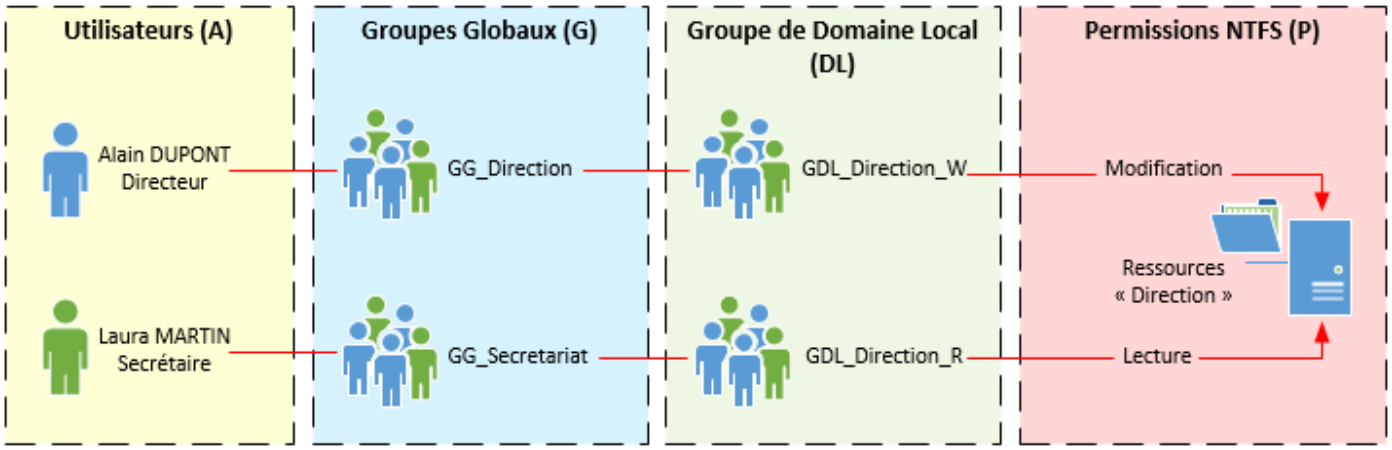
Active Directory

AGDLP

AGDLP

Comprendre la gestion des droits dans Active Directory



Introduction

Aujourd'hui, dans les systèmes informatiques, bien gérer qui a accès à quoi est super important pour éviter les erreurs et garantir la sécurité. Microsoft propose une méthode appelée **AGDLP** pour organiser les droits d'accès dans **Active Directory** (AD). Cette méthode permet de séparer les utilisateurs des droits qu'ils ont sur les fichiers ou dossiers, tout en gardant une structure claire et facile à gérer.

AGDLP, c'est un acronyme :

- **A** pour *Account* (les utilisateurs),
- **G** pour *Global Group* (groupes globaux),
- **DL** pour *Domain Local Group* (groupes locaux de domaine),
- **P** pour *Permissions* (les droits d'accès).

Chaque élément a un rôle bien défini dans la gestion des accès.

Origine et contexte

Avant AGDLP, on donnait souvent les droits directement aux utilisateurs, ce qui rendait les choses compliquées à maintenir et risquait de créer des erreurs. AGDLP propose une méthode plus propre : on donne les droits aux groupes, pas aux personnes directement.

Ce système s'inspire du modèle **RBAC** (*Role-Based Access Control*), où les rôles (et non les individus) déterminent les accès.

Objectifs d'AGDLP

AGDLP a plusieurs buts :

- **Centraliser** la gestion des droits pour éviter les erreurs.
- **Séparer** les utilisateurs des permissions : un utilisateur n'a jamais de droits directs, il les obtient via les groupes.
- **Faciliter l'ajout ou le départ d'un utilisateur** : on le met dans le bon groupe, et il a automatiquement les bons accès.
- **Rendre les accès plus clairs et traçables** : on peut facilement voir qui a accès à quoi en regardant les groupes.

Comment ça fonctionne

Voici comment AGDLP est structuré :

1. Les **utilisateurs** (A) sont placés dans des **groupes globaux** (G) selon leur service ou rôle.
2. Ces groupes globaux sont ensuite ajoutés à des **groupes locaux de domaine** (DL).
3. Les groupes locaux reçoivent les **permissions** (P) sur les ressources (fichiers, dossiers, imprimantes...).

Exemple :
Alice travaille dans les RH. Elle est dans le groupe global **GG_Paie_2025**. Ce groupe est ajouté au groupe local **GDL_01-01-2025_LECTURE**, qui a le droit de lire les fichiers du dossier `C:\Partage\RH\Paie\2025`.
Si quelqu'un doit modifier les fichiers, il sera dans un autre groupe local, comme **GDL_01-01-2025_ECRITURE**, qui a les droits d'écriture.

Chaque dossier peut avoir un groupe pour la lecture et un autre pour l'écriture, ce qui permet de bien contrôler les accès.

Avantages

AGDLP a plein de points positifs :

- Une **organisation claire** des droits.
- Une **meilleure sécurité**, car les droits sont donnés aux groupes, pas aux individus.
- Une **gestion plus simple** : ajouter un utilisateur ou un dossier ne demande pas de tout reconfigurer.
- Une **bonne traçabilité** : on sait facilement qui a accès à quoi.
- Les droits peuvent **se propager automatiquement** aux sous-dossiers, ce qui fait gagner du temps.

Limites

Mais AGDLP a aussi quelques inconvénients :

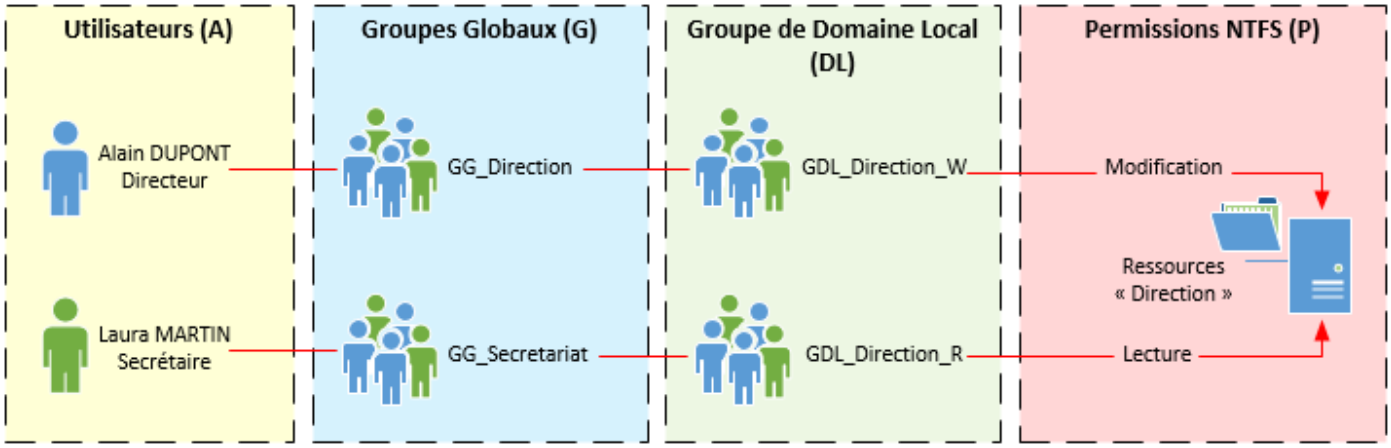
- Il faut **bien planifier** au début, sinon ça peut devenir compliqué.
- Pour ceux qui ne connaissent pas bien le système, **comprendre les droits peut être difficile**.
- Dans les grandes entreprises, **les changements peuvent mettre du temps à se propager**.
- Une mauvaise organisation au départ peut rendre la suite plus difficile.

Conclusion

AGDLP est une méthode efficace pour gérer les accès dans Active Directory. Elle sépare clairement les utilisateurs, les groupes et les droits, ce qui rend le système plus sécurisé et plus facile à gérer.

Même si sa mise en place demande un peu de travail, les bénéfices à long terme sont importants. Pour toute entreprise qui veut une gestion des accès solide et évolutive, AGDLP est une très bonne solution.

Création de l'arborescence



Introduction

Comme vu plus tôt dans ce chapitre, il est très long et complexe de mettre en place l'AGDLP car il faut des GDL (Groupe de Domaine Local)
C'est pourquoi j'ai mis en place une série de script, qui permet d'automatiser la création des dossiers, si on part de zéro, mais aussi la création des GDL, la parentalité des GDL, ainsi que l'application des droits sur les répertoires !

Structure et nomenclature

Commençons par définir la structure de notre arborescence dans un petit fichier csv :
Il y aura plusieurs colonnes :

- **NIVEAU1-8** représentent les différents niveau de répertoires gérés par le SI de l'organisation
- **FolderPath** sera généré par un script, ce sera la concaténation de NIVEAU1-8 pour former un chemin absolu
- **CREER_GROUPE** permet de définir si on à besoin de GDL pour ce répertoire, si c'est le niveau final souhaité ici
- **GG_ECRITURE** on liste les groupes de personnes qui auront un accès en écriture à partir de ce répertoire, et tous les enfants
- **GG_LECTURE** on liste les groupes de personnes qui auront un accès en lecture à partir de ce répertoire, et tous les enfants

Voici le tableau CSV :

NIVEAU 1	NIVEAU 2	NIVEAU 3	NIVEAU 4	NIVEAU 5	NIVEAU 6	NIVEAU 7	NIVEAU 8	FolderPath	CREER_GROU	GG_ECRITURE	GG_LECTURE
01-DIRECTION										ORG_GG_DIRECTION	
02-FINANCE										ORG_GG_FINANCE	ORG_GG_DIRECTION
02-FINANCE	01-BUDGET									ORG_GG_FINANCE	ORG_GG_DIRECTION
02-FINANCE	02-FACTURES									ORG_GG_FINANCE	ORG_GG_DIRECTION
03-RH										ORG_GG_RH	ORG_GG_DIRECTION
03-RH	01-CONTRATS									ORG_GG_RH	ORG_GG_DIRECTION
03-RH	02-PAIE								NON		
04-IT										ORG_GG_IT	ORG_GG_DIRECTION
04-IT	01-INFRA									ORG_GG_IT	ORG_GG_DIRECTION
04-IT	02-PROJETS									ORG_GG_IT	ORG_GG_DIRECTION
04-IT	02-PROJETS	2026							NON		

```
NIVEAU 1;NIVEAU 2;NIVEAU 3;NIVEAU 4;NIVEAU 5;NIVEAU 6;NIVEAU 7;NIVEAU
8;FolderPath;CREER_GROUPES;GG_ECRITURE;GG_LECTURE
01-DIRECTION;;;;;;;;;ORG_GG_DIRECTION;
02-FINANCE;;;;;;;;;ORG_GG_FINANCE;ORG_GG_DIRECTION
02-FINANCE;01-BUDGET;;;;;;;;ORG_GG_FINANCE;ORG_GG_DIRECTION
02-FINANCE;02-FACTURES;;;;;;;;ORG_GG_FINANCE;ORG_GG_DIRECTION
03-RH;;;;;;;;;ORG_GG_RH;ORG_GG_DIRECTION
03-RH;01-CONTRATS;;;;;;;;ORG_GG_RH;ORG_GG_DIRECTION
03-RH;02-PAIE;;;;;;;NON;;
04-IT;;;;;;;;;ORG_GG_IT;ORG_GG_DIRECTION
04-IT;01-INFRA;;;;;;;;ORG_GG_IT;ORG_GG_DIRECTION
04-IT;02-PROJETS;;;;;;;;ORG_GG_IT;ORG_GG_DIRECTION
04-IT;02-PROJETS;2026;;;;;;;NON;;
```

Avec cette structure de fichier, et de dossier, il sera plus simple de procéder pour la suite.

J'ai fait le choix, pour des questions de simplicité, de numéroter l'ensemble des répertoire, ce qui permet de choisir l'ordre de ces derniers, et rendra bien plus lisible les GDL, et leur gestion automatisée. Nous verrons cela dans les pages suivantes.

Il faut bien penser à créer une ligne par répertoire, car chaque ligne représente ces derniers, avec leur permissions qui leurs sont propres.

Création des dossiers

La structure du fichier, avec des chemins absolus, permet de faciliter l'écriture du script.
En effet, chaque ligne représentant un chemin absolu donc chaque nom de répertoire est séparé des autres par un "\".
Il suffit alors de créer les dossiers en lisant le fichier comme cela.

