Documentation scripts PowerShell

Date	Version	Description
13/02/2026	1.0	Première version

Prérequis pour le lancement

Les scripts doivent être exécutés sur la machine **AD**, depuis un compte disposant de droits **Administrateur de domaine**.

Avant de lancer les scripts, il faut :

- Placer les 4 fichiers `.ps1` et le fichier `.csv` dans **le même répertoire**
- Avoir déjà créé un **dossier partagé** sur le serveur de fichiers
- Ce dossier partagé doit être **accessible depuis la machine où les scripts PowerShell sont lancés (testez l'accès UNC avant de commencer : `\\serveur\nom-du-partage`)

Structure

Le script se compose de quatre fichiers PS1 : trois scripts et un menu pour les lancer (successivement ou individuellement).

- Menu.ps1
- CreateStructure.ps1
- CreateGroups.ps1
- AssignUserGroups.ps1

Tous les scripts reposent sur un seul fichier `.csv`, générable depuis l'interface web.
Ce fichier `.csv` doit impérativement respecter le format suivant :

Dossier;SousDossier;NomPrenom;Droit

Exemple de ligne valide :
`00-DIRECTION;00-01-sous-dossier 1;Kad Merad;M`

- `Dossier` : dossier racine, doit commencer par `XX-` (ex. `00-DIRECTION`)
- `SousDossier` : sous-dossier, doit commencer par `XX-YY-` (ex. `00-01-sous-dossier 1`)
- `NomPrenom` : nom complet de l'utilisateur (ex. `Kad Merad`)
- `Droit` :
  - Vide → accès racine uniquement
  - `L` → lecture au niveau du sous-dossier
  - `M` → modification au niveau du sous-dossier

CreateStructure.ps1

Ce script lit le fichier CSV et crée automatiquement l'arborescence de dossiers correspondante.

- Le script cherche automatiquement un unique fichier **CSV** dans le même dossier que le script et utilise le premier trouvé s'il y en a plusieurs.
- Il affiche le nom du CSV retenu puis demande le chemin du dossier racine dans lequel l'arborescence sera créée, en proposant de le créer s'il n'existe pas.
- Le CSV est importé avec les colonnes `Dossier;SousDossier;NomPrenom;Droit` (sans en-tête dans le fichier), la séparation se faisant sur le `;`.
- Chaque ligne est vérifiée : `Dossier` doit commencer par `XX-` et `SousDossier` par `XX-YY-` (par exemple `00-DIRECTION` et `00-01-sous-dossier 1`), sinon la ligne est ignorée.
- Pour chaque entrée valide, le script crée d'abord le dossier principal (`Dossier`) sous le répertoire de base, en évitant les doublons grâce à un HashSet.
- Le sous-dossier est créé en supprimant le premier préfixe `XX-` du champ `SousDossier` pour ne garder que la partie relative (ex. `00-01-sous-dossier 1` devient `01-sous-dossier 1` sous `00-DIRECTION`).
- En fin d'exécution, le script affiche le nombre total de dossiers créés ainsi que le chemin de la racine.

CreateGroups.ps1

Ce script parcourt l'arborescence de dossiers et crée les groupes AD nécessaires, puis applique les droits NTFS RO/RW de manière hiérarchique.

- Le script commence par vérifier la présence du module **ActiveDirectory**, puis demande le chemin UNC du répertoire de base (ex. `\\ST550-FICHIERS\Test2$`) et le DN de l'OU dans laquelle les groupes seront créés.
- Il parcourt récursivement tous les dossiers sous ce répertoire, calcule le chemin relatif et impose que chaque niveau commence par un préfixe `XX-` (sinon le dossier est ignoré avec un message de format invalide).
- À partir de la liste des segments de chemin (`00-DIRECTION`, `01-SERVICE`, etc.), il extrait les numéros et construit un préfixe de groupe de type `GG-00-01-...`.
- Pour chaque dossier, il détermine le niveau dans l'arborescence : au niveau 1, seul un groupe `RO` est créé, à partir du niveau 2, des groupes `RO` et `RW` sont gérés.
- Pour chaque suffixe (`RO` ou `RW`), le script vérifie si le groupe AD existe déjà, le crée en groupe global de sécurité dans l'OU indiquée si besoin, avec une description incluant le type d'accès et les segments de chemin.
- Avant d'appliquer les droits NTFS, il supprime toute ACE existante correspondant à ce groupe sur le dossier ciblé pour repartir sur une configuration propre.
- Les groupes `RO` reçoivent des droits `ReadAndExecute` + `Synchronize`, au niveau 1 sans héritage (droits uniquement sur ce

dossier), et aux niveaux inférieurs avec héritage sur les sous-dossiers/fichiers via les flags d'héritage.

- Les groupes `RW` reçoivent des droits `Modify` + `Synchronize` avec héritage sur les dossiers et fichiers en dessous (équivalent d'un `(OI)(CI)(M)` classique).
- À la fin, le script affiche le nombre total de groupes créés et signale les éventuelles erreurs de création ou d'application de permissions.

