API Graph : Créer un dossier avec un accès spécifique pour un utilisateur

Principes

- Utilisation de Microsoft Graph :
  - localisation du dossier **Fichiers** d'un canal Teams
  - création d'un sous-dossier.
- Utilisation de l'API REST SharePoint (avec le même enregistrement d'application) pour :
- rompre l'héritage des autorisations sur ce sous-dossier,
- supprimer les droits des Membres/Visiteurs, de sorte que seuls les Propriétaires conservent un contrôle total,
- accorder un droit de modification (**Edit**) à un compte Entra ID précis.

> **Pourquoi combiner Graph + REST SharePoint ?**
>
> Microsoft Graph expose bien l'API pour cibler le dossier de fichiers d'un canal et créer des dossiers (filesFolder + Drive/children), mais ne propose pas (de façon fiable) une action pour "casser" l'héritage des permissions ; cette opération reste du ressort de l'API REST SharePoint.
>
> - learn.microsoft.com

Pré-requis

- Enregistrement d'application (app-only) dans Entra ID avec consentement administrateur :
  - Microsoft Graph (Application) : Files.ReadWrite.All, Sites.Read.All (ou Sites.ReadWrite.All)
  - SharePoint (Application) : Sites.FullControl.All Ces portées permettent de :
- récupérer filesFolder d'un canal Teams et créer le dossier via Graph,
- rompre l'héritage et gérer les affectations de rôles via SharePoint REST.
- Récupérez TenantId, ClientId, ClientSecret de l'app.

Script Powershell

Paramètres à renseigner

- $TenantId : ID de votre locataire
- $ClientId / $ClientSecret : de l'appli Entra ID
- $TeamId : ID de l'équipe (ID du groupe Microsoft 365)
- $ChannelId : ID du canal (ex. 19:xxxxx@thread.tacv2)
- $FolderName : nom du sous-dossier à créer
- $TargetUser : compte qui doit avoir modification (UPN ou e-mail)