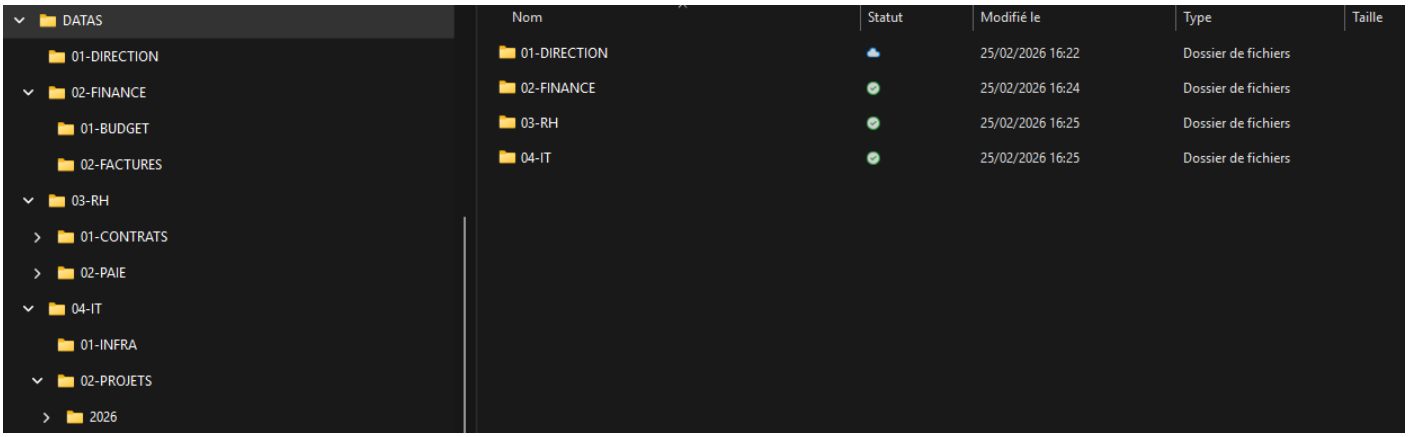
```
<#
================================================================================
SCRIPT : 1.1 - Creation_Arborescence.ps1
OBJET  : Creation de l arborescence de dossiers a partir du CSV source
================================================================================

DESCRIPTION :
  - Ce script lit le fichier CSV contenant les colonnes NIVEAU 1 .. NIVEAU 8
  - Construit un chemin "FolderPath" en concatenant les niveaux non vides
  - Remplace les caracteres interdits Windows et force les noms en majuscules
  - Cree les dossiers physiquement dans le partage cible
  - Met a jour la colonne FolderPath dans le CSV

Toutes les chaines visibles sont sans accents pour compatibilite maximum.
Les commentaires interne sont en francais normal.
================================================================================
#>

Param(
    [string]$CsvPath    = "C:\AGDLP\ORG_ARBO_GDL.csv",
    [string]$Destination = "\\SRV-DATAS\datas\"
)

$ErrorActionPreference = "Stop"

# ------------------------------------------------------------------------------
# 1. Verification des pre-requis
# ------------------------------------------------------------------------------

if (-not (Test-Path $CsvPath)) {
    Write-Error "Le fichier CSV est introuvable : $CsvPath"
    exit 1
}

if (-not (Test-Path $Destination)) {
    Write-Error "Le dossier de destination est introuvable : $Destination"
    exit 1
}

# ------------------------------------------------------------------------------
```

```
# 2. Lecture du CSV
# ------------------------------------------------------------------------

$rows  = Import-Csv -Path $CsvPath -Delimiter ";" -Encoding UTF8
$total = $rows.Count
$index = 0

Write-Host "Creation des dossiers a partir du fichier : $CsvPath"
Write-Host "Dossier cible : $Destination"
Write-Host ""

# ------------------------------------------------------------------------
# 3. Boucle principale
# ------------------------------------------------------------------------

foreach ($row in $rows) {
    $index++

    # Mise a jour de la barre de progression
    Write-Progress `
        -Activity "Creation des dossiers" `
        -Status "$index / $total" `
        -PercentComplete (($index / $total) * 100)

    # Construction du chemin logique a partir des colonnes NIVEAU X
    $niveaux = @()
    for ($i = 1; $i -le 8; $i++) {
        $col = "NIVEAU $i"
        if ($row.PSObject.Properties.Name -contains $col) {
            $val = $row.$col
            if ($val -and $val.Trim() -ne "") {
                $niveaux += $val.Trim()
            }
        }
    }

    # Si la ligne ne contient aucun niveau, on passe
    if ($niveaux.Count -eq 0) { continue }

    # Nettoyage du chemin : caracteres interdits + majuscules
```

```
    $raw       = $niveaux -join "\"
    $parts     = $raw.Split("\")
    $safeParts = foreach ($p in $parts) {
        ($p -replace '[:*?"<>|]', "_").ToUpper()
    }

    $folderPath = $safeParts -join "\"

    # Ajout ou mise a jour de FolderPath dans la ligne CSV
    if ($row.PSObject.Properties.Name -contains "FolderPath") {
        $row.FolderPath = $folderPath
    }
    else {
        $row | Add-Member -Name FolderPath -MemberType NoteProperty -Value $folderPath -Force
    }

    # Creation physique des dossiers
    $current = $Destination
    foreach ($sp in $safeParts) {
        $current = Join-Path $current $sp

        if (-not (Test-Path -LiteralPath $current)) {
            try {
                New-Item -Path $current -ItemType Directory -Force | Out-Null
                Write-Host "Cree :    $current"
            }
            catch {
                Write-Warning "Erreur lors de la creation : $current -> $_"
            }
        }
        else {
            Write-Host "Existe :  $current"
        }
    }
}

# Effacer la barre de progression
Write-Progress -Activity "Creation des dossiers" -Completed

# ------------------------------------------------------------------------
```

```
# 4. Sauvegarde du CSV mis a jour
# ------------------------------------------------------------------------

$utf8 = New-Object System.Text.UTF8Encoding($true)
$content = $rows | ConvertTo-Csv -Delimiter ";" -NoTypeInformation
[System.IO.File]::WriteAllLines($CsvPath, $content, $utf8)

Write-Host ""
Write-Host "CSV mis a jour : $CsvPath" -ForegroundColor Green
Write-Host "Creation arborescence terminee."
```

Avec cela, nous avons une arborescence toute neuve et propre (presque, celle-ci n'est évidemment la que pour le test)



Apres

Voici à quoi ressemble le CSV après le premier script :

NIVEAU 1	NIVEAU 2	NIVEAU 3	NIVEAU 4	NIVEAU 5	NIVEAU 6	NIVEAU 7	NIVEAU 8	FolderPath	CREER_GROUPES	GG_ECRITURE	GG_LECTURE
01-DIRECTION								01-DIRECTION		ORG_GG_DIRECTION	
02-FINANCE								02-FINANCE		ORG_GG_FINANCE	ORG_GG_DIRECTION
02-FINANCE	01-BUDGET							02-FINANCE\01-BUDGET		ORG_GG_FINANCE	ORG_GG_DIRECTION
02-FINANCE	02-FACTURES							02-FINANCE\02-FACTURES		ORG_GG_FINANCE	ORG_GG_DIRECTION
03-RH								03-RH		ORG_GG_RH	ORG_GG_DIRECTION
03-RH	01-CONTRATS							03-RH\01-CONTRATS		ORG_GG_RH	ORG_GG_DIRECTION
03-RH	02-PAIE							03-RH	NON		
04-IT								04-IT		ORG_GG_IT	ORG_GG_DIRECTION
04-IT	01-INFRA							04-IT\01-INFRA		ORG_GG_IT	ORG_GG_DIRECTION
04-IT	02-PROJETS							04-IT\02-PROJETS		ORG_GG_IT	ORG_GG_DIRECTION
04-IT	02-PROJETS	2026						04-IT\02-PROJETS\2026	NON		

```
NIVEAU 1;NIVEAU 2;NIVEAU 3;NIVEAU 4;NIVEAU 5;NIVEAU 6;NIVEAU 7;NIVEAU
8;FolderPath;CREER_GROUPES;GG_ECRITURE;GG_LECTURE
01-DIRECTION;;;;;;;;01-DIRECTION;;ORG_GG_DIRECTION;
02-FINANCE;;;;;;;;02-FINANCE;;ORG_GG_FINANCE;ORG_GG_DIRECTION
02-FINANCE;01-BUDGET;;;;;;;02-FINANCE\01-BUDGET;;ORG_GG_FINANCE;ORG_GG_DIRECTION
```

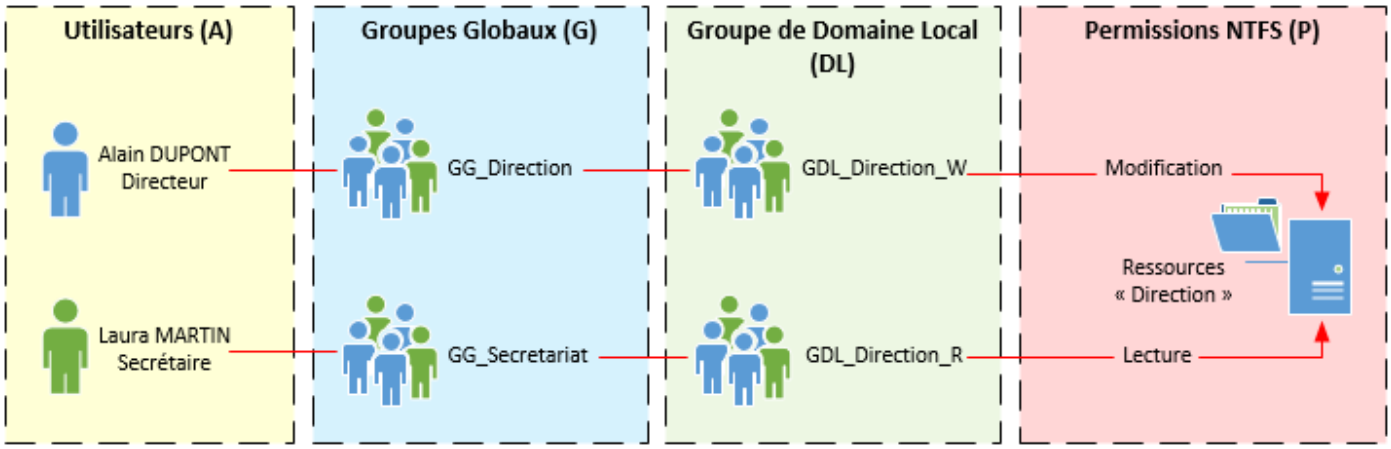
```
02-FINANCE;02-FACTURES;;;;;;;02-FINANCE\02-FACTURES;;ORG_GG_FINANCE;ORG_GG_DIRECTION
03-RH;;;;;;;;03-RH;;ORG_GG_RH;ORG_GG_DIRECTION
03-RH;01-CONTRATS;;;;;;;03-RH\01-CONTRATS;;ORG_GG_RH;ORG_GG_DIRECTION
03-RH;02-PAIE;;;;;;;03-RH;NON;;
04-IT;;;;;;;;04-IT;;ORG_GG_IT;ORG_GG_DIRECTION
04-IT;01-INFRA;;;;;;;04-IT\01-INFRA;;ORG_GG_IT;ORG_GG_DIRECTION
04-IT;02-PROJETS;;;;;;;04-IT\02-PROJETS;;ORG_GG_IT;ORG_GG_DIRECTION
04-IT;02-PROJETS;2026;;;;;;04-IT\02-PROJETS\2026;NON;;
```

La colonne FolderPath est remplie.

Passons à la suite, avec les GDL !

AGDLP

Création des GDL



Introduction

On le redit encore, mais le plus long pour l'AGDLP, c'est la mise en place, d'abord des dossiers, et ensuite des groupes GDL, ce que nous allons voir maintenant !

Reprenons notre fichier modèle :

NIVEAU 1	NIVEAU 2	NIVEAU 3	NIVEAU 4	NIVEAU 5	NIVEAU 6	NIVEAU 7	NIVEAU 8	FolderPath	CREER_GROUPES	GG_ECRITURE	GG_LECTURE
01-DIRECTION								01-DIRECTION		ORG_GG_DIRECTION	
02-FINANCE								02-FINANCE		ORG_GG_FINANCE	ORG_GG_DIRECTION
02-FINANCE	01-BUDGET							02-FINANCE\01-BUDGET		ORG_GG_FINANCE	ORG_GG_DIRECTION
02-FINANCE	02-FACTURES							02-FINANCE\02-FACTURES		ORG_GG_FINANCE	ORG_GG_DIRECTION
03-RH								03-RH		ORG_GG_RH	ORG_GG_DIRECTION
03-RH	01-CONTRATS							03-RH\01-CONTRATS		ORG_GG_RH	ORG_GG_DIRECTION
03-RH	02-PAIE							03-RH	NON		
04-IT								04-IT		ORG_GG_IT	ORG_GG_DIRECTION
04-IT	01-INFRA							04-IT\01-INFRA		ORG_GG_IT	ORG_GG_DIRECTION
04-IT	02-PROJETS							04-IT\02-PROJETS		ORG_GG_IT	ORG_GG_DIRECTION
04-IT	02-PROJETS	2026						04-IT\02-PROJETS\2026	NON		

Et en vue CSV :

```
NIVEAU 1;NIVEAU 2;NIVEAU 3;NIVEAU 4;NIVEAU 5;NIVEAU 6;NIVEAU 7;NIVEAU
8;FolderPath;CREER_GROUPES;GG_ECRITURE;GG_LECTURE
01-DIRECTION;;;;;;;;01-DIRECTION;;ORG_GG_DIRECTION;
02-FINANCE;;;;;;;;02-FINANCE;;ORG_GG_FINANCE;ORG_GG_DIRECTION
02-FINANCE;01-BUDGET;;;;;;;02-FINANCE\01-BUDGET;;ORG_GG_FINANCE;ORG_GG_DIRECTION
02-FINANCE;02-FACTURES;;;;;;;02-FINANCE\02-FACTURES;;ORG_GG_FINANCE;ORG_GG_DIRECTION
03-RH;;;;;;;;03-RH;;ORG_GG_RH;ORG_GG_DIRECTION
03-RH;01-CONTRATS;;;;;;;03-RH\01-CONTRATS;;ORG_GG_RH;ORG_GG_DIRECTION
```

```
03-RH;02-PAIE;;;;;;;03-RH;NON;;
04-IT;;;;;;;;04-IT;;ORG_GG_IT;ORG_GG_DIRECTION
04-IT;01-INFRA;;;;;;;04-IT\01-INFRA;;ORG_GG_IT;ORG_GG_DIRECTION
04-IT;02-PROJETS;;;;;;;04-IT\02-PROJETS;;ORG_GG_IT;ORG_GG_DIRECTION
04-IT;02-PROJETS;2026;;;;;;04-IT\02-PROJETS\2026;NON;;
```

Pour bien faire, il faut créer trois GDL par répertoire, un pour afficher le contenu seul, un lecture seule récursif, et un lecture/écriture récursif.

C'est donc ce qu'on va faire, automatiquement, suivant ce nommage : GDL_<numérotation-répertoire>_VUE, GDL_<numérotation-répertoire>_LECTURE et GDL_<numérotation-répertoire>_ECRITURE.

De plus, nous allons préparer la parentalité des GDL, c'est à dire que chaque enfant GDL_ECRITURE et GDL_LECTURE soit membre de son parent GDL_VUE direct, et nous allons également les disposer dans un miroir de l'arborescence dans l'AD.

Cela permet une gestion plus simple car, si nous voulons ajouter des droits sur un dossier N-15 sous dossiers, il est plus simple de naviguer directement dans l'arborescence miroir de l'AD et ainsi ajouté le bon GG à la GDL finale.
Grâce à la parentalité, les droits seront automatiquement les bons.

