

## AD ENTREPRISE

PS C:\Windows\system32> **Get-PSRepository**

```
PS C:\Windows\system32> Get-PSRepository

Name                InstallationPolicy SourceLocation
----                -
PSGallery           Untrusted         https://www.powershellgallery.com/api/v2
```

PS C:\Windows\system32> **Set-PSRepository -Name "PSGallery" -InstallationPolicy Trusted**

PS C:\Windows\system32> **Get-PSRepository**

```
PS C:\Windows\system32> Get-PSRepository

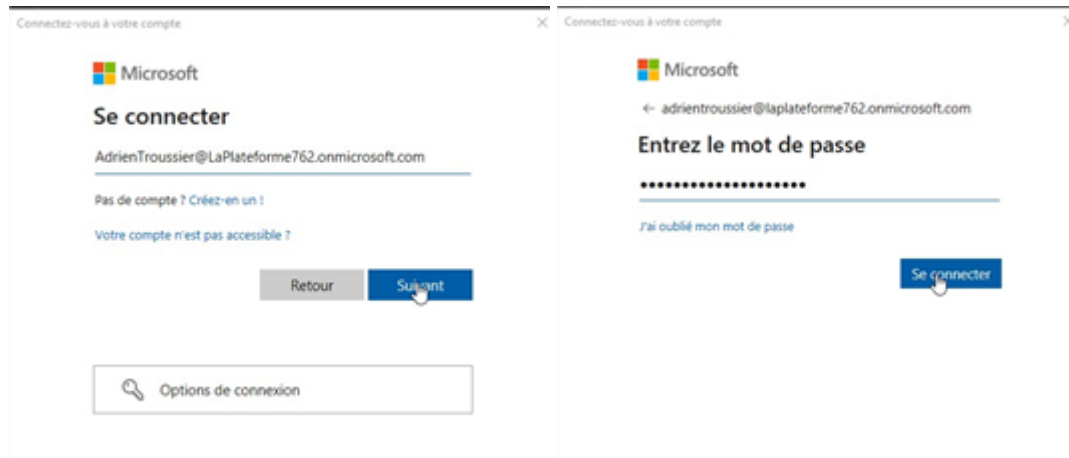
Name                InstallationPolicy SourceLocation
----                -
PSGallery           Trusted          https://www.powershellgallery.com/api/v2
```

PS C:\Windows\system32> **Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned**

PS C:\Windows\system32> **Install-Module -Name Microsoft.Graph**

PS C:\Windows\system32> **Update-Module -Name Microsoft.Graph**

PS C:\Windows\system32> **Connect-MgGraph -Scopes "User.ReadWrite.All, Directory.ReadWrite.All, Group.ReadWrite.All, Directory.AccessAsUser.All Policy.ReadWrite.All"**



- 1) S'authentifier via le navigateur <https://microsoft.com/devicelogin> sur une app mobile (Authenticator)

## Microsoft Authenticator

### Scanner le code QR

Utilisez l'application Microsoft Authenticator pour scanner le code QR. Ceci permet de connecter l'application Microsoft Authenticator à votre compte.

Après avoir scanné le code QR, cliquez sur « Suivant ».



Impossible de numériser l'image ?

Précédent

Suivant

## 2) Scanner le Qrcode avec votre téléphone

### Opération réussie

Bravo ! Vous avez correctement configuré vos informations de sécurité. Cliquez sur « Terminé » pour poursuivre la connexion.

#### Méthode de connexion par défaut :



Microsoft Authenticator

Terminé

## Accès Conditionnelle



## Sécurité | Prise en main

Rechercher



### Prise en main



Diagnostiquer et résoudre les problèmes



Protéger



Accès conditionnel



Identity Protection



Centre de sécurité



ID vérifié

# Stratégies ...

- + Nouvelle stratégie
- + Nouvelle stratégie à partir de modèles
- ↑ Charger un fichier de stratégie
- What If
- Actualiser
- Fonctionnalités d'évaluation
- ...

