**Documentation**

**Technique**

ligne horizontale

**Script CRUD PowerShell**

ligne horizontale

# 

**31/03/2025**

**Gabriel PFISTER, Quentin PARC**

**Sommaire**

ligne horizontale

[**Contexte 2**](#_yqc3zntowln1)

[**Mise en place PowerShell CRUD pour Active Directory 2**](#_80qomgufoadg)

[Prérequis 2](#_beeclz2h9pya)

[1. Script ActiveDirectoryCRUD.ps1 3](#_jzgghj18g7df)

[2. Script ActiveDirectory\_Ajouter\_Supprimer\_Tous.ps1 4](#_hbtu2fyyeb48)

# Contexte



Un script CRUD (Create, Read, Update, Delete) en PowerShell permet d'automatiser la gestion des utilisateurs Active Directory. Ces scripts facilitent l'ajout, la suppression, la modification et la lecture des informations utilisateurs d'un domaine Active Directory.

L'objectif est de simplifier l'administration des utilisateurs en utilisant des commandes PowerShell personnalisées et interactives.

# **Mise en place PowerShell CRUD pour Active Directory**

## 

## **Prérequis**

Avant d’utiliser le script PowerShell pour Créer, Lire, Mettre à jour et Supprimer des utilisateurs dans l’Active Directory, assurez-vous d’avoir