OU et GDL

Voici donc le script qui nous permet de faire tout cela :

```
<#
================================================================================
SCRIPT : 2.1 - Creation_GDL.ps1
OBJET  : Creation des OU et des GDL (ECRITURE / LECTURE / VUE) a partir du CSV
================================================================================

DESCRIPTION :
  - Pour chaque ligne du CSV ayant un FolderPath et CREER_GROUPES <> "NON" :
      * Construit le "code numerique" a partir des segments du FolderPath
        (ex : "02-03-05" a partir de "02-ADMIN_G\03-PARTAGE\05-...").
      * Cree l arborescence d OU sous l OU racine $BaseOU.
      * Cree 3 groupes AD (Global/Security) :
            GDL_<code>_ECRITURE
            GDL_<code>_LECTURE
            GDL_<code>_VUE
```

```
      * Calcule le GDL parent "VUE" (Parent_GDL) en fonction du dossier parent.
      * Met a jour le CSV : GDL_ECRITURE, GDL_LECTURE, GDL_VUE, Parent_GDL.

================================================================================
#>

Param(
    [string]$CsvPath = "C:\AGDLP\ORG_ARBO_GDL.csv",
    [string]$BaseOU  = "OU=GDL,OU=AGDLP,DC=domain,DC=local"
)

$ErrorActionPreference = "Stop"

# ------------------------------------------------------------------------------
# 0. Module ActiveDirectory
# ------------------------------------------------------------------------------
try {
    Import-Module ActiveDirectory -ErrorAction Stop
}
catch {
    Write-Error "Module ActiveDirectory introuvable. Installer RSAT / AD PowerShell."
    exit 1
}

# ------------------------------------------------------------------------------
# 1. Fonctions utilitaires
# ------------------------------------------------------------------------------

function Get-NumericCodeFromFolderPath {
    <#
      Extrait pour chaque segment du FolderPath le prefixe numerique (^d+),
      puis joint par un tiret. Exemple :
        "02-ADMIN_G\03-PARTAGE\01-TRUC" -> "02-03-01"
    #>
    param([string]$FolderPath)

    if ([string]::IsNullOrWhiteSpace($FolderPath)) { return $null }

    $segments = $FolderPath -split '\\' | Where-Object { $_ -and $_.Trim() -ne '' }
    $nums = foreach ($s in $segments) {
```

```
        if ($s -match '^\d+') { $Matches[0] }
    }
    if ($nums.Count -eq 0) { return $null }
    return ($nums -join '-')
}

function Ensure-OU {
    <#
      Cree une OU si elle n existe pas. Retourne le DN de l OU.
    #>
    param(
        [string]$Name,
        [string]$ParentDN
    )

    $ouDN = "OU=$Name,$ParentDN"
    $existing = Get-ADOrganizationalUnit -LDAPFilter "(distinguishedName=$ouDN)" -ErrorAction
SilentlyContinue
    if ($existing) {
        Write-Host "  OU existe deja : $ouDN"
        return $ouDN
    }

    New-ADOrganizationalUnit -Name $Name -Path $ParentDN -ProtectedFromAccidentalDeletion
$false | Out-Null
    Write-Host "  OU creee : $ouDN"
    return $ouDN
}

function Ensure-Group {
    <#
      Cree un groupe AD Global/Security si absent.
    #>
    param(
        [string]$Name,
        [string]$OU
    )

    $existing = Get-ADGroup -Filter "Name -eq '$Name'" -ErrorAction SilentlyContinue
    if ($existing) {
```

```
        Write-Host "  GDL existe deja : $Name"
        return
    }

    New-ADGroup `
        -Name $Name `
        -SamAccountName $Name `
        -GroupScope Global `
        -GroupCategory Security `
        -Path $OU | Out-Null

    Write-Host "  GDL cree : $Name"
}

# ------------------------------------------------------------------------
# 2. Lecture du CSV
# ------------------------------------------------------------------------
if (-not (Test-Path $CsvPath)) {
    Write-Error "CSV introuvable : $CsvPath"
    exit 1
}

$rows  = Import-Csv -Path $CsvPath -Delimiter ";" -Encoding UTF8
$total = $rows.Count
$index = 0

Write-Host "Creation des GDL (ECRITURE/LECTURE/VUE) a partir : $CsvPath"
Write-Host "OU de base : $BaseOU"
Write-Host ""

# ------------------------------------------------------------------------
# 3. Traitement principal
# ------------------------------------------------------------------------
foreach ($row in $rows) {
    $index++

    # Progression unique
    Write-Progress -Activity "Creation OU et GDL" -Status "$index / $total" -PercentComplete
(($index / $total) * 100)
```

```
# Donnees de base
$folderPath = $row.FolderPath
if ([string]::IsNullOrWhiteSpace($folderPath)) { continue }

# Gestion du drapeau de pilotage "CREER_GROUPES"
$creer = $null
if ($row.PSObject.Properties.Name -contains "CREER_GROUPES") {
    $creer = $row.CREER_GROUPES
}
$creer = if ($creer) { $creer.Trim().ToUpper() } else { "" }

if ($creer -eq "NON") {
    Write-Host "Skip (CREER_GROUPES=NON) : $folderPath"
    continue
}

# 3.1 Code numerique a partir du FolderPath
$code = Get-NumericCodeFromFolderPath -FolderPath $folderPath
if ([string]::IsNullOrWhiteSpace($code)) {
    Write-Warning "Aucun code numerique pour : $folderPath -> ligne ignoree"
    continue
}

# 3.2 Noms des GDL (en MAJUSCULE)
$GDL_E = ("GDL_{0}_ECRITURE" -f $code).ToUpper()
$GDL_L = ("GDL_{0}_LECTURE"  -f $code).ToUpper()
$GDL_V = ("GDL_{0}_VUE"      -f $code).ToUpper()

Write-Host "Traitement : $folderPath"
Write-Host "  GDL_ECRITURE : $GDL_E"
Write-Host "  GDL_LECTURE  : $GDL_L"
Write-Host "  GDL_VUE      : $GDL_V"

# 3.3 Creation de la chaine d OU mirroir du FolderPath
$parts = $folderPath -split '\\' | Where-Object { $_ -and $_.Trim() -ne '' }
$currentOU = $BaseOU
foreach ($part in $parts) {
    $currentOU = Ensure-OU -Name $part -ParentDN $currentOU
}
```

```
    # 3.4 Creation des 3 groupes dans l OU finale
    Ensure-Group -Name $GDL_E -OU $currentOU
    Ensure-Group -Name $GDL_L -OU $currentOU
    Ensure-Group -Name $GDL_V -OU $currentOU

    # 3.5 Calcul du Parent_GDL = GDL_<codeParent>_VUE
    $Parent_GDL = $null
    $parentFolder = Split-Path $folderPath -Parent
    if ($parentFolder -and $parentFolder.Trim() -ne '') {
        $parentCode = Get-NumericCodeFromFolderPath -FolderPath $parentFolder
        if ($parentCode) {
            $Parent_GDL = ("GDL_{0}_VUE" -f $parentCode).ToUpper()
        }
    }

    # 3.6 Injection / mise a jour des colonnes dans le CSV
    foreach ($col in @("GDL_ECRITURE","GDL_LECTURE","GDL_VUE","Parent_GDL")) {
        if (-not ($row.PSObject.Properties.Name -contains $col)) {
            $row | Add-Member -Name $col -MemberType NoteProperty -Value $null -Force
        }
    }
    $row.GDL_ECRITURE = $GDL_E
    $row.GDL_LECTURE  = $GDL_L
    $row.GDL_VUE      = $GDL_V
    $row.Parent_GDL   = $Parent_GDL
}

# Effacer la barre de progression
Write-Progress -Activity "Creation OU et GDL" -Completed

# ------------------------------------------------------------------------
# 4. Sauvegarde CSV
# ------------------------------------------------------------------------
$rows | Export-Csv -Path $CsvPath -Delimiter ";" -NoTypeInformation -Encoding UTF8
```

Le tableau CSV qui en ressort devrait ressembler à ça :

NIVEAU 1	NIVEAU 2	NIVEAU 3	NIVEAU 4	NIVEAU 5	NIVEAU 6	NIVEAU 7	NIVEAU 8	FolderPath	CREER_GROUPES	GG_ECRITURE	GG_LECTURE	GDL_ECRITURE	GDL_LECTURE	GDL_VUE	Parent_GDL
01-DIRECTION								01-DIRECTION		ORG_GG_DIRECTION		GDL_01_ECRITURE	GDL_01_LECTURE	GDL_01_VUE	
02-FINANCE								02-FINANCE		ORG_GG_FINANCE	ORG_GG_DIRECTION	GDL_02_ECRITURE	GDL_02_LECTURE	GDL_02_VUE	
02-FINANCE	01-BUDGET							02-FINANCE\01-BUDGET		ORG_GG_FINANCE	ORG_GG_DIRECTION	GDL_02-01_ECRITURE	GDL_02-01_LECTURE	GDL_02-01_VUE	GDL_02_VUE
02-FINANCE	02-FACTURES							02-FINANCE\02-FACTURES		ORG_GG_FINANCE	ORG_GG_DIRECTION	GDL_02-02_ECRITURE	GDL_02-02_LECTURE	GDL_02-02_VUE	GDL_02_VUE
03-RH								03-RH		ORG_GG_RH	ORG_GG_DIRECTION	GDL_03_ECRITURE	GDL_03_LECTURE	GDL_03_VUE	
03-RH	01-CONTRATS							03-RH\01-CONTRATS		ORG_GG_RH	ORG_GG_DIRECTION	GDL_03-01_ECRITURE	GDL_03-01_LECTURE	GDL_03-01_VUE	GDL_03_VUE
03-RH	02-PAIE							03-RH	NON						
04-IT								04-IT		ORG_GG_IT	ORG_GG_DIRECTION	GDL_04_ECRITURE	GDL_04_LECTURE	GDL_04_VUE	
04-IT	01-INFRA							04-IT\01-INFRA		ORG_GG_IT	ORG_GG_DIRECTION	GDL_04-01_ECRITURE	GDL_04-01_LECTURE	GDL_04-01_VUE	GDL_04_VUE
04-IT	02-PROJETS							04-IT\02-PROJETS		ORG_GG_IT	ORG_GG_DIRECTION	GDL_04-02_ECRITURE	GDL_04-02_LECTURE	GDL_04-02_VUE	GDL_04_VUE

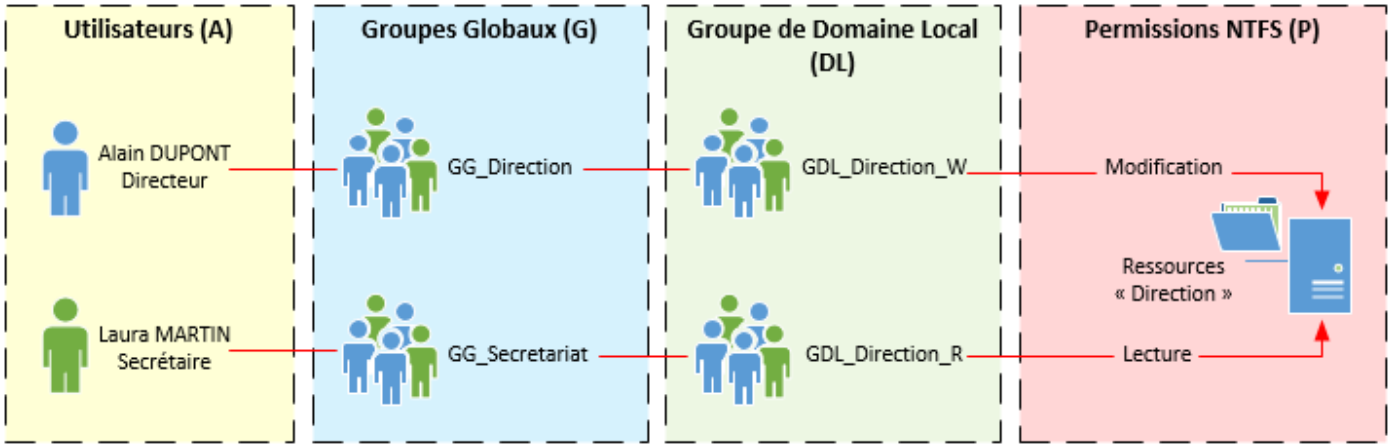
Et en version CSV :

```
NIVEAU 1;NIVEAU 2;NIVEAU 3;NIVEAU 4;NIVEAU 5;NIVEAU 6;NIVEAU 7;NIVEAU
8;FolderPath;CREER_GROUPES;GG_ECRITURE;GG_LECTURE;GDL_ECRITURE;GDL_LECTURE;GDL_VUE;Parent_GDL
01-DIRECTION;;;;;;;;01-DIRECTION;;ORG_GG_DIRECTION;;GDL_01_ECRITURE;GDL_01_LECTURE;GDL_01_VUE;
02-FINANCE;;;;;;;;02-
FINANCE;;ORG_GG_FINANCE;ORG_GG_DIRECTION;GDL_02_ECRITURE;GDL_02_LECTURE;GDL_02_VUE;
02-FINANCE;01-BUDGET;;;;;;;02-FINANCE\01-BUDGET;;ORG_GG_FINANCE;ORG_GG_DIRECTION;GDL_02-
01_ECRITURE;GDL_02-01_LECTURE;GDL_02-01_VUE;GDL_02_VUE
02-FINANCE;02-FACTURES;;;;;;;02-FINANCE\02-FACTURES;;ORG_GG_FINANCE;ORG_GG_DIRECTION;GDL_02-
02_ECRITURE;GDL_02-02_LECTURE;GDL_02-02_VUE;GDL_02_VUE
03-RH;;;;;;;;03-RH;;ORG_GG_RH;ORG_GG_DIRECTION;GDL_03_ECRITURE;GDL_03_LECTURE;GDL_03_VUE;
03-RH;01-CONTRATS;;;;;;;03-RH\01-CONTRATS;;ORG_GG_RH;ORG_GG_DIRECTION;GDL_03-
01_ECRITURE;GDL_03-01_LECTURE;GDL_03-01_VUE;GDL_03_VUE
03-RH;02-PAIE;;;;;;;03-RH;NON;;;;;;
04-IT;;;;;;;;04-IT;;ORG_GG_IT;ORG_GG_DIRECTION;GDL_04_ECRITURE;GDL_04_LECTURE;GDL_04_VUE;
04-IT;01-INFRA;;;;;;;04-IT\01-INFRA;;ORG_GG_IT;ORG_GG_DIRECTION;GDL_04-01_ECRITURE;GDL_04-
01_LECTURE;GDL_04-01_VUE;GDL_04_VUE
04-IT;02-PROJETS;;;;;;;04-IT\02-PROJETS;;ORG_GG_IT;ORG_GG_DIRECTION;GDL_04-02_ECRITURE;GDL_04-
02_LECTURE;GDL_04-02_VUE;GDL_04_VUE
04-IT;02-PROJETS;2026;;;;;;04-IT\02-PROJETS\2026;NON;;;;;;
```

Bien ! A présent nous avons des groupes bien nommés dans des OU bien classées.
Nous allons passer à la mise en place de la hiérarchie des groupes !

