GOPAS

Functions, scripts, modules

Ing. Ondřej Ševeček | GOPAS a.s. |
MCSM:Directory2012 | MCM:Directory2008 | MVP:Enterprise Security | CEH | CHFI | CISA |
ondrej@sevecek.com | www.sevecek.com |

GOPAS: info@gopas.cz | www.gopas.cz | www.facebook.com/P.S.GOPAS

1

Execution policy

```
Get-ExecutionPolicy -List
```

- machine policy, user policy
  - GPO enforcement cannot be overwritten by the CMD/BAT usage of -ExecutionPolicy parameter of powershell.exe or pwsh.exe

GOPAS

2

#1 start powershell as Administrator and disable ExecutionPolicy

- start Windows PowerShell as Administrator

```
Set-ExecutionPolicy bypass
```

- restart PowerShell_ISE
- since now you can create individual .PS1 files for each task and run the scripts repeatedly by simply pressing F5

GOPAS

3

Functions simplest

```
function Calculate-Nonsense ( $oneNumber, $secondNumber )
{
  $computation = ($oneNumber * $secondNumber - 5) % 7
  return $computation
}

Calculate-Nonsense -oneNumber 97 -secondNumber 381

# you can call the function without parameter names, but I would
not strongly recommend against such a practice
Calculate-Nonsense 97 13
```

GOPAS

4

Functions and default parameter values

```
function Calculate-Nonsense ( $oneNumber, $secondNumber = 381 )
{
  $computation = ($oneNumber * $secondNumber - 5) % 7
  return $computation
}

Calculate-Nonsense -oneNumber 97
```

GOPAS

5

Functions and cast attributes

```
# defining the so-called switch parameter
# supplied parameter values get converted to the declared types
function Calculate-Nonsense ( [double] $oneNumber, [double]
$secondNumber, [switch] power2 )
{
  $computation = ($oneNumber * $secondNumber - 5) % 7

  if ($power2) {

    $computation = $computation * $computation
  }

  return $computation
}

# using the so-called switch parameter
Calculate-Nonsense -oneNumber 97 -secondNum 381 -power2
```

GOPAS

6

#2 create a function that downloads weather JSON data for a city and test it with Prague and London

- create a new function Check-Weather for the task
- use Invoke-WebRequest to download JSON data that identify the city (such as Prague or London)
  - https://www.metaweather.com/api/location/search/?query=prague
- use ConvertFrom-Json to obtain the woeid value for the particular city
- use New-Object to create an output [PSObject] object

```
New-Object -TypeName PSObject -Property @{ .... }
```

- use Invoke-WebRequest again to download JSON weather data from the URL specific for the city defined by its woeid
  - https://www.metaweather.com/api/location/<woeid>
  - such as for Prague particularly
  - https://www.metaweather.com/api/location/796597

