

Mémento PowerShell



Table des matières

Notions de base

Notions de base	2
Exécution d'un script PowerShell	2
Parcours des enregistrements d'un fichier .csv	3

Traitement des fichiers et dossiers

Traitement des fichiers et dossiers.....	4
Principaux cmdlets.....	4
Création d'un répertoire (directory).....	4
Création d'un fichier (file)	5
Parcours des fichiers et des sous-répertoires contenus dans un répertoire.....	6
Partage d'un répertoire (directory).....	7
Modification de l'héritage des autorisations NTFS d'un répertoire (directory)	8
Suppression de toutes les autorisations NTFS d'un groupe pour un répertoire (directory)	8
Ajout d'autorisations NTFS à un répertoire (directory)	9

Traitement des objets de l'Active Directory

Traitement des objets de l'Active Directory.....	10
Principaux objets et attributs d'objets dans l'Active Directory	10
Création d'une OU (Organizational Unit)(Unité d'Organisation)	12
Création d'un groupe d'utilisateurs (group)	13
Ajout d'un utilisateur à un groupe d'utilisateurs (group)	13
Création d'un utilisateur (user)	14
Parcours d'une collection d'objets Active Directory	16

Notions de base

Exécution d'un script PowerShell

La politique d'exécution des scripts PowerShell est gérée par la cmdlet [Set-ExecutionPolicy](#).

Pour exécuter des scripts Powershell (fichiers d'extension `.ps1`) sur une machine, il faut que la politique d'exécution du système le permette. Cette politique n'est pas vraiment un élément de sécurité mais juste un dispositif pour éviter d'exécuter du code Powershell par inadvertance.

Pour connaître la politique d'exécution des scripts powershell sur la machine locale :

`Get-ExecutionPolicy`

Pour modifier la politique d'exécution sur la machine locale et autoriser l'exécution de tous les scripts écrits localement (avec les droits Administrateurs) :

`Set-ExecutionPolicy RemoteSigned`

Pour afficher la politique d'exécution (*ExecutionPolicy*) courante en vigueur pour chaque champ d'application (*scope*) :

`Get-ExecutionPolicy -List`

```
PS C:\WINDOWS\system32> Get-ExecutionPolicy
Restricted

PS C:\WINDOWS\system32> Set-ExecutionPolicy RemoteSigned

PS C:\WINDOWS\system32> Get-ExecutionPolicy
RemoteSigned

PS C:\WINDOWS\system32> Get-ExecutionPolicy -List

Scope ExecutionPolicy
-----
MachinePolicy Undefined
UserPolicy Undefined
Process Undefined
CurrentUser Undefined
LocalMachine RemoteSigned
```

Les valeurs possibles de la politique d'exécution des scripts PowerShell sont :

- **Restricted** : on ne peut pas exécuter des scripts Powershell
- **AllSigned** : pour s'exécuter les scripts doivent être signés par un éditeur de confiance.
- **RemoteSigned** : les scripts écrits localement sont autorisés ; en revanche les scripts téléchargés doivent être signés par un éditeur de confiance. Il est possible d'autoriser les scripts téléchargés et non signés en utilisant la cmdlet `Unblock-File`.
- **Unrestricted** : tous les scripts peuvent être exécutés. Un message d'avertissement est affiché lors de l'exécution de scripts téléchargés informant des risques.
- **ByPass** : aucune vérification n'est effectuée, tous les scripts peuvent être exécutés.

Les valeurs possibles de la portée d'une politique d'exécution sont :

- **Process** : portée du processus Powershell
- **CurrentUser** : portée de l'utilisateur actuel
- **LocalMachine** : portée pour tous les utilisateurs de la machine
- **UserPolicy** : portée du groupe d'utilisateurs pour l'utilisateur actuel
- **MachinePolicy** : portée du groupe d'utilisateurs pour tous les utilisateurs de la machine.

Quelques remarques en préambule à l'exécution d'un script :

- Powershell n'est pas sensible à la casse

- Pour commenter du code, il faut utiliser le caractère `#`

`# Code PowerShell commenté`

- Pour effacer l'écran d'exécution :

`Clear-Host`

Parcours des enregistrements d'un fichier .csv

Le parcours des enregistrements d'un fichier .csv se fait avec la cmdlet [Import-Csv](#)

```
# Parcours des enregistrements d'un fichier .csv

# Le fichier utilisateurs.csv est stocké dans le même dossier que le script PowerShell
# Le nom de ce dossier (exemple : C:\Scripts-PowerShell\Tests ) est stocké dans la variable
# $PSScriptRoot

Set-Location $PSScriptRoot

$myFile=Import-Csv -Path "utilisateurs.csv" -Delimiter ";"

foreach($myRecord in $myFile)
{
    write "Nom : $($myRecord.nom)    Prénom : $($myRecord.prenom)"
}
```

```
Nom : Leloup    Prénom : Pierre
Nom : Dupont    Prénom : Michel
```



The screenshot shows a Notepad window with the title 'utilisateurs.csv X'. The content of the file is as follows:

```
1 prenom;nom;login;password;groupe;ou
2 Pierre;Leloup;pleloup;secret1A!;Etudiants;salleR209
3 Michel;Dupont;mdupont;secret1A!;Etudiants;salleR209
```