AGDLP

Hiérarchie des GDL



Introduction

Nous reprenons une nouvelle fois notre fichier CSV :

NIVEAU 1	NIVEAU 2	NIVEAU 3	NIVEAU 4	NIVEAU 5	NIVEAU 6	NIVEAU 7	NIVEAU 8	FolderPath	CREER_GROUPES	GG_ECRITURE	GG_LECTURE	GDL_ECRITURE	GDL_LECTURE	GDL_VUE	Parent_GDL
01-DIRECTION								01-DIRECTION		ORG_GG_DIRECTION		GDL_01_ECRITURE	GDL_01_LECTURE	GDL_01_VUE	
02-FINANCE								02-FINANCE		ORG_GG_FINANCE	ORG_GG_DIRECTION	GDL_02_ECRITURE	GDL_02_LECTURE	GDL_02_VUE	
02-FINANCE	01-BUDGET							02-FINANCE\01-BUDGET		ORG_GG_FINANCE	ORG_GG_DIRECTION	GDL_02-01_ECRITURE	GDL_02-01_LECTURE	GDL_02-01_VUE	GDL_02_VUE
02-FINANCE	02-FACTURES							02-FINANCE\02-FACTURES		ORG_GG_FINANCE	ORG_GG_DIRECTION	GDL_02-02_ECRITURE	GDL_02-02_LECTURE	GDL_02-02_VUE	GDL_02_VUE
03-RH								03-RH		ORG_GG_RH	ORG_GG_DIRECTION	GDL_03_ECRITURE	GDL_03_LECTURE	GDL_03_VUE	
03-RH	01-CONTRATS							03-RH\01-CONTRATS		ORG_GG_RH	ORG_GG_DIRECTION	GDL_03-01_ECRITURE	GDL_03-01_LECTURE	GDL_03-01_VUE	GDL_03_VUE
03-RH	02-PAIE							03-RH	NON						
04-IT								04-IT		ORG_GG_IT	ORG_GG_DIRECTION	GDL_04_ECRITURE	GDL_04_LECTURE	GDL_04_VUE	
04-IT	01-INFRA							04-IT\01-INFRA		ORG_GG_IT	ORG_GG_DIRECTION	GDL_04-01_ECRITURE	GDL_04-01_LECTURE	GDL_04-01_VUE	GDL_04_VUE
04-IT	02-PROJETS							04-IT\02-PROJETS		ORG_GG_IT	ORG_GG_DIRECTION	GDL_04-02_ECRITURE	GDL_04-02_LECTURE	GDL_04-02_VUE	GDL_04_VUE

```
NIVEAU 1;NIVEAU 2;NIVEAU 3;NIVEAU 4;NIVEAU 5;NIVEAU 6;NIVEAU 7;NIVEAU
8;FolderPath;CREER_GROUPES;GG_ECRITURE;GG_LECTURE;GDL_ECRITURE;GDL_LECTURE;GDL_VUE;Parent_GDL
01-DIRECTION;;;;;;;;01-DIRECTION;;ORG_GG_DIRECTION;;GDL_01_ECRITURE;GDL_01_LECTURE;GDL_01_VUE;
02-FINANCE;;;;;;;;02-
FINANCE;;ORG_GG_FINANCE;ORG_GG_DIRECTION;GDL_02_ECRITURE;GDL_02_LECTURE;GDL_02_VUE;
02-FINANCE;01-BUDGET;;;;;;;02-FINANCE\01-BUDGET;;ORG_GG_FINANCE;ORG_GG_DIRECTION;GDL_02-
01_ECRITURE;GDL_02-01_LECTURE;GDL_02-01_VUE;GDL_02_VUE
02-FINANCE;02-FACTURES;;;;;;;02-FINANCE\02-FACTURES;;ORG_GG_FINANCE;ORG_GG_DIRECTION;GDL_02-
02_ECRITURE;GDL_02-02_LECTURE;GDL_02-02_VUE;GDL_02_VUE
03-RH;;;;;;;;03-RH;;ORG_GG_RH;ORG_GG_DIRECTION;GDL_03_ECRITURE;GDL_03_LECTURE;GDL_03_VUE;
03-RH;01-CONTRATS;;;;;;;03-RH\01-CONTRATS;;ORG_GG_RH;ORG_GG_DIRECTION;GDL_03-
01_ECRITURE;GDL_03-01_LECTURE;GDL_03-01_VUE;GDL_03_VUE
03-RH;02-PAIE;;;;;;;03-RH;NON;;;;;;
```

```
04-IT;;;;;;;;04-IT;;ORG_GG_IT;ORG_GG_DIRECTION;GDL_04_ECRITURE;GDL_04_LECTURE;GDL_04_VUE;
04-IT;01-INFRA;;;;;;;04-IT\01-INFRA;;ORG_GG_IT;ORG_GG_DIRECTION;GDL_04-01_ECRITURE;GDL_04-
01_LECTURE;GDL_04-01_VUE;GDL_04_VUE
04-IT;02-PROJETS;;;;;;;04-IT\02-PROJETS;;ORG_GG_IT;ORG_GG_DIRECTION;GDL_04-02_ECRITURE;GDL_04-
02_LECTURE;GDL_04-02_VUE;GDL_04_VUE
04-IT;02-PROJETS;2026;;;;;;04-IT\02-PROJETS\2026;NON;;;;;;
```

Ici, il faut que le GDL "04-IT\02-PROJETS" qui est lecture ou écriture donc "**GDL_04-02_ECRITURE**" ou "**GDL_04-02_LECTURE** soient membre de "04-IT" vue donc "**GDL_04_VUE**" etc etc

De plus, il faut, au contraire que "**GDL_01_ECRITURE**"ou "**GDL_04-02_LECTURE**" par exemple aient des droits récursifs sur l'ensemble des enfants, mais cela sera pour une prochaine partie.

Mise en place de la hiérarchie

On en vient donc au script qui permet de faire ça.
Il va simplement lire les colonnes GDL_ECRITURE, GDL_LECTURE, GDL_VUE et Parent_GDL, ce qui lui donnes toutes les infos nécessaires :

```
<#
================================================================================
SCRIPT : 3.1 - Creation_Parentalite.ps1
OBJET  : Chainer les GDL de chaque dossier vers le GDL_VUE de leur parent
================================================================================

DESCRIPTION :
  - Pour chaque ligne du CSV :
        * Si Parent_GDL est renseigne :
             - Ajouter GDL_ECRITURE   -> Parent_GDL
             - Ajouter GDL_LECTURE    -> Parent_GDL
             - Ajouter GDL_VUE        -> Parent_GDL
  - Tous les messages visibles sont sans accents.
  - Commentaires en francais pour clarifier chaque etape.

NOTE :
  - Ce script n effectue PAS de creation de groupes.
    Il se contente de realiser le chainage entre niveaux.
  - Fonctionne sur les noms de groupes crees dans le script 2.1.
```

```
================================================================================
#>

Param(
    [string]$CsvPath = "C:\AGDLP\ORG_ARBO_GDL.csv"
)

$ErrorActionPreference = "Stop"

# ------------------------------------------------------------------------------
# Module ActiveDirectory
# ------------------------------------------------------------------------------
try {
    Import-Module ActiveDirectory -ErrorAction Stop
}
catch {
    Write-Error "Module ActiveDirectory introuvable. Installer RSAT AD PowerShell."
    exit 1
}

# ------------------------------------------------------------------------------
# Lecture du CSV
# ------------------------------------------------------------------------------
if (-not (Test-Path $CsvPath)) {
    Write-Error "CSV introuvable : $CsvPath"
    exit 1
}

$rows  = Import-Csv -Path $CsvPath -Delimiter ";" -Encoding UTF8
$total = $rows.Count
$index = 0

Write-Host "Application de la parentalite GDL a partir du fichier : $CsvPath"
Write-Host ""

# ------------------------------------------------------------------------------
# Boucle principale
# ------------------------------------------------------------------------------
foreach ($row in $rows) {
    $index++
```

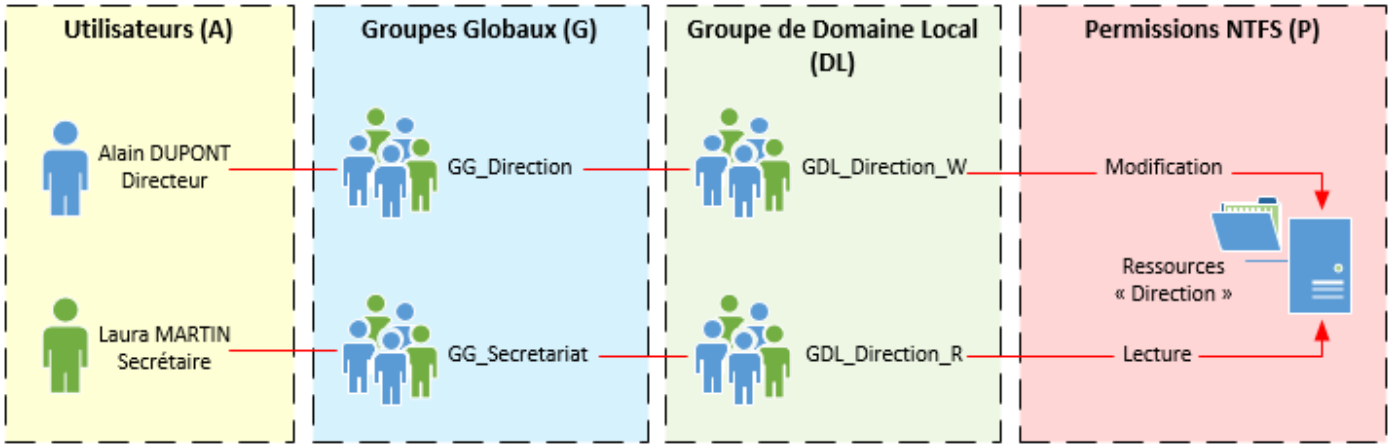
```
# Mettre a jour la progression
Write-Progress -Activity "Chainage GDL enfant -> Parent_GDL" `
               -Status "$index / $total" `
               -PercentComplete (($index/$total)*100)

# Recuperation du parent
$parent = $row.Parent_GDL
if ([string]::IsNullOrWhiteSpace($parent)) {
    # Aucun parent pour cette ligne
    continue
}

# Verifier que le parent existe dans l AD
$parentGroup = Get-ADGroup -Filter "Name -eq '$parent'" -ErrorAction SilentlyContinue
if (-not $parentGroup) {
    Write-Warning "Parent GDL introuvable dans AD : $parent"
    continue
}

# Groupes enfants a chainer
$childGroups = @()

if ($row.GDL_ECRITURE -and $row.GDL_ECRITURE.Trim() -ne "") {
    $childGroups += $row.GDL_ECRITURE.Trim()
}
if ($row.GDL_LECTURE -and $row.GDL_LECTURE.Trim() -ne "") {
    $childGroups += $row.GDL_LECTURE.Trim()
}
if ($row.GDL_VUE -and $row.GDL_VUE.Trim() -ne "") {
    $childGroups += $row.GDL_VUE.Trim()
}

# Ajouter les groupes enfants dans le groupe parent
foreach ($cg in $childGroups) {

    $exists = Get-ADGroup -Filter "Name -eq '$cg'" -ErrorAction SilentlyContinue
    if (-not $exists) {
        Write-Warning "GDL enfant introuvable : $cg"
        continue
```

```
        }

        try {
            Add-ADGroupMember -Identity $parent -Members $cg -ErrorAction Stop
            Write-Host "Chaine : $cg -> $parent"
        }
        catch {
            Write-Warning "Erreur chainage : $cg -> $parent   ($_ )"
        }
    }
}

# Effacer la barre de progression
Write-Progress -Activity "Chainage GDL enfant -> Parent_GDL" -Completed

Write-Host ""
Write-Host "Chainage parental GDL termine." -ForegroundColor Green
```

Avec cela, chaque enfant VUE, LECTURE et ECRITURE est membre de son parent VUE direct. Prochaine étape appliquer les ACL aux répertoires du partage !

AGDLP

Appliquer les ACL



Introduction

Nous reprenons une nouvelle fois notre fichier CSV :

NIVEAU 1	NIVEAU 2	NIVEAU 3	NIVEAU 4	NIVEAU 5	NIVEAU 6	NIVEAU 7	NIVEAU 8	FolderPath	CREER_GROUPES	GG_ECRITURE	GG_LECTURE	GDL_ECRITURE	GDL_LECTURE	GDL_VUE	Parent_GDL
01-DIRECTION								01-DIRECTION		ORG_GG_DIRECTION		GDL_01_ECRITURE	GDL_01_LECTURE	GDL_01_VUE	
02-FINANCE								02-FINANCE		ORG_GG_FINANCE	ORG_GG_DIRECTION	GDL_02_ECRITURE	GDL_02_LECTURE	GDL_02_VUE	
02-FINANCE	01-BUDGET							02-FINANCE\01-BUDGET		ORG_GG_FINANCE	ORG_GG_DIRECTION	GDL_02-01_ECRITURE	GDL_02-01_LECTURE	GDL_02-01_VUE	GDL_02_VUE
02-FINANCE	02-FACTURES							02-FINANCE\02-FACTURES		ORG_GG_FINANCE	ORG_GG_DIRECTION	GDL_02-02_ECRITURE	GDL_02-02_LECTURE	GDL_02-02_VUE	GDL_02_VUE
03-RH								03-RH		ORG_GG_RH	ORG_GG_DIRECTION	GDL_03_ECRITURE	GDL_03_LECTURE	GDL_03_VUE	
03-RH	01-CONTRATS							03-RH\01-CONTRATS		ORG_GG_RH	ORG_GG_DIRECTION	GDL_03-01_ECRITURE	GDL_03-01_LECTURE	GDL_03-01_VUE	GDL_03_VUE
03-RH	02-PAIE							03-RH	NON						
04-IT								04-IT		ORG_GG_IT	ORG_GG_DIRECTION	GDL_04_ECRITURE	GDL_04_LECTURE	GDL_04_VUE	
04-IT	01-INFRA							04-IT\01-INFRA		ORG_GG_IT	ORG_GG_DIRECTION	GDL_04-01_ECRITURE	GDL_04-01_LECTURE	GDL_04-01_VUE	GDL_04_VUE
04-IT	02-PROJETS							04-IT\02-PROJETS		ORG_GG_IT	ORG_GG_DIRECTION	GDL_04-02_ECRITURE	GDL_04-02_LECTURE	GDL_04-02_VUE	GDL_04_VUE

```
NIVEAU 1;NIVEAU 2;NIVEAU 3;NIVEAU 4;NIVEAU 5;NIVEAU 6;NIVEAU 7;NIVEAU
8;FolderPath;CREER_GROUPES;GG_ECRITURE;GG_LECTURE;GDL_ECRITURE;GDL_LECTURE;GDL_VUE;Parent_GDL
01-DIRECTION;;;;;;;;01-DIRECTION;;ORG_GG_DIRECTION;;GDL_01_ECRITURE;GDL_01_LECTURE;GDL_01_VUE;
02-FINANCE;;;;;;;;02-
FINANCE;;ORG_GG_FINANCE;ORG_GG_DIRECTION;GDL_02_ECRITURE;GDL_02_LECTURE;GDL_02_VUE;
02-FINANCE;01-BUDGET;;;;;;;02-FINANCE\01-BUDGET;;ORG_GG_FINANCE;ORG_GG_DIRECTION;GDL_02-
01_ECRITURE;GDL_02-01_LECTURE;GDL_02-01_VUE;GDL_02_VUE
02-FINANCE;02-FACTURES;;;;;;;02-FINANCE\02-FACTURES;;ORG_GG_FINANCE;ORG_GG_DIRECTION;GDL_02-
02_ECRITURE;GDL_02-02_LECTURE;GDL_02-02_VUE;GDL_02_VUE
03-RH;;;;;;;;03-RH;;ORG_GG_RH;ORG_GG_DIRECTION;GDL_03_ECRITURE;GDL_03_LECTURE;GDL_03_VUE;
03-RH;01-CONTRATS;;;;;;;03-RH\01-CONTRATS;;ORG_GG_RH;ORG_GG_DIRECTION;GDL_03-
01_ECRITURE;GDL_03-01_LECTURE;GDL_03-01_VUE;GDL_03_VUE
03-RH;02-PAIE;;;;;;;03-RH;NON;;;;;;
```

```
04-IT;;;;;;;;04-IT;;ORG_GG_IT;ORG_GG_DIRECTION;GDL_04_ECRITURE;GDL_04_LECTURE;GDL_04_VUE;
04-IT;01-INFRA;;;;;;;04-IT\01-INFRA;;ORG_GG_IT;ORG_GG_DIRECTION;GDL_04-01_ECRITURE;GDL_04-
01_LECTURE;GDL_04-01_VUE;GDL_04_VUE
04-IT;02-PROJETS;;;;;;;04-IT\02-PROJETS;;ORG_GG_IT;ORG_GG_DIRECTION;GDL_04-02_ECRITURE;GDL_04-
02_LECTURE;GDL_04-02_VUE;GDL_04_VUE
04-IT;02-PROJETS;2026;;;;;;04-IT\02-PROJETS\2026;NON;;;;;;
```

