PowerShell

Toutes les ressources pour PowerShell

- Retirer les accents, cédilles, … d’une chaîne de caractères
- Exemples de commandes
- Script de création des comptes dans un active Directory et ouverture des sessions RDS.

Retirer les accents, cédilles, … d’une chaîne de caractères

Méthode plus simple, plus compact et plus efficace que de faire des tableaux de correspondances et des "replace".

```
$string = "Ici votre texte accentué de façon à faire un test"
[Text.Encoding]::ASCII.GetString([Text.Encoding]::GetEncoding("Cyrillic").GetBytes($string))
Write-Host "> " $string
> Ici votre texte accentue de facon a faire un test
```

Source

Exemples de commandes

`Get-help` -> Obtenir l'aide PowerShell, et faire la mise à jour

`get-process` -> Lister les processus en cours

`Get-Help Get-Process -parameter *` → Demander de l'aide pour voir les paramètres possibles de Get-Process

`Get-Help Get-Process -example` → Voir des exemples de la commande Get-Process

`clear` -> Nettoyer l'affichage

`Get-Process | Where-Object {$_.WorkingSet -gt 10000}` Récupérer les processus de plus de 10 Mo

`$nom="Cecile"` → Créer une variable

`$nom` → Afficher la variable

`$nom="Cecile","Coco"` -> Créer un tableau

`$nom.GetType()` -> Renvoyer le type de la variable (string, float..)

`Get-alias` → Lister les alias

`new-alias -Name "so" -Value "sort-object"` -> Créer un alias avec comme nom *so* et *sort-object* en commande

`Get-ChildItem env:` → Lister les variables d'environnement

`Get-PSProvider` → Lister les fournisseurs PowerShell

`Get-PSDrive` → Lister les disques de la session actuelle (Physique et Provider)

`add-windowsfeature RSAT-AD-Powershell` → Installer les modules de gestion d'Active Directory à distance

`Get-ADUser -filter 'name -like "damien*"' -properties *'` → Voir les propriétés des utilisateurs dont le nom commence par damien

`Get-ADObject -filter {(objectclass -eq "computer")} -properties SamAccountName` → Lister les ordinateurs et les trier par nom

`Get-ADObject -filter {(objectclass -eq "computer")} -properties SamAccountName | export-csv -Path "C:\.."` → Exporter le résultat dans un fichier .csv

`Get-ADObject -filter {(objectclass -eq "user")} -properties SamAccountName | export-csv -Path "C:\..\export.csv"` → Exporter tous les utilisateurs dans un .csv

`Get-ADUser -filter ''passwordneverexpires -eq "True"'` → Voir les utilisateurs qui ont un mot de passe qui n'expire jamais

Script de création des comptes dans un active Directory et ouverture des sessions RDS.

Le script ci-dessous doit être passé avec le chemin d'un fichier csv en argument.

