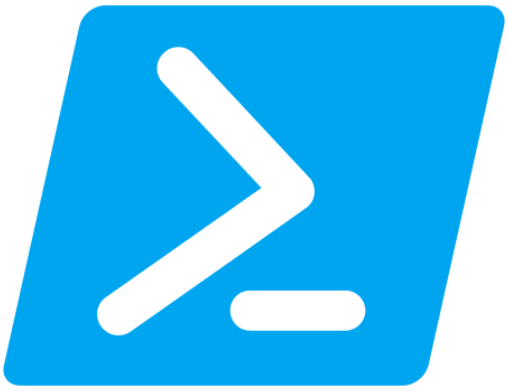
BENZERGUA FARES

POWERSHELL : Script de gestion des utilisateurs d'un AD



Script Powershell sous Windows Serveur 2022

1. On va vérifier la version de Powershell installée sur votre serveur avec la commande $PsVersionTable

```
PS C:\Users\Administrateur> $PsVersionTable

Name                           Value
----                           -----
PSVersion                      5.1.20348.320
PSEdition                      Desktop
PSCompatibleVersions           {1.0, 2.0, 3.0, 4.0...}
BuildVersion                   10.0.20348.320
CLRVersion                     4.0.30319.42000
WSManStackVersion              3.0
PSRemotingProtocolVersion      2.3
SerializationVersion           1.1.0.1
```

2. Nous allons relever la politique d'exécution des scripts avec la commande Get-ExecutionPolicy

```
PS C:\Users\Administrateur.WINSRV> Get-ExecutionPolicy
RemoteSigned
PS C:\Users\Administrateur.WINSRV>
```

3. Depuis Powershell , nous allons changer la politique d'exécution des scripts avec la commande Set-ExecutionPolicy Unrestricted

```
PS C:\Users\Administrateur> Set-ExecutionPolicy Unrestricted

Modification de la stratégie d'exécution
La stratégie d'exécution permet de vous prémunir contre les scripts que vous jugez non fiables. En modifiant la
stratégie d'exécution, vous vous exposez aux risques de sécurité décrits dans la rubrique d'aide
about_Execution_Policies à l'adresse https://go.microsoft.com/fwlink/?LinkID=135170. Voulez-vous modifier la stratégie
d'exécution ?
[O] Oui  [T] Oui pour tout  [N] Non  [U] Non pour tout  [S] Suspendre  [?] Aide (la valeur par défaut est « N ») :
```

```
PS C:\Users\Administrateur> Get-ExecutionPolicy
Unrestricted
```

4. Nous allons adapter la fonction « ListeEmployes » pour les paramètres de votre annuaire AD

```
function listeEmployes{
    Write-Host `n
    Write-Host "Affichage de la liste des employés" -ForegroundColor Red

    #A modifier au niveau du DN
    Get-ADUser -Filter * -SearchBase "OU=People,DC=win,DC=menuimetal,DC=fr" | Format-table GivenName,SurName,SamAccountName

    Write-Host "Appuyez sur la touche <ENTREE> afin de revenir au menu"
    read-Host
    clear
}
```

Get-ADUser -Filter * -SearchBase "OU=People,DC=win,DC=menuimetal,DC=fr" | Format-table GivenName,SurName,SamAccountName

5. Créez le corps de la fonction « CreerEmploye »

```
function CreerEmployes{
    write-host "----------------Création  de compte employé----------------"

    # Infos du compte
    $nom = Read-Host "Entrez votre nom : "
    $prenom = Read-Host "Entrez votre prénom : "
    $OU = Read-Host "Entrez l'OU où créer le compte : "

    # Vérification si l'OU existe
    if (Get-ADOrganizationalUnit -Filter "Name -eq '$OU'") {
     Write-Host "L'OU spécifiée existe. "
    }
    else {
    Write-Host "l'OU spécifiée n'existe pas."
    }
    # Vérification si l'utlisateur existe
   if (Get-ADUser -Filter {GivenName -eq $prenom -and Surname -eq $nom}) {
     Write-Host "L'utilisateur $prenom $nom existe déjà."
     }

    else {
     Write-Host "L'utilisateur $prenom $nom n'existe pas."
     }

    # Création du compte
    if ($nom.Length -gt 7) {
     $SamMAccountName = $prenom[0].ToString().ToUpper() + $nom.Substring(0,7).ToLower()
     }

     else {
     $SamMAccountName = $prenom[0].ToString().ToUpper() + $nom.ToLower()
     }

    # Création du mot de passe
    if ($nom.Length -gt 7) {
     $MDP = $prenom[0].ToString().ToUpper() + $nom.Substring(0,7).ToLower() + "_2024"
     }

     else {
     $MDP = $prenom[0].ToString().ToUpper() + $nom.ToLower() + "_2024"
     }

    # Cryptage du mot de passe
     $MDPCrypte = ConvertTo-SecureString $MDP -AsPlainText -Force

    # Création du nouvel utilisateur dans Active Directory
     New-ADUser -SamAccountName $SamAccountName -UserPrincipalName "$SamAccountName@menuimetal.fr" -Name "$prenom $nom" ` -GivenName $prenom -Surname $nom -Path $OU -AccountPassword $MDPCrypte -Enabled $true

     Write-Host "L'utilisateur $SamAccountName a été créé avec succès."


    Write-Host "Appuyez sur la touche <ENTREE> afin de revenir au menu"
    read-Host
    clear
}
```


le script est composé de 7 parties qu'on décomposera un à un

Partie 1 : Infos du compte

```
87
88      # Infos du compte
89      $nom = Read-Host "Entrez votre nom : "
90      $prenom = Read-Host "Entrez votre prénom : "
91      $OU = Read-Host "Entrez l'OU où créer le compte : "
92
```

On a ici 3 variables, $nom $prenom et $OU dans lesquelles seront stockés le prénom le nom et l'ou entrés par l'utilisateur suivi de la commande Read-Host qui permet de demander à l'utilisateur de saisir une donnée dans la console.

