

# Configurer une application enregistrée dans Entra ID pour gérer une équipe Sharepoint

## Inscription d'une application

Une inscription d'application dans Entra se fait :

- depuis le **portail Azure**,
- en **CLI/script** avec l'**API Microsoft Graph** par exemple en utilisant **Powershell**.

## Installer le module Microsoft.Graph de Powershell

### Installer Powershell 7

### Installer Microsoft.Graph de Powershell

- prérequis :
  - permissions API (Microsoft Graph) : Sites.Selected (Application)
- ouvrir une session Powershell en tant, qu'administrateur

```
Install-Module Microsoft.Graph
Import-Module Microsoft.Graph
```

## Préparation Azure AD (OAuth2)

### Depuis l'interface d'administration du portail Azure

- création d'une **Inscription d'applications** Entra ID :
  - Portail Azure ⇒ Entra ID
  - Inscription d'applications ⇒ Nouvelle inscription
  - Nom : app
  - Locataire unique seulement
  - S'inscrire
- Récupérer :
  - Tenant ID
  - Client ID
  - l'ajout du certificat se fera ultérieurement (dans Certificates & Secrets)
- Ajouter la permission Microsoft Graph :
  - Autorisations d'application (et non Autorisations déléguées) :
    - Sites.Selected

La permission **Sites.Selected** restreint l'accès au site explicitement autorisé via Graph.

Une autorisation d'application ne nécessite pas de compte utilisateur :

- L'application agit en autonomie (service, script, automatisation).
  - Nécessite un consentement administrateur.
- 
- Demander le consentement de l'administrateur.

### En CLI

- Connexion avec les droits nécessaires sur le tenant Entra ID

```
Connect-MgGraph -Scopes "Application.ReadWrite.All","AppRoleAssignment.ReadWrite.All"
Welcome to Microsoft Graph!
```

```
Connected via delegated access using 14d82eec-204b-4c2f-b7e8-296a70dab67e
```

```
Readme: https://aka.ms/graph/sdk/powershell
SDK Docs: https://aka.ms/graph/sdk/powershell/docs
API Docs: https://aka.ms/graph/docs
```

NOTE: You can use the `-NoWelcome` parameter to suppress this message.

- Création d'une nouvelle application

```
$app = New-MgApplication -DisplayName "Application"
```

- Création du Principal de service associé

```
$sp = New-MgServicePrincipal -AppId $app.AppId
```

## Autoriser l'app enregistrée à mettre à jour les dossiers de l'équipe Teams

Cela ne peut pas se faire depuis l'interface d'administration Entra ID mais en utilisant Powershell.

### Utilisation d'un certificat

- Création d'un certificat

```
$cert = New-SelfSignedCertificate `
-Subject "CN=applicaton" `
-CertStoreLocation "Cert:\CurrentUser\My" `
-KeySpec Signature `
-KeyUsage DigitalSignature `
-KeyExportPolicy Exportable `
-HashAlgorithm SHA256 `
-KeyLength 2048 `
-NotAfter (Get-Date).AddYears(4)
```

- Notez l'empreinte du certificat

```
$cert.Thumbprint
```

- Vérifier que le certificat est BIEN présent au BON endroit dans Cert:\CurrentUser\My

```
Get-Item "Cert:\CurrentUser\My\<TON_THUMBPRINT>"
```

- Vérifier que la CLÉ PRIVÉE est présente dans le certificat

```
$cert = Get-Item "Cert:\CurrentUser\My\<THUMBPRINT>"
$cert.HasPrivateKey
```

`$cert.HasPrivateKey` doit renvoyer True

### Export du certificat PUBLIC (.cer) à importer dans Entra ID

```
Export-Certificate `
-Cert "Cert:\CurrentUser\My\${$cert.Thumbprint}" `
-FilePath "application.cer"
```

- importer le certificat dans l'application à partir du portail Azure
- Test avec PowerShell (certificat local)

```
Connect-MgGraph `
-TenantId "<TENANT_ID>" `
-ClientId "<APP_ID>" `
-CertificateThumbprint "<THUMBPRINT>"
```

=> Résultat attendu  
Plain TextWelcome To Microsoft Graph!