Traitement des fichiers et dossiers

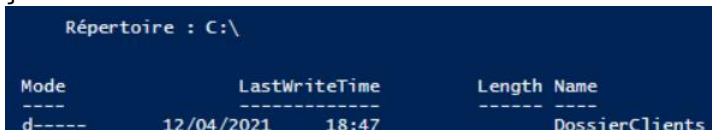
Principaux cmdlets

Test-Path	Test de l'existence d'un fichier ou d'un dossier
New-Item	Création de fichiers et dossiers
Get-ChildItem	Affichage de la liste de tous les fichiers et dossiers contenus dans un dossier
Copy-Item	Copie d'un fichier ou d'un dossier
Remove-Item	Suppression d'un fichier ou d'un dossier
New-PSDrive	Mappage d'un dossier en tant que lecteur
Get-Location	Affichage du dossier courant
Set-Location	Désignation d'un dossier pour en faire le dossier courant
Move-Item	Déplacement d'un fichier ou d'un dossier
Rename-Item	Renommage d'un fichier ou d'un dossier

Création d'un répertoire (directory)

Un répertoire est créé avec la cmdlet [New-Item](#).

```
# Création d'un répertoire
$myDirectoryFullName="C:\DossierClients"
if (!(Test-Path "$myDirectoryFullName"))
{
    New-Item -ItemType Directory -Path "$myDirectoryFullName"
}
else
{
    write-Host "Attention : le répertoire $myDirectoryFullName existe déjà !"
}
```



Répertoire : C:\

Mode	LastWriteTime	Length	Name
d-----	12/04/2021 18:47		DossierClients

Remarque : si le chemin d'accès du répertoire, et son nom sont dans deux variables différentes, on peut aussi utiliser :

```
$myDirectoryPath="C:\" # Terminer le chemin par le caractère \
$myDirectoryName="DossierClients"
New-Item -ItemType Directory -Path "$myDirectoryPath" -Name "$myDirectoryName"
```

Création d'un fichier (file)

Un fichier est créé avec la cmdlet [New-Item](#).

Création d'un fichier

```
$myFileFullName="C:\DossierClients\FichierClients.txt"
if (!(Test-Path "$myFileFullName"))
{
    New-Item -ItemType File -Path "$myFileFullName"
}
else
{
    write-Host "Attention : le fichier $myFileFullName existe déjà !"
}
```

```
Répertoire : C:\DossierClients

Mode                LastWriteTime         Length Name
----                -
-a-----         12/04/2021    18:47             0 FichierClients.txt
```

Remarque : si le chemin d'accès du fichier, et son nom sont dans deux variables différentes, on peut aussi utiliser :

```
$myFilePath="C:\DossierClients\"    # Terminer le chemin par le caractère \
$myFileName="FichierClients.txt"

New-Item -ItemType File -Path "$myFilePath" -Name "$myFileName"
```

Remarque : le contenu d'un répertoire peut être obtenu avec la cmdlet [Get-Item](#).

Affichage du contenu complet du répertoire C:\ImagesOS
(fichiers et sous-répertoires)

```
Get-Item C:\ImagesOS\*
```

Affichage des fichiers du répertoire C:\ImagesOS
(fichiers seulement)

```
Get-Item C:\ImagesOS\*. *
Get-Item C:\ImagesOS
```

```
PS C:\Scripts-PowerShell\Tests> Get-Item C:\ImagesOS\*

Répertoire : C:\ImagesOS

Mode                LastWriteTime         Length Name
----                -
d-----         15/04/2021    13:23             ISO
-a-----         05/03/2021     13:51          13 Corrigé1.txt
-a-----         05/03/2021     13:51          13 Sujet1.txt

PS C:\Scripts-PowerShell\Tests> Get-Item C:\ImagesOS\*. *

Répertoire : C:\ImagesOS

Mode                LastWriteTime         Length Name
----                -
-a-----         05/03/2021     13:51          13 Corrigé1.txt
-a-----         05/03/2021     13:51          13 Sujet1.txt
```

Parcours des fichiers et des sous-répertoires contenus dans un répertoire

Le parcours des éléments contenus dans un répertoire (fichiers avec ou sans les sous-répertoires) se fait avec la cmdlet [Get-ChildItem](#)

Liste des fichiers et sous-dossiers d'un dossier

```
$myDirectoryFullName="C:\ImagesOS"
```

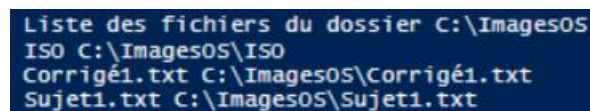
#On récupère la liste des fichiers de ce répertoire

```
$myFiles = Get-ChildItem -Path "$myDirectoryFullName"
```

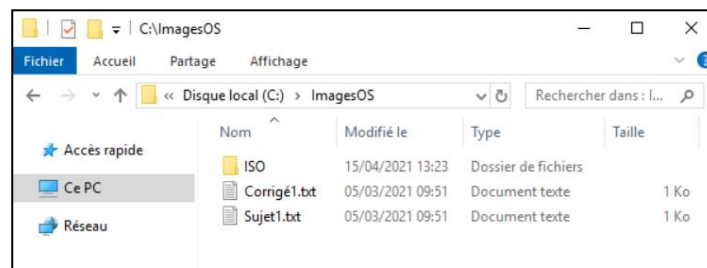
```
Write-Host "Liste des fichiers du dossier $myDirectoryFullName"
```

```
foreach ($aFile in $myFiles)
```

```
{  
    Write-Host "$($aFile.name) $($aFile.FullName)" # Nom du fichier et chemin d'accès complet  
}
```



```
Liste des fichiers du dossier C:\ImagesOS  
ISO C:\ImagesOS\ISO  
Corrigé1.txt C:\ImagesOS\Corrigé1.txt  
Sujet1.txt C:\ImagesOS\Sujet1.txt
```



Remarque importante :

On peut aussi obtenir les fichiers et dossiers contenus dans chacun des répertoires listés si on utilise le paramètre *-Recurse* pour parcourir récursivement ces dossiers :

```
$myFiles = Get-ChildItem -Path "$myDirectoryFullName" -Recurse
```

Remarque : on peut sélectionner les fichiers d'un dossier avec [Where-Object](#).