Application des droits

Voici le script qui permet d'appliquer les droits à chaque GDL homonymique de son répertoire, ainsi que l'héritage des groupes RW à l'ensemble des enfants.
Encore une fois, le script ne ce sert que des colonnes utiles dans le cas présent : FolderPath, GDL_RW et GDL_RO :

```
<#
================================================================================
SCRIPT : 4.1 - Appliquer_ACL.ps1
OBJET  : Appliquer les ACL NTFS tri-niveaux (ECRITURE / LECTURE / VUE)
================================================================================

DESCRIPTION :
  Pour chaque dossier present dans le CSV :
    * Desactiver l'heritage NTFS
    * Supprimer toutes les ACL existantes
    * Appliquer les droits :
         - GDL_ECRITURE : Modify (herite)
         - GDL_LECTURE  : ReadAndExecute (herite)
         - GDL_VUE      : ReadAndExecute (non herite)
         - Administrateurs : Modify (herite)
         - Admins du domaine : Modify (herite)
         - SYSTEM : Modify (herite)

REMARQUE :
  - Le script suppose que 2.1 et 3.1 ont deja genere les GDL
    et les chainages Parent_GDL.
================================================================================
#>
```

```
Param(
    [string]$RootPath = "\\SRV-DATAS\datas\",
    [string]$CsvPath  = "C:\AGDLP\ORG_ARBO_GDL.csv"
)

$ErrorActionPreference = "Stop"

# ------------------------------------------------------------------------
# 1. Verification des droits administrateur (indispensable pour Set-Acl)
# ------------------------------------------------------------------------
$principal = New-Object
Security.Principal.WindowsPrincipal([Security.Principal.WindowsIdentity]::GetCurrent())
if (-not $principal.IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)) {
    Write-Warning "Ce script doit etre lance en tant qu administrateur."
    exit 1
}

# ------------------------------------------------------------------------
# 2. Lecture du CSV
# ------------------------------------------------------------------------
if (-not (Test-Path $CsvPath)) {
    Write-Error "CSV introuvable : $CsvPath"
    exit 1
}

$rows  = Import-Csv -Path $CsvPath -Delimiter ";" -Encoding UTF8
$total = $rows.Count
$index = 0

Write-Host "Application des ACL NTFS a partir du fichier : $CsvPath"
Write-Host "Racine de donnees : $RootPath"
Write-Host ""

# ------------------------------------------------------------------------
# 3. Comptes administratifs ayant toujours Modify (herite)
# ------------------------------------------------------------------------
$RWAdmins = @(
    "Administrateurs",
    "Admins du domaine",
```

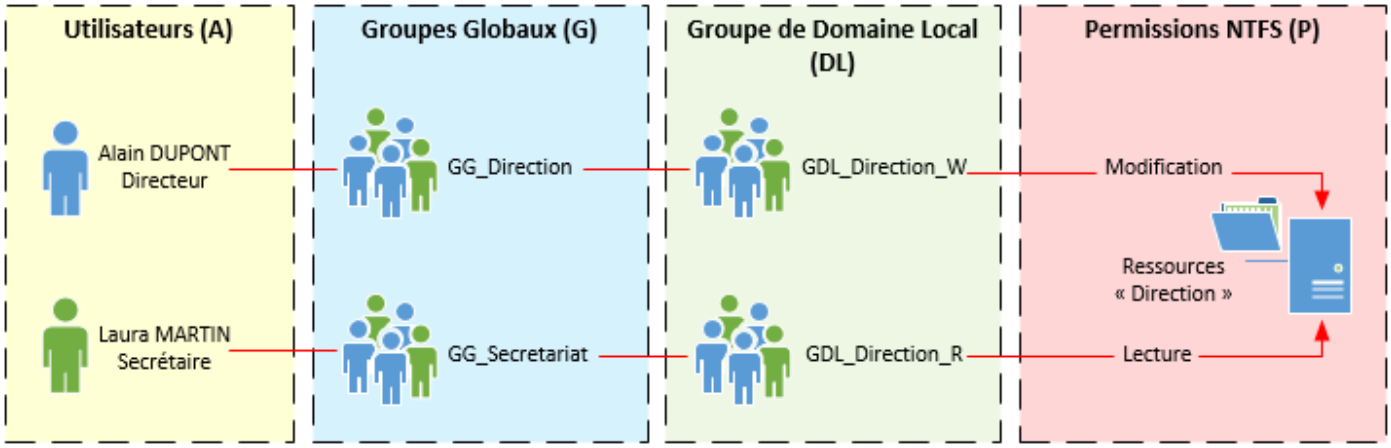
```
    "Administrateur-lcc",
    "SYSTEM"
)

# ------------------------------------------------------------------------
# 4. Traitement principal
# ------------------------------------------------------------------------
foreach ($entry in $rows) {
    $index++

    # Mise a jour de la barre de progression
    Write-Progress -Activity "Application ACL" `
                   -Status "$index / $total" `
                   -PercentComplete (($index / $total) * 100)

    # Verifier la presence de FolderPath
    if ([string]::IsNullOrWhiteSpace($entry.FolderPath)) { continue }

    # Construit le chemin complet
    $folderFullPath = Join-Path -Path $RootPath -ChildPath $entry.FolderPath

    if (-not (Test-Path -LiteralPath $folderFullPath)) {
        Write-Warning "Dossier introuvable : $folderFullPath"
        continue
    }

    try {
        # ------------------------------------------------------------------
        # 4.1 Recuperation des ACL actuelles et desactivation de l heritage
        # ------------------------------------------------------------------
        $acl = Get-Acl -LiteralPath $folderFullPath
        $acl.SetAccessRuleProtection($true, $false)   # Disable inheritance, remove inherited
rules

        # Suppression de toutes les ACL existantes
        foreach ($rule in @($acl.Access)) {
            [void]$acl.RemoveAccessRule($rule)
        }

        # Types et drapeaux utiles
```

```
        $Allow  = [System.Security.AccessControl.AccessControlType]::Allow
        $CI_OI  = [System.Security.AccessControl.InheritanceFlags] "ContainerInherit,
ObjectInherit"
        $NonePF = [System.Security.AccessControl.PropagationFlags]::None

        # ---------------------------------------------------------------------
        # 4.2 GDL_ECRITURE : Modify (herite)
        # ---------------------------------------------------------------------
        if ($entry.GDL_ECRITURE -and $entry.GDL_ECRITURE.Trim() -ne "") {
            $ruleE = New-Object System.Security.AccessControl.FileSystemAccessRule(
                $entry.GDL_ECRITURE,
                [System.Security.AccessControl.FileSystemRights]::Modify,
                $CI_OI, $NonePF, $Allow
            )
            [void]$acl.AddAccessRule($ruleE)
        }

        # ---------------------------------------------------------------------
        # 4.3 GDL_LECTURE : ReadAndExecute (herite)
        # ---------------------------------------------------------------------
        if ($entry.GDL_LECTURE -and $entry.GDL_LECTURE.Trim() -ne "") {
            $ruleL = New-Object System.Security.AccessControl.FileSystemAccessRule(
                $entry.GDL_LECTURE,
                [System.Security.AccessControl.FileSystemRights]::ReadAndExecute,
                $CI_OI, $NonePF, $Allow
            )
            [void]$acl.AddAccessRule($ruleL)
        }

        # ---------------------------------------------------------------------
        # 4.4 GDL_VUE : ReadAndExecute (non herite)
        # ---------------------------------------------------------------------
        if ($entry.GDL_VUE -and $entry.GDL_VUE.Trim() -ne "") {
            $ruleV = New-Object System.Security.AccessControl.FileSystemAccessRule(
                $entry.GDL_VUE,
                [System.Security.AccessControl.FileSystemRights]::ReadAndExecute,
                [System.Security.AccessControl.InheritanceFlags]::None,
                $NonePF,
                $Allow
            )
```

```
            [void]$acl.AddAccessRule($ruleV)
        }

        # ---------------------------------------------------------------
        # 4.5 Comptes administratifs : Modify (herite)
        # ---------------------------------------------------------------
        foreach ($admin in $RWAdmins) {
            $ruleA = New-Object System.Security.AccessControl.FileSystemAccessRule(
                $admin,
                [System.Security.AccessControl.FileSystemRights]::Modify,
                $CI_OI, $NonePF, $Allow
            )
            [void]$acl.AddAccessRule($ruleA)
        }

        # ---------------------------------------------------------------
        # 4.6 Application finale des ACL
        # ---------------------------------------------------------------
        Set-Acl -LiteralPath $folderFullPath -AclObject $acl
        Write-Host "ACL appliquees : $folderFullPath"

    } catch {
        Write-Warning "Erreur ACL sur $folderFullPath -> $_"
    }
}

# Effacer la barre de progression
Write-Progress -Activity "Application ACL" -Completed

Write-Host ""
Write-Host "Application des ACL terminee." -ForegroundColor Green
```

Parfait ! Nous avons à présent mis en place l'AGDLP pour une gestion des droits beaucoup plus fine, facile, et modulable dans le temps.
De plus, ce script s'appuie sur un répertoire racine qui peut etre modifié, si jamais une nouvelle branche devait être créé dans le partage.

Création des GG



Introduction

Nous allons attaquer la partie des groupes globaux, les créer automatiquement via un script, et y ajouter des membres avec le même principe.

Tableau modèle

Il nous faut d'abord le fichier qui servira de modèle, que l'on mettra à jour, pour faire les modifications.
Le gros avantage est que ce fichier peut-être divisé par service, et distribué aux responsables et directeurs de services pour avoir des infos toujours à jour, sans donner accès à une console d'administration.

GG	Membre
ORG_GG_DIRECTION	
ORG_GG_FINANCE	
ORG_GG_RH	
ORG_GG_IT	
ORG_GG_DIRECTION	Alice MARTIN
ORG_GG_DIRECTION	Thomas ROBERT
ORG_GG_FINANCE	Julie LAMBERT
ORG_GG_FINANCE	Nicolas PETIT
ORG_GG_RH	Sophie DURAND
ORG_GG_RH	Marc LEFEVRE
ORG_GG_IT	Lucas MOREAU
ORG_GG_IT	Emma GARNIER
ORG_GG_IT	Hugo CHEVALIER
ORG_GG_COMMUNICATION	
ORG_GG_COMMUNICATION	Clara BERTRAND
ORG_GG_LOGISTIQUE	
ORG_GG_LOGISTIQUE	Antoine RENAUD

Il faut ensuite exporter, ou convertir ce tableau en csv, encodage UTF8, séparateur ";" point virgule.

On obtient alors ce fichier :

```
GG;Membre
ORG_GG_DIRECTION;
ORG_GG_FINANCE;
ORG_GG_RH;
ORG_GG_IT;
ORG_GG_DIRECTION;Alice MARTIN
ORG_GG_DIRECTION;Thomas ROBERT
ORG_GG_FINANCE;Julie LAMBERT
ORG_GG_FINANCE;Nicolas PETIT
ORG_GG_RH;Sophie DURAND
ORG_GG_RH;Marc LEFEVRE
ORG_GG_IT;Lucas MOREAU
ORG_GG_IT;Emma GARNIER
ORG_GG_IT;Hugo CHEVALIER
ORG_GG_COMMUNICATION;
ORG_GG_COMMUNICATION;Clara BERTRAND
ORG_GG_LOGISTIQUE;
ORG_GG_LOGISTIQUE;Antoine RENAUD
```

Création des GG

Voici à présent le script qui permet de créer tous les GG, dans un seul et même endroit, dans OU=GG,OU=AGDLP,DC=domain,DC=local :

```
<#
================================================================================
SCRIPT : 5.1 - Creation_GG.ps1  (Version 2-passes, robuste)
OBJET  : Creation des Groupes Globaux (GG) + alignement de leurs membres
================================================================================

PRINCIPE :
  PASS A : Creer tous les groupes globaux (GG et sous-GG) avant ajout des membres
  PASS B : Vider contenu de chaque GG puis injecter les membres (GG puis utilisateurs)

AVANTAGES :
  - Plus aucune erreur "groupe introuvable"
  - Les sous-GG peuvent apparaitre apres les parents dans le CSV
  - Nettoyage des caracteres parasites (nbsp, guillemets, espaces)
  - Sorties visibles sans accents (compatibilite console)
================================================================================
#>

Param(
    [string]$CSVPath = "C:\AGDLP\ORG_ARBO_GDL.csv",
    [string]$OUPath  = "OU=GG,OU=AGDLP,DC=domain,DC=local",
    [string]$LogFile = "C:\AGDLP\AD_Group_Update_Errors.txt"
)

$ErrorActionPreference = "Stop"

# ------------------------------------------------------------------------------
# 0. Module AD
# ------------------------------------------------------------------------------
try { Import-Module ActiveDirectory -ErrorAction Stop }
catch {
    Write-Error "Module ActiveDirectory non disponible. Installer RSAT."
    exit 1
}

# ------------------------------------------------------------------------------
# 1. Reset fichier log (UTF-8 BOM)
```

```
# ------------------------------------------------------------------------
$utf8Bom = New-Object System.Text.UTF8Encoding($true)
[System.IO.File]::WriteAllText($LogFile, "", $utf8Bom)

# ------------------------------------------------------------------------
# 2. Lecture CSV GG
# ------------------------------------------------------------------------
if (-not (Test-Path $CSVPath)) {
    Write-Error "Fichier CSV introuvable : $CSVPath"
    exit 1
}

$data = Import-Csv -Path $CSVPath -Delimiter ";" |
        Where-Object { $_.GG -and $_.GG.Trim() -ne "" }

if ($data.Count -eq 0) {
    Write-Host "Aucun groupe global a traiter."
    exit 0
}

# ------------------------------------------------------------------------
# 3. Petite fonction de nettoyage (supprime espaces HTML, nbsp, guillemets)
# ------------------------------------------------------------------------
function Clean-Cell([string]$s) {
    if (-not $s) { return "" }
    $s = $s -replace " ", " "
    $s = $s -replace "\u00A0", " "
    $s = $s -replace "\u200B", ""
    $s = $s -replace "[«»“”‘’]", ""
    return ($s -replace "\s+", " ").Trim()
}

foreach ($row in $data) {
    $row.GG     = Clean-Cell $row.GG
    $row.Membre = Clean-Cell $row.Membre
}

# ------------------------------------------------------------------------
# 4. PASS A : CREATION DE TOUS LES GROUPES (GG + sous-GG references)
# ------------------------------------------------------------------------
```

```
# Ensemble de tous les groupes a creer
$allGG = New-Object System.Collections.Generic.HashSet[string]
([StringComparer]::OrdinalIgnoreCase)

# Tous les GG declares
$data | ForEach-Object { $null = $allGG.Add($_.GG) }

# Tous les GG references comme Membre
foreach ($row in $data) {
    if ($row.Membre -and ($row.Membre -like "*_GG_*")) {
        $null = $allGG.Add($row.Membre)
    }
}

# Correction : pas de .ToArray(), on convertit proprement en tableau PowerShell
$allGGList = @($allGG)

$totalA = $allGGList.Count
$indexA = 0

Write-Host "Pass A : Creation des GG (sans membres) - Total: $totalA"
foreach ($ggName in $allGGList) {
    $indexA++
    Write-Progress -Activity "Pass A - Creation GG" -Status "$indexA / $totalA" -
PercentComplete (($indexA/$totalA)*100)

    if ([string]::IsNullOrWhiteSpace($ggName)) { continue }

    $exists = Get-ADGroup -Filter "SamAccountName -eq '$ggName'" `
                      -SearchBase $OUPath `
                      -ErrorAction SilentlyContinue

    if ($exists) { continue }

    try {
        New-ADGroup -Name $ggName -SamAccountName $ggName -Path $OUPath `
            -GroupScope Global -GroupCategory Security -ErrorAction Stop | Out-Null
        Write-Host "Cree : $ggName"
    }
```

```
    catch {
        $msg = "Erreur creation du groupe $ggName : $_`r`n"
        [System.IO.File]::AppendAllText($LogFile, $msg, $utf8Bom)
        Write-Warning "Creation impossible : $ggName (voir log)"
    }
}

Write-Progress -Activity "Pass A - Creation GG" -Completed

# ------------------------------------------------------------------------
# 5. PASS B : VIDER et RE-INJECTER les membres
# ------------------------------------------------------------------------

$groups = $data | Group-Object -Property GG

$totalB = $groups.Count
$indexB = 0

Write-Host ""
Write-Host "Pass B : Injection des membres dans chaque GG"

foreach ($grp in $groups) {
    $indexB++
    Write-Progress -Activity "Pass B - Inj. membres" `
        -Status "$indexB / $totalB - $($grp.Name)" `
        -PercentComplete (($indexB/$totalB)*100)

    $GroupName = Clean-Cell $grp.Name
    if (-not $GroupName) { continue }

    # ------------------------------------------------------
    # 5.1 Reset membres du GG
    # ------------------------------------------------------
    try {
        $existing = Get-ADGroup -Identity $GroupName -ErrorAction Stop
        $members = Get-ADGroupMember -Identity $GroupName -ErrorAction SilentlyContinue
        if ($members) {
            Remove-ADGroupMember -Identity $GroupName -Members $members `
                -Confirm:$false -ErrorAction SilentlyContinue
        }
```