- Puis :

```
Get-MgContext
```

=> on doit voir :

```

ClientId           : <id client>
TenantId           : <ID tenant>
Scopes             : {Sites.Selected}
AuthType           : AppOnly
TokenCredentialType : ClientCertificate
CertificateThumbprint : <empreinte>

```

### Exporter le certificat pour obtenir un PFX PKCS#1 (clé privée incluse)

```

$thumb = "EMPREINTE_DU_CERTIFICAT"

$cert = Get-Item "Cert:\CurrentUser\My\$thumb"

Export-PfxCertificate `
  -Cert $cert `
  -FilePath "graph-auth.pfx" `
  -Password (ConvertTo-SecureString -String "MotDePasseFort123!" -AsPlainText -Force)

```

### Importer un PFX sur une autre machine

```

Import-PfxCertificate `
  -FilePath "fichier.pfx" `
  -CertStoreLocation Cert:\CurrentUser\My `
  -Password (ConvertTo-SecureString -String "MotDePasseFort123!" -AsPlainText)

```

### Générer une clé PKCS#8

- Recharger le PFX

```

$pfXPath = "c:\temp\fichier.pfx"
$pfxPassword = "TempPfxPassword!ChangeMe"

$cert = New-Object System.Security.Cryptography.X509Certificates.X509Certificate2(
  $pfXPath,
  $pfxPassword,
  [System.Security.Cryptography.X509Certificates.X509KeyStorageFlags]::Exportable
)

```

- Récupérer la clé privée RSA

```
$rsa = $cert.PrivateKey
```

- Exporter la clé AU FORMAT PKCS#8

```

# Récupère les paramètres
$params = $rsa.ExportParameters($true)

# Crée un RSA moderne
$rs2 = [System.Security.Cryptography.RSA]::Create()
$rs2.ImportParameters($params)

# Export PKCS#8
$pkcs8 = $rs2.ExportPkcs8PrivateKey()

$base64 = [System.Convert]::ToBase64String(
  $pkcs8,
  [System.Base64FormattingOptions]::InsertLineBreaks
)

$pem = @"
-----BEGIN PRIVATE KEY-----
$base64
-----END PRIVATE KEY-----
"@

```

```
$pem | Set-Content `
-Path C:\Temp\fichier.key `
-Encoding ASCII
```

### Résumé .cer .pfx

Un fichier .cer contient uniquement la **clé publique du certificat**. Cela permet :

- de déclarer le certificat dans Entra ID
- permet à Microsoft de vérifier les signatures

La clé privée est déjà stockée dans Windows :

- dans **Cert:\LocalMachine\My** ou **Cert:\CurrentUser\My**
- aucun PFX n'est nécessaire tant que l'on se connecte depuis le PC où se trouve la clé privée. En indiquant l'empreinte de la clé (thumbprint), cela permet de sélectionner la bonne clé privée.

Un fichier .cer ne permet pas :

- de s'authentifier
- de signer

Un fichier .pfx (format PFX / PKCS#12) contient :

- la clé publique
- la clé privée
- (optionnel) la chaîne de certificats
- et est protégé par une passphrase

C'est le seul format portable qui permet :

- d'importer un certificat avec clé privée
- d'authentifier une application sur une autre machine
- d'utiliser le certificat sur Linux / AWX / Docker / CI/CD

Format	Contient	Usage
.cer	clé publique	Entra ID, SharePoint
.pfx	clé publique + clé privée	Authentification
.pfx + mdp	portable sécurisé	Serveur, automatisation

### Donner accès à l'application sur le site Sharepoint

- l'administrateur doit donner l'accès au site voulu

```
# deconnexion
Disconnect-MgGraph

# Connexion admin
Connect-MgGraph -Scopes "Sites.FullControl.All"

# Récupérer le site
$site = Get-MgSite -SiteId "mondomaine.sharepoint.com/sites/MonSite"

# Donner accès controle total à l'application (fullcontrol / read / write)
New-MgSitePermission `
-SiteId $site.Id `
-Roles "fullcontrol" `
-GrantedToIdentities @{
  Application = @{
    Id = "ID application"
    DisplayName = "application"
  }
}
```

- Se reconnecter en AppOnly :

```
Connect-MgGraph `
-TenantId "<TENANT_ID>" `
```

```
-ClientId "<APP_ID>" `
-CertificateThumbprint "<THUMBPRINT>"
```