Exemple : pour ne parcourir que les fichiers dont le nom commence par Cor :

```
$myFiles = Get-ChildItem -Path "$myDirectoryFullName" | where-Object {$_.Name -like 'Cor*'}
```

Partage d'un répertoire (directory)

Un répertoire est partagé avec la cmdlet [New-SmbShare](#).

```
# Partager un dossier avec la cmdlet New-SmbShare
# Ici, on partage avec Contrôle Total (FullAccess) à Tout le monde

$myDirectoryPath="C:\" # Terminer le chemin par le caractère \
$myDirectoryName="DossierClients"
$myDirectoryFullName=$myDirectoryPath+$myDirectoryName
$myUNCDirectoryPath="//SERVEUR1\"+$myDirectoryName

if (Test-Path $myDirectoryFullName)
{
    if (Test-Path $myUNCDirectoryPath)
    {Write-Host "Le nom de partage $myUNCDirectoryPath existe déjà"
    }
    else
    {New-SmbShare -Name $myDirectoryName -Path "$myDirectoryFullName" -FullAccess "Tout le
monde"
    }
}
else
{Write-Host "Le dossier $myDirectoryFullName n'existe pas"
}
```

Name	ScopeName	AccountName	AccessControlType	AccessRight
C:\DossierClients	*	Tout le monde	Allow	Full

Remarque : l'ensemble des ressources partagées est obtenu avec la cmdlet [Get-SmbShare](#), et les autorisations de partage d'un répertoire peuvent être obtenues avec la cmdlet [Get-SmbShareAccess](#).

Affichage de toutes les ressources partagées

[Get-SmbShare](#)

Name	ScopeName	Path	Description
ADMIN\$	*	C:\Windows	Administration à distance
Bulletins	*	C:\Bulletins	
C\$	*	C:\	Partage par défaut
Calendrier	*	C:\Calendrier	
dernierTest	*	C:\dernierTest	
DocCommerciaux	*	C:\DocCommerciaux	
DocTech	*	C:\DocTech	
DOCUMENTATION	*	C:\DOCUMENTATION	
DossierClients	*	C:\DossierClients	
EXEMPLE	*	C:\EXEMPLE	
Formation	*	C:\Formation	
HPLaserJet5N	*	HPLaserJet5N,Local\$plonly	HPLaserJet5N
ImagesOS	*	C:\ImagesOS	
IPC\$	*		IPC distant
NETLOGON	*	C:\Windows\SYSTEM32\sysvol\DOMAINE2019.local\SCRIPTS	Partage de serveur d'accès
print\$	*	C:\Windows\system32\spool\drivers	Pilotes d'imprimantes
Public	*	C:\Public	
REPBASES	*	C:\REPBASES	
REPBASES2	*	C:\REPBASES2	
Société	*	C:\Société	
SYSVOL	*	C:\Windows\SYSTEM32\sysvol	Partage de serveur d'accès
test	*	C:\test	

Affichage de l'état de partage pour un nom de partage

[\\$myShareName="DossierClients"](#)

[Get-SmbShareAccess -Name "\\$myShareName"](#)

```
AccessControlType : Allow
AccessRight       : Full
AccountName      : Tout le monde
Name             : DossierClients
ScopeName        : *
PSComputerName   :
```

Modification de l'héritage des autorisations NTFS d'un répertoire (directory)

```
# Pour modifier les propriétés d'héritage d'un objet, il faut d'abord récupérer les règles ACL
# existantes du dossier ou du fichier avec Get-ACL, puis utiliser la méthode
# SetAccessRuleProtection(isProtected, preserveInheritance) :
#   Le booléen isProtected définit si le dossier hérite ou non des autorisations d'accès :
#   $true désactive l'héritage,
#   $false réactive l'héritage
#   Le booléen preserveInheritance permet de copier ou supprimer les autorisations héritées :
#   $true pour copier les autorisations héritées en autorisations explicites,
#   $false pour supprimer
# Enfin, on applique cet ensemble d'autorisations ACL au fichier ou au dossier existant avec Set-ACL

# Suppression de l'héritage d'un dossier avec
# copie des autorisations héritées en autorisations explicites

$myDirectoryFullName="C:\DossierClients"

if (Test-Path "$myDirectoryFullName")
{
    $ACL = Get-ACL -Path "$myDirectoryFullName"

    $ACL.SetAccessRuleProtection($true, $true)

    $ACL | Set-Acl -Path "$myDirectoryFullName"
}
```

Suppression de toutes les autorisations NTFS d'un groupe pour un répertoire (directory)

```
# Pour supprimer les autorisations d'un objet, il faut d'abord récupérer les règles ACL
# existantes du dossier ou du fichier avec Get-ACL, puis utiliser la méthode PurgeAccessRules :
# PurgeAccessRules ne fonctionne pas avec un nom d'utilisateur en chaîne, elle fonctionne
# uniquement avec des SID.
# Donc nous avons utilisé la classe « Ntaccount » pour convertir le nom de compte d'utilisateur de
# chaîne en SID.
# PurgeAccessRules ne fonctionne qu'avec des autorisations explicites (désactiver l'héritage avant
# si nécessaire.
# Enfin, on applique cet ensemble d'autorisations ACL au fichier ou au dossier existant avec Set-ACL

$myDirectoryFullName="C:\Test"
$myGroupOrUserLogin="Utilisateurs"

if (Test-Path "$myDirectoryFullName")
{
    $ACL = Get-ACL -Path "$myDirectoryFullName"

    $usersid = New-Object System.Security.Principal.Ntaccount ($myGroupOrUserLogin)

    $acl.PurgeAccessRules($usersid)

    $ACL | Set-Acl -Path "$myDirectoryFullName"
}
else
{
    Write-Host "Le dossier $myDirectoryFullName n'existe pas"
}
```