Partie 2 : Vérification si l'OU existe

```
92
93      # Vérification si l'OU existe
94      if (Get-ADOrganizationalUnit -Filter "Name -eq '$OU'") {
95       Write-Host "L'OU spécifiée existe. "
96      }
97      else {
98      Write-Host "l'OU spécifiée n'existe pas."
99      }
```

Sur cette partie nous avons une boucle if/else, suivie de la commande Get-ADOrganizationalUnit qui permet de récupérer les unités d'organisation (OU) dans Active Directory. le -Filter spécifie qu'on recherche une OU dont le Nom (propriété Name) correspond à la valeur stockée dans la variable $OU. Le nom de l'OU est donc comparé à cette valeur pour voir si elle existe dans l'annuaire. Si l'OU existe donc, le script affiche un message avec Write-Host pour indiquer que l'OU existe, en revanche si elle n'existe pas elle passe à la condition else

Partie 3 : Vérification si l'utilisateur existe

```
100     # Vérification si l'utlisateur existe
101     if (Get-ADUser -Filter {GivenName -eq $prenom -and Surname -eq $nom}) {
102       Write-Host "L'utilisateur $prenom $nom existe déjà."
103       }
104
105     else {
106       Write-Host "L'utilisateur $prenom $nom n'existe pas."
107       }
```

Nous avons encore une fois une boucle if/else avec la commande Get-ADUser qui est utilisée pour récupérer un ou plusieurs utilisateurs dans Active Directory. -Filter {GivenName (le prénom) -eq $prenom -and Surname (le nom de famille) -eq $nom}. Si l'utilisateur existe, le script affiche un message avec Write-Host pour indiquer que celui-ci existe, en revanche s'il n'existe pas elle passe à la condition else

Partie 4 : Création du compte et du mot de passe

```
108
109     # Création du compte
110     if ($nom.Length -gt 7) {
111      $SamMAccountName = $prenom[0].ToString().ToUpper() + $nom.Substring(0,7).ToLower()
112      }
113
114      else {
115      $SamMAccountName = $prenom[0].ToString().ToUpper() + $nom.ToLower()
116      }
117
118     # Création du mot de passe
119     if ($nom.Length -gt 7) {
120      $MDP = $prenom[0].ToString().ToUpper() + $nom.Substring(0,7).ToLower() + "_2024"
121      }
122
123      else {
124      $MDP = $prenom[0].ToString().ToUpper() + $nom.ToLower() + "_2024"
125      }
126
```

Pour la création du compte et du mot de passe nous utilisons encore la même structure (if/else), dans nos boucles on utilise la commande $nom.Length suivi de l'opérateur -gt signifiant "greater than" (supérieur à) qui vérifie si le nom de famille contient plus de 7 caractères pour limiter la longueur de l'identifiant

Nous allons décortiquer la variable $SamMAccountName :

1. $prenom[0] : Cela récupère la première lettre du prénom (index 0 de la chaîne $prenom)
2. ToString().ToUpper() : La première lettre du prénom est convertie en une chaîne de caractères et mise en majuscule
3. $nom.Substring(0,7) : Cela récupère les 7 premiers caractères du nom de famille
4. ToLower() : Les 7 premiers caractères du nom de famille sont convertis en minuscules

Par exemple : Si $prenom = "Marie" et $nom = "Dupontel", le sAMAccountName sera : "Mduponte" (la première lettre du prénom + les 7 premières lettres du nom de famille)

Dans le cas où le nom de famille contient 7 caractères ou moins, l'identifiant sera constitué de :

1. $prenom[0].ToString().ToUpper() : La première lettre du prénom, convertie en majuscule.
2. $nom.ToLower() : Le nom de famille en entier, converti en minuscules.

Par exemple : Si $prenom = "Luc" et $nom = "Durand", le $sAMAccountName sera : "Ldurand" (la première lettre du prénom + le nom de famille entier en minuscules)

Pour la variable $MDP (Mot de passe) le principe est le même, nous ajoutons seulement "_2024" à la fin.

Partie 5 : Cryptage du mot de passe

```
126
127     # Cryptage du mot de passe
128      $MDPCrypte = ConvertTo-SecureString $MDP -AsPlainText -Force
129
```

Pour la variable $MDPCrypte :

1. ConvertTo-SecureString : Convertit un texte clair en un mot de passe sécurisé
2. $MDP : Contient le mot de passe en texte clair (ex. "Jdupont_2024")
3. -AsPlainText : Indique que le texte fourni est en clair.-Force : Force la conversion malgré l'utilisation de texte clair

Partie 6 : Création du nouvel utilisateur dans Active Directory

```
# Création du nouvel utilisateur dans Active Directory
 New-ADUser -SamAccountName $SamAccountName -UserPrincipalName "$SamAccountName@menuimetal.fr" -Name "$prenom $nom" ` -Give

 Write-Host "L'utilisateur $SamAccountName a été créé avec succès."
```

New-ADUser -SamAccountName $sAMAccountName -UserPrincipalName "$sAMAccountName@menuimetal.fr" -Name "$prenom $nom" ` -GivenName $prenom -Surname $nom -Path $ou -AccountPassword $MDP -Enabled $true