* Un Active Directory actif
* Les droits d’administrateur sur votre machine

# Scripts

| **# Importation du module Active Directory Import-Module ActiveDirectory  # Définition des constantes $Domain = "SISR.sl" $OU\_Employes = "OU=Employes,DC=SISR,DC=sl" $CSV\_Path = "C:\Users\Administrateur\Desktop\utilisateurs.csv" $CSV\_Output\_Path = "C:\Users\Administrateur\Desktop\utilisateurs\_modifies.csv" $Partage\_Parent = "\\Serveur\Utilisateurs"  # Vérifier et créer l'OU Employes si elle n'existe pas if (-not (Get-ADOrganizationalUnit -Filter { Name -eq "Employes" })) {  New-ADOrganizationalUnit -Name "Employes" -Path "DC=SISR,DC=sl" }  # Fonction pour créer un partage réseau function Creer-Partage($SamAccountName) {  $Dossier = "$Partage\_Parent\$SamAccountName"  if (-Not (Test-Path $Dossier)) {  New-Item -Path $Dossier -ItemType Directory | Out-Null  icacls $Dossier /grant "$SamAccountName`:(OI)(CI)F" /T  Write-Host "📂 Dossier réseau créé pour $SamAccountName"  } else {  Write-Host "⚠️ Dossier réseau déjà existant pour $SamAccountName"  } }  # Fonction principale function Gerer-Utilisateurs {  if (-Not (Test-Path $CSV\_Path)) {  Write-Host "❌ Fichier CSV non trouvé : $CSV\_Path"  return  }   $UtilisateursCSV = Import-Csv -Path $CSV\_Path -Delimiter ','  $UtilisateursModifies = @()   foreach ($Utilisateur in $UtilisateursCSV) {  $AncienSamAccountName = $Utilisateur.AncienSamAccountName  $Prenom = $Utilisateur.Prenom  $Nom = $Utilisateur.Nom  $SamAccountName = $Utilisateur.SamAccountName  $MotDePasse = $Utilisateur.MotDePasse  $Groupe = $Utilisateur.Groupe  $Action = $Utilisateur.Action   switch ($Action) {  "Ajouter" {  if (-not (Get-ADUser -Filter "SamAccountName -eq '$SamAccountName'" -ErrorAction SilentlyContinue)) {  New-ADUser -SamAccountName $SamAccountName -UserPrincipalName "$SamAccountName@$Domain" `  -Name "$Prenom $Nom" -GivenName $Prenom -Surname $Nom -Path $OU\_Employes `  -AccountPassword (ConvertTo-SecureString $MotDePasse -AsPlainText -Force) `  -Enabled $true -ChangePasswordAtLogon $true   Add-ADGroupMember -Identity $Groupe -Members $SamAccountName  Creer-Partage $SamAccountName   Write-Host "✅ Utilisateur $Prenom $Nom ajouté."  $UtilisateursModifies += $Utilisateur  } else {  Write-Host "⚠️ L'utilisateur $SamAccountName existe déjà."  }  }   "Modifier" {  if ($AncienSamAccountName -eq "" -or -not (Get-ADUser -Filter "SamAccountName -eq '$AncienSamAccountName'" -ErrorAction SilentlyContinue)) {  Write-Host "❌ AncienSamAccountName $AncienSamAccountName introuvable."  } else {  $User = Get-ADUser -Identity $AncienSamAccountName -Properties MemberOf, DistinguishedName  Rename-ADObject -Identity $User.DistinguishedName -NewName "$Prenom $Nom"   Set-ADUser -Identity $AncienSamAccountName `  -SamAccountName $SamAccountName `  -UserPrincipalName "$SamAccountName@$Domain" `  -GivenName $Prenom `  -Surname $Nom `  -DisplayName "$Prenom $Nom" `  -ChangePasswordAtLogon $true   if ($MotDePasse) {  Set-ADAccountPassword -Identity $SamAccountName -NewPassword (ConvertTo-SecureString $MotDePasse -AsPlainText -Force) -Reset  Write-Host "🔑 Mot de passe mis à jour."  }   if ($Groupe) {  $CurrentGroups = $User.MemberOf | Get-ADGroup | Select-Object -ExpandProperty Name  foreach ($CurrentGroup in $CurrentGroups) {  Remove-ADGroupMember -Identity $CurrentGroup -Members $AncienSamAccountName -Confirm:$false  }  Add-ADGroupMember -Identity $Groupe -Members $SamAccountName  Write-Host "✅ Groupe mis à jour : $SamAccountName → $Groupe"  }   Write-Host "✅ Utilisateur modifié : $AncienSamAccountName → $SamAccountName."  $Utilisateur.Action = "Modifié"  $UtilisateursModifies += $Utilisateur  }  }   "Supprimer" {  if (Get-ADUser -Filter "SamAccountName -eq '$SamAccountName'" -ErrorAction SilentlyContinue) {  Remove-ADUser -Identity $SamAccountName -Confirm:$false  Write-Host "❌ Utilisateur $SamAccountName supprimé."  $Utilisateur.Action = "Supprimé"  $UtilisateursModifies += $Utilisateur  } else {  Write-Host "⚠️ L'utilisateur $SamAccountName n'existe pas."  }  }   default {  Write-Host "⚠️ Action inconnue pour $SamAccountName."  }  }  }   $UtilisateursModifies | Export-Csv -Path $CSV\_Output\_Path -NoTypeInformation  Write-Host "🎯 Modifications enregistrées dans : $CSV\_Output\_Path" }  # Menu interactif do {  Clear-Host  Write-Host "========= MENU ========="  Write-Host "1 - Ajouter des utilisateurs (modifiez le CSV et ajoutez 'Ajouter')"  Write-Host "2 - Lister un utilisateur spécifique"  Write-Host "3 - Modifier des utilisateurs (modifiez le CSV et ajoutez 'Modifier')"  Write-Host "4 - Supprimer des utilisateurs (modifiez le CSV et ajoutez 'Supprimer')"  Write-Host "5 - Lancer toutes les actions du CSV"  Write-Host "6 - Quitter"  Write-Host "========================"   $Choix = Read-Host "Entrez votre choix"   switch ($Choix) {  "1" { Write-Host "Ajoutez des utilisateurs en modifiant le CSV avec 'Ajouter' et relancez l'option 5." }  "2" {   $Sam = Read-Host "Entrez le SamAccountName de l'utilisateur"  Get-ADUser -Identity $Sam -Properties \* | Select-Object SamAccountName, Name, Enabled, DistinguishedName  }  "3" { Write-Host "Modifiez le CSV avec 'Modifier' et relancez l'option 5." }  "4" { Write-Host "Modifiez le CSV avec 'Supprimer' et relancez l'option 5." }  "5" { Gerer-Utilisateurs }  "6" { Write-Host "Fermeture du script..."; exit }  default { Write-Host "❌ Option invalide, réessayez." }  }   Pause } while ($Choix -ne "6")** |
| --- |

### **2. Script ActiveDirectory\_Ajouter\_Supprimer\_Tous.ps1**

Ce script est conçu pour gérer en masse les utilisateurs et groupes Active Directory.