Les stratégies d'accès conditionnel de Microsoft Entra sont utilisées pour appliquer des contrôles d'accès afin d'assurer la sécurité de votre organisation. [En savoir plus](#)

Toutes les stratégies

Stratégies gérées par Microsoft

1

0

sur 1

Total

Rechercher

Ajouter un filtre

1 sur 1 stratégie trouvée

Nom de stratégie	État	Date de création	Date de modification	
<a href="#">mfa pour les supérieurs</a>	Rapport uniq...	07/12/2024 01:28:50		...

Contrôlez l'accès en fonction des personnes auxquelles la stratégie s'applique, telles que des utilisateurs et des groupes, des identités de charge de travail, des rôles d'annuaire ou des invités externes en incluant tous les utilisateurs .

Contrôle strict sur l'accès des utilisateurs pour répondre à des niveaux de risque de connexion spécifiques .

What if pour tester

## Activer l'authentification SAML pour les applications critiques.

- 1 Création d'application dans application d'entreprise ( une app déjà déployée ou sa propre app )
- 2 Configurer l'authentification unique avec le protocole SAML (id, url de connexion, url de réponse)
- 3 Ajout d'utilisateur et de groupes
- 4 chargez le fichier de métadonnées qui se trouve dans la partie certificats saml
- 5 test de connexion

## Configuration SAML de base

 Enregistrer |  Des commentaires ?

### Identificateur (ID d'entité) \* ⓘ

ID unique qui identifie votre application à Microsoft Entra ID. Cette valeur doit être unique dans toutes les applications de votre locataire Microsoft Entra. L'identificateur par défaut sera l'audience de la réponse SAML pour l'authentification unique initiée par IDP.

		Par défaut	
<input type="text" value="https://www.mysalesforce.com"/>	✓	<input checked="" type="checkbox"/>	 
<input type="text" value="mysalesforce"/>	✓	<input type="checkbox"/>	 

[Ajouter un identificateur](#)

**Modèles :** https://\*.my.salesforce.com

### URL de réponse (URL Assertion Consumer Service) \* ⓘ

L'URL de réponse correspond à l'emplacement où l'application est supposée recevoir le jeton d'authentification. Cette URL est parfois appelée « Assertion Consumer Service » (ACS) dans SAML.

		Index	Par défaut	
<input type="text" value="https://www.mysalesforce.com/saml_login"/>	✓	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	 

[Ajouter une URL de réponse](#)

**Modèles :** https://MYDOMAIN.my.salesforce.com/<ENTITYID>

### URL de connexion \*

L'URL d'authentification est utilisée si vous souhaitez effectuer une authentification unique initiée par le fournisseur de services. Cette valeur est l'URL de la page de connexion pour votre application. Ce champ n'est pas nécessaire si vous voulez effectuer une authentification unique initiée par le fournisseur d'identité.

\*

✓

**Modèles :** https://MYDOMAIN.my.salesforce.com

## Attribution des utilisateurs/groupes à l'application

### Ajouter une attribution ...

Default Directory

 Les groupes ne sont pas disponibles pour l'attribution en raison du niveau de votre plan Active Directory. Vous pouvez affecter des utilisateurs individuels à l'application.

Utilisateurs

2 utilisateurs sélectionnés.

Sélectionner un rôle \*

System Administrator

## Dropbox Business | Utilisateurs et groupes

Application d'entreprise



× « + Ajouter un utilisateur/groupe ✎ Modifier l'affectation 🗑 Supprimer l'affectation ⋮

- Vue d'ensemble
- Plan de déploiement
- Diagnostic et résoudre les problèmes
- ✓ Gérer
  - Propriétés
  - Propriétaires
  - Rôles et administrateurs
  - Utilisateurs et groupes**
  - Authentification unique
  - Approvisionnement

ⓘ L'application apparaît dans Mes applications pour les utilisateurs attribués. Définissez « Visible pour les utilisateurs ? » sur No empêcher ceci.

Attribuez ici des utilisateurs et des groupes à des rôles d'application pour votre application. Pour créer des rôles d'application utilisez l'[inscription d'application](#)

🔍 Les 200 premiers sont affichés, effectuer un...