Ajout d'autorisations NTFS à un répertoire (directory)

Voir aussi [Changing NTFS Security Permissions using PowerShell - blue42](#) et [Set-Acl](#)

```
# Il faut d'abord récupérer les règles ACL existantes du dossier ou du fichier avec Get-ACL.  
# Pour ajouter une autorisation, il faut créer un nouveau FileSystemAccessRule avec la méthode  
# constructeur en spécifiant : Chaîne d'identité, FileSystemRights, AccessControlType  
# puis ajouter cette nouvelle autorisation ACL à l'ensemble des autorisations avec SetAccessRule  
# ou supprimer cette autorisation avec RemoveAccessRule  
# puis appliquer cet ensemble des autorisations ACL au fichier ou au dossier existant avec Set-ACL
```

```
# Ajout de l'autorisation Modifier à un utilisateur, pour un dossier
```

```
$myDirectoryFullName="C:\DossierClients"
```

```
$myGroupOrUserLogin="mdupont"
```

```
$myAuthorisations="Modify"
```

```
$myControlType="Allow"
```

```
if (Test-Path "$myDirectoryFullName")  
{  
    $ACL = Get-ACL -Path "$myDirectoryFullName"  
  
    $AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule  
"$myGroupOrUserLogin", "$myAuthorisations", "ContainerInherit, ObjectInherit", "None",  
"$myControlType"  
  
    $ACL.SetAccessRule($AccessRule)  
  
    $ACL | Set-Acl -Path "$myDirectoryFullName"  
}
```

Remarque : le descripteur de sécurité pour le partage d'un répertoire est obtenu avec la cmdlet [Get-Acl](#).

```
# Affichage du descripteur de sécurité avec Get-Acl
```

```
# Ce cmdlet permet d'afficher les 3 propriétés path, owner, et access list, ainsi que le groupe de  
sécurité du propriétaire, l'audit et le sddl (descripteur de sécurité sous forme d'une chaîne de  
caractères exprimé en format Security Descriptor Definition Language.)
```

```
$myDirectoryFullName="C:\DossierClients"
```

```
Get-Acl -Path "$myDirectoryFullName" |  
Format-List
```

```
Path : Microsoft.PowerShell.Core\FileSystem::C:\DossierClients  
Owner : BUILTIN\Administrateurs  
Group : DOMAINE2019\Utilisateurs du domaine  
Access : CREATEUR PROPRIETAIRE Allow 268435456  
         AUTORITE NT\Système Allow FullControl  
         BUILTIN\Administrateurs Allow FullControl  
         BUILTIN\Utilisateurs Allow CreateFiles, AppendData  
         BUILTIN\Utilisateurs Allow ReadAndExecute, Synchronize  
         DOMAINE2019\mdupont Allow Modify, Synchronize  
  
Audit :  
Sddl : O:BAG:DUD:PAI(A;OICIIO;GA;;;CO)(A;OICI;FA;;;SY)(A;OICI;FA;;;BA)(A;CI;DCLC;;  
;BU)(A;OICI;0x1200a9;;;BU)(A;OICI;0x1301bf;;;S-1-5-21-3086836198-3415409897  
-3203873499-1283)
```

Traitement des objets de l'Active Directory

Principaux objets et attributs d'objets dans l'Active Directory

Principales classes d'objets les plus utilisées

Classe (<i>ObjectClass</i>)	Description
Computer	ordinateurs intégrés au domaine : clients, serveurs membres, contrôleurs de domaine (il peut y en avoir plusieurs) mémorisant l'AD
User	utilisateurs qui peuvent s'authentifier sur le domaine, et accéder aux ressources du domaine
Group	groupes : permet de regrouper des objets au sein d'un groupe, notamment pour simplifier l'administration (exemple : un groupe d'utilisateurs contient des utilisateurs)
Container	conteneurs créés d'origine : Users , Computers , Builtin , ...
OrganizationalUnit	unités d'organisation arborescentes qui permettent d'organiser les objets (exemple : OU contenant des ordinateurs, OU contenant des utilisateurs)
PrintQueue	ressources de type « imprimante »

Active Directory intègre déjà des containers (**Users**, **Computers**, **Builtin**, etc) qui, à la différence des unités d'organisation natives, ne peuvent pas se voir appliquer des stratégies de groupe (GPO).

Les identifiants uniques : DistinguishedName et GUID

Chaque objet dispose d'identifiants uniques qui sont représentés par deux attributs : le *DistinguishedName* et le *GUID*.

A. Le DistinguishedName (DN)

Cet identifiant unique permet de décrire complètement un objet dans l'annuaire Active Directory ; il contient notamment le nom de l'objet, ainsi que son chemin d'accès complet dans la hiérarchie des conteneurs et unités d'organisation de l'AD.

