**SharePoint Farm Information**

|  |  |  |
| --- | --- | --- |
| SharePoint Farm Information | 1 | Check SharePoint Farm Status and details |
|  | 2 | List Alternate Sites and their Access Mapping URL's |
|  | 3 | Security Token Service web service status |
|  | 4 | Check if SharePoint farm is customized |
|  | 5 | Check status of SharePoint services |
|  | 6 | Check if Document Library/List exceeds Threshold limits (5000 items in one view) |
|  | 7 | Check the size of a document library/List |
|  | 8 | Check IIS Application Pools and Websites - Started/Stopped |
|  | 9 | Check if IIS/SQL service (SharePoint instance) is stopped |
|  | 10 | SharePoint Permissions |
|  | 11 | SharePoint Quotas - List |
|  | 12 | SharePoint DSN |
|  | 13 | Check Size Of SharePoint Databases and Log Files |
|  | 14 | SharePoint Databases Read-only |

This script will only pull and display information for the SharePoint farm which is useful for Projects where we migrate from SharePoint on premise to another on premise version or cloud.

1. Check SharePoint Farm Status and details – In this module we will fetch information about the OS, Service pack level, OS Version and build, Total memory, HDD Free space, SharePoint farm build, upgrade and database status. E.g. $farmBuild = (Get-SPFarm).BuildVersion and $(Get-SPFarm).NeedsUpgrade. ***Note: This requires the Microsoft.SharePoint.PowerShell snapin***.
2. List Sites and their Alternate Access Mapping url's – Please refer the below commands

$farm.AlternateUrlCollections | ForEach-Object {

Write-Output "IncomingUrl: $($\_.IncomingUrl)"

Write-Output "Zone : $($\_.Zone)"

Write-Output "Public URL : $($\_.publicurl)`n"

}

1. Security Token Service web service status

$sts = Get-SPServiceApplication | ?{$\_ -match "Security"}

$sts.Status

1. Check if SharePoint farm is customized e.g.

$solutions = $farm.Solutions

foreach ($solution in $solutions) {

$obj | Add-Member -Type NoteProperty -Name "$($solution.DisplayName)" -Value "$($solution.Id)"

}

1. Check status of SharePoint services

$spServices = get-spserviceinstance -server $spServer

$obj = New-Object PSObject

Write-Host "`n=============================="

Write-Host "Services on $($spServer.Name)"

Write-Host "=============================="

foreach ($spservice in $spServices) {

$service = get-spserviceinstance | where-object { $\_.TypeName -eq $spservice.TypeName }

$obj | Add-Member -Type NoteProperty -Name "$($spservice.TypeName)" -Value "$($service.Status)"

}

Write-Output $obj

1. Check if Document Library/List exceeds Threshold limits (5000 items in one view) – This will loop through all web applications and sites inside them and report if any list exceeds the set threshold limit in Central Administration. Function below:

Function Get-ListThreshold([string]$WebAppURL) {

$SPWebApp = Get-SPWebApplication $WebAppURL

$Threshold = $SPWebApp.MaxItemsPerThrottledOperation

$Warning = $SPWebApp.MaxItemsPerThrottledOperation \* (50 / 100)

$Critical = $SPWebApp.MaxItemsPerThrottledOperation \* (75 / 100)

foreach ($SPsite in $SPWebApp.Sites) {

foreach ($SPweb in $SPsite.AllWebs) {

foreach ($SPlist in $SPweb.Lists) {

if ($SPlist.ItemCount -gt $Threshold -or $SPlist.ItemCount -gt $Warning -or $SPlist.ItemCount -gt $Critical) {

$obj = New-Object PSObject

$obj | Add-Member -Type NoteProperty -Name "Title" -Value "$($SPlist.Title)"

$obj | Add-Member -Type NoteProperty -Name "URL"-Value "$($SPweb.URL)"

$obj | Add-Member -Type NoteProperty -Name "Count" -Value "$($SPlist.ItemCount)"

$obj | Add-Member -Type NoteProperty -Name "Threshold" -Value "$($Threshold)"

if ($SPlist.ItemCount -gt $Threshold) {

$obj | Add-Member -Type NoteProperty -Name "Level" -Value "Exceeded Limit"

}

elseif ($SPlist.ItemCount -gt $Critical) {

$obj | Add-Member -Type NoteProperty -Name "Level" -Value "Critical (Above 75%)"

}

elseif ($SPlist.ItemCount -gt $Warning) {

$obj | Add-Member -Type NoteProperty -Name "Level" -Value "Warning (Above 50%)"

}

Write-Host "`n================================"

Write-Host "Large List"

Write-Host "================================`n"

Write-Output $obj

}

}

$SPweb.Dispose()

}

$SPsite.Dispose()

}

}

1. Check the size of a document library/List – Similar to the one above, loop through all web applications and all sites within each web application and report document libraries greater than 500 MB and files greater than 50 MB. We need two text boxes in JSON with **Document library greater** than and **File greater** than for capturing input