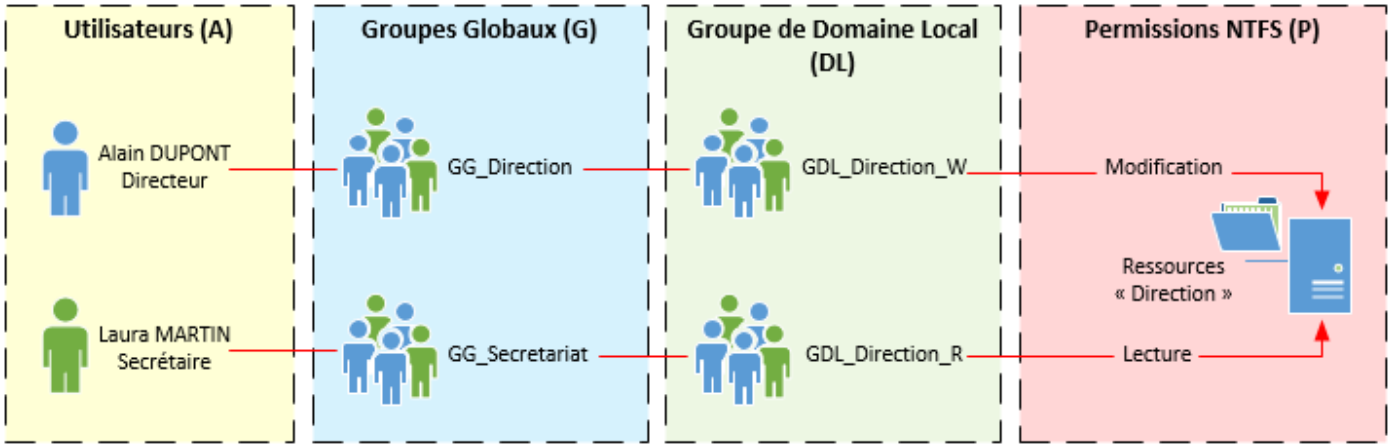
```
}
catch {
    # Si pour une raison obscure il n existe pas, on tente creation tardive
    try {
        New-ADGroup -Name $GroupName -SamAccountName $GroupName -Path $OUPath `
            -GroupScope Global -GroupCategory Security -ErrorAction Stop | Out-Null
    }
    catch {
        $msg = "Erreur creation tardive du groupe $GroupName : $_`r`n"
        [System.IO.File]::AppendAllText($LogFile, $msg, $utf8Bom)
        Write-Warning "Impossible de traiter $GroupName (voir log)"
        continue
    }
}

# ------------------------------------------------------
# 5.2 Ajouter les membres GROUPE (tous existent depuis Pass A)
# ------------------------------------------------------
foreach ($line in $grp.Group) {
    $m = Clean-Cell $line.Membre
    if (-not $m) { continue }
    if ($m -notlike "*_GG_*") { continue }

    $child = Get-ADGroup -Identity $m -ErrorAction SilentlyContinue
    if (-not $child) {
        $msg = "Groupe membre introuvable (apres Pass A) : $m (parent: $GroupName)`r`n"
        [System.IO.File]::AppendAllText($LogFile, $msg, $utf8Bom)
        Write-Warning "GG introuvable : $m -> ignore"
        continue
    }

    Add-ADGroupMember -Identity $GroupName -Members $child -ErrorAction SilentlyContinue
    Write-Host " + GG : $m"
}

# ------------------------------------------------------
# 5.3 Ajouter les membres UTILISATEUR
# ------------------------------------------------------
foreach ($line in $grp.Group) {
    $m = Clean-Cell $line.Membre
```

```
        if (-not $m) { continue }
        if ($m -like "*_GG_*") { continue }

        $user = Get-ADUser -Filter { DisplayName -eq $m } -ErrorAction SilentlyContinue
        if (-not $user) {
            $msg = "Utilisateur introuvable : $m (groupe : $GroupName)`r`n"
            [System.IO.File]::AppendAllText($LogFile, $msg, $utf8Bom)
            Write-Warning "Utilisateur introuvable : $m -> ignore"
            continue
        }

        Add-ADGroupMember -Identity $GroupName -Members $user.SamAccountName `
            -ErrorAction SilentlyContinue
        Write-Host " + USR: $m"
    }
}

Write-Progress -Activity "Pass B - Inj. membres" -Completed

Write-Host ""
Write-Host "Traitement GG termine avec succes. Voir le log si necessaire : $LogFile" -
ForegroundColor Green
```

Si les utilisateurs existes, il se retrouves dans les groupes GG qui leurs correspondant dans le tableau.
J'ai pendant un moment pensé à faire une ligne par GG et séparer les utilisateurs par une ",", mais cela n'aurait pas été pratique pour les éventuels utilisateurs d'autres services.

AGDLP

Liaison GG-GDL



Introduction

Il s'agit de la dernière étape pour mettre en place des droits AGDLP, ajout les GG comme membres des GDLP afin que les utilisateurs puissent accéder aux ressources partagées.

Introduction

Nous reprenons une nouvelle fois notre fichier CSV :

NIVEAU 1	NIVEAU 2	NIVEAU 3	NIVEAU 4	NIVEAU 5	NIVEAU 6	NIVEAU 7	NIVEAU 8	FolderPath	CREER_GROUPES	GG_ECRITURE	GG_LECTURE	GDL_ECRITURE	GDL_LECTURE	GDL_VUE	Parent_GDL
01-DIRECTION								01-DIRECTION		ORG_GG_DIRECTION		GDL_01_ECRITURE	GDL_01_LECTURE	GDL_01_VUE	
02-FINANCE								02-FINANCE		ORG_GG_FINANCE	ORG_GG_DIRECTION	GDL_02_ECRITURE	GDL_02_LECTURE	GDL_02_VUE	
02-FINANCE	01-BUDGET							02-FINANCE\01-BUDGET		ORG_GG_FINANCE	ORG_GG_DIRECTION	GDL_02-01_ECRITURE	GDL_02-01_LECTURE	GDL_02-01_VUE	GDL_02_VUE
02-FINANCE	02-FACTURES							02-FINANCE\02-FACTURES		ORG_GG_FINANCE	ORG_GG_DIRECTION	GDL_02-02_ECRITURE	GDL_02-02_LECTURE	GDL_02-02_VUE	GDL_02_VUE
03-RH								03-RH		ORG_GG_RH	ORG_GG_DIRECTION	GDL_03_ECRITURE	GDL_03_LECTURE	GDL_03_VUE	
03-RH	01-CONTRATS							03-RH\01-CONTRATS		ORG_GG_RH	ORG_GG_DIRECTION	GDL_03-01_ECRITURE	GDL_03-01_LECTURE	GDL_03-01_VUE	GDL_03_VUE
03-RH	02-PAIE							03-RH	NON						
04-IT								04-IT		ORG_GG_IT	ORG_GG_DIRECTION	GDL_04_ECRITURE	GDL_04_LECTURE	GDL_04_VUE	
04-IT	01-INFRA							04-IT\01-INFRA		ORG_GG_IT	ORG_GG_DIRECTION	GDL_04-01_ECRITURE	GDL_04-01_LECTURE	GDL_04-01_VUE	GDL_04_VUE
04-IT	02-PROJETS							04-IT\02-PROJETS		ORG_GG_IT	ORG_GG_DIRECTION	GDL_04-02_ECRITURE	GDL_04-02_LECTURE	GDL_04-02_VUE	GDL_04_VUE

```
NIVEAU 1;NIVEAU 2;NIVEAU 3;NIVEAU 4;NIVEAU 5;NIVEAU 6;NIVEAU 7;NIVEAU
8;FolderPath;CREER_GROUPES;GG_ECRITURE;GG_LECTURE;GDL_ECRITURE;GDL_LECTURE;GDL_VUE;Parent_GDL
01-DIRECTION;;;;;;;01-DIRECTION;;ORG_GG_DIRECTION;;GDL_01_ECRITURE;GDL_01_LECTURE;GDL_01_VUE;
02-FINANCE;;;;;;;02-
FINANCE;;ORG_GG_FINANCE;ORG_GG_DIRECTION;GDL_02_ECRITURE;GDL_02_LECTURE;GDL_02_VUE;
02-FINANCE;01-BUDGET;;;;;;;02-FINANCE\01-BUDGET;;ORG_GG_FINANCE;ORG_GG_DIRECTION;GDL_02-
01_ECRITURE;GDL_02-01_LECTURE;GDL_02-01_VUE;GDL_02_VUE
02-FINANCE;02-FACTURES;;;;;;;02-FINANCE\02-FACTURES;;ORG_GG_FINANCE;ORG_GG_DIRECTION;GDL_02-
```

```
02_ECRITURE;GDL_02-02_LECTURE;GDL_02-02_VUE;GDL_02_VUE
03-RH;;;;;;;;03-RH;;ORG_GG_RH;ORG_GG_DIRECTION;GDL_03_ECRITURE;GDL_03_LECTURE;GDL_03_VUE;
03-RH;01-CONTRATS;;;;;;;03-RH\01-CONTRATS;;ORG_GG_RH;ORG_GG_DIRECTION;GDL_03-
01_ECRITURE;GDL_03-01_LECTURE;GDL_03-01_VUE;GDL_03_VUE
03-RH;02-PAIE;;;;;;;03-RH;NON;;;;;;
04-IT;;;;;;;;04-IT;;ORG_GG_IT;ORG_GG_DIRECTION;GDL_04_ECRITURE;GDL_04_LECTURE;GDL_04_VUE;
04-IT;01-INFRA;;;;;;;04-IT\01-INFRA;;ORG_GG_IT;ORG_GG_DIRECTION;GDL_04-01_ECRITURE;GDL_04-
01_LECTURE;GDL_04-01_VUE;GDL_04_VUE
04-IT;02-PROJETS;;;;;;;04-IT\02-PROJETS;;ORG_GG_IT;ORG_GG_DIRECTION;GDL_04-02_ECRITURE;GDL_04-
02_LECTURE;GDL_04-02_VUE;GDL_04_VUE
04-IT;02-PROJETS;2026;;;;;;04-IT\02-PROJETS\2026;NON;;;;;;
```

Ajouter les GG dans les GDL

Pour cette dernière étape, le script va utiliser les colonnes GG_RW qui va ajouter comme membre du GDL_RW correspondant, idem pour GG_RO avec GDL_RO :

```
<#
================================================================================
SCRIPT : 6.1 - Ajout_G_dans_GDL.ps1
OBJET  : Aligne les GDL avec les GG listes dans le CSV (GG_ECRITURE / GG_LECTURE)
================================================================================

DESCRIPTION :
  - Pour chaque ligne du CSV :
      * Nettoie les membres "groupes non GDL" des GDL cibles
      * Ajoute les GG_ECRITURE dans GDL_ECRITURE
      * Ajoute les GG_LECTURE  dans GDL_LECTURE
  - Ignore proprement les cellules vides
  - Logue les groupes AD introuvables (UTF8-BOM)
  - Une seule barre de progression Write-Progress
  - Messages sans accents (compatibilite)

PREREQUIS :
  - 2.1 : GDL crees (GDL_ECRITURE / GDL_LECTURE renseignes dans CSV)
  - 5.1 : GG crees (version 2-passes recommande)

================================================================================
#>
```

