Powershell

Contient les ressources Powershell pour windows.

Powershell - codes popup

Icons

16 – “stop mark”

32 – “question mark”

48 – “Exclamation mark”

64 – “Information mark”

Buttonset

0 OK

1 OK, Cancel

2 stop, retry skip

3 yes, no, cancel

4 yes, no

5 retry, cancel

Return values

- -1 — timeout;
- 1 — OK button;
- 3 — Stop button;
- 4 — Retry button;
- 5 — Skip button;
- 6 — Yes button;
- 7 — No button.

Intégration Code

```
#creating object os WScript
$wshell = New-Object -ComObject Wscript.Shell -ErrorAction Stop
```

```
#invoking the POP method using object
$wshell.Popup("Are you want to continue from here?",<time in second>,"Hello
User?",<icon>+<buttonsSet>)
```

Powershell - Conversion image/texte

```
#Convertir un fichier image en fichier texte.

$FilePath = "C:\jumbor.jpg"
$SaveTo = "C:\jumbor.txt"
[byte[]]$Data = Get-Content $FilePath -Encoding Byte
[system.convert]::ToBase64String($Data) | Set-Content $SaveTo

#Fin.
```

Powershell - Créer / Modifier des variables d'environement

Pour créer une variable d’environnement temporaire :

```
$env:<variableName> = "<value>"
```

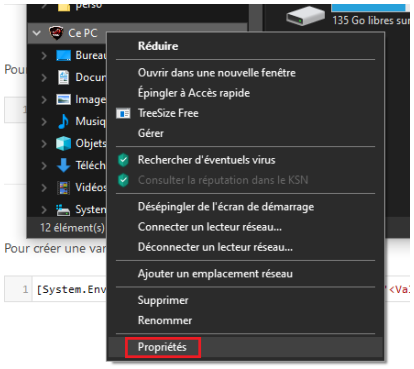
Pour créer une variable d’environnement utilisateur :

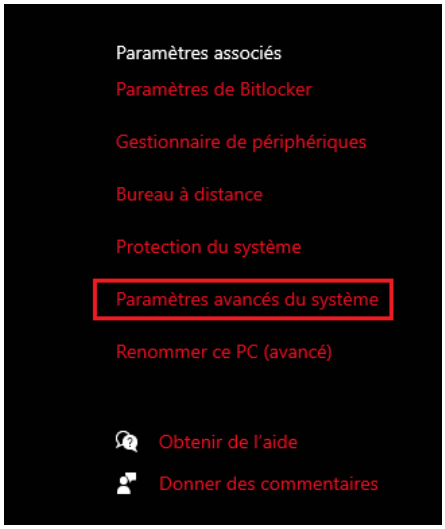
```
[System.Environment]::SetEnvironmentVariable('<Name>','<Value>',[System.EnvironmentVariableTarget]::User)
```

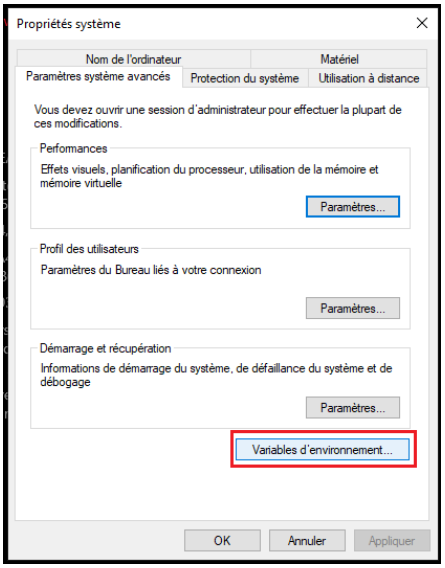
Pour créer une variable d'environnement système :