GOPAS

7

#2 answer create a function that downloads weather JSON data for a city and test it with Prague and London

```
function Check-Weather ( $city )
{
  Write-Host ('Obtain city information: {0}' -f $city)

  $cityInfoURI = ' https://www.metaweather.com/api/location/search/?query={0}' -f $city
  Write-Host ('HTTP query: {0}' -f $cityInfoURI)

  $cityInfoJSON = Invoke-WebRequest -Uri $cityInfoURI
  Write-Host ('Query status: http = {0} | json = {1} chars' -f $cityInfoJSON.StatusCode, $cityInfoJSON.Content.Length)

  $cityInfo = ConvertFrom-Json -Input $cityInfoJSON.Content
  Write-Host ('WOEID for the city: {0} | {1}' -f $cityInfo.title, $cityInfo.woeid)

  $weatherURI = 'https://www.metaweather.com/api/location/{0}' -f $cityInfo.woeid
  Write-Host ('HTTP query: {0}' -f $weatherURI)

  $weatherJSON = Invoke-WebRequest -Uri $weatherURI
  Write-Host ('Query status: http = {0} | json = {1} chars' -f $weatherJSON.StatusCode, $weatherJSON.Content.Length)

  $weather = ConvertFrom-Json -Input $weatherJSON

  $sunrise = Get-Date $weather.sun_rise
  $sunset = Get-Date $weather.sun_set

  $todayData = $weather.consolidated_weather | ? applicable_date -eq (Get-Date).ToString('yyyy-MM-dd')
  $todayOverall = $todayData.weather_state_name

  $tomorrowData = $weather.consolidated_weather | ? applicable_date -eq (Get-Date).AddDays(1).ToString('yyyy-MM-dd')
  $tomorrowOverall = $tomorrowData.weather_state_name

  Write-Host ('Sunrise: {0}' -f $sunrise)
  Write-Host ('Sunset: {0}' -f $sunset)
  Write-Host ('Today: {0}' -f $todayOverall)
  Write-Host ('Tomorrow: {0}' -f $tomorrowOverall)

  $outWeatherHash = @{
                      Sunrise = $sunrise;
                      Sunset = $sunset;
                      Today = $todayOverall;
                      Tomorrow = $tomorrowOverall
                    }

  $outWeather = New-Object -TypeName PSObject -Property $outWeatherHash

  return $outWeather
}
```

GOPAS

8

#3 create a function that would create a home folder for a particular user

- input parameters should be
  - $folderRoot
  - $account
- use Test-Path and if to verify folder existence such as in the following folder C:\TEMP\Training\Homes\<account>
- use New-Item to create the home folder if it does not exist
- use ICACLS to grant exclusive access to the user
  - `icacls /C /L /inheritance:r /grant "Administrators:(OI)(CI)F"`
  - `icacls /C /L /setowner "Administrators"`
  - `icacls /C /L /grant "kamil:(OI)(CI)RX"`
  - `icacls /C /L /grant "kamil:(WD,AD)"`
  - `icacls /C /L /grant "kamil:(IO)(OI)(CI)F"`

GOPAS

9

#3 answer create a function that would create a home folder for a particular user ...

```
# the "if" keyword works like anywhere else :-)
function Create-HomeFolder ($folderRoot, $account)
{
  $homeFolder = Join-Path $folderRoot $account

  if (-not (Test-Path $homeFolder)) {

    Write-Host ('Creating the folder: {0}' -f $homeFolder)
    New-Item $homeFolder -ItemType Directory -Force | Out-Null
  }

  icacls $folderRoot /C /L /inheritance:r /grant "Administrators:(OI)(CI)F"
  icacls $folderRoot /C /L /setowner "Administrators"
  icacls $folderRoot /C /L /grant "$($account):(OI)(CI)RX"
  icacls $folderRoot /C /L /grant "$($account):(WD,AD)"
  icacls $folderRoot /C /L /grant "$($account):(IO)(OI)(CI)F"
}
```

GOPAS

10

#3 answer create a function that would create a home folder for a particular user ...

```
# we need to cope with the ICACLS output which get piped out of the function
function Create-HomeFolder ($folderRoot, $account)
{
  $homeFolder = Join-Path $folderRoot $account

  if (-not (Test-Path $homeFolder)) {

    Write-Host ('Creating the folder: {0}' -f $homeFolder)
    New-Item $homeFolder -ItemType Directory -Force | Out-Null
  }

  $cmdOut = icacls $folderRoot /C /L /inheritance:r /grant
"Administrators:(OI)(CI)F"
  $cmdOut = icacls $folderRoot /C /L /setowner "Administrators"
  $cmdOut = icacls $folderRoot /C /L /grant "$($account):(OI)(CI)RX"
  $cmdOut = icacls $folderRoot /C /L /grant "$($account):(WD,AD)"
  $cmdOut = icacls $folderRoot /C /L /grant "$($account):(IO)(OI)(CI)F"
}
```