Nom d'affichage		Type d'objet
<input type="checkbox"/>	 Tommy Johnson	User
<input type="checkbox"/>	 mansour boumediene	User

*Tester la connexion SSO (uniquement avec un abonnement payant)*

# Script Ajouter Utilisateur

<# Ce script permet de se connecter a Microsoft AzureAD via le api MgGrap ensuite il importe un csv et il nous invite a entré un mot de passe (le MasterPassword de Keepass) il génère un mot de passe aléatoire qu'il sécurise puis ensuite il créer des utilisateur si dans AzureAD puis il créer un entré pour chaque utilisateur dans keepass permettant de stocker les mot de passe de manière sécurisé et affiche un message prouvant que l'utilisateur a bien été créer dans l'azureAD et pour finir il ferme keepass puis se déconnecte de l'AzureAD #>

```
function Generate-Password {
    # Définir la longueur du mot de passe
    $Length = 22

    # Définir les caractères qui peuvent être utilisés dans le mot de passe
    $Chars =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890!@#%&
*()"

    # Générer un mot de passe en sélectionnant des caractères aléatoires
    # Chaque caractère est choisi aléatoirement à partir du jeu de caractères défini
    $Password = -join ((1..$Length) | ForEach-Object { $Chars[(Get-Random -Minimum 0
-Maximum $Chars.length)] })

    # Retourner le mot de passe généré
    return $Password
}

# Se connecter à Microsoft Graph avec les permissions spécifiées pour gérer les utilisateurs
Connect-MgGraph -Scopes "User.ReadWrite.All"

# Importer les informations des recrues depuis un fichier CSV
$Recrues = Import-Csv -Path "C:\starfleet\MembresStarfleet.csv"

# Demander à l'utilisateur d'entrer son mot de passe principal en toute sécurité
$MasterPassword = Read-Host "Veuillez insérer votre mot de passe : " -AsSecureString

# Parcourir chaque recrue dans la liste importée
foreach ($Recrue in $Recrues) {
    try {
        # Générer un mot de passe sécurisé pour la recrue
        $newPassword = Generate-Password

        # Convertir le mot de passe en chaîne sécurisée
        $newPassword = ConvertTo-SecureString -String $newPassword -AsPlainText
        -Force
    }
}
```

```

# Vérifier si le prénom de la recrue est vide ou nul
# Si le prénom est nul ou vide, l'utilisateur est créé sans prénom
if ($Recrue.Surname = 0) {
# Créer un utilisateur dans Microsoft Graph avec les informations fournies et un mot
de passe temporaire
    New-MgUser -UserPrincipalName $Recrue.UserPrincipalName `
        -PasswordProfile @{ ForceChangePasswordNextSignIn=$true;
Password="$newPassword" }`
        -DisplayName $Recrue.DisplayName `
        -GivenName $Recrue.GivenName `
        -JobTitle $Recrue.JobTitle `
        -Department $Recrue.Department `
        -MailNickName $Recrue.MailNickName `
        -AccountEnabled:$true
    }
else {
# Créer un utilisateur dans Microsoft Graph en incluant à la fois le prénom et le nom
de famille
    New-MgUser -UserPrincipalName $Recrue.UserPrincipalName `
        -PasswordProfile @{ ForceChangePasswordNextSignIn=$true;
Password="$newPassword" }`
        -DisplayName $Recrue.DisplayName `
        -GivenName $Recrue.GivenName `
        -Surname $Recrue.Surname `
        -JobTitle $Recrue.JobTitle `
        -Department $Recrue.Department `
        -MailNickName $Recrue.MailNickName `
        -AccountEnabled:$true
    }

# Ajouter une entrée dans KeePass pour l'utilisateur avec le mot de passe généré
New-KeePassEntry -DatabaseProfileName "starfleet" `
    -KeePassEntryGroupPath "starfleet/MemberStarfleet" `
    -Title "Entra" `
    -UserName $Recrue.UserPrincipalName `
    -KeePassPassword $newPassword `
    -MasterKey $MasterPassword `

# Afficher un message confirmant la création de l'utilisateur et son mot de passe
Write-Host "Utilisateur $($Recrue.DisplayName): $_ `nmdp:$newPassword`n"
}
catch {
# Afficher un message d'erreur si la création de l'utilisateur échoue
Write-Host "Erreur lors de la création de $($Recrue.DisplayName): $_"
}
}

