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]")))
```

```
    #Resturn the password
    return $fctPassword
}
```

Revision #2
Created 2026-07-17 14:57:04 UTC by Olivier
Updated 2026-07-17 15:00:09 UTC by Olivier