Script

```
# ============================
# 1) Paramètres
# ============================
$TenantId     = "<YOUR_TENANT_ID>"
$ClientId     = "<YOUR_CLIENT_ID>"
$ClientSecret = "<YOUR_CLIENT_SECRET>"

$TeamId       = "<TEAM_ID>"
$ChannelId    = "<CHANNEL_ID>"
$FolderName   = "Dossier-Projet"
$TargetUser   = "prenom.nom@votre-domaine.fr"   # UPN/email Entra ID

# ============================
# 2) Fonctions utilitaires
# ============================
```

```
function Get-OAuthToken {
    param(
        [Parameter(Mandatory=$true)][string]$TenantId,
        [Parameter(Mandatory=$true)][string]$ClientId,
        [Parameter(Mandatory=$true)][string]$ClientSecret,
        [Parameter(Mandatory=$true)][string]$Scope      # ex: "https://graph.microsoft.com/.default"
    )

    $body = @{
        client_id     = $ClientId
        client_secret = $ClientSecret
        scope         = $Scope
        grant_type    = "client_credentials"
    }
    $uri = "https://login.microsoftonline.com/$TenantId/oauth2/v2.0/token"
    $resp = Invoke-RestMethod -Method POST -Uri $uri -Body $body -ContentType "application/x-www-form-
urlencoded"
    return $resp.access_token
}

function Invoke-Graph {
    param(
        [Parameter(Mandatory=$true)][string]$Method,
        [Parameter(Mandatory=$true)][string]$Uri,     # relatif à graph.microsoft.com/v1.0
        [Parameter()][hashtable]$Headers,
        [Parameter()][object]$Body
    )
    if (-not $script:GraphToken) {
        throw "Token Graph absent."
    }
    $h = @{ "Authorization" = "Bearer $($script:GraphToken)" }
    if ($Headers) { $Headers.GetEnumerator() | ForEach-Object { $h[$_.Key] = $_.Value } }

    $full = "https://graph.microsoft.com/v1.0$Uri"
    if ($Body) {
        return Invoke-RestMethod -Method $Method -Uri $full -Headers $h -Body ($Body | ConvertTo-Json -
Depth 10) -ContentType "application/json"
    } else {
        return Invoke-RestMethod -Method $Method -Uri $full -Headers $h
    }
}

function Invoke-SPRest {
    param(
        [Parameter(Mandatory=$true)][string]$Method,
        [Parameter(Mandatory=$true)][string]$Url,     # URL complète SharePoint (_api/...)
        [Parameter()][object]$Body,
        [Parameter()][hashtable]$Headers
    )
    if (-not $script:SPToken) {
        throw "Token SharePoint absent."
    }
    $h = @{
        "Authorization" = "Bearer $($script:SPToken)"
        "Accept"        = "application/json;odata=verbose"
    }
    if ($Headers) { $Headers.GetEnumerator() | ForEach-Object { $h[$_.Key] = $_.Value } }

    if ($Body) {
        $json = $Body | ConvertTo-Json -Depth 12
        return Invoke-RestMethod -Method $Method -Uri $Url -Headers $h -ContentType
"application/json;odata=verbose" -Body $json
    } else {
        return Invoke-RestMethod -Method $Method -Uri $Url -Headers $h
    }
}

# ============================
# 3) Authentification : Graph + SharePoint
# ============================
```

```
$script:GraphToken = Get-OAuthToken -TenantId $TenantId -ClientId $ClientId -ClientSecret $ClientSecret
-Scope "https://graph.microsoft.com/.default"

# On obtiendra l'URL du site SharePoint à partir du DriveItem, puis on demandera un jeton SharePoint
pour ce host.
# Pour l'instant on ne connaît pas encore le host SP (tenant). On génèrera le token SP après avoir
récupéré webUrl/siteId.

# ============================
# 4) Localiser le dossier "Fichiers" du canal, et créer le sous-dossier
# ============================
# 4.1 - Obtenir le filesFolder du canal (driveItem contenant parentReference.driveId + id)
$filesFolder = Invoke-Graph -Method GET -Uri "/teams/$TeamId/channels/$ChannelId/filesFolder"

$parentDriveId = $filesFolder.parentReference.driveId
$parentItemId  = $filesFolder.id

if (-not $parentDriveId -or -not $parentItemId) {
    throw "Impossible de récupérer filesFolder (driveId / itemId non trouvés)."
}

# 4.2 - Créer le nouveau sous-dossier
$createBody = @{
    name   = $FolderName
    folder = @{}
}
$newFolder = Invoke-Graph -Method POST -Uri "/drives/$parentDriveId/items/$parentItemId/children" -Body
$createBody

$folderItemId = $newFolder.id
if (-not $folderItemId) { throw "Échec de la création du dossier." }

