

}

else

{

}

}

## get-dsk

The below powershell script block fetches the disk drive information like DiskIS, Partition, Drive letter, size, freespace, Percent free space details.

function get-dsk ($servername)

{

get-WmiObject Win32\_DiskDrive -ComputerName $servername | % {

$disk = $\_

$partitions = "ASSOCIATORS OF " +

"{Win32\_DiskDrive.DeviceID='$($disk.DeviceID)'} " +

"WHERE AssocClass = Win32\_DiskDriveToDiskPartition"

Get-WmiObject -ComputerName $servername -Query $partitions | % {

$partition = $\_

$drives = "ASSOCIATORS OF " +

"{Win32\_DiskPartition.DeviceID='$($partition.DeviceID)'} " +

"WHERE AssocClass = Win32\_LogicalDiskToPartition"

Get-WmiObject -ComputerName $servername -Query $drives | % {

$part1 = $partition.Name

$part2 = $part1 -split ', ' `

-split ','

$diskid = ($part2[0]) -replace '#', ''

$part = ($part2[1]) -replace '#', ''

New-Object -Type PSCustomObject -Property @{

Disk = $disk.DeviceID

TotalSize = $disk.Size

DiskModel = $disk.Model

DiskID = $diskid

Partition = $part

RawSize = $partition.Size

DriveLetter = $\_.DeviceID

VolumeName = $\_.VolumeName

Size = $\_.Size

FreeSpace = $\_.FreeSpace

}

}

}

}

}

## get-localgroupmembers

The below script block fetches the user group membership information.

Function Get-LocalGroupMembers

{

param (

[Parameter(ValuefromPipeline = $true)]

[array]$server = $env:computername,

$GroupName = $null

)

PROCESS

{

$finalresult = @()

$computer = [ADSI]"WinNT://$server"

if (!($groupName))

{

$Groups = $computer.psbase.Children | Where { $\_.psbase.schemaClassName -eq "group" } | select -expand name

}

else

{

$groups = $groupName

}

$CurrentDomain = [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain().GetDirectoryEntry() | select name, objectsid

$domain = $currentdomain.name

$SID = $CurrentDomain.objectsid

$DomainSID = (New-Object System.Security.Principal.SecurityIdentifier($sid[0], 0)).value

foreach ($group in $groups)

{

$gmembers = $null

$LocalGroup = [ADSI]("WinNT://$server/$group,group")

$GMembers = $LocalGroup.psbase.invoke("Members")

$GMemberProps = @{ Server = "$server"; "Local Group" = $group; Name = ""; Type = ""; ADSPath = ""; Domain = ""; SID = "" }

$MemberResult = @()

if ($gmembers)

{

foreach ($gmember in $gmembers)

{

$membertable = new-object psobject -Property $GMemberProps

$name = $gmember.GetType().InvokeMember("Name", 'GetProperty', $null, $gmember, $null)

$sid = $gmember.GetType().InvokeMember("objectsid", 'GetProperty', $null, $gmember, $null)

$UserSid = New-Object System.Security.Principal.SecurityIdentifier($sid, 0)

$class = $gmember.GetType().InvokeMember("Class", 'GetProperty', $null, $gmember, $null)

$ads = $gmember.GetType().InvokeMember("adspath", 'GetProperty', $null, $gmember, $null)

$MemberTable.name = "$name"

$MemberTable.type = "$class"

$MemberTable.adspath = "$ads"

$membertable.sid = $usersid.value

if ($userSID -like "$domainsid\*")

{

$MemberTable.domain = "$domain"

}

$MemberResult += $MemberTable

}

}

$finalresult += $MemberResult

}

$finalresult | select server, "local group", name, type, domain

}

}

$luserinfo = Get-LocalGroupMembers

$luserinfo | Export-Csv C:\Serverinfo\details\UserGroup\_pre.csv -NoTypeInformation

$user\_group\_html=$luserinfo |ConvertTo-Html -Fragment -As Table -PreContent "<h2>USER GROUP DETAILS</h2>" | Out-String

# Windows Management instrumentation classes(WMI)

## Win32\_ComputerSystem

The **Win32\_ComputerSystem** [WMI class](https://docs.microsoft.com/en-us/windows/desktop/WmiSdk/retrieving-a-class) represents a computer system running Windows.

## Win32\_Quickfixengineering

The **Win32\_QuickFixEngineering** [WMI class](https://docs.microsoft.com/en-us/windows/win32/wmisdk/retrieving-a-class) represents a small system-wide update, commonly referred to as a quick-fix engineering (QFE) update, applied to the current operating system. This class returns only the updates supplied by Component Based Servicing (CBS). These updates are not listed in the registry. Updates supplied by Microsoft Windows Installer (MSI) or the Windows update site ([https://update.microsoft.com](https://update.microsoft.com/)) are not returned by **Win32\_QuickFixEngineering**.

## Win32\_Product

The **Win32\_Product** [WMI class](https://msdn.microsoft.com/en-us/library/aa393244(v=vs.85)) represents products as they are installed by Windows Installer. A product generally correlates to one installation package.

## Win32\_service

The **Win32\_Service** [WMI class](https://docs.microsoft.com/en-us/windows/win32/wmisdk/retrieving-a-class) represents a service on a computer system running Windows.

## Win32\_Operatingsystem

The **Win32\_OperatingSystem** [WMI class](https://docs.microsoft.com/en-us/windows/win32/wmisdk/retrieving-a-class) represents a Windows-based operating system installed on a computer.

# Powershell execution

## requirement

In order to execute the powershell script you much have local admin rights on local computer.

Always use the run as administrator while opening the powershell prompt

## powershell execution on single computer

## powershell execution for multiple computers