- Puis Tester l'accès au site précis:

```
Get-MgSite -SiteId "mondomaine.sharepoint.com:/sites/MonSite"
```

## Créer un dossier dans le site SharePoint

- Identifier le site SharePoint en utilisant le chemin direct :

```
$site = Get-MgSite -SiteId "mondomaine.sharepoint.com:/sites/MonSite"
$site.Id
```

```
=> Afficher le GUID du site du type :
xxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx
```

- Récupérer le drive du site (Documents) : chaque site a une document library principale.

```
$drive = Get-MgSiteDrive -SiteId $site.Id
$drive.Id
```

```
=> Affiche un DriveId
Type : documentLibrary
```

- Créer un dossier À LA RACINE (test simple).

```
New-MgDriveRootChild `
-DriveId $drive.Id `
-AdditionalProperties @{
  "name" = "Test-Dossier"
  "folder" = @{}
  "@microsoft.graph.conflictBehavior" = "rename"
}
```

```
=> Résultat attendu
Id : 01ABCDEF...
Name : Test-Dossier
=> Dossier créé avec succès à vérifier dans SharePoint
```

- lister les dossier à la racine

```
$items = Get-MgDriveRootChild -DriveId $drive.Id
```

- Récupérer le dossier General

```
$parent = $items | Where-Object { $_.Name -eq "General" }
$parent.Id
```

- création du sous-dossier

```
New-MgDriveItemChild `
-DriveId $drive.Id `
-DriveItemId $parent.Id `
-AdditionalProperties @{
  "name" = "Test-SousDossier"
  "folder" = @{}
  "@microsoft.graph.conflictBehavior" = "rename"
}
```

## Installer le module module PnP.PowerShell

PnP.PowerShell supporte :

- L'authentification Graph App-only
- La commande Grant-PnPazureADAppSitePermission
- Les opérations moderne de gestion des Sites.Selected

```
Install-Module PnP.PowerShell -Scope CurrentUser
```

## Se connecter

- génération d'un certificat autosigné

```
$cert = New-SelfSignedCertificate `
-Subject "CN=MinioS3Sync" `
-KeyAlgorithm RSA `
-KeyLength 2048 `
-CertStoreLocation "Cert:\CurrentUser\My" `
-NotAfter (Get-Date).AddYears(3)
```

- Exporter le certificat .CER (public)

```
Export-Certificate `
-Cert $cert `
-FilePath "./minio-s3.cer"
```

- Exporter la clé privée .PFX

```
$password = Read-Host "Mot de passe du certificat" -AsSecureString
```

```
Export-PfxCertificate `
-Cert $cert `
-FilePath "./minio-s3.pfx" `
-Password $password
``
```

- Uploader le fichier minio-s3.cer dans Entra ID
  - Entra ID → App Registration → Ton App → Certificates & secrets → Upload certificate → sélectionne minio-s3.cer
- se connecter

```
Connect-PnPOnline `
-Url "https://educvaladonlimogesfr-admin.sharepoint.com" `
-ClientId "ad39ff78-4116-43d6-86cf-3dfa8f120aa3" `
-Tenant "educvaladonlimogesfr.onmicrosoft.com" `
-CertificatePath "./minio-s3.pfx" `
-CertificatePassword $password
```

## Donner l'autorisation

```
Grant-PnPAzureADAppSitePermission `
-AppId "ad39ff78-4116-43d6-86cf-3dfa8f120aa3" `
-DisplayName "Minio-S3c" `
-Site "https://educvaladonlimogesfr.sharepoint.com/sites/Signaturenumrique" `
-Permissions Write
```

## Tester la commande

```
Connect-PnPOnline -Url "https://<tenant>-admin.sharepoint.com" -ClientId "<appID>" -ClientSecret "<secret>"
```

## Script bash pour configurer l'accès à l'équipe Sharepoint associée à l'équipe Teams

Le script :

- génère un token OAuth2 (client\_credentials)
- Récupère automatiquement le {site-id} via Graph API
- Récupère automatiquement le {drive-id} (bibliothèque SharePoint)
- Assigne les permissions Sites.Selected (write) à l'application inscrite
- Teste l'accès via rclone (liste des fichiers)
- Prérequis :

```
apt install jq
```