GOPAS

11

#3 answer create a function that would create a home folder for a particular user

```
# we should also improve stability on various OS localizations
function Create-HomeFolder ($folderRoot, $account)
{
  $homeFolder = Join-Path $folderRoot $account

  if (-not (Test-Path $homeFolder)) {

    Write-Host ('Creating the folder: {0}' -f $homeFolder)
    New-Item $homeFolder -ItemType Directory -Force | Out-Null
  }

  $administrators = (New-Object Security.Principal.SecurityIdentifier 'S-1-5-32-
544').Translate('Security.Principal.NTAccount').Value

  $cmdOut = icacls $folderRoot /C /L /inheritance:r /grant
"$($administrators):(OI)(CI)F"
  $cmdOut = icacls $folderRoot /C /L /setowner $administrators
  $cmdOut = icacls $folderRoot /C /L /grant "$($account):(OI)(CI)RX"
  $cmdOut = icacls $folderRoot /C /L /grant "$($account):(WD,AD)"
  $cmdOut = icacls $folderRoot /C /L /grant "$($account):(IO)(OI)(CI)F"
}
```

GOPAS

12

#4 call your function to create home folders for all AD users that have email address

- use Get-ADUser to obtain all users that have email address
- use Foreach-Object to call your function

GOPAS

13

#4 answer call your function to create home folders for all AD users that have email address

```
# the -Filter parameter accepts just a string which might be similar to the
PowerShell operator syntax, but is parsed internally by the Active Directory
Web Service on a DC in a slightly different way
# for example the following would not work
#    -Filter 'Mail'
#    -Filter 'Mail -ne ""'
# also the following would find ALL accounts in PowerShell because the "*"
wildcard matches empty strings as well in PowerShell
Get-ADUser -Filter 'Mail -like "*"' -Prop SamAccountName |
  % {
      Create-HomeFolder -folderRoot C:\temp\training\Homes -account
$_.SamAccountName
    }
```

GOPAS

14

Functions that accept pipe input

```
function Create-HomeFolder ($folderRoot, [Parameter(ValueFromPipeline)] $account)
{
  $homeFolder = Join-Path $folderRoot $account

  if (-not (Test-Path $homeFolder)) {

    Write-Host ('Creating the folder: {0}' -f $homeFolder)
    New-Item $homeFolder -ItemType Directory -Force | Out-Null

  } else {

    Write-Host ('Folder already exists: {0}' -f $homeFolder)
  }

  $administrators = (New-Object Security.Principal.SecurityIdentifier 'S-1-5-32-
544').Translate('Security.Principal.NTAccount').Value

  $cmdOut = icacls $homeFolder /C /L /inheritance:r /grant "$($administrators):(OI)(CI)F"
  $cmdOut = icacls $homeFolder /C /L /setowner $administrators
  $cmdOut = icacls $homeFolder /C /L /grant "$($account):(OI)(CI)RX"
  $cmdOut = icacls $homeFolder /C /L /grant "$($account):(WD,AD)"
  $cmdOut = icacls $homeFolder /C /L /grant "$($account):(IO)(OI)(CI)F"
}


Get-ADUser -Filter 'Mail -like "*"' -Prop SamAccountName | select -Expand SamAccountName |
Create-HomeFolder -folderRoot c:\temp\training\homes
```

GOPAS

15

Functions with mandatory parameters

```
function Create-HomeFolder (
  [Parameter(Mandatory = $true)] $folderRoot,
  [Parameter(Mandatory = $true, ValueFromPipeline)] $account
  )
{
...
```

GOPAS

16

#5 create PS module file for both functions and use it from a script

- put all the internal and public functions into a common training.psm1 file
- use the Export-ModuleMember to export the functions from the module file

```
# last command in the module usually
Export-ModuleMember -Function Create-HomeFolder, Check-Weather
```

- import the module with Import-Module

```
Import-Module c:\temp\training\training.psm1 -
DisableNameChecking
```