AssignUserGroups.ps1

Ce script lit le fichier CSV, crée les utilisateurs AD si nécessaire, génère un mot de passe selon une politique prédéfinie, puis les ajoute aux bons groupes GG-* en fonction des chemins et des droits.

- Le script vérifie le module **ActiveDirectory**, détecte automatiquement l'unique fichier CSV présent dans le répertoire courant et refuse de continuer s'il n'y en a aucun ou plusieurs.
- Il demande ensuite le DN de l'OU où créer les utilisateurs et un suffixe UPN (par ex. `@srv-st550.local`), puis vérifie l'existence de l'OU fournie.
- Une fonction de normalisation supprime les accents et met tout en minuscules pour générer un `SamAccountName` au format `prenom.nom`.
- Une fonction génère un mot de passe complexe au format `#Xxx@NnnnN!JJ`, basé sur le prénom, le nom et le jour du mois.
- Pour chaque ligne du CSV, le script extrait `Dossier`, `SousDossier` et `NomPrenom`, contrôle le format des chemins (`Dossier` en `XX-...`, `SousDossier` en `XX-YY-...`), et ignore la ligne si le

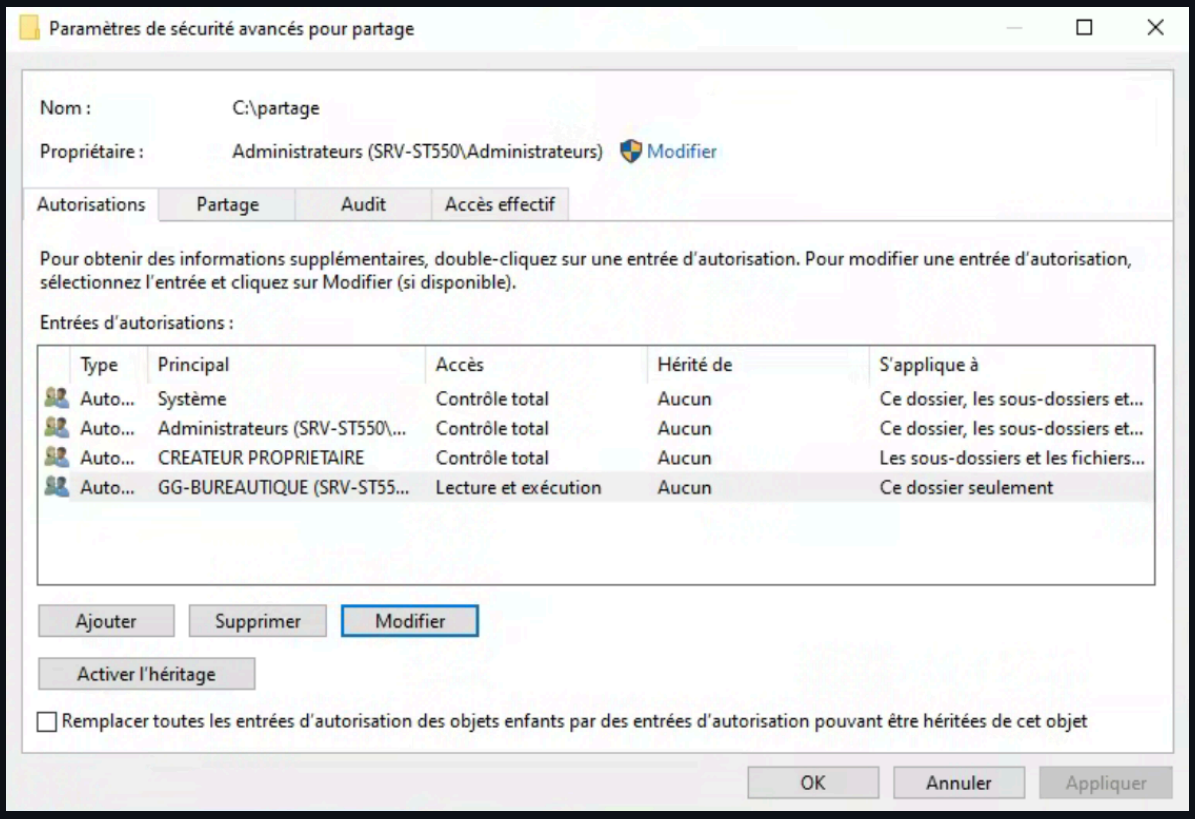
format n’est pas valide.