Function Get-LargeDocuments([string] $WebAppURL) {

$WebApp = Get-SPWebApplication $WebAppURL

foreach ($SPSite in $WebApp.Sites) {

foreach ($SPWeb in $SPSite.AllWebs) {

foreach ($SPList in $SPWeb.Lists) {

# Get Document Libraries

if ($SPList.BaseType -eq "DocumentLibrary") {

foreach ($item in $SPList.Items) {

$data = @{

"Site" = $SPSite.Url

"Web" = $SPWeb.Url

"list" = $SPList.Title

"Item URL" = $item.Url

"Item Name" = $item.Name

"Item Created" = $item["Created"]

"Item Modified" = $item["Modified"]

"Size (MB)" = $item.File.Length / 1MB

}

# add files larger than 50 MB to the Output

if ($item.File.Length -gt 50MB) {

Write-Host "`n================================"

Write-Host "Large Document"

Write-Host "================================`n"

Write-Host($SPSite.Url + "/" + $item.Url)

Write-Output $data

}

}

}

$SPWeb.Dispose()

}

$SPSite.Dispose()

}

}

}

1. Check IIS Application Pools and Websites - Started/Stopped – This will report the status of IIS Application pools and sites. This will require the webadministration module

Application pools - Get-ChildItem IIS:\AppPools | Format-List

Web Sites - Get-ChildItem IIS:\Sites | Select-Object Name,ID,State,PhysicalPath | Format-List

1. Check if IIS/SQL service (SharePoint instance) is stopped – For this we will report the status of SQL service for SharePoint instance and IIS services using WMI

Write-Host "IIS Admin Service ( IISADMIN ) - "$(Get-Service -Name "IISADMIN" | Select -ExpandProperty Status)

Write-Host "Windows Process Activation Service ( WAS ) - "$(Get-Service -Name "WAS" | Select -ExpandProperty Status)

Write-Host "World Wide Web Publishing Service ( W3SVC ) - "$(Get-Service -Name "W3SVC" | Select - ExpandProperty Status)

The SQL instance for SharePoint can be found using the below command.

$db = (Get-SPDatabase)[0]

$db.Server

1. SharePoint Permissions – We need to get a permissions report for SharePoint Web Application which will be passed as a parameter. Script will be similar to

<https://gallery.technet.microsoft.com/office/SharePoint-Permissions-f42ea9db>

1. SharePoint Quotas – List

$t = [Microsoft.SharePoint.Administration.SPWebService]::ContentService.quotatemplates

$tFound = $false

$webApp = Get-SPWebApplication http://pocs.onertn.ray.com | %{$\_.Sites} | Get-SPSite

$webApp | fl Url, @{n="Storage Used/1MB";e={[int]($\_.Usage.Storage/1MB)}},

@{n="Storage Available Warning/1MB"; e={[int](($\_.Quota).StorageWarningLevel/1MB)}},

@{n="Storage Available Maximum/1MB"; e={[int](($\_.Quota).StorageMaximumLevel/1MB)}},

@{n="Sandboxed Resource Points Warning";e={[int](($\_.Quota).UserCodeWarningLevel)}},

@{n="Sandboxed Resource Points Maximum";e={[int](($\_.Quota).UserCodeMaximumLevel)}},

@{n="Quota Name"; e={ foreach($qt in $t){if($qt.QuotaId -eq [int](($\_.Quota).QuotaID)){$qt.Name; $tFound = $true}} if($tFound -eq $false){"No Template Applied"}$tFound=$false;}} >> C:\scripts\SiteCollectionList.txt

if($parent) {$webApp.Dispose(); $t.Dispose()}

1. SharePoint Dsn

$db = (Get-SPDatabase)[0]

$db.DatabaseConnectionString

1. Check Size Of SharePoint Databases and Log Files

$Sites = Get-SPSite $WebApp.Url -Limit All

foreach ($Site in $Sites) {

$SizeInMB = [math]::Round($Site.Usage.Storage / 1MB, 2)

$obj = New-Object PSObject

$obj | Add-Member -Type NoteProperty -Name "Title" -Value "$($Site.RootWeb.Title)"

$obj | Add-Member -Type NoteProperty -Name "Site URL" -Value "$($Site.URL)"

$obj | Add-Member -Type NoteProperty -Name "ContentDB" -Value "$($Site.ContentDatabase.Name)"

$obj | Add-Member -Type NoteProperty -Name "Size" -Value "$SizeInMB MB"

$Result += $obj

}

1. SharePoint Databases Read-only – This will report any content databases set in read only mode.  
   Get-SpContentDatabase | % {$\_.IsReadOnly}

**Pre-requisites:**

1. **We need to detect installed SharePoint version from registry – 12 for MOSS, 14 for 2010, 15 for 2013 and 16 for 2016 & 2019. If we get a value SharePoint which is equal to “Installed” then SharePoint is installed.**

**$array = @("12.0", "14.0", "15.0", "16.0")**

**$regPath = "HKLM:\Software\Microsoft\Shared Tools\Web Server Extensions\"**

**$basePath = Get-ChildItem $regPath**

**$basePath | ForEach-Object {if($\_.GetValue("SharePoint") -eq "Installed") {return $true}}**

1. **If SharePoint is installed then we need to add its snapin using: Add-PSSnapin Microsoft.SharePoint.PowerShell**
2. **We will skip MOSS as by default we cannot load Snapin on SharePoint 2007 – 2010 to 2019 only**