Identification de l'élément	Description
CN	CommonName – Nom commun – Nom de l'objet final ciblé
OU	OrganizationalUnit – Unité d'organisation contenant l'objet
CN	ContaiNer – Conteneur –contenant l'objet : <i>Users</i> (utilisateurs), <i>Builtin</i> (groupes de sécurité), ...
DC	DomainComponent – Composant de domaine – Utilisé pour indiquer le domaine cible, avec un élément « dc » par partie du domaine

Exemple pour un utilisateur se trouvant dans une OU :

cn=Michel Dupont,ou=SalleR209,dc=DOMAINE2019,dc=LOCAL

Common Name Organizational Unit Domain Component

Exemple pour un utilisateur se trouvant dans le conteneur *Users* :

cn=Alice Nevers,cn=Users,dc=DOMAINE2019,dc=LOCAL

Common Name Container Domain Component

B. Le GUID

Le GUID (*Globally Unique Identifier*) est un identificateur global unique qui permet d'identifier un objet d'un annuaire Active Directory. Il correspond à l'attribut « **ObjectGUID** » dans le schéma Active Directory.

Il est attribué à l'objet dès sa création et ne change jamais, même si l'objet est déplacé ou modifié. Le GUID suit un objet de la création jusqu'à la suppression.

Codé sur 128 bits, le GUID d'un objet est **unique au sein d'une forêt** et il est généré par un algorithme qui **garantit son unicité**.

Les principaux attributs indispensables

Nom de l'attribut dans le schéma (<i>PropertyName</i>)	Description
<i>DistinguishedName</i>	Désignation complète d'un objet dans l'annuaire Active Directory, sous la forme CN=...,OU=...,DC= ,DC=...
<i>SamAccountName</i>	Login (nom d'ouverture de session) de l'utilisateur Nom que devra utiliser l'utilisateur pour s'authentifier sur le domaine
<i>UserPrincipalName</i>	Nom d'ouverture de session de l'utilisateur concaténé au nom du domaine sous la forme « @domaine.local » : nom complet de l'utilisateur avec le domaine inclus. Également appelé UPN
<i>ObjectClass</i>	Classe de l'objet : <i>user, group, computer, container, organizationalUnit, ...</i>
<i>Name</i>	Nom complet qui sera affiché pour cet utilisateur
<i>GivenName</i>	Prénom de l'utilisateur
<i>SurName</i>	Nom de l'utilisateur
<i>SID</i>	Identifiant de sécurité unique qui permet d'identifier un objet

```
PS C:\Scripts-PowerShell\Tests> Get-ADUser -Filter * -SearchBase "OU=SalleR209,DC=domaine2019,DC=local"

DistinguishedName : CN=Michel Dupont,OU=SalleR209,DC=DOMAINE2019,DC=local
Enabled           : True
GivenName        : Michel
Name             : Michel Dupont
ObjectClass      : user
ObjectGUID       : cc545ba4-7364-4536-875d-0228de3a60ce
SamAccountName   : mdupont
SID              : S-1-5-21-3086836198-3415409897-3203873499-1283
Surname          : Dupont
UserPrincipalName : mdupont@domaine2019.local
```

```
PS C:\Scripts-PowerShell\Tests> Get-ADObject -Filter * -SearchBase "CN=Computers,DC=domaine2019,DC=local"

DistinguishedName      Name      ObjectClass ObjectGUID
-----
CN=Computers,DC=DOMAINE2019,DC=local Computers container 53509ccf-b0b6-432c-bc53-056a379cfdc9
CN=DEBIAN,CN=Computers,DC=DOMAINE2019,DC=local DEBIAN computer 3d34630a-b296-4976-b40b-a1cf51835812
CN=SERVEUR2,CN=Computers,DC=DOMAINE2019,DC=local SERVEUR2 computer 9cf1e092-0c97-47ef-976d-dd5c281ccd16

PS C:\Scripts-PowerShell\Tests> Get-ADObject -Filter * -SearchBase "CN=Users,DC=domaine2019,DC=local"

DistinguishedName      Name
-----
CN=Users,DC=DOMAINE2019,DC=local Users
CN=Professeurs,CN=Users,DC=DOMAINE2019,DC=local Professeurs
CN=Dupois,CN=Users,DC=DOMAINE2019,DC=local Dupois
CN=krbtgt,CN=Users,DC=DOMAINE2019,DC=local krbtgt
CN=Ordinateurs du domaine,CN=Users,DC=DOMAINE2019,DC=local Ordinateurs du domaine
CN=Contrôleurs de domaine,CN=Users,DC=DOMAINE2019,DC=local Contrôleurs de domaine
CN=Administrateurs du schéma,CN=Users,DC=DOMAINE2019,DC=local Administrateurs du schéma
```

Remarques : les attributs DisplayName, FirstName, LastName n'existent pas.

Création d'une OU (Organizational Unit)(Unité d'Organisation)

Une OU est créée avec la cmdlet [Dsadd ou](#).

Création d'une OU dans une autre OU conteneur

```
$myDomainTLD="local"           # Top Level Domain
$myDomainSLD="domaine2019"     # Second Level Domain
$myOUConteneur="Salles Infos"

$myOU="SalleR212"

if (!(Dsquery ou -name $myOU "ou=$myOUConteneur,dc=$myDomainSLD,dc=$myDomainTLD"))
{
    dsadd ou "ou=$myOU,ou=$myOUConteneur,dc=$myDomainSLD,dc=$myDomainTLD"
}
else
{
    write-Host "Attention : l'OU $myOU existe déjà dans ou=$myOUConteneur,dc=$myDomainSLD,dc=$myDomainTLD"
}
```

Remarque : la liste des OU peut être obtenue avec la cmdlet [Get-ADOrganizationalUnit](#).