Fonctionnalités :

1. Création de groupes AD (Chefs de service, Administrateurs réseau, Employés, Invités, Stagiaires).
2. Ajout de plusieurs utilisateurs automatiquement dans chaque groupe.
   * Exemple : 10 chefs de service (chef1, chef2...), 63 employés (employe1, employe2...).
3. Suppression de tous les utilisateurs de l’UO (Unité Organisationnelle) Employes.
4. Menu interactif permettant d’ajouter ou supprimer tous les utilisateurs.

**Script ActiveDirectoryCRUD.ps1**

| # Importation du module Active Directory Import-Module ActiveDirectory  # Définition du domaine AD $Domain = "SISR.sl" # Définition de l'unité organisationnelle principale $OU\_Employes = "OU=Employes,DC=SISR,DC=sl"  # Vérifier et créer l'UO Employes si elle n'existe pas if (-not (Get-ADOrganizationalUnit -Filter { Name -eq "Employes" })) {  New-ADOrganizationalUnit -Name "Employes" -Path "DC=SISR,DC=sl" }  # Création des groupes $Groups = @{  "AdminsReseaux" = "Groupe Administrateur Réseau";  "Salaries" = "Groupe Salariés";  "Service" = "Groupe Service";  "Stagiaires" = "Groupe Stagiaires";  "Invites" = "Groupe Invites"; }  foreach ($Group in $Groups.Keys) {  if (-not (Get-ADGroup -Filter { Name -eq $Group })) {  New-ADGroup -Name $Group -GroupScope Global -Path $OU\_Employes -SamAccountName $Group -Description $Groups[$Group]  } }  # Fonctions CRUD pour les utilisateurs function Ajouter-Utilisateur {  param (  [string]$Prenom,  [string]$Nom,  [string]$MotDePasse,  [string]$Groupe  )  $SamAccountName = "$Prenom.$Nom"  $NomComplet = "$Prenom $Nom"  New-ADUser -Server SISR.sl -SamAccountName $SamAccountName -UserPrincipalName "$SamAccountName@SISR.sl" -Name $NomComplet -GivenName $Prenom -Surname $Nom -Path $OU\_Employes -AccountPassword (ConvertTo-SecureString $MotDePasse -AsPlainText -Force) -Enabled $true -ChangePasswordAtLogon $true  Add-ADGroupMember -Identity $Groupe -Members $SamAccountName  Write-Host "Utilisateur $NomComplet ajouté avec succès sur SISR.sl et ajouté au groupe $Groupe." }  function Lire-Utilisateur {  param ([string]$SamAccountName)  Get-ADUser -Server SISR.sl -Identity $SamAccountName -Properties \* | Select-Object Name, SamAccountName, UserPrincipalName, Enabled, GivenName, Surname, DistinguishedName }  function Modifier-Utilisateur {  param (  [string]$SamAccountName,  [string]$NouveauNom,  [string]$NouveauPrenom,  [string]$NouveauMotDePasse  )   # Récupérer l'utilisateur dans AD  $User = Get-ADUser -Identity $SamAccountName -Properties GivenName, Surname, UserPrincipalName, DistinguishedName   if ($User) {  # Si le prénom ou le nom est vide, utiliser l'ancien prénom ou nom  $NouveauPrenom = if ($NouveauPrenom) { $NouveauPrenom } else { $User.GivenName }  $NouveauNom = if ($NouveauNom) { $NouveauNom } else { $User.Surname }   # Générer le nouveau SamAccountName et Nom Complet  $NouveauSamAccountName = "$NouveauPrenom.