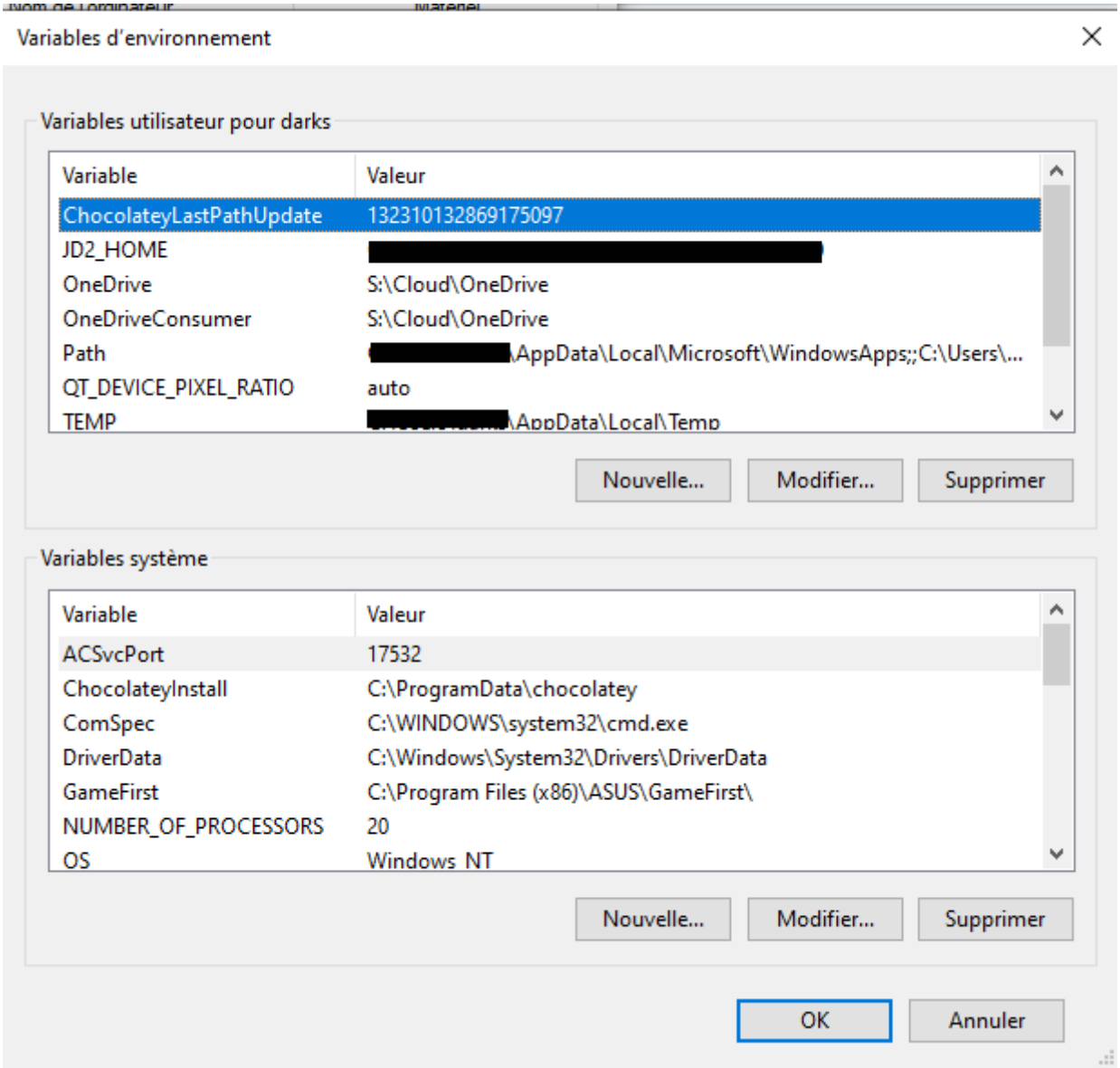
```
[System.Environment]::SetEnvironmentVariable('<Name>','<Value>',[System.EnvironmentVariableTarget]::Machine)
```

> Pour vérifier les variables d'environnement existantes : "***Ordinateur, Clic droit --> propriétés --> Avancées***"


Ce PC
Réduire
Ouvrir dans une nouvelle fenêtre
Épingler à Accès rapide
TreeSize Free
Gérer
Rechercher d'éventuels virus
Consulter la réputation dans le KSN
Désépingler de l'écran de démarrage
Connecter un lecteur réseau...
Déconnecter un lecteur réseau...
Ajouter un emplacement réseau
Supprimer
Renommer
Propriétés