Le contenu du fichier csv sera détaillé après le script.

```
# Force le type d'execution
Set-ExecutionPolicy Unrestricted

$ErrorActionPreference = "SilentlyContinue"
$host.ui.RawUI.WindowTitle = "Script de création d'utilisateurs dans l'Active Directory"
$host.ui.RawUI.ForegroundColor = 'Green'
$host.ui.RawUI.BackgroundColor = 'Black'

Set-PSDebug -Off

<#
.SYNOPSIS
    Script permettant l'inscription des nouveaux utilisateurs dans l'Active Directory.
.DESCRIPTION
    Script permettant l'inscription des nouveaux utilisateurs dans l'Active Directory.
.PARAMETER FichierCSV
    Chemin complet du fichier .CSV
.EXAMPLE
    ActiveDirectoryUsersCreator.ps1 "D:\Users\user\Scripts\Active Directory\fichier.csv"
.INPUTS

.OUTPUTS
```

```
.NOTES
    Le script suit le processus suivant :
        - Analyse du fichier CSV entré en paramètre du script
            * Le fichier CSV doit contenir les colonnes suivantes :
                + Nom
                + Prenom
                + NomComplet
                + Utilisateur
                + MotDePasse
                + Groupe (dans lequel sera ajouté l'utilisateur)
                + UniteOrganisation (dans laquelle sera ajouté l'utilisateur)
                + UniteOrganisationRacine (dans laquelle sera ajouté l'Unité Organisationnelle
précédente)
                + Domaine (Domaine dans lequel créer l'utilisateur)
                + TLD (TLD du domaine)
                + Serveur (Serveur sur lequel l'utilisateur se connectera)
                + Passerelle (Serveur de passerelle de bureau à distance)
                + Administrateur (Compte administrateur du domaine)
                + AdminPassword (Mot de passe du compte administrateur)
                + ExecutablePath (Chemin de l'exécutable dont on créera le raccourci)
                + Logiciel (Nom du logiciel qui sera donné au raccourci)
        - Création du dossier qui recevra les fichiers RDP destinés aux utilisateurs pour se
connecter sur le serveur qui leur a été attribué.
        - Création de l'Unité Organisationnelle dans laquelle seront inscrits les nouveaux
utilisateurs si celle-ci n'existe pas déjà.
        - Création du Groupe de sécurité dans lequel seront inscrits les nouveaux utilisateurs
si celui-ci n'existe pas déjà.
        - Création du compte utilisateur dans l'Unité Organisationnelle.
        - Intégration du compte utilisateur dans le groupe dont il dépend.
        - Génération du fichier RDP.
        - Ouverture de session de l'utilisateur.
        - Création du raccourci de l'application sur le bureau de la session.
#>

# Importation du module Active Directory

Import-Module ActiveDirectory

# Prise en compte du fichier CSV dont le chemin est passé en argument
```

```
#Param(
#[string]$FichierCSV
#)

$FichierCSV = $args[0]

#### Déclaration des fonctions utilisées dans le script ####

# Fonction permettant une pause jusqu'à l'appui sur une touche

function Pause ($Message="Appuyer sur une touche pour continuer..."){
Write-Host ""
Write-Host -NoNewLine $Message
$null = $Host.UI.RawUI.ReadKey("NoEcho,IncludeKeyDown")
Write-Host ""
}

# Fonction utilisée pour la création d'un nouveau raccourci.

Function New-Shortcut{
param(
   [parameter(Mandatory=$true)][string]$ShortcutFullName,
   [parameter(Mandatory=$true)][string]$ShortcutTarget
)
$ShortcutObject = New-Object -comObject WScript.Shell
$Shortcut = $ShortcutObject.CreateShortcut($ShortcutFullName)
$Shortcut.TargetPath = $ShortcutTarget
$Shortcut.Save()
}

##### Script #####


# Analyse du fichier csv en spécifiant le caractère de séparation.

$UtilisateursCSV = Import-csv -Path $FichierCSV -Delimiter ";"

# Enregistrement des variables issues de l'analyse du fichier CSV passé en argument.

foreach($ligne in $UtilisateursCSV)
```

```
{
$Nom = $ligne.Nom
$Prenom = $ligne.Prenom
$NomComplet = $ligne.NomComplet
$NomUtilisateur = $ligne.Utilisateur
$MotDePasse = $ligne.MotDePasse
$Groupe = $ligne.Groupe
$UniteOrganisation = $ligne.UniteOrganisation
$UniteOrganisationRoot = $ligne.UniteOrganisationRacine
$Domaine = $ligne.Domaine
$TLD = $ligne.TLD
$RDSServer = $ligne.Serveur
$Passerelle = $ligne.Passerelle

# Vérification et création du dossier dans lequel seront déposés les fichiers RDP

$RDPDir = "C:\RDP_$Groupe"
If(!(test-path $RDPDir))
{
      New-Item -ItemType Directory -Force -Path $RDPDir
}

# Vérification de l'existence de l'OU à créer dans la base Active Directory

$OUExist = Get-ADOrganizationalUnit -LDAPFilter "(Name=_$UniteOrganisation)" | Select-Object
Name | Foreach { $_.Name }

if ( "$OUExist" -eq "_$UniteOrganisation" ) {
    # Si l’OU existe déjà, afficher un message d’avertissement

    Write-Warning "`n L'Unite Organisationnelle _$UniteOrganisation existe deja."

    }
        else
    {
    # Si l'OU n'existe pas, création de l'OU

    Write-Warning "`n L'Unite Organisationnelle _$UniteOrganisation n'existe pas."
    New-ADOrganizationalUnit -Name _$UniteOrganisation -Path
"OU=$UniteOrganisationRoot,DC=$Domaine,DC=$TLD"
```

```
}

# Vérification de l'existence du groupe à créer dans la base Active Directory

$GroupExist = Get-ADGroup -LDAPFilter "(Name=$Groupe)" | Select-Object Name | Foreach {
$_.Name }

 if ( "$GroupExist" -eq "$Groupe" ) {
    # Si l'utilisateur existe déjà, afficher un message d'avertissement

    Write-Warning "`n Le groupe $Groupe existe deja dans l'annuaire."

    }
 else
    {
    # Si le groupe n'existe pas, création du groupe.

    Write-Warning "`n Le groupe $Groupe n'existe pas dans l'annuaire."
    New-ADGroup $Groupe -Path
"OU=_$UniteOrganisation,OU=$UniteOrganisationRoot,DC=$Domaine,DC=$TLD" -GroupCategory Security
-GroupScope Global -PassThru –Verbose
 }

# Vérification de l'existence de l'utilisateur à créer dans la base Active Directory

if (Get-ADUser -F {SamAccountName -eq "$NomUtilisateur"} ) {