$NouveauNom"  $NomComplet = "$NouveauPrenom $NouveauNom"   # Récupérer l'ancien DN et générer le nouveau  $NewDN = "CN=$NomComplet," + ($User.DistinguishedName -replace '^CN=[^,]+,')   # Renommer l'objet dans AD  Rename-ADObject -Identity $User.DistinguishedName -NewName $NomComplet   # Mettre à jour les attributs (identifiant, prénom, nom, affichage)  Set-ADUser -Identity $User.SamAccountName `  -SamAccountName $NouveauSamAccountName `  -UserPrincipalName "$NouveauSamAccountName@SISR.sl" `  -GivenName $NouveauPrenom `  -Surname $NouveauNom `  -DisplayName $NomComplet   # Si un nouveau mot de passe est fourni, le modifier  if ($NouveauMotDePasse -ne "") {  $SecurePassword = ConvertTo-SecureString $NouveauMotDePasse -AsPlainText -Force  Set-ADAccountPassword -Identity $User.SamAccountName -NewPassword $SecurePassword -Reset  Write-Host "Mot de passe mis à jour pour $NouveauSamAccountName." -ForegroundColor Yellow  } else {  Write-Host "Le mot de passe n'a pas été modifié." -ForegroundColor Cyan  }   Write-Host " Utilisateur mis à jour : $SamAccountName → $NouveauSamAccountName ($NomComplet) sur SISR.sl." -ForegroundColor Green  } else {  Write-Host " Erreur : Utilisateur $SamAccountName introuvable." -ForegroundColor Red  } }  function Supprimer-Utilisateur {  param ([string]$SamAccountName)  Remove-ADUser -Server SISR.sl -Identity $SamAccountName -Confirm:$false  Write-Host "Utilisateur $SamAccountName supprimé avec succès de SISR.sl." }  function Lister-Utilisateurs {  Write-Host "Liste des utilisateurs Active Directory sur SISR.sl :"  Get-ADUser -Server SISR.sl -Filter \* | Select-Object Name, SamAccountName, Enabled | Format-Table -AutoSize }  # Menu interactif do {  Clear-Host  Write-Host "=== Gestion des Utilisateurs AD sur SISR.sl ==="  Write-Host "1. Ajouter un utilisateur"  Write-Host "2. Lire les informations d'un utilisateur"  Write-Host "3. Modifier un utilisateur"  Write-Host "4. Supprimer un utilisateur"  Write-Host "5. Lister tous les utilisateurs"  Write-Host "6. Quitter"  $choix = Read-Host "Choisissez une option"  switch ($choix) {  1 {  $Prenom = Read-Host "Prénom"  $Nom = Read-Host "Nom"  $MotDePasse = Read-Host "Mot de passe"  Write-Host "Choisissez un groupe : 1. Administrateurs Réseaux, 2. Salariés, 3. Service, 4. Stagiaires, 5. Invites"  $choixGroupe = [int](Read-Host "Numéro du groupe")  Write-Host "Vous avez choisi : $choixGroupe" -ForegroundColor Yellow  $Groupe = switch ($choixGroupe) {  1 { "AdminsReseaux" }  2 { "Salaries" }  3 { "Service" }  4 { "Stagiaires" }  5 { "Invites" }  default { "Salaries" }  }  Write-Host "L'utilisateur sera ajouté au groupe : $Groupe" -ForegroundColor Cyan  Ajouter-Utilisateur -Prenom $Prenom -Nom $Nom -MotDePasse $MotDePasse -Groupe $Groupe  }  2 {  $SamAccountName = Read-Host "Identifiant (prenom.nom)"  Lire-Utilisateur -SamAccountName $SamAccountName | Format-Table -AutoSize  }  3 {  $SamAccountName = Read-Host "Identifiant (prenom.nom)"  $NouveauPrenom = Read-Host "Nouveau prénom (laisser vide pour conserver l'ancien)"  $NouveauNom = Read-Host "Nouveau nom (laisser vide pour conserver l'ancien)"  $NouveauMotDePasse = Read-Host "Nouveau mot de passe (laisser vide si inchangé)"  Modifier-Utilisateur -SamAccountName $SamAccountName -NouveauNom $NouveauNom -NouveauPrenom $NouveauPrenom -NouveauMotDePasse $NouveauMotDePasse  }  4 {  $SamAccountName = Read-Host "Identifiant (prenom.nom)"  Supprimer-Utilisateur -SamAccountName $SamAccountName  }  5 {  Lister-Utilisateurs  }  6 {  Write-Host "Fin du programme."  exit  }  default {  Write-Host "Choix invalide, veuillez réessayer."  }  }  Pause } while ($true) |
| --- |