Liste de toutes les OU

[Get-ADOrganizationalUnit](#) -Filter * | [Format-Table](#) -AutoSize

City	Country	DistinguishedName	LinkedGroupPolicyObjects
		OU=Domain Controllers,DC=DOMAINE2019,DC=local	{CN={6AC1786C-016F-11D2-945F-00C04F8984F9},CN=Policies,CN=System,DC=DOMAINE2019,D...
		OU=Production,DC=DOMAINE2019,DC=local	{}
		OU=Utilisateurs,OU=Production,DC=DOMAINE2019,DC=local	{cn={24203286-7176-477A-B2DB-DC441A4F0070},cn=policies,cn=system,DC=DOMAINE2019,D...
		OU=PC,OU=Production,DC=DOMAINE2019,DC=local	{cn={7D5771C8-B87C-4A11-92AD-F3467F7B6E73},cn=policies,cn=system,DC=DOMAINE2019,D...
		OU=SIO,DC=DOMAINE2019,DC=local	{}
		OU=SalleR209,DC=DOMAINE2019,DC=local	{}

Création d'un groupe d'utilisateurs (group)

Un groupe est créé avec la cmdlet [Dsadd group](#).

Création d'un groupe de domaine local dans une OU

```
$myGroup="Etudiants"

$myDomainTLD="local"      # Top Level Domain
$myDomainSLD="domaine2019" # Second Level Domain
$myOU="Saller209"

if (!(Dsquery group -samid "$myGroup"))
{
    dsadd group -scope l "cn=$myGroup,ou=$myOU,dc=$myDomainSLD,dc=$myDomainTLD"
}
else
{
    write-Host "Attention : le groupe $myGroup existe déjà !"
}
```

Attention : si on crée un groupe dans le conteneur *Users*, il faut remplacer `ou=$myOU` par `cn=Users`

Le scope utilisé peut être l (domaine local), g (global), ou u (universal).

Remarque : la liste des groupes peut être obtenue avec la cmdlet [Get-ADGroup](#).

Liste de tous les groupes

```
Get-ADGroup -Filter * | Format-Table -AutoSize
```

DistinguishedName	GroupCategory	GroupScope	Name
CN=Utilisateurs DHCP,CN=Users,DC=DOMAINE2019,DC=local	Security	DomainLocal	Utilisateurs DHCP
CN=Administrateurs DHCP,CN=Users,DC=DOMAINE2019,DC=local	Security	DomainLocal	Administrateurs DHCP
CN=Administrateurs,CN=Builtin,DC=DOMAINE2019,DC=local	Security	DomainLocal	Administrateurs
CN=Utilisateurs,CN=Builtin,DC=DOMAINE2019,DC=local	Security	DomainLocal	Utilisateurs
CN=Invités,CN=Builtin,DC=DOMAINE2019,DC=local	Security	DomainLocal	Invités
CN=Opérateurs d'impression,CN=Builtin,DC=DOMAINE2019,DC=local	Security	DomainLocal	Opérateurs d'impression
CN=Opérateurs de sauvegarde,CN=Builtin,DC=DOMAINE2019,DC=local	Security	DomainLocal	Opérateurs de sauvegarde
CN=Etudiants,OU=Saller209,DC=DOMAINE2019,DC=local	Security	DomainLocal	Etudiants
CN=Professeurs,CN=Users,DC=DOMAINE2019,DC=local	Security	DomainLocal	Professeurs

Ajout d'un utilisateur à un groupe d'utilisateurs (group)

Un utilisateur est ajouté à un groupe avec la cmdlet [Add-ADGroupMember](#).

Ajout d'un utilisateur à un groupe

```
$myGroup="Juridique"
$myUserLogin="anevers"

$myMembersGroup=Get-ADGroupMember -Identity "$myGroup" | select -ExpandProperty SamAccountName
if ($myMembersGroup -contains $myUserLogin)
{
    write-Host "Attention : l'utilisateur $myUserLogin est déjà dans le groupe $myGroup !"
}
else
{
    Add-ADGroupMember -Identity "$myGroup" -Members "$myUserLogin"
}
```

Remarque : la liste des utilisateurs d'un groupe peut être obtenue avec la cmdlet [Get-ADGroupMember](#).

Liste des utilisateurs d'un groupe

```
$myGroup="Juridique"

write-Host "Liste de tous les utilisateurs du groupe $myGroup"
Get-ADGroupMember $myGroup | Format-Table -AutoSize
```

Liste de tous les utilisateurs du groupe Juridique						
distinguishedName	name	objectClass	objectGUID	SamAccountName	SID	
CN=Charles Dupont,CN=Users,DC=DOMAINE2019,DC=local	Charles Dupont	user	4939e88b-8756-4c22-8041-608386a8af82	cdupont	S-1-5-21-308683619...	
CN=Alice Nevers,CN=Users,DC=DOMAINE2019,DC=local	Alice Nevers	user	a3d0fc31-bb75-407d-aeb4-d4eede062e14	anevers	S-1-5-21-308683619...	

Création d'un utilisateur (user)

Un utilisateur est créé avec la cmdlet [Dsadd user](#).