```

```
Update-KeePassDatabaseConfiguration -DatabaseProfileName "starfleet"
# Fermer KeePass en forçant l'arrêt du processus si nécessaire
Stop-Process -Name "KeePass" -Force

# Se déconnecter de Microsoft Graph
Disconnect-MgGraph
```

## Script Suppression Utilisateur

```
Connect-MgGraph -Scopes "User.ReadWrite.All"

$keepassDbPath = "C:\Users\Adrien\Documents\Database.kdbx"
$Recrues = Import-CSV "C:\starfleet\MembresStarfleet.csv"

foreach ($Recrue in $Recrues) {
    try {
        Remove-MgUser -UserId $Recrue.UserPrincipalName
        Write-Host "Utilisateur supprimer $($Recrue.DisplayName): $_"
    }
    catch {
        Write-Host "Erreur lors de la suppression de $($Recrue.DisplayName): $_"
    }
}

Get-MgUser
Disconnect-MgGraph
```

## Script CreationGroups

```
Connect-MgGraph "Group.ReadWrite.All"

$Groups = Import-CSV "C:\starfleet\GroupesStarfleet.csv"

foreach($Group in $Groups) {
    try
    {
        New-MgGroup -DisplayName $Group.DisplayName `
        -Description $Group.Description `
        -MailNickname $Group.MailNickname `
        -GroupType @() `
        -SecurityEnabled:$true `
        -MailEnabled:$false

        Write-Host "Group $($Group.DisplayName): $_"
    }
}
```



```

        catch {
            Write-Host "Erreur lors de la création de $($Group.DisplayName):$_"
        }
    }
    Get-MgGroup
    Disconnect-MgGraph

```

## Script Suppression Groups

```

Connect-MgGraph "Group.ReadWrite.All"

$Groups = Get-MgGroup

foreach($Group in $Groups) {
    try
    {
        Remove-MgGroup -GroupId $Group.Id

        Write-Host "Group $($Group.DisplayName): $_"
    }
    catch{
        Write-Host "Erreur lors de la suppression de $($Group.DisplayName):$_"
    }
}

Disconnect-MgGraph

```

## Script Management Groups

```

<#
    Ce script permet de se connecter à AzureAD via l'API Microsoft Graph (MgGraph).
    Il affiche ensuite un menu qui demande à l'utilisateur s'il souhaite supprimer ou
    ajouter des membres dans un groupe spécifique.
    Selon le choix de l'utilisateur, le script exécute l'action correspondante.
#>

# Connexion à l'API Microsoft Graph avec les autorisations nécessaires
Connect-MgGraph "User.ReadWrite.All, Group.ReadWrite.All"

# Récupération des groupes et des utilisateurs dans AzureAD
$Groups = Get-MgGroup
$Users = Get-MgUser -All | Select-Object -Property DisplayName, Id, JobTitle

# Affichage du menu d'action
Write-Host "1 Ajouter"
Write-Host "2 Supprimer`n"