Paramètres associés
Paramètres de Bitlocker
Gestionnaire de périphériques
Bureau à distance
Protection du système
Paramètres avancés du système
Renommer ce PC (avancé)
Obtenir de l'aide
Donner des commentaires


Propriétés système
Nom de l'ordinateur
Matériel
Paramètres système avancés
Protection du système
Utilisation à distance
Vous devez ouvrir une session d'administrateur pour effectuer la plupart de ces modifications.
Performances
Effets visuels, planification du processeur, utilisation de la mémoire et mémoire virtuelle
Paramètres...
Profil des utilisateurs
Paramètres du Bureau liés à votre connexion
Paramètres...
Démarrage et récupération
Informations de démarrage du système, de défaillance du système et de débogage
Paramètres...
Variables d'environnement...
OK
Annuler
Appliquer


Variables d'environnement
Variables utilisateur pour darks
Variable
Valeur
ChocolateyLastPathUpdate
132310132869175097
JD2_HOME
[redacted]
OneDrive
S:\Cloud\OneDrive
OneDriveConsumer
S:\Cloud\OneDrive
Path
[redacted]\AppData\Local\Microsoft\WindowsApps;;C:\Users\...
QT_DEVICE_PIXEL_RATIO
auto
TEMP
[redacted]\AppData\Local\Temp
Nouvelle...
Modifier...
Supprimer
Variables système
Variable
Valeur
ACSvcPort
17532
ChocolateyInstall
C:\ProgramData\chocolatey
ComSpec
C:\WINDOWS\system32\cmd.exe
DriverData
C:\Windows\System32\Drivers\DriverData
GameFirst
C:\Program Files (x86)\ASUS\GameFirst\
NUMBER_OF_PROCESSORS
20
OS
Windows_NT
Nouvelle...
Modifier...
Supprimer
OK
Annuler

Powershell - Fonctions utiles

Get-AdminRights

```
function Get-AdminRights(){#Check if script is launched with admin rights, else relaunch
it with good rights.
    <#Function of VOID type. No parameters necessary.
        you can comment or uncomment the lines marcked with "#!" by removing "#!" to adapt
the behavior #>

    #Get the identity of the current account based on its admin group membreship
    $IsAdmin=[Security.Principal.WindowsIdentity]::GetCurrent().Groups -contains 'S-1-5-
32-544'

    #Is identity of current user member of local admin ?
    if ($IsAdmin -eq $false){#if no

        #Write event in windows log
        #!Write-EventLog -LogName Application -Source 'Automation Engine' -EntryType
Warning -EventId 201 -Message "Script not launched with admin rights."

        #Write event in console
        Write-Host "Script not launched with admin privileges..." -ForegroundColor Red
        # OR display an arror popup
         #!$result = New-Object -ComObject Wscript.Shell -ErrorAction Stop
         #!$result.Popup("Script Not launched with admin rights",0,"Warning",48+0)

        #Relaunch as admin
         #!Start-Process powershell.exe -Verb RunAs -ArgumentList ('-noprofile -noexit -
file "{0}" -elevated' -f ($myinvocation.MyCommand.Definition))
        # OR pause and exit
        pause
        exit
    }
}
```

Get-FileLocation

```
function Get-FileLocation(){#Browse on computer and return the path of file.
    <#Function of VOID type. No parameters necessary.
        use it in a STRING variable to get the selected path when clicking "OK"#>

    #Create browse window.
    $FileBrowser = New-Object System.Windows.Forms.OpenFileDialog -Property @{
        Multiselect = $false # Multiple files can be chosen
        Filter = 'Exports (*.csv)|*.csv' # Specified file types
    }

    #call the browse window
    [void]$FileBrowser.ShowDialog()

    # Is there file(s) selected by user ?
    If($FileBrowser.FileNames -like "*\*") {#If yes,

        #Lists selected files (optional)
        $FileBrowser.FileName

    }else{#If no,
        #Cancel the operation
        Write-Host "Cancelled by user"
    }
}
```