**Script ActiveDirectory\_Ajouter\_Supprimer\_Tous.ps1**

| # Importation du module Active Directory Import-Module ActiveDirectory  # Définition du domaine AD $Domain = "SISR.sl" # Définition de l'unité organisationnelle principale $OU\_Employes = "OU=Employes,DC=SISR,DC=sl"  # Vérifier et créer l'UO Employes si elle n'existe pas if (-not (Get-ADOrganizationalUnit -Filter { Name -eq "Employes" })) {  New-ADOrganizationalUnit -Name "Employes" -Path "DC=SISR,DC=sl" }  # Création des groupes $Groups = @{  "ChefsService" = 10;  "AdminsReseaux" = 3;  "Employes" = 63;  "Invites" = 2;  "Stagiaires" = 2; }  foreach ($Group in $Groups.Keys) {  if (-not (Get-ADGroup -Filter { Name -eq $Group })) {  New-ADGroup -Name $Group -GroupScope Global -Path $OU\_Employes -SamAccountName $Group -Description "Groupe des $Group"  } }  # Fonction pour créer des utilisateurs et les ajouter au groupe correspondant function Create-ADUserAndAddToGroup {  param (  [string]$BaseName,  [int]$Count,  [string]$Group  )  for ($i = 1; $i -le $Count; $i++) {  $UserName = "$BaseName$i"  New-ADUser -Name $UserName -SamAccountName $UserName -UserPrincipalName "$UserName@$Domain" -Path $OU\_Employes -AccountPassword (ConvertTo-SecureString "P@ssword123" -AsPlainText -Force) -Enabled $true  Add-ADGroupMember -Identity $Group -Members $UserName  } }  # Fonction pour supprimer tous les utilisateurs de l'UO Employes function Supprimer-TousUtilisateurs {  Get-ADUser -Filter \* -SearchBase $OU\_Employes | ForEach-Object {  Remove-ADUser -Identity $\_.SamAccountName -Confirm:$false  }  Write-Host "Tous les utilisateurs ont été supprimés avec succès." }  # Menu interactif do {  Clear-Host  Write-Host "=== Gestion des Utilisateurs AD ==="  Write-Host "1. Ajouter tous les utilisateurs"  Write-Host "2. Supprimer tous les utilisateurs"  Write-Host "3. Quitter"  $choix = Read-Host "Choisissez une option"  switch ($choix) {  1 {  # Création des utilisateurs et ajout aux groupes seulement après avoir choisi l'option 1  Create-ADUserAndAddToGroup "chef" $Groups["ChefsService"] "ChefsService"  Create-ADUserAndAddToGroup "adminReseau" $Groups["AdminsReseaux"] "AdminsReseaux"  Create-ADUserAndAddToGroup "employe" $Groups["Employes"] "Employes"  Create-ADUserAndAddToGroup "invite" $Groups["Invites"] "Invites"    # Création des stagiaires individuellement  New-ADUser -Name "stagiaireInfo" -SamAccountName "stagiaireInfo" -UserPrincipalName "stagiaireInfo@$Domain" -Path $OU\_Employes -AccountPassword (ConvertTo-SecureString "P@ssword123" -AsPlainText -Force) -Enabled $true  Add-ADGroupMember -Identity "Stagiaires" -Members "stagiaireInfo"   New-ADUser -Name "stagiaire" -SamAccountName "stagiaire" -UserPrincipalName "stagiaire@$Domain" -Path $OU\_Employes -AccountPassword (ConvertTo-SecureString "P@ssword123" -AsPlainText -Force) -Enabled $true  Add-ADGroupMember -Identity "Stagiaires" -Members "stagiaire"   Write-Host "Tous les utilisateurs ont été ajoutés avec succès."  }  2 {  Write-Host " Attention : Tous les utilisateurs vont être supprimés !"  $confirmation = Read-Host "Êtes-vous sûr ? (oui/non)"  if ($confirmation -eq "oui") {  Supprimer-TousUtilisateurs  } else {  Write-Host "Annulation de la suppression."  }  }  3 {  Write-Host "Fin du programme."  exit  }  default {  Write-Host "Choix invalide, veuillez réessayer."  }  }  Pause } while ($true) |
| --- |

**Script CSV**

Création d’un script prenant en entré un fichier CSV afin de Créer, Lire, Modifier et Supprimer un utilisateur,