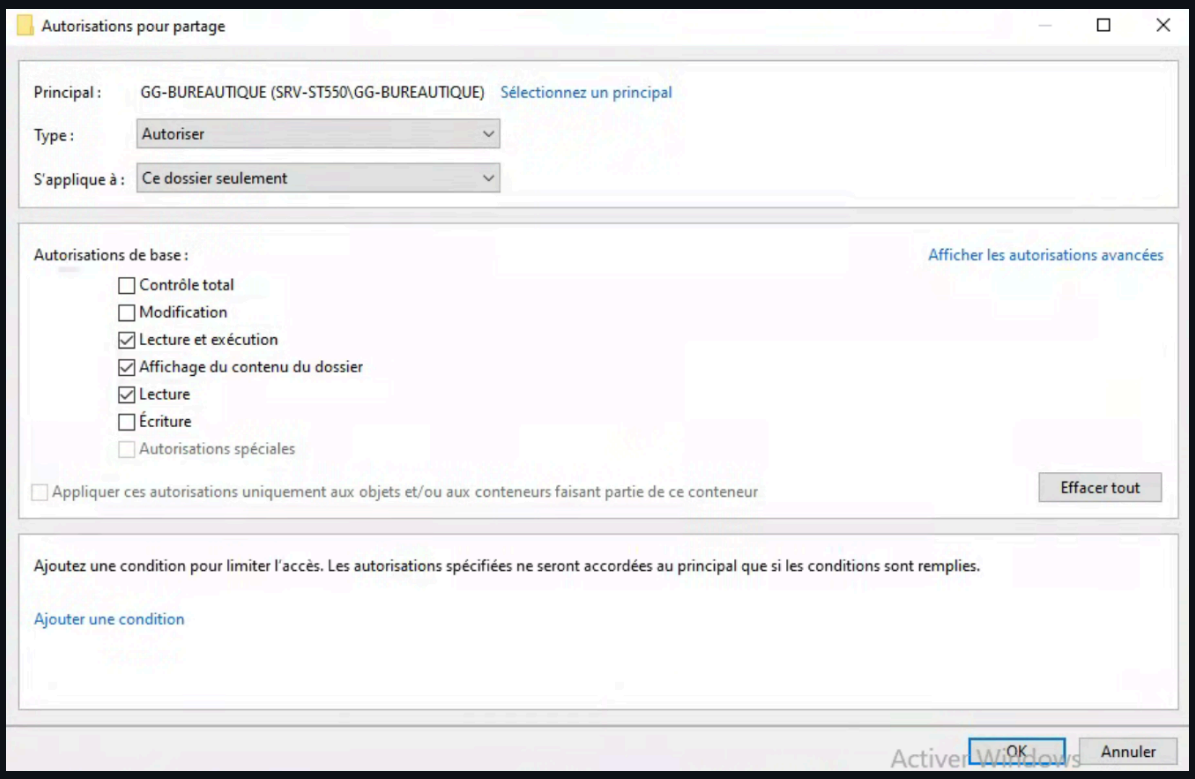
- Le champ `NomPrenom` est découpé pour reconstruire un affichage de type `Prenom NOM`, utilisé comme `DisplayName`, `GivenName` et `Surname` formatés.
- Si le compte existe déjà (même SamAccountName), il met à jour les propriétés de base et signale que l’utilisateur existe ; sinon il crée le compte dans l’OU indiquée, activé, avec mot de passe non expirant et sans demande de changement au premier logon.
- Le script calcule ensuite les noms de groupes à cibler : `GG-XX-RO` pour la racine, `GG-XX-YY-RO` et `GG-XX-YY-RW` pour le sous-dossier, à partir des numéros extraits des noms de dossiers.
- Chaque utilisateur est toujours ajouté au groupe racine `GG-XX-RO` ainsi qu’au groupe `GG-BUREAUTIQUE` (dont l’existence est vérifiée au lancement du script).
- Le champ `Droit` détermine l’ajout supplémentaire :
  - vide → seulement `GG-XX-RO` + `GG-BUREAUTIQUE`
  - `L` → ajout à `GG-XX-YY-RO`
  - `M` → ajout à `GG-XX-YY-RW`
  - les autres valeurs étant ignorées avec un message
- Un récapitulatif final affiche le fichier traité, le nombre de lignes lues, d’utilisateurs uniques, de comptes créés, d’ajouts aux groupes et de lignes ignorées/erreurs.

Note importante sur le dossier racine partagé

Le dossier racine partagé (celui qui contient tous les dossiers `xx-...`) doit avoir les permissions NTFS suivantes après exécution des scripts :

- **Désactiver l'héritage** des permissions (option : « Convertir les permissions héritées en permissions explicites sur cet objet »)
- **Supprimer toutes les entrées** correspondant aux utilisateurs du domaine.
- Ajouter le groupe **GG-BUREAUTIQUE** avec le droit **Affichage du contenu du dossier / Lecture des données** (List folder contents / Read Data), appliqué uniquement sur **Ce dossier uniquement** (ne pas cocher « Appliquer aux sous-dossiers et fichiers »)





Cela permet aux utilisateurs de voir la liste des dossiers racine sans pouvoir y accéder directement (leurs droits réels proviennent des groupes GG-XX-RO / GG-XX-YY-* créés par le script).