```

```

# Demande à l'utilisateur de choisir une action
$Action = Read-Host "Ajouter ou Supprimer"

# Affichage du choix utilisateur
Switch($Action)
{
    '1' {Write-Host "Vous avez choisi d'ajouter"}
    '2' {Write-Host "Vous avez choisi de supprimer "}
}

# Fonction pour ajouter des membres aux groupes en fonction de leur intitulé de poste
function AjouterMembre{
    # Parcours de chaque utilisateur récupéré
    foreach ($User in $Users) {
        try {
            # Conditions pour affecter les utilisateurs à des groupes selon leur JobTitle
            if ($User.JobTitle -like "**medical**") {
                $Group = Get-MgGroup -Filter "displayName eq 'Service Medical'"
                New-MgGroupMember -GroupId $Group.Id -DirectoryObjectId $User.Id
            }
            elseif ($User.JobTitle -like "**communications**") {
                $Group = Get-MgGroup -Filter "displayName eq 'Communication'"
                New-MgGroupMember -GroupId $Group.Id -DirectoryObjectId $User.Id
            }
            elseif ($User.JobTitle -like "**scientifique**") {
                $Group = Get-MgGroup -Filter "displayName eq 'Scientifiques'"
                New-MgGroupMember -GroupId $Group.Id -DirectoryObjectId $User.Id
            }
            elseif ($User.JobTitle -like "**securite**") {
                $Group = Get-MgGroup -Filter "displayName eq 'Securite'"
                New-MgGroupMember -GroupId $Group.Id -DirectoryObjectId $User.Id
            }
            elseif ($User.JobTitle -like "**conseiller**") {
                $Group = Get-MgGroup -Filter "displayName eq 'Conseillers'"
                New-MgGroupMember -GroupId $Group.Id -DirectoryObjectId $User.Id
            }
            elseif ($User.JobTitle -like "**ingenieur**") {
                $Group = Get-MgGroup -Filter "displayName eq 'Ingenierie'"
                New-MgGroupMember -GroupId $Group.Id -DirectoryObjectId $User.Id
            }
            elseif ($User.JobTitle -like "**navire**" -or $User.JobTitle -like "**timonnier**") {
                $Group = Get-MgGroup -Filter "displayName eq 'Marins'"
                New-MgGroupMember -GroupId $Group.Id -DirectoryObjectId $User.Id
            }
            elseif ($User.JobTitle -notlike "**capitaine**" -and $User.JobTitle -notlike
            "**commandant**" -and $User.JobTitle -notlike "**officier**") {
                $Group = Get-MgGroup -Filter "displayName eq 'Autres'"
            }
        }
    }
}

```

```

        New-MgGroupMember -GroupId $Group.Id -DirectoryObjectId $User.Id
    }
    else {
        Write-Host "$User.DisplayName a été ajouté au groupe :
$($Group.DisplayName):$_"
    }
    # Confirmation de l'ajout
    Write-Host "$User.DisplayName a été ajouté au groupe :
$($Group.DisplayName):$_"
    }
    catch {
        # Gestion des erreurs en cas d'échec
        Write-Host "$User.DisplayName n'a pas été ajouté au groupe :
$($Group.DisplayName):$_"
    }
}

# Ajout des officiers supérieurs
foreach ($User in $Users) {
    try {
        if ($User.JobTitle -like "*capitaine*" -or $User.JobTitle -like "*commandant*" -or
$User.JobTitle -like "*officier*") {
            $Group = Get-MgGroup -Filter "displayName eq 'Officiers Superieurs'"
            New-MgGroupMember -GroupId $Group.Id -DirectoryObjectId $User.Id
            Write-Host "$User.DisplayName a été ajouté au groupe :
$($Group.DisplayName)"
        }
        else {
            Write-Host "$User.DisplayName n'a pas été ajouté au groupe :
$($Group.DisplayName):$_"
        }
    }
    catch {
        Write-Host "$User.DisplayName n'a pas été ajouté au groupe :
$($Group.DisplayName): $_"
    }
}
}

```

```

# Fonction pour supprimer tous les membres des groupes
function SupprimerMembre {
    foreach ($Group in $Groups) {
        # Récupération des membres du groupe
        $Members = Get-MgGroupMember -GroupId $Group.Id
        foreach ($Member in $Members){
            try {
                # Suppression des membres du groupe
            }
        }
    }
}

```

```

        Remove-MgGroupMemberByRef -GroupId $Group.Id -DirectoryObjectId
$Member.Id
        Write-Host "Les utilisateurs du groupe $($Group.DisplayName):$_ ont été
supprimés avec succès."
    }
    catch {
        Write-Host "Les utilisateurs du groupe $($Group.DisplayName):$_ n'ont pas
été supprimés."
    }
}
}
}
}
}

```

```

# Exécution de l'action en fonction du choix de l'utilisateur
if ($Action -eq 1) {
    AjouterMembre
}
elseif ($Action -eq 2) {
    SupprimerMembre
}
else {
    Write-Host "Je n'ai pas compris votre demande."
}

```