Création d'un utilisateur dans une OU, qui sera membre d'un groupe, et qui aura un dossier personnel ; Attention : la création du dossier personnel n'est pas faite dans ce script

```
$myDomainTLD="local"      # Top Level Domain
$myDomainSLD="domaine2019" # Second Level Domain
```

```
$myOU="SalleR209"
$myGroup="Etudiants"
```

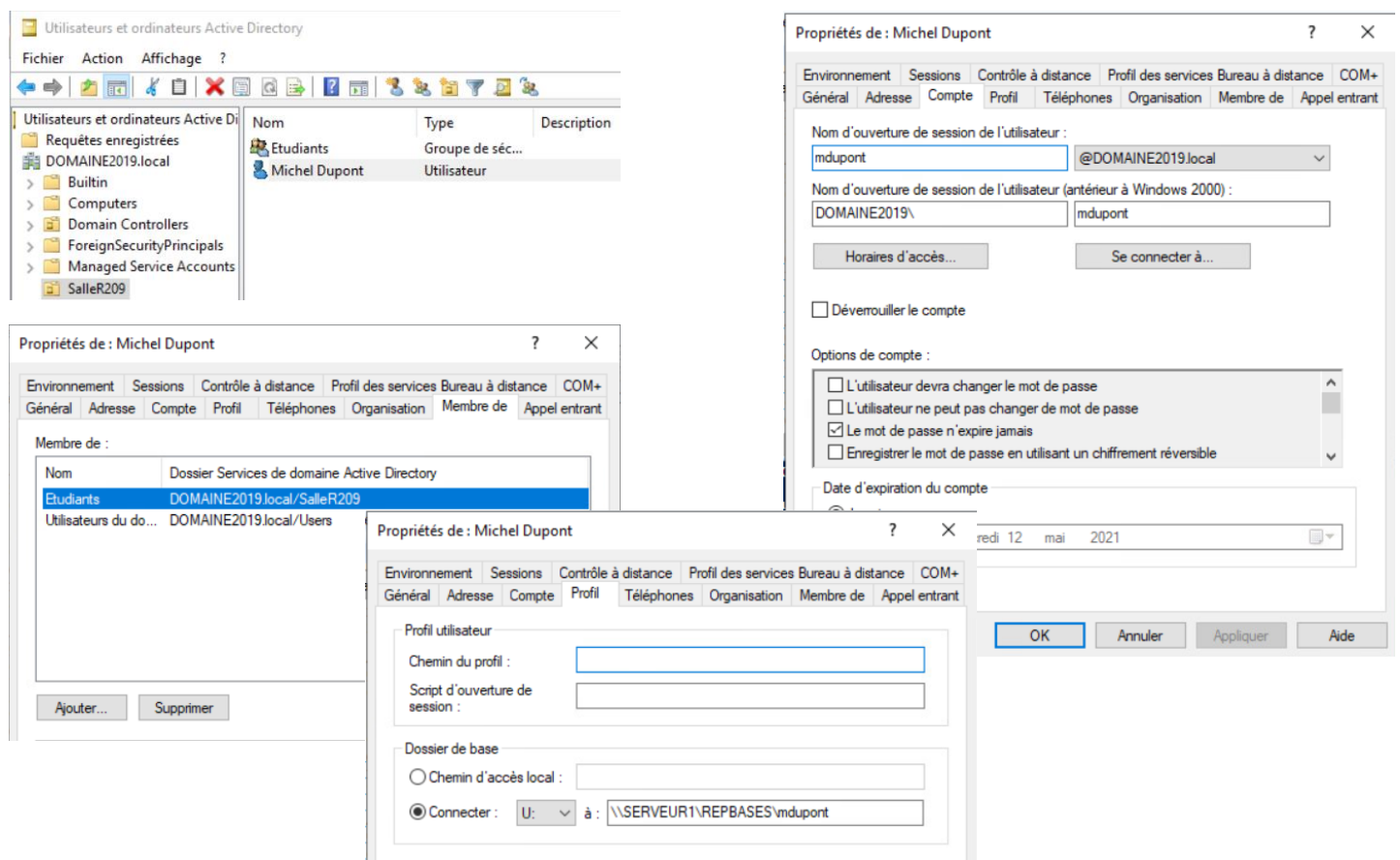
```
$myUserFirstName="Michel"
$myUserLastName="Dupont"
$myUserLogin="mdupont"
$myPassword="secret1A!"
```

```
$myDirectoryPath="\\SERVER1\REPBASES\$myUserLogin"
```

```
if (!(Dsquery user -samid "$myUserLogin"))
{
    dsadd user cn="$myUserFirstName $myUserLastName,ou=$myOU,dc=$myDomainSLD,dc=$myDomainTLD" `
    -samid "$myUserLogin" `
    -upn "$myUserLogin@$myDomainSLD.$myDomainTLD" `
    -fn "$myUserFirstName" `
    -ln "$myUserLastName" `
    -display "$myUserFirstName $myUserLastName" `
    -pwd "$myPassword" `
    -mustchpwd no `
    -pwdneverexpires yes `
    -disabled no `
    -memberof "cn=$myGroup,ou=$myOU,dc=$myDomainSLD,dc=$myDomainTLD" `
    -hmdrv U: `
    -hmdir "$myDirectoryPath"
}
else
{
    write-Host " Attention : l'utilisateur $myUserFirstName $myUserLastName existe déjà !"
}
```

Attention : si on crée un utilisateur dans le conteneur *Users*, il faut remplacer `ou=$myOU` par `cn=Users`

Attention : la création du dossier personnel n'est pas faite dans ce script



Remarque : la liste des utilisateurs peut être obtenue avec la cmdlet [Get-ADUser](#).

