Upload files & folders via Graph API

> Requirements: You need to work through the "Preparations" part of this manual: Download & read files ... | LNC DOCS (lucanoahcaprez.ch)

With this tutorial, files can be written to SharePoint Online repositories automatically and without user interaction. This can be especially valuable when you need an affordable place to automatically save files. It is also useful if you want to build an automation in to an existing file structure.

Preparations

Work according to this manual and complete the steps up to and including "Preparations". Afterwards, the app registration can write to the appropriately authorized SharePoint Online site without user interaction.

Getting the Graph API Authentication

With these preparations you will then be able to upload data to SharePoint Online using the Graph API. It is important that all variables are filled in appropriately and correctly.

```
$TenantId = "<yourtenantid>"
$ClientId = "<yourappregistrationid>"
$ClientSecret = "<yourclientsecret>"

$SiteID = "<yoursharepointsiteid>"

$LibraryName = "Documents"
$InPathRoot = "<yourinputfolderforuploadingdata>"
$SharePointPath = "<yourdestinationfolderonthissharepointsite>"

# Get Bearer Token for authentication against Graph API
```

```
$Body = @{
    "tenant"        = $TenantId
    "client_id"     = $ClientId
    "scope"         = "https://graph.microsoft.com/.default"
    "client_secret" = $ClientSecret
    "grant_type"    = "client_credentials"
}

$Params = @{
    "Uri"         = "https://login.microsoftonline.com/$TenantId/oauth2/v2.0/token"
    "Method"      = "Post"
    "Body"        = $Body
    "ContentType" = "application/x-www-form-urlencoded"
}
$AuthResponse = Invoke-RestMethod @Params
$Header = @{
    "Authorization" = "Bearer $($AuthResponse.access_token)"
}

# Get Drive ID of the Site
$Params = @{
    "Method"      = "Get"
    "Uri"         = "https://graph.microsoft.com/v1.0/sites/$SiteID/drives"
    "Headers"     = $Header
    "ContentType" = "application/json"
}
$DriveID = ($(Invoke-RestMethod @Params).value | where { $_.name -eq $LibraryName }).id
```

Upload folder structure and data

This is the final code snippet which allows you to upload the files in the specified folder.

```
# Upload Folderstructure and Files from Local FileSystem
$files = Get-ChildItem $InPathRoot -File -Recurse
foreach($file in $files){
    $filepath = ((Split-Path $file.FullName -Parent)+"\").Replace("$InPathRoot", "")
    $HTTPMethod = "PUT"
    if($filepath){
        $UploadUrl =
```

```
"https://graph.microsoft.com/v1.0/drives/$DriveID/items/root:/$($SharePointPath+"\"+$filepath+$file.Name):/content"
    }else{
        $UploadUrl = "https://graph.microsoft.com/v1.0/drives/$DriveID/items/root:/$($SharePointPath+$file.Name):/content"
    }
    $Params = @{
        "Method"      = "$HTTPMethod"
        "Uri"         = "$UploadUrl"
        "Headers"     = $Header
        "ContentType" = "multipart/form-data"
        "InFile"      = $file.FullName
    }
    if((((Get-Item $file.FullName).length / 1MB) -gt 59)){
        $HTTPMethod = "POST"
        $UploadUrl = "https://graph.microsoft.com/v1.0/drives/$DriveID/items/root:/$($SharePointPath+"\"+$file.Name):/createUploadSession"
        $Params = @{
            "Method"      = "$HTTPMethod"
            "Uri"         = "$UploadUrl"
            "Headers"     = $Header
            "ContentType" = "application/json"
        }
    }
    if(((Get-Item $file.FullName).length / 1MB) -gt 59 -and $filepath){
        $Params["UploadUrl"] = "https://graph.microsoft.com/v1.0/drives/$DriveID/items/root:/$($SharePointPath+"\"+$filepath+$file.Name):/createUploadSession"
    }
    $uploadSession = Invoke-RestMethod @Params
    if($uploadSession.uploadUrl){
        $fileInBytes = [System.IO.File]::ReadAllBytes($file.fullname)
        $fileLength = $file.Length

        # Split the file up
        $PartSizeBytes = 50000 * 1024
        $index = 0
        $start = 0
```

```
        $end = 0

        # Upload each junck of chunk of the file
        while ($fileLength -gt ($end + 1)) {
            $start = $index * $PartSizeBytes
            if (($start + $PartSizeBytes - 1) -lt $fileLength) {
                $end = ($start + $PartSizeBytes - 1)
            }
            else {
                $end = ($start + ($fileLength - ($index * $PartSizeBytes)) - 1)
            }

            [byte[]]$body = $fileInBytes[$start..$end]
            $headers = @{
                'Content-Length' = $body.Length.ToString()
                'Content-Range'  = "bytes $start-$end/$fileLength"
            }
            write-Host "bytes $start-$end/$fileLength | Index: $index and ChunkSize:
$PartSizeBytes"
            $response = Invoke-WebRequest -Method Put -Uri $uploadSession.uploadUrl -Body
$body -Headers $headers

            $index++
        }
    }
}
```

Revision #13
Created 2022-12-22 09:13:26 UTC
Updated 2026-08-22 22:52:54 UTC by Luca Noah Caprez