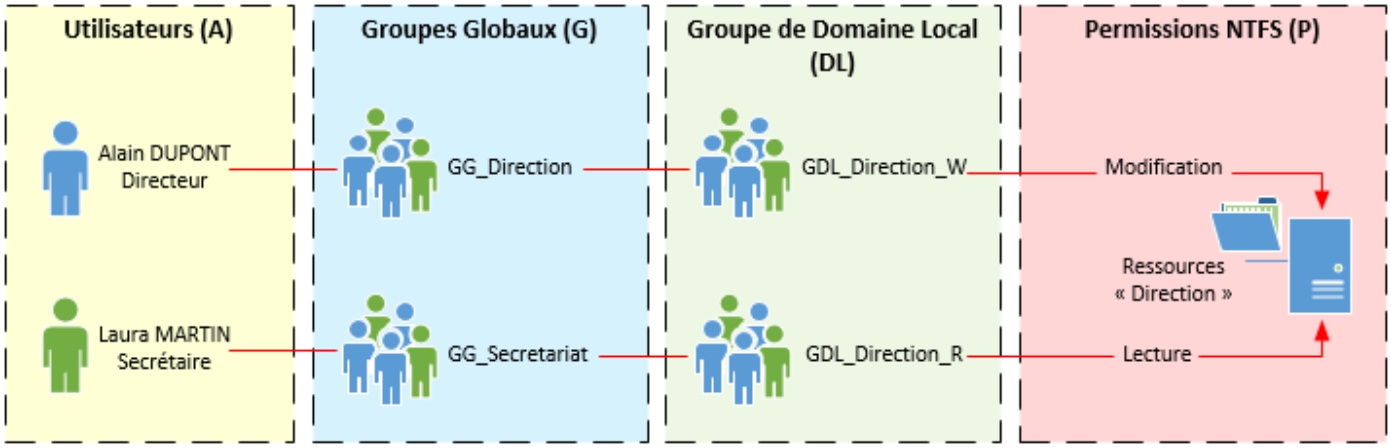
```powershell
Param(
    [string]$CsvPath = "C:\AGDLP\ORG_ARBO_GDL.csv",
    [string]$LogFile = "C:\AGDLP\GG_to_GDL_Errors.txt"
)

$ErrorActionPreference = "Stop"

# ------------------------------------------------------------------------
# 0) Chargement module Active Directory
# ------------------------------------------------------------------------
try { Import-Module ActiveDirectory -ErrorAction Stop }
catch { Write-Error "Module ActiveDirectory introuvable (installer RSAT)."; exit 1 }

# ------------------------------------------------------------------------
# 1) Initialisation log
# ------------------------------------------------------------------------
$utf8Bom = New-Object System.Text.UTF8Encoding($true)
[System.IO.File]::WriteAllText($LogFile, "", $utf8Bom)

# ------------------------------------------------------------------------
# 2) Lecture / validations de base
# ------------------------------------------------------------------------
if (-not (Test-Path $CsvPath)) { Write-Error "CSV introuvable : $CsvPath"; exit 1 }

$rows  = Import-Csv -Path $CsvPath -Delimiter ";" -Encoding UTF8
$total = $rows.Count
$index = 0

Write-Host "Ajout des GG dans les GDL a partir : $CsvPath"
Write-Host "Log : $LogFile"
Write-Host ""

# ------------------------------------------------------------------------
# 3) Fonctions utilitaires
# ------------------------------------------------------------------------

# Nettoyage leger des cellules (supprime NBSP, guillemets typographiques, espaces parasites)
function Clean-Cell([string]$s) {
    if (-not $s) { return "" }
```

```
    $s = $s -replace " ", " "
    $s = $s -replace "\u00A0", " "  # NBSP
    $s = $s -replace "\u200B", ""   # ZWSP
    $s = $s -replace "[«»“”‘’]", ""
    return ($s -replace "\s+", " ").Trim()
}

# Transforme "A, B , C" -> tableau ["A","B","C"] en nettoyant chaque element
function Parse-GGList([string]$cell) {
    $cell = Clean-Cell $cell
    if (-not $cell) { return @() }
    return $cell.Split(",") | ForEach-Object { Clean-Cell $_ } | Where-Object { $_ -ne "" }
}

# ------------------------------------------------------------------------
# 4) Parcours des lignes
# ------------------------------------------------------------------------
foreach ($row in $rows) {
    $index++
    Write-Progress -Activity "Ajout GG -> GDL" -Status "$index / $total" -PercentComplete
(($index / $total) * 100)

    # Recup GDL cibles
    $gdlE = Clean-Cell $row.GDL_ECRITURE
    $gdlL = Clean-Cell $row.GDL_LECTURE

    # Rien a faire si pas de GDL cible
    if ([string]::IsNullOrWhiteSpace($gdlE) -and [string]::IsNullOrWhiteSpace($gdlL)) {
continue }

    # Parse des listes de GG declarees
    $ggE = Parse-GGList $row.GG_ECRITURE
    $ggL = Parse-GGList $row.GG_LECTURE

    # ---------------------------
    # 4.1) Nettoyage GDL_ECRITURE
    # ---------------------------
    if ($gdlE) {
        # On ne retire que les "groupes non GDL_*" (on conserve chainages/parents GDL)
        $currentE = Get-ADGroupMember -Identity $gdlE -ErrorAction SilentlyContinue |
```

```
                    Where-Object { $_.objectClass -eq 'group' -and $_.Name -notlike "GDL_*" }
        if ($currentE) {
            try {
                Remove-ADGroupMember -Identity $gdlE -Members $currentE -Confirm:$false -
ErrorAction SilentlyContinue
                Write-Host "Nettoye : $gdlE (groupes non GDL)"
            }
            catch {
                $msg = "Nettoyage impossible GDL_ECRITURE=$gdlE : $_`r`n"
                [System.IO.File]::AppendAllText($LogFile, $msg, $utf8Bom)
                Write-Warning "Nettoyage partiel : $gdlE (voir log)."
            }
        }
    }

    # -------------------------
    # 4.2) Nettoyage GDL_LECTURE
    # -------------------------
    if ($gdlL) {
        $currentL = Get-ADGroupMember -Identity $gdlL -ErrorAction SilentlyContinue |
                    Where-Object { $_.objectClass -eq 'group' -and $_.Name -notlike "GDL_*" }
        if ($currentL) {
            try {
                Remove-ADGroupMember -Identity $gdlL -Members $currentL -Confirm:$false -
ErrorAction SilentlyContinue
                Write-Host "Nettoye : $gdlL (groupes non GDL)"
            }
            catch {
                $msg = "Nettoyage impossible GDL_LECTURE=$gdlL : $_`r`n"
                [System.IO.File]::AppendAllText($LogFile, $msg, $utf8Bom)
                Write-Warning "Nettoyage partiel : $gdlL (voir log)."
            }
        }
    }

    # --------------------------------
    # 4.3) Ajout des GG_ECRITURE -> GDL_ECRITURE
    # --------------------------------
    if ($gdlE -and $ggE.Count -gt 0) {
        foreach ($m in $ggE) {
```

```
            $grp = Get-ADGroup -Identity $m -ErrorAction SilentlyContinue
            if (-not $grp) {
                $msg = "GG_ECRITURE introuvable : $m (cible : $gdlE)`r`n"
                [System.IO.File]::AppendAllText($LogFile, $msg, $utf8Bom)
                Write-Warning "GG introuvable : $m -> ignore"
                continue
            }
            Add-ADGroupMember -Identity $gdlE -Members $grp -ErrorAction SilentlyContinue
            Write-Host "Ajoute : $m -> $gdlE"
        }
    }

    # ------------------------------
    # 4.4) Ajout des GG_LECTURE -> GDL_LECTURE
    # ------------------------------
    if ($gdlL -and $ggL.Count -gt 0) {
        foreach ($m in $ggL) {
            $grp = Get-ADGroup -Identity $m -ErrorAction SilentlyContinue
            if (-not $grp) {
                $msg = "GG_LECTURE introuvable : $m (cible : $gdlL)`r`n"
                [System.IO.File]::AppendAllText($LogFile, $msg, $utf8Bom)
                Write-Warning "GG introuvable : $m -> ignore"
                continue
            }
            Add-ADGroupMember -Identity $gdlL -Members $grp -ErrorAction SilentlyContinue
            Write-Host "Ajoute : $m -> $gdlL"
        }
    }
}

Write-Progress -Activity "Ajout GG -> GDL" -Completed

Write-Host ""
Write-Host "Alignement GG -> GDL termine. Voir le log si des avertissements apparaissent."
```

Script Master



Introduction

L'ensemble des scripts vu précédemment est découpé d'une façon logique.
Ainsi, il est possible de modifier le fichier source CSV et relancer les scripts sans devoir relancer toute la machine.

Pour simplifier encore plus tout cela, on peut faire un script master qui gère tout ça.
Voici ce qu'il peut faire :

- Lancer tout les script déjà vu
- Lancer chaque étape individuellement
- Préparer le CSV (nettoyage, tri, export)