Get-FolderLocation

```
function Get-FolderLocation(){#Browse on computer and return the path of folder.
    <#Function of VOID type. No parameters necessary.
        use it in a STRING variable to get the selected path when clicking "OK"#>

    #Create the browse window
    [Reflection.Assembly]::LoadWithPartialName("System.Windows.Forms") | Out-Null
    [System.Windows.Forms.Application]::EnableVisualStyles()
```

```
    #Create the window elements
    $browse = New-Object System.Windows.Forms.FolderBrowserDialog
    $browse.RootFolder = [System.Environment+SpecialFolder]'MyComputer'
    $browse.ShowNewFolderButton = $false
    $browse.Description = "Choose a directory"



    #While nothing is clicked, display the window
    while($result -ne "Cancel" -and $browse.ShowDialog() -ne "OK")
    {
        $result = [System.Windows.Forms.MessageBox]::Show("You clicked Cancel. Try again
or exit script?", "Choose a directory",
[System.Windows.Forms.MessageBoxButtons]::RetryCancel)
    }

    if($result -ne "Cancel")
    {
        $browse.SelectedPath
        $browse.Dispose()
    }

}
```

New-LogSource

```
function New-LogSource([string]$LogSourceName){
    <# Function of VOID Type. Take a STRING in entry containing the Log Source Name
        This function will check if the source already exist and create it if not (require
administative rights)
    #>

    #Get the Log source events
    Get-EventLog -LogName Application -Source $LogSourceName -ErrorAction SilentlyContinue
| Format-Table

    #Is previous command succesfull ?
    if ($? -eq $false){#if no, Log source does'nt exist
```

```
        #Say log source currently not exist
        Write-Host "The logsource $LogSourceName does'nt exist. It will be created..." -
ForegroundColor Yellow

        #Create the log source
        [System.Diagnostics.EventLog]::CreateEventSource($LogSourceName, "Application")

        #Spawn a log entry saying that the log source is now created
        Write-Host "Log source $LogSourceName Created Successfully." -ForegroundColor
Green

        #Say in the logs that the log source is now created
        Write-EventLog -LogName Application -Source $LogSourceName -EntryType Information
-EventId 9999 -Message "Log source $LogSourceName Created Successfully."

    }else{#if yes, log source exist

        #Just say that the source already exist
        Write-Host "Log source $LogSourceName already exist. It will not be created
again." -ForegroundColor Yellow

    }
}
```

New-Password

```
function New-Password([int]$lenght){#Generate a password composed by A-Z, a-z, 0-9 &
special caracters.
    <# Function of regular type
        take INT, mandatory : lenght of the password to be generated
        function return STRING : password generated#>

    #Generate a random caracter untill the full password is generated
    do {
        $fctPassword = -join((33..47) +  (48..57) + (65..90) + (97..122) + (58..64) | Get-
Random -Count $lenght | Foreach-Object {[char]$_})
    }
    while (!(($fctPassword -cmatch '[a-z]') -and ($fctPassword -cmatch '[A-Z]') -and
```

```
($fctPassword -match '\d') -and ($fctPassword -match "[^\w]")))

    #Return the password
    return $fctPassword
}
```

Powershell - Journalisation windows

> Par convention, nous utiliserons "PSAutomationEngine" comme nom de source.

Créer une source d'évènements (requiert les droits admin) :

```
[System.Diagnostics.EventLog]::CreateEventSource("PSAutomationEngine", "Application")
```

La première valeur est le nom de la source.

La seconde valeur est le journal dans lequel enregistrer la source.

Les valeurs possibles sont :

- Application
- Security
- Setup
- System

Récupérer les journaux générés par la source :

```
Get-EventLog -LogName "Application" -Source "PSAutomationEngine"
```

La première valeur est le nom du journal.

La seconde valeur est le nom de la source.

Créer un évènement :

```
[System.Diagnostics.EventLog]::WriteEntry("PSAutomationEngine","This is a test entry","information",9)
```

La première valeur est le nom de la source.

La seconde valeur est le message de l'évènement.

La troisième valeur est le type d'évènement.

Les différentes valeurs possibles sont :

- information
- warning
- error

La quatrième valeur est le code d'évènement.