[setup\\_sharepoint\\_access.sh](#)

```
#!/usr/bin/env bash

### -----
### CONFIGURATION À RENSEIGNER
### -----

TENANT_ID="XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
CLIENT_ID="XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
CLIENT_SECRET="YOUR_APP_SECRET"

HOSTNAME="contoso.sharepoint.com"          # domaine SharePoint
SITE_PATH="DocumensoSite"                 # nom du site SharePoint
DRIVE_NAME="Documents"                   # nom de la bibliothèque (ex: "Documents")

RCLONE_REMOTE="sharepoint"                # nom du remote rclone
RCLONE_MINIO="s3minio:documenso"         # bucket minio
RCLONE_SHAREPOINT="${RCLONE_REMOTE}:${DRIVE_NAME}"

### -----
echo "1. Obtention du token OAuth2..."

ACCESS_TOKEN=$(
curl -s -X POST \
-H "Content-Type: application/x-www-form-urlencoded" \
-d "client_id=${CLIENT_ID}" \
-d "scope=https://graph.microsoft.com/.default" \
-d "client_secret=${CLIENT_SECRET}" \
-d "grant_type=client_credentials" \
"https://login.microsoftonline.com/${TENANT_ID}/oauth2/v2.0/token" \
| jq -r .access_token
)

if [[ "$ACCESS_TOKEN" == "null" || -z "$ACCESS_TOKEN" ]]; then
    echo "❌ Impossible de générer un token ! Vérifie ton CLIENT_ID / SECRET."
    exit 1
fi

echo " Token obtenu."

### -----
echo "2. Récupération automatique du site-id..."

SITE_INFO=$(curl -s -X GET \
-H "Authorization: Bearer $ACCESS_TOKEN" \
"https://graph.microsoft.com/v1.0/sites/${HOSTNAME}/sites/${SITE_PATH}?$select=id,webUrl")

SITE_ID=$(echo "$SITE_INFO" | jq -r '.id')

if [[ "$SITE_ID" == "null" || -z "$SITE_ID" ]]; then
    echo "❌ Impossible de récupérer le site-id !"
    echo "Réponse brute:"
    echo "$SITE_INFO"
    exit 1
fi

echo " SITE_ID trouvé : $SITE_ID"

### -----
echo "3. Récupération automatique du drive-id de la bibliothèque $DRIVE_NAME..."

DRIVES_LIST=$(curl -s -X GET \
-H "Authorization: Bearer $ACCESS_TOKEN" \
"https://graph.microsoft.com/v1.0/sites/${SITE_ID}/drives")

DRIVE_ID=$(echo "$DRIVES_LIST" | jq -r ".value[] | select(.name==\"$DRIVE_NAME\") | .id")

if [[ -z "$DRIVE_ID" ]]; then
    echo "❌ Impossible de trouver la bibliothèque '$DRIVE_NAME'"
    echo "Réponse brute:"
    echo "$DRIVES_LIST"

```

```
    exit 1
fi

echo "DRIVE_ID trouvé : $DRIVE_ID"

### -----
echo "4. Attribution de la permission Sites.Selected (write) à l'application..."

ASSIGN_RESPONSE=$(curl -s -X POST \
-H "Authorization: Bearer $ACCESS_TOKEN" \
-H "Content-Type: application/json" \
-d "{
  \"roles\": [\"write\"],
  \"grantedToIdentities\": [
    {
      \"application\": {
        \"id\": \"$CLIENT_ID\"
      }
    }
  ]
}" \
"https://graph.microsoft.com/v1.0/sites/$SITE_ID/permissions")

echo " Permission appliquée."
echo "Réponse API :"
echo "$ASSIGN_RESPONSE"

### -----
echo "5. Test d'accès SharePoint avec rclone..."

docker exec -it rclone-sync rclone lsd "$RCLONE_SHAREPOINT"

echo " Configuration terminée avec succès !"
echo "SITE_ID = $SITE_ID"
echo "DRIVE_ID = $DRIVE_ID"
```

- exécuter le script

```
bash setup_sharepoint_access.sh
```

From:  
[/- Les cours du BTS SIO](#)

Permanent link:  
[/doku.php/reseau/cloud/azure/configurerappoursharepoint](#)

Last update: **2026/04/10 22:24**