Powershell

Voici donc le script :

```
<#
================================================================================
SCRIPT : 0 - Maitre_AGDLP.ps1
OBJET  : Piloter l'ensemble du deploiement AGDLP + utilitaires CSV GG
================================================================================

FONCTIONS MAJEURES
------------------
- Matrice de progression multi-etapes (barre ASCII 40)
```

```
- Validations souples (colonnes minimales uniquement avant 1.1/2.1)
- Menu :
    1) Tout lancer (1.1 -> 2.1 -> 3.1 -> 4.1 -> 5.1 -> 6.1)
    2..7) Etapes individuelles
    8) Audit / Dry-run (dossiers, groupes AD manquants)
    9) Preparer CSV GG (nettoyage + tri topologique + export)
   10) Quitter
- Audit GG : groupes references (dans Arbo) introuvables dans AD, suggestions
- Sorties visibles sans accents (compatibilite), commentaires en francais

REMARQUES
---------
- Les scripts 1.1 a 6.1 doivent exister dans $ScriptsDir

================================================================================
#>

Param(
    [string]$CsvArbo     = "C:\Scripts\Arbo\LCC_Arbo_DATAS.csv",
    [string]$CsvGG       = "C:\Scripts\Arbo\GG_LCC_SVC.csv",
    [string]$Share       = "\\10.0.10.182\datas\",
    [string]$BaseOU_GDL  = "OU=GDL,OU=AGDLP,DC=ccpl,DC=local",
    [string]$BaseOU_GG   = "OU=GG,OU=AGDLP,DC=ccpl,DC=local",
    [string]$ScriptsDir  = "C:\Scripts\Arbo",
    [string]$OutDir      = "C:\Scripts\Arbo\Out"
)

$ErrorActionPreference = "Stop"

# ================================================================================
#  UTILITAIRES AFFICHAGE (barre ASCII 40 + matrice)
# ================================================================================
$AsciiBarLen = 40

function New-AsciiBar {
    param([double]$Percent)
    $Percent = [Math]::Max(0, [Math]::Min(100, $Percent))
    $filled  = [Math]::Floor(($Percent/100) * $AsciiBarLen)
    if ($filled -gt $AsciiBarLen) { $filled = $AsciiBarLen }
    $hash = "#" * $filled
```

```
    $dash = "-" * ($AsciiBarLen - $filled)
    return "[{0}{1}] {2}%" -f $hash, $dash, ([Math]::Round($Percent))
}

# Etapes
$Steps = @(
    @{ Id=1; Name="1/6  Arborescence";   File="1.1-Creation_Arborescence.ps1"; Args={@("-
CsvPath `"$CsvArbo`"","-Destination `"$Share`"")};  State="EN ATTENTE"; Pct=0 },
    @{ Id=2; Name="2/6  GDL + OU";       File="2.1-Creation_GDL.ps1";          Args={@("-
CsvPath `"$CsvArbo`"","-BaseOU `"$BaseOU_GDL`"")};  State="EN ATTENTE"; Pct=0 },
    @{ Id=3; Name="3/6  Parentalite";    File="3.1-Creation_Parentalite.ps1";  Args={@("-
CsvPath `"$CsvArbo`"")};                            State="EN ATTENTE"; Pct=0 },
    @{ Id=4; Name="4/6  ACL";            File="4.1-Appliquer_ACL.ps1";         Args={@("-
RootPath `"$Share`"","-CsvPath `"$CsvArbo`"")};    State="EN ATTENTE"; Pct=0 },
    @{ Id=5; Name="5/6  Creation GG";    File="5.1-Creation_GG.ps1";           Args={@("-
CSVPath `"$CsvGG`"","-OUPath `"$BaseOU_GG`"")};    State="EN ATTENTE"; Pct=0 },
    @{ Id=6; Name="6/6  GG -> GDL";      File="6.1-Ajout_GG_dans_GDL.ps1";      Args={@("-
CsvPath `"$CsvArbo`"")};                            State="EN ATTENTE"; Pct=0 }
)

function Draw-Matrix {
    Clear-Host
    Write-Host "====================== PROGRESSION AGDLP ======================"
    foreach ($s in $Steps) {
        $bar = New-AsciiBar -Percent $s.Pct
        Write-Host ("{0,-26} {1,-4} {2}" -f $s.Name, $s.State, $bar)
    }
    Write-Host "================================================================"
}

# ============================================================================
#  OUTILS VALIDATION / AUDIT
# ============================================================================

function Test-Admin {
    return ([Security.Principal.WindowsPrincipal]
[Security.Principal.WindowsIdentity]::GetCurrent()
    ).IsInRole([Security.Principal.WindowsBuiltInRole]::Administrator)
}
```

```
function Require-ModuleAD {
    try { Import-Module ActiveDirectory -ErrorAction Stop; return $true }
    catch { Write-Warning "Module ActiveDirectory manquant."; return $false }
}

function Test-CsvColumns {
    param([string]$Path,[string[]]$RequiredCols)
    if (!(Test-Path $Path)) { return @("Fichier introuvable: $Path") }
    $rows = Import-Csv -Path $Path -Delimiter ';' -Encoding UTF8
    if (!$rows) { return @("CSV vide: $Path") }
    $cols = ($rows | Select-Object -First 1).PSObject.Properties.Name
    $missing = $RequiredCols | Where-Object { $_ -notin $cols }
    return $missing
}

# Nettoyage leger cellules (NBSP, ZWSP, guillemets typographiques, espaces)
function Clean-Cell([string]$s) {
    if (-not $s) { return "" }
    $s = $s -replace " ", " "
    $s = $s -replace "\u00A0", " "
    $s = $s -replace "\u200B", ""
    $s = $s -replace "[«»“”‘’]", ""
    return ($s -replace "\s+", " ").Trim()
}

# Calcul distance d'edition (Levenshtein) 100% ASCII
function Get-EditDistance {
    param(
        [string]$a,
        [string]$b
    )

    if ($null -eq $a) { $a = "" }
    if ($null -eq $b) { $b = "" }

    $la = $a.Length
    $lb = $b.Length

    # matrice (la+1) x (lb+1)
    $d = New-Object 'int[,]' ($la + 1), ($lb + 1)
```

```
    for ($i = 0; $i -le $la; $i++) { $d[$i,0] = $i }
    for ($j = 0; $j -le $lb; $j++) { $d[0,$j] = $j }

    for ($i = 1; $i -le $la; $i++) {
        for ($j = 1; $j -le $lb; $j++) {

            # PAS de ternaire ici. If clair et propre :
            if ($a[$i-1] -eq $b[$j-1]) {
                $cost = 0
            }
            else {
                $cost = 1
            }

            $delete  = $d[$i-1, $j] + 1
            $insert  = $d[$i, $j-1] + 1
            $replace = $d[$i-1, $j-1] + $cost

            $d[$i,$j] = [Math]::Min([Math]::Min($delete, $insert), $replace)
        }
    }

    return $d[$la,$lb]
}
# Validation generale legere
function Validate-Core {
    $ok=$true
    if (!(Test-Admin)) { Write-Warning "Doit etre lance en administrateur."; $ok=$false }
    if (!(Test-Path $Share)) { Write-Warning "Partage introuvable: $Share"; $ok=$false }
    if (!(Require-ModuleAD)) { $ok=$false }
    if (!(Test-Path $CsvArbo)) { Write-Warning "CSV Arbo introuvable: $CsvArbo"; $ok=$false }
    if (!(Test-Path $CsvGG)) { Write-Warning "CSV GG introuvable: $CsvGG"; $ok=$false }
    if (!(Test-Path $ScriptsDir)) { Write-Warning "ScriptsDir introuvable: $ScriptsDir";
$ok=$false }

    $reqMin = @("NIVEAU 1","NIVEAU 2","NIVEAU 3","NIVEAU 4","NIVEAU 5","NIVEAU 6","NIVEAU
7","NIVEAU 8","CREER_GROUPES")
    $missMin = Test-CsvColumns -Path $CsvArbo -RequiredCols $reqMin
    if ($missMin.Count -gt 0) { Write-Warning "Colonnes minimales manquantes (Arbo):
```

```
$($missMin -join ', ')"; $ok=$false }

    return $ok
}

# Validation specifique par etape (souple)
function Validate-ForStep {
    param([ValidateSet("ARBO","GDL","PARENT","ACL","GG","GGtoGDL","PREP_GG","AUDIT")]
[string]$Step)

    switch ($Step) {
        "ARBO"   { return $true }
        "GDL"    { return $true }
        "PARENT" {
            $need=@("FolderPath","GDL_ECRITURE","GDL_LECTURE","GDL_VUE","Parent_GDL")
            $missing = Test-CsvColumns -Path $CsvArbo -RequiredCols $need
            if ($missing.Count -gt 0) { Write-Warning "Manque pour 3.1: $($missing -join ',
')";
                $ans=Read-Host "Continuer quand meme ? (O/N)"; return ($ans -match '^(?i)o') }
            return $true
        }
        "ACL"    {
            $need=@("FolderPath","GDL_ECRITURE","GDL_LECTURE","GDL_VUE")
            $missing = Test-CsvColumns -Path $CsvArbo -RequiredCols $need
            if ($missing.Count -gt 0) { Write-Warning "Manque pour 4.1: $($missing -join ',
')";
                $ans=Read-Host "Continuer quand meme ? (O/N)"; return ($ans -match '^(?i)o') }
            return $true
        }
        "GG"     {
            $need=@("GG","Membre")
            $missing = Test-CsvColumns -Path $CsvGG -RequiredCols $need
            if ($missing.Count -gt 0) { Write-Warning "Manque pour 5.1: $($missing -join ',
')";
                $ans=Read-Host "Continuer quand meme ? (O/N)"; return ($ans -match '^(?i)o') }
            return $true
        }
        "GGtoGDL" {
            $need=@("FolderPath","GDL_ECRITURE","GDL_LECTURE")
            $missing = Test-CsvColumns -Path $CsvArbo -RequiredCols $need
```

```
            if ($missing.Count -gt 0) { Write-Warning "Manque pour 6.1: $($missing -join ',
')";
                $ans=Read-Host "Continuer quand meme ? (O/N)"; return ($ans -match '^(?i)o') }
            return $true
        }
        "PREP_GG" { return $true }
        "AUDIT"   { return $true }
    }
}

# ================================================================================
#  LANCEMENT D'UNE ETAPE
# ================================================================================
function Run-Step {
    param([ValidateSet("ARBO","GDL","PARENT","ACL","GG","GGtoGDL")] [string]$Step)

    if (-not (Validate-ForStep $Step)) { return }

    # recuperer la structure etape
    $s = switch ($Step) {
        "ARBO"    { $Steps[0] }
        "GDL"     { $Steps[1] }
        "PARENT"  { $Steps[2] }
        "ACL"     { $Steps[3] }
        "GG"      { $Steps[4] }
        "GGtoGDL" { $Steps[5] }
    }

    $s.State = "EN COURS"; $s.Pct = 10; Draw-Matrix

    $path = Join-Path $ScriptsDir $s.File
    if (!(Test-Path $path)) {
        $s.State="ERREUR"; $s.Pct=0; Draw-Matrix
        Write-Warning "Script introuvable: $path"
        return
    }

    try {
        & powershell.exe -ExecutionPolicy Bypass -File $path @(& $s.Args)
        $s.State = "TERMINE"; $s.Pct = 100; Draw-Matrix
```

```
    }
    catch {
        $s.State = "ERREUR"; $s.Pct = 0; Draw-Matrix
        Write-Warning "Erreur a l'execution: $path -> $_"
    }
}

# ================================================================================
#  AUDIT / DRY-RUN
# ================================================================================
function Audit-Only {
    Write-Host ""
    Write-Host "=== AUDIT (dry-run) ==="

    # 1) Dossiers manquants
    $rows = Import-Csv -Path $CsvArbo -Delimiter ';' -Encoding UTF8
    $missingFolders = @()
    foreach ($r in $rows) {
        if ([string]::IsNullOrWhiteSpace($r.FolderPath)) { continue }
        $fp = Join-Path $Share $r.FolderPath
        if (!(Test-Path -LiteralPath $fp)) { $missingFolders += $fp }
    }
    Write-Host "Dossiers manquants : $($missingFolders.Count)"
    if ($missingFolders.Count -gt 0) { $missingFolders | Select-Object -Unique | ForEach-
Object { "  - $_" } }

    # 2) Groupes GDL manquants (si colonnes presentes)
    $cols = ($rows | Select-Object -First 1).PSObject.Properties.Name
    if (@("GDL_ECRITURE","GDL_LECTURE","GDL_VUE","Parent_GDL") | ForEach-Object { $_ -in $cols
} | Where-Object {$_} | Measure-Object | Select-Object -ExpandProperty Count) {
        $needGroups = @()
        foreach ($r in $rows) {
            foreach ($g in @($r.GDL_ECRITURE,$r.GDL_LECTURE,$r.GDL_VUE,$r.Parent_GDL)) {
                if ([string]::IsNullOrWhiteSpace($g)) { continue }
                if (!(Get-ADGroup -Filter "Name -eq '$g'" -ErrorAction SilentlyContinue)) {
$needGroups += $g }
            }
        }
        Write-Host "GDL manquants : $($needGroups.Count)"
        if ($needGroups.Count -gt 0) { $needGroups | Select-Object -Unique | ForEach-Object {
```

```
"  - $_" } }
    } else {
        Write-Host "(Colonnes GDL_* absentes pour l'instant — lancer 2.1 pour les creer.)"
    }

    # 3) Audit GG references dans CSV Arbo vs AD
    $ggRefs = @()
    foreach ($r in $rows) {
        foreach ($cell in @($r.GG_ECRITURE,$r.GG_LECTURE)) {
            $cell = Clean-Cell $cell
            if (-not $cell) { continue }
            $ggRefs += ($cell.Split(",") | ForEach-Object { Clean-Cell $_ } | Where-Object {
$_ -ne "" })
        }
    }
    $ggRefs = $ggRefs | Select-Object -Unique
    $missingGG = @()
    foreach ($g in $ggRefs) {
        if (-not (Get-ADGroup -Identity $g -ErrorAction SilentlyContinue)) { $missingGG += $g
}
    }
    Write-Host "GG references mais introuvables dans AD : $($missingGG.Count)"
    if ($missingGG.Count -gt 0) {
        # suggestions
        $allKnown = Get-ADGroup -LDAPFilter "(cn=LCC_GG_*)" -SearchScope Subtree -ErrorAction
SilentlyContinue | Select-Object -ExpandProperty Name
        foreach ($m in $missingGG) {
            $best=""; $bestScore=9999
            foreach ($k in $allKnown) {
                $d = Get-EditDistance $m $k
                if ($d -lt $bestScore) { $bestScore=$d; $best=$k }
            }
            if ($best) { Write-Host ("  - {0}  (suggestion: {1})" -f $m, $best) } else {
Write-Host ("  - {0}" -f $m) }
        }
    }
    Read-Host "Fin audit - Entrer pour continuer" | Out-Null
}

# ================================================================================
```

```
#  PREPARATION CSV GG (nettoyage + tri topologique + export)
# ============================================================================

function Prepare-GG-CSV {
    if (!(Test-Path $CsvGG)) { Write-Warning "CSV GG introuvable: $CsvGG"; return }

    if (!(Test-Path $OutDir)) { New-Item -ItemType Directory -Path $OutDir -Force | Out-Null }
    $out = Join-Path $OutDir "GG_LCC_SVC.sorted.csv"

    Write-Host "Lecture et nettoyage de $CsvGG ..."
    $df = Import-Csv -Path $CsvGG -Delimiter ';' | ForEach-Object {
        [pscustomobject]@{
            GG     = (Clean-Cell $_.GG)
            Membre = (Clean-Cell $_.Membre)
        }
    }

    # Supprimer lignes totalement vides
    $df = $df | Where-Object { $_.GG -or $_.Membre }

    # Construire dependances : parent depend de chaque Membre *_GG_*
    $parents = @{}
    $refs    = @{}
    foreach ($r in $df) {
        if (-not $r.GG) { continue }
        if (-not $parents.ContainsKey($r.GG)) { $parents[$r.GG] = New-Object
System.Collections.Generic.HashSet[string] ([StringComparer]::OrdinalIgnoreCase) }
        if ($r.Membre -and $r.Membre -like "*_GG_*" -and $r.Membre -ne $r.GG) {
            if (-not $refs.ContainsKey($r.Membre)) { $refs[$r.Membre] = $true }
            [void]$parents[$r.GG].Add($r.Membre)
        }
    }

    # Ensemble des noeuds
    $all = New-Object System.Collections.Generic.HashSet[string]
([StringComparer]::OrdinalIgnoreCase)
    foreach ($k in $parents.Keys) { [void]$all.Add($k) }
    foreach ($k in $refs.Keys)    { [void]$all.Add($k) }

    # Calcul indegree pour Kahn
```

```
$indeg = @{}
foreach ($n in $all) { $indeg[$n] = 0 }
foreach ($p in $parents.Keys) {
    foreach ($c in $parents[$p]) {
        if (-not $indeg.ContainsKey($c)) { $indeg[$c] = 0 }
        $indeg[$p] += 0 # ensure key
        $indeg[$p] = $indeg[$p]       # parent indeg sera incremente ci-dessous
    }
}
# indegree = nb de dependances (c -> p) : parent a +1 par enfant reference
foreach ($p in $parents.Keys) {
    foreach ($c in $parents[$p]) {
        $indeg[$p] = $indeg[$p] + 1
    }
}

# File des noeuds indeg 0
$queue = New-Object System.Collections.Queue
foreach ($n in $all) { if ($indeg[$n] -eq 0) { $queue.Enqueue($n) } }

$order = New-Object System.Collections.ArrayList
while ($queue.Count -gt 0) {
    $n = $queue.Dequeue()
    [void]$order.Add($n)
    foreach ($p in $parents.Keys) {
        if ($parents[$p].Contains($n)) {
            $indeg[$p] = $indeg[$p] - 1
            [void]$parents[$p].Remove($n)
            if ($indeg[$p] -eq 0) { $queue.Enqueue($p) }
        }
    }
}

# Cycles residuels
$cyclic = @()
foreach ($n in $indeg.Keys) { if ($indeg[$n] -gt 0) { $cyclic += $n } }

# Reconstruire CSV trie : pour chaque GG dans l'ordre topo, garder ses lignes
$declares = ($df | Select-Object -ExpandProperty GG -Unique)
$blocks = New-Object System.Collections.ArrayList
```

```
    foreach ($g in $order) {
        if ($declares -contains $g) {
            $blk = $df | Where-Object { $_.GG -eq $g }
            foreach ($r in $blk) { [void]$blocks.Add($r) }
        }
    }
    # Ajouter ceux eventuellement manques
    foreach ($g in $declares) {
        if (-not ($order -contains $g)) {
            $blk = $df | Where-Object { $_.GG -eq $g }
            foreach ($r in $blk) { [void]$blocks.Add($r) }
        }
    }

    # Export
    $blocks | Export-Csv -Path $out -Delimiter ';' -NoTypeInformation -Encoding UTF8
    Write-Host "CSV GG trie ecrit : $out"

    if ($cyclic.Count -gt 0) {
        Write-Warning "Attention: dependances cycliques detectees:"
        $cyclic | Select-Object -Unique | ForEach-Object { "  - $_" }
    }

    $ans = Read-Host "Remplacer le CSV GG d'origine par la version triee ? (O/N)"
    if ($ans -match '^(?i)o') {
        Copy-Item -Path $out -Destination $CsvGG -Force
        Write-Host "Remplacement effectue."
    }
    Read-Host "Fin preparation GG - Entrer pour continuer" | Out-Null
}

# ================================================================================
#  MENU PRINCIPAL
# ================================================================================
function Main-Menu {
    do {
        Draw-Matrix
        Write-Host ""
        Write-Host "=== MENU AGDLP ==="
        Write-Host "1) Tout lancer (1.1 -> 2.1 -> 3.1 -> 4.1 -> 5.1 -> 6.1)"
```

```
        Write-Host "2) 1.1 - Creer arborescence"
        Write-Host "3) 2.1 - Creer GDL + OU"
        Write-Host "4) 3.1 - Parentalite GDL"
        Write-Host "5) 4.1 - Appliquer ACL"
        Write-Host "6) 5.1 - Creer GG"
        Write-Host "7) 6.1 - Ajouter GG -> GDL"
        Write-Host "8) Audit / Dry-run"
        Write-Host "9) Preparer CSV GG (nettoyage + tri topo)"
        Write-Host "10) Quitter"

        $coreOK = Validate-Core
        $choice = Read-Host "Choix"

        if (-not $coreOK -and $choice -notin @('8','9','10')) {
            Write-Warning "Pre-requis generaux non valides. Corriger puis relancer."
            Read-Host "Entrer pour continuer" | Out-Null
            continue
        }

        switch ($choice) {
            '1' { Run-Step ARBO; Run-Step GDL; Run-Step PARENT; Run-Step ACL; Run-Step GG;
Run-Step GGtoGDL; Read-Host "Flux complet termine - Entrer pour continuer" | Out-Null }
            '2' { Run-Step ARBO;     Read-Host "Entrer pour continuer" | Out-Null }
            '3' { Run-Step GDL;      Read-Host "Entrer pour continuer" | Out-Null }
            '4' { Run-Step PARENT;   Read-Host "Entrer pour continuer" | Out-Null }
            '5' { Run-Step ACL;      Read-Host "Entrer pour continuer" | Out-Null }
            '6' { Run-Step GG;       Read-Host "Entrer pour continuer" | Out-Null }
            '7' { Run-Step GGtoGDL;  Read-Host "Entrer pour continuer" | Out-Null }
            '8' { Audit-Only }
            '9' { Prepare-GG-CSV }
            '10' { return }
            default { }
        }
    } while ($true)
}

# Lancement
Main-Menu
```

Utilisation

On peut ouvrir le script dans Powershell ISE ou Powershell basique, et au moment de l'exécution, nous avons ce retour :

```
====================== PROGRESSION AGDLP ======================
1/6  Arborescence          EN ATTENTE [--------------------------------------] 0%
2/6  GDL + OU              EN ATTENTE [--------------------------------------] 0%
3/6  Parentalite           EN ATTENTE [--------------------------------------] 0%
4/6  ACL                   EN ATTENTE [--------------------------------------] 0%
5/6  Creation GG           EN ATTENTE [--------------------------------------] 0%
6/6  GG -> GDL             EN ATTENTE [--------------------------------------] 0%
================================================================


=== MENU AGDLP ===
1) Tout lancer (1.1 -> 2.1 -> 3.1 -> 4.1 -> 5.1 -> 6.1)
2) 1.1 - Creer arborescence
3) 2.1 - Creer GDL + OU
4) 3.1 - Parentalite GDL
5) 4.1 - Appliquer ACL
6) 5.1 - Creer GG
7) 6.1 - Ajouter GG -> GDL
8) Audit / Dry-run
9) Preparer CSV GG (nettoyage + tri topo)
10) Quitter
Choix :
```

On peut alors choisir les actions à mener, une barre de progression s'affiche pour chacun des 6 scripts principaux afin de voir ou se trouve l'éventuel blocage.