Liste de tous les utilisateurs

Get-ADUser -Filter * | Format-Table -AutoSize

DistinguishedName	Enabled	GivenName	Name	ObjectClass	ObjectGUID
CN=Administrateur,CN=Users,DC=DOMAINE2019,DC=local	True		Administrateur	user	5895fe3d-f8a7-4ab6-90a9-b4f7d17...
CN=Invité,CN=Users,DC=DOMAINE2019,DC=local	False		Invité	user	0e1a36e4-18a3-4eb4-96ae-a9b5e0b...
CN=krbtgt,CN=Users,DC=DOMAINE2019,DC=local	False		krbtgt	user	7dca5905-ed24-453e-b4ed-1c3511d...
CN=User_VPN_LDAP,CN=Users,DC=DOMAINE2019,DC=local	True		User_VPN_LDAP	user	689b578b-f128-4a7a-9470-9cce2b3...
CN=Alice Nevers,CN=Users,DC=DOMAINE2019,DC=local	True	Alice	Alice Nevers	user	a3d0fc31-bb75-407d-aeb4-d4eede0...
CN=Martin Lebrun,CN=Users,DC=DOMAINE2019,DC=local	True	Martin	Martin Lebrun	user	2eae2b6d-7667-4921-9aa8-c7c341f...
CN=Charles Dupont,CN=Users,DC=DOMAINE2019,DC=local	True	Charles	Charles Dupont	user	4939e88b-8756-4c22-8041-608386a...
CN=Albert Dubois,CN=Users,DC=DOMAINE2019,DC=local	True	Albert	Albert Dubois	user	e2a08650-ee36-4539-a9dc-294fd98...

Parcours d'une collection d'objets Active Directory

Exemples de collections que l'on peut parcourir :

- liste des OU d'un domaine, avec la cmdlet [Get-ADOrganizationalUnit](#)
- liste des groupes d'utilisateurs d'un domaine, avec la cmdlet [Get-ADGroup](#)
- liste des utilisateurs d'un groupe, avec la cmdlet [Get-ADGroupMember](#)
- liste des utilisateurs d'un domaine, avec la cmdlet [Get-ADUser](#)

Liste des utilisateurs avec Get-ADUser

```
$myDomainTLD="local"          # Top Level Domain
$myDomainSLD="domaine2019"    # Second Level Domain

Write-Host "Liste des utilisateurs"

$myUsers= Get-ADUser -Filter * -SearchBase "DC=$myDomainSLD,DC=$myDomainTLD"

foreach ($aUser in $myUsers)
{
    Write-Host $aUser.Name, $aUser.SamAccountName, $aUser.DistinguishedName
}
```

```
Liste des utilisateurs
Administrateur Administrateur CN=Administrateur,CN=Users,DC=DOMAINE2019,DC=local
Invité Invité CN=Invité,CN=Users,DC=DOMAINE2019,DC=local
krbtgt krbtgt CN=krbtgt,CN=Users,DC=DOMAINE2019,DC=local
User_VPN_LDAP User_VPN_LDAP CN=User_VPN_LDAP,CN=Users,DC=DOMAINE2019,DC=local
Alice Nevers anevers CN=Alice Nevers,CN=Users,DC=DOMAINE2019,DC=local
Martin Lebrun mlebrun CN=Martin Lebrun,CN=Users,DC=DOMAINE2019,DC=local
```

Test de l'appartenance d'un utilisateur à un groupe avec Get-ADGroupMember

ce test peut être effectué en fonction du login de l'utilisateur (samaccountname), son nom (name),
ou son distinguishedname

```
$myUserLogin="anevers"
$myGroupName="Juridique"
if ((Get-ADGroupMember -Identity $myGroupName | select -ExpandProperty samaccountname) -contains
$myUserLogin)
{
    Write-Host "trouvé avec samaccountname", $myUserLogin, $myGroupName
}
else
{
    Write-Host "non trouvé avec samaccountname", $myUserLogin, $myGroupName
}

$myUserName="Alice Nevers"
$myGroupName="Juridique"
if ((Get-ADGroupMember -Identity $myGroupName | select -ExpandProperty name) -contains $myUserName)
{
    Write-Host "trouvé avec name", $myUserName, $myGroupName
}
else
{
    Write-Host "non trouvé avec name", $myUserName, $myGroupName
}

$myUserDN="cn=Alice Nevers,cn=Users,dc=domaine2019,dc=local"
# Attention : "cn=anevers,cn=Users,dc=domaine2019,dc=local" n'est pas un distinguishedname correct
$myGroupName="Juridique"
if ((Get-ADGroupMember -Identity $myGroupName | select -ExpandProperty distinguishedname) -contains
$myUserDN)
{
    Write-Host "trouvé avec distinguishedname", $myUserDN, $myGroupName
}
else
{
    Write-Host "non trouvé avec distinguishedname", $myUserDN, $myGroupName
}
```

```
trouvé avec samaccountname anevers Juridique
trouvé avec name Alice Nevers Juridique
trouvé avec distinguishedname cn=Alice Nevers,cn=Users,dc=domaine2019,dc=local Juridique
```


On peut aussi parcourir les éléments de n'importe quelle collection d'objets Active Directory (sites, OU, ...), avec la cmdlet [Get-ADObject](#).

```
# Liste des éléments contenus dans un conteneur, une OU, ... avec Get-ADObject

$myDomainTLD="local"           # Top Level Domain
$myDomainSLD="domaine2019"    # Second Level Domain

$myContainer="Computers"

write-Host "Contenu de $myContainer"

$myObjects = Get-ADObject -Filter * -SearchBase "CN=$myContainer,DC=$myDomainSLD,DC=$myDomainTLD"

foreach ($AnObject in $myObjects)
{
    write-Host $AnObject.name, $AnObject.DistinguishedName, $AnObject.ObjectClass
}
```

```
Computers container
DEBIAN computer
SERVEUR14 computer
PC2 computer
SERVEUR2 computer
```