> Voir : Powershell - Matrice des codes d'états

```
  Index Time          EntryType   Source                InstanceID Message
  ----- ----          ---------   ------                ---------- -------
1145375 Jan 04 09:52  Information PSAutomationEngine            9 This is a test entry
```

Supprimer une source (requiert les droits admin) :

```
[System.Diagnostics.EventLog]::DeleteEventSource("PSAutomationEngine")
```

La première valeur est le nom de la source.

Powershell - Stocker un mot de passe

Pour récupérer le mot de passe et le stocker dans un fichier :

```
[string]$DestFile="$env:USERPROFILE\Documents\password.secret"
read-host "Enter Password" -AsSecureString | ConvertFrom-SecureString | Out-File $DestFile
```

Pour utiliser le mot de passe en clair :

```
[string]$EncryptedPasswordFile="$env:USERPROFILE\Documents\password.secret"
$PasswordSecureString=ConvertTo-Securestring (Get-content $EncryptedPasswordFile)

$PlainTextPassword =
[System.Runtime.InteropServices.Marshal]::PtrToStringBSTR([System.Runtime.InteropServices.Mars
hal]::SecureStringToBSTR($PasswordSecureString))
```

Pour utiliser le mot de passe dans un objet credential :

```
[string]$EncryptedPasswordFile="$env:USERPROFILE\Documents\password.secret"

[string]$Username="MyUserName"
$PasswordSecureString=ConvertTo-Securestring (Get-content $EncryptedPasswordFile)

$credentials=New-object System.Management.Automation.PSCredential ($Username,
$PasswordSecureString)
```

Powershell - Secure winRM

I. Paramétrage du service

Vérifier les listener :

```
winrm e winrm/config/listener
```

Vérifier l'état de winRM :

```
winrm get winrm/config
```

II. Paramétrage trusted host

Lister les TrustedHosts :

```
Get-Item WSMan:\localhost\Client\TrustedHosts | Select-Object Value | Format-Table
```

Ajouter le host dans les TrustedHosts :

```
Set-Item WSMan:\localhost\Client\TrustedHosts -Value '<ip des serveurs> ou *'
```

III. Configuration Firewall

Ajouter une règle autorisant winRM :

```
New-NetFirewallRule -DisplayName "Windows Remote Management (HTTP-In)" -Profile Domain -
Direction Inbound -Action Allow -RemoteAddress '<ipadress> or CIDR network' -Protocol TCP -
LocalPort 5986
```

Powershell - Template

```
<#############################################################################
# Title       :                                                             #
# Version     :                                                             #
# Author      :                                                             #
# For         :                                                             #
# Last change :                                                             #
# Description :                                                             #
# Require     :                                                             #
#                                                                           #
#    Distributed under CC-BY-NC Licence                                     #
#############################################################################>

<# changelog
V0.1 :
- ...
V0.2 :
- ...
#>

#region ---------- Header ----------
#Add-Type -AssemblyName system.windows.forms
#Add-Type -AssemblyName system.drawing
#Add-Type -AssemblyName System.XML
#Add-Type -AssemblyName PresentationFrameworkion
#endregion ------- Header ----------

#region --------- Arguments ---------
param(

    #keep this parameter as it can call the Help page for your script
    [Parameter(Mandatory=$false)][switch]$Help,

    #insert additionals parameters here
)
#endregion ------ Arguments ---------
```

```
#region ----------- Help ------------
if($help){

    Write-Host "
   "
    exit

}
#endregion -------- Help ------------

#region ----------- Title -----------
[String]$AppName="" # Report application Name here
[String]$Version="" # Report application Version Here
[string]$Title=$AppName+" V. "+$Version
$host.UI.RawUI.WindowTitle = $Title
#endregion -------- Title -----------

#region ---------- Classes ----------
#endregion ------- Classes ----------

#region --------- Variables ---------
#endregion ------ Variables ---------

#region --------- Functions ---------
#endregion ------ Functions ---------

#region ------------ GUI ------------
if (!$NoGUI){}
#endregion --------- GUI ------------

#region ----------- Script ----------
#endregion -------- Script ----------
```

Powershell - ressources microsoft

Verbes approuvés :

Approved Verbs for PowerShell Commands - PowerShell | Microsoft Learn