# 4.3 - Récupérer les identifiants SharePoint (listId, listItemId, siteId) + webUrl du dossier créé
$spIds = Invoke-Graph -Method GET -Uri
"/drives/$parentDriveId/items/$folderItemId?`$select=sharepointIds,webUrl"
$sharepointIds = $spIds.sharepointIds
$folderWebUrl  = $spIds.webUrl

if (-not $sharepointIds.listId -or -not $sharepointIds.listItemId -or -not $sharepointIds.siteId -or -
not $folderWebUrl) {
    throw "sharepointIds/webUrl non trouvés pour le dossier créé."
}

# 4.4 - Obtenir l'URL racine du site SharePoint (depuis siteId Graph)
$siteInfo = Invoke-Graph -Method GET -Uri "/sites/$($sharepointIds.siteId)?`$select=webUrl"
$siteUrl  = $siteInfo.webUrl
if (-not $siteUrl) { throw "Impossible de déterminer l'URL du site SharePoint." }

# Maintenant que l'on connaît le host SharePoint, on récupère un jeton pour SharePoint Online.
# Le scope .default côté SharePoint consommera les permissions d'application que vous avez consenties
(Sites.FullControl.All).
# Remarque : utilisez bien le host du tenant (ex: https://contoso.sharepoint.com)
$spHost = ([Uri]$siteUrl).Scheme + "://" + ([Uri]$siteUrl).Host
$script:SPToken = Get-OAuthToken -TenantId $TenantId -ClientId $ClientId -ClientSecret $ClientSecret -
Scope "$spHost/.default"

# ============================
# 5) Rompre l'héritage & verrouiller aux Propriétaires
# ============================
# On cible l'élément de liste correspondant au dossier via listId/listItemId
# 5.1 - Break role inheritance (copyRoleAssignments=true pour dupliquer les rôles actuels :
Owners/Members/Visitors)
$breakUri =
"$siteUrl/_api/web/lists(guid'$($sharepointIds.listId)')/items($($sharepointIds.listItemId))/breakrolein
heritance(copyRoleAssignments=true, clearSubscopes=true)"
Invoke-SPRest -Method POST -Url $breakUri | Out-Null

# 5.2 - Récupérer les groupes associés (Members/Visitors) pour les supprimer de ce dossier
$assocMember = Invoke-SPRest -Method GET -Url
"$siteUrl/_api/web/associatedmembergroup?`$select=Id,Title"
```

```
$assocVisitor = $null
try { $assocVisitor = Invoke-SPRest -Method GET -Url
"$siteUrl/_api/web/associatedvisitorgroup?`$select=Id,Title" } catch {}

# 5.3 - Supprimer les attributions de rôles des groupes "Membres" (et "Visiteurs" s'il existe) au niveau
DU DOSSIER
function Remove-RoleAssignmentByPrincipalId {
    param([int]$PrincipalId)
    if ($PrincipalId) {
        $delUri =
"$siteUrl/_api/web/lists(guid'$($sharepointIds.listId)')/items($($sharepointIds.listItemId))/roleassignm
ents/getbyprincipalid(principalid=$PrincipalId)/deleteobject"
        Invoke-SPRest -Method POST -Url $delUri | Out-Null
    }
}

if ($assocMember.d.Id)  { Remove-RoleAssignmentByPrincipalId -PrincipalId $assocMember.d.Id }
if ($assocVisitor -and $assocVisitor.d.Id) { Remove-RoleAssignmentByPrincipalId -PrincipalId
$assocVisitor.d.Id }

# À ce stade : seuls les Propriétaires (groupe associé) conservent le contrôle total au niveau du
dossier.

# ============================
# 6) Accorder "Modif" au compte Entra ID ciblé
# ============================
# 6.1 - S'assurer que l'utilisateur existe dans le site (ensureUser) et récupérer son principalId (Id
côté site)
$ensureBody = @{ logonName = $TargetUser }   # e-mail/UPN accepté
$ensureResp = Invoke-SPRest -Method POST -Url "$siteUrl/_api/web/ensureuser" -Body $ensureBody
$principalId = $ensureResp.d.Id
if (-not $principalId) { throw "Impossible de résoudre l'utilisateur '$TargetUser' dans le site
SharePoint." }

# 6.2 - Ajouter une affectation de rôle "Edit" (RoleDefId = 1073741830) pour ce principal au niveau du
dossier
# Réf. principaux RoleDefId : Full Control=1073741829, Edit=1073741830, Contribute=1073741827,
Read=1073741826
$addRoleUri =
"$siteUrl/_api/web/lists(guid'$($sharepointIds.listId)')/items($($sharepointIds.listItemId))/roleassignm
ents/addroleassignment(principalid=$principalId, roledefid=1073741830)"
Invoke-SPRest -Method POST -Url $addRoleUri | Out-Null

Write-Host "OK - Dossier '$FolderName' créé dans le canal, héritage rompu."
Write-Host "    - Seuls les Propriétaires du site ont un contrôle total."
Write-Host "    - L'utilisateur '$TargetUser' dispose d'un accès 'Modification'."
Write-Host "URL du dossier : $folderWebUrl"
``
```