    # Si l'utilisateur existe déjà, afficher un message d'avertissement

    Write-Warning "`n Le compte $NomUtilisateur existe deja dans l'annuaire."

    }
else
    {
    #Création de l'utilisateur

    Write-Warning "`n Le compte $NomUtilisateur n'existe pas dans l'annuaire."
    New-ADuser `
     -Name "$NomComplet" `
     -GivenName "$Prenom" `
     -Surname "$Nom" `
```

```
      -SamAccountName "$NomUtilisateur" `
      -Path "OU=_$UniteOrganisation,OU=$UniteOrganisationRoot,DC=$Domaine,DC=$TLD" `
      -AccountPassword (ConvertTo-SecureString "$MotDePasse" -AsPlainText -force) `
      -passThru `
      -Enabled $True `
      -ChangePasswordAtLogon $False `
      -CannotChangePassword $True

    #Ajout de l'utilisateur dans le groupe dont il dépend

    Add-ADGroupMember -Identity "$Groupe" -Members "$NomUtilisateur"
}

##### Création du fichier RDP. #####

# Vérification de l'existence du fichier rdp.

if ( ( Test-Path "$RDPDir\$NomUtilisateur.rdp" ) -eq $False ) {
# Si le fichier n'existe pas, afficher un message et créer le fichier
Write-Host "`n Le fichier $NomUtilisateur n'existe pas dans le répertoire $RDPDir `n Création
du fichier"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "screen mode id:i:2"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "use multimon:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "desktopwidth:i:1920"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "desktopheight:i:1080"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "session bpp:i:16"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "winposstr:s:0,1,-1816,21,-952,777"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "compression:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "keyboardhook:i:2"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "audiocapturemode:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "videoplaybackmode:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "connection type:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "networkautodetect:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "bandwidthautodetect:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "displayconnectionbar:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "enableworkspacereconnect:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "disable wallpaper:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "allow font smoothing:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "allow desktop composition:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "disable full window drag:i:1"
```

```
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "disable menu anims:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "disable themes:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "disable cursor setting:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "bitmapcachepersistenable:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "full address:s:$RDSServer"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "audiomode:i:2"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "redirectprinters:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "redirectcomports:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "redirectsmartcards:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "redirectclipboard:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "redirectposdevices:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "autoreconnection enabled:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "authentication level:i:2"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "prompt for credentials:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "negotiate security layer:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "remoteapplicationmode:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "alternate shell:s:"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "shell working directory:s:"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "gatewayhostname:s:$Passerelle"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "gatewayusagemethod:i:2"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "gatewaycredentialssource:i:4"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "gatewayprofileusagemethod:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "promptcredentialonce:i:1"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "gatewaybrokeringtype:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "use redirection server name:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "rdgiskdcproxy:i:0"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "kdcproxyname:s:"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value "drivestoredirect:s:"
    ADD-content -path "$RDPDir\$NomUtilisateur.rdp" -value
"username:s:$Domaine.$TLD\$NomUtilisateur"
    }
else
    {
    # Si le fichier existe déjà, afficher un message d’avertissement
    Write-Warning "`n Le fichier $NomUtilisateur.rdp existe deja dans le répertoire $RDPDir"
}

##### Ouverture des sessions de bureau à distance. #####

# Enregistrement des infos de connexions
```

```
cmdkey /add:$RDSServer /user:$NomUtilisateur@$Domaine.$TLD /pass:$MotDePasse

# Connexion au serveur par l'utilisateur

mstsc "$RDPDir\$NomUtilisateur.rdp"
# Suppression des infos de connexions de l'utilisateur de la mémoire système

cmdkey /delete:$RDSServer

# Fin de la boucle
Pause
}

##### Fin du script #####

$host.ui.RawUI.ForegroundColor = 'Red'
Write-Host "`n Au revoir !!!"
$host.ui.RawUI.ForegroundColor = 'Green'
Pause
```

Contenu du fichier csv :

```
Nom;Prenom;NomComplet;Utilisateur;MotDePasse;Groupe;UniteOrganisation;UniteOrganisationRacine;
Domaine;TLD;Serveur;Passerelle
```