GOPAS

17

Script local source folder (only when running .PS1 or .PSM1 actually)

```
# the following is not populated correctly when not actually
running from within a .PS1 or .PSM1 file
$scriptFile = $MyInvocation.MyCommand.Definition
$scriptFolder = Split-Path -Parent $scriptFile

Import-Module (Join-Path $scriptFolder training.psm1)
```

GOPAS

18

Simple XML files (edit within ISE)

```
<mailingRoot>

  <!-- note the &quot; and &lt; and &gt; escape sequences -->
  <basic subject="HR business intelligence"
         from="&quot;Human resources department&quot; &lt;humanresources@gopas.cz&gt;"
         />

  <smtp>data.gopas.virtual</smtp>

  <attach>

    <one>T:\Samples\Doc\*.pdf</one>
    <one>T:\Samples\Photo\s*</one>

  </attach>

  <userFilter><![CDATA[

  ($_.SamAccountName -like 'k*') -or ($_.SamAccountName -like 'm*') -or ($_.Country -eq 'CZ')

  ]]></userFilter>

</mailingRoot>
```

GOPAS

19

Reading simple XML files

```
$xmlConfig = [XML] (Get-Content c:\temp\training\smtp-
mailing\mailingparams.xml)

# attribute
$xmlConfig.mailingRoot.basic.subject
# attribute
$xmlConfig.mailingRoot.basic.from
# element with plain text
$xmlConfig.mailingRoot.smtp

# collection of child elements
$xmlConfig.mailingRoot.attach.one | % { Write-Host $_ }

# element with CDATA text
$xmlConfig.mailingRoot.userFilter.InnerText
```

GOPAS

20

Warning! Pipe output inside function ...

```
# NOT a good implementation of the function but try it first this simply :-)
# the goal of the function is to return process ID of the SNMP service
function Restart-SnmpTrap ()
{
  Write-Host ('Stop the service first')
  gwmi Win32_Service | ? Name -eq 'SNMPTRAP' | % { $_.StopService() }

  Write-Host ('Start the service back again')
  gwmi Win32_Service | ? Name -eq 'SNMPTRAP' | % { $_.StartService() }

  Write-Host ('Service has been started. Determine its ProcessId')
  $processId = gwmi Win32_Service | ? Name -eq 'SNMPTRAP' | select -Expand
ProcessId

  Write-Host ('Process ID of the service: {0}' -f $processId)

  return $processId
}

# may look legitimate yet
Restart-SnmpTrap
```

GOPAS

21

... Warning! Pipe output inside function

```
# now the problem should be more apparent

$snmpProcId = Restart-SnmpTrap

$snmpProcId

Restart-SnmpTrap | measure
Restart-SnmpTrap | gm


Write-Host # writes something on console only (+coloring)
echo <=> Write-Output # sends data to output pipe, do not use
unless you know what you are doing
return # sends data to output pipe and exits the function
[void], Out-Null # always void/null the output of anything that
you do not need
```

GOPAS

22

Warning! Example commands that return values even if you usually do not handle them

```
[Collections.ArrayList] $names = @('Ondrej', 'Kamil')
# returns the index at which it inserted the new object
$names.Add('Jitka')


# returns the newly created item [DirectoryInfo], [FileInfo]
or [RegistryKey]
New-Item HKLM:\Software\GOPAS


# return their textual output as [string[]]
ipconfig /flushdns
certutil -pulse
gpupdate
```

GOPAS

23

#6 complex solution

- take Query-Sql, SMTP-Mailing and Update-XlsSheet contents and functions
- create a solution that
  - queries the SQL table HumanResources.vEmployee from data.gopas.virtual\INFOSYS AdventureWorks database
  - writes the found items into an .XLSX file
  - uploads the created .XLSX file together with some pictures from T:\Samples\Photo as attachments to SMTP email server data.gopas.virtual using the HTML template for the email content
    - implement it using Send-MailMessage cmdlet
    - find all email parameters inside the supplied .XML file

GOPAS

24