Microsoft SharePoint

- Sync multiple sites to explorer via PowerShell
- Read SharePoint file information via Graph API
- Download & read files via Graph API
- Upload files & folders via Graph API

Sync multiple sites to explorer via PowerShell

Prepare CSV

Create CSV with these columns. This information is needed to start the Sync Process (HTTP-Call on odopen://, see in PowerShell script) along with the UsersID and UsersUPN.

Column (Property)	Example	How to get value
url	https://lucanoahcaprez.sharepoint.com/sites/teamsite1234	Export from SharePoint Admincenter
siteid	39r18c7a-11a1-5acf-813e-784339a741de	Export from SharePoint Admincenter
webid	61b9a58e-30e1-4069-9417-843a0938a6a1	Always the same (static)
listid	0b947fb2-6cde-4be2-a801-61a7642a5186	Always the same (static)
sitename	Team Site 1234	Export from SharePoint Admincenter **Attention:** This has to be UTF-7 encoded (no ä, ö, ü, é, è, etc. included).

Copy template here.

```
url;siteid;webid;listid;sitename
https://lucanoahcaprez.sharepoint.com/sites/teamsite1234;39r18c7a-11a1-5acf-813e-784339a741de;61b9a58e-30e1-4069-9417-843a0938a6a1;0b947fb2-6cde-4be2-a801-61a7642a5186;Team Site 1234
```

Run script on client

This script can be run on the client to sync the provided sites in the csv.

```
$SiteList = Import-Csv -Delimiter ";" -Path "C:\LocalData\MultipleSiteSync.csv"

$userid = "<userid>"
$userEmail = "<userupn>"

Foreach($Site in $SiteList){
    $SiteId = $Site.siteid
    $WebId = $Site.webid
    $Listid = $Site.listid
    $WebTitle = $Site.sitename
    $WebUrl = $Site.url

    Start-Process
"odopen://sync?userId=$userid&userEmail=$UserEmail&isSiteAdmin=0&siteId=$SiteId&webId=$WebId&w
ebTitle=$WebTitle&webTemplate=64&webUrl=$WebUrl&onPrem=0&libraryType=3&listId=$ListId&listTitl
e=Dokumente&scope=OPENLIST"
    Start-Sleep -seconds 10
}
```

Read SharePoint file information via Graph API

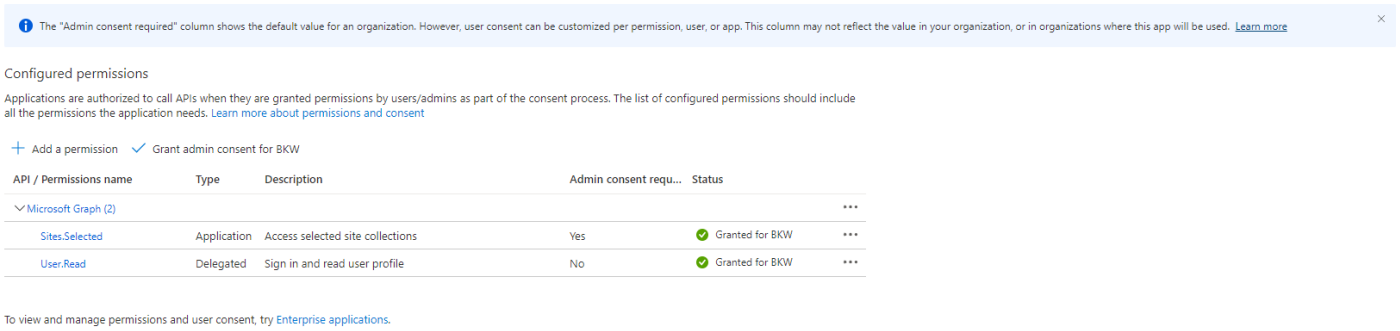
> Requirements: Graph Explorer knowledge needed.

Create App Registration

An App Registration will be used to authenticate against Microsoft Graph API. For creating an App Registration there is an other detailed guide.

Permissions

With API Permission "Sites.Selected" you can specify on which SharePoint-Sites the App Registration will have permissions to read/write files and properties.



Get Site ID

To get the site id of your sharepoint sites to to the following link in a Webbrowser:

GET https://<tenantname>.sharepoint.com/sites/<sitename>/_api/site/id

Example: https://lucanoahcaprez.sharepoint.com/sites/SPS-LNC-WebFiles-PROD/_api/site/id

Example XML Response in Browser:



Get Folder ID

Folder IDs can be found using the graph explorer and the Site ID:

GET https://graph.microsoft.com/v1.0/sites/<SiteID>/drives
Example: https://graph.microsoft.com/v1.0/sites/22c1c748-f7e0-4f10-97f4-64b51694a0ca/drives

Example JSON Response from Graph API


```
{
   "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#drives",
   "value": [
      {
         "createdDateTime": "2021-04-03T23:11:15Z",
         "description": "",
         "id": "b!SMfBIuD3EE-X9GS1FpSgypE4hAdzBPVAhzjowjy6CTAFMPrY5yGSRYYXW7Jhy97f",
         "lastModifiedDateTime": "2022-05-06T07:25:15Z",
         "name": "Dokumente",
         "webUrl":
"https://<tenantname>.sharepoint.com/sites/<sitename>/Freigegebene%20Dokumente",
         "driveType": "documentLibrary",
         "createdBy": {
            "user": {
               "displayName": "Systemkonto"
            }
         },
         "lastModifiedBy": {
            "user": {
               "displayName": "Systemkonto"
            }
         },
         "owner": {
            "group": {
               "email": "<sitename>@<tenantname>.onmicrosoft.com",
               "id": "bab76dba-25aa-4804-8cad-4fde62da958b",
               "displayName": "Besitzer von <sitename>"
            }
         },
         "quota": {
            "deleted": 0,
            "remaining": 27487376431088,
            "state": "normal",
            "total": 27487790694400,
            "used": 414263312
```

```
      }
    },
    {
      "createdDateTime": "2021-04-06T13:42:16Z",
      "description": "",
      "id": "b!SMfBIuD3EE-X9GS1FpSgypE4hAdzBPVAhzjowjy6CTA3sb9Pi_NxRJ3lfYXRyKTz",
      "lastModifiedDateTime": "2021-10-02T21:16:08Z",
      "name": "Vertragsablage",
      "webUrl": "https://<tenantname>.sharepoint.com/sites/<sitename>/<foldername>",
      "driveType": "documentLibrary",
      "createdBy": {
        "user": {
          "displayName": "Systemkonto"
        }
      },
      "lastModifiedBy": {
        "user": {
          "displayName": "Systemkonto"
        }
      },
      "owner": {
        "group": {
          "email": "<sitename>@<tenantname>.onmicrosoft.com",
          "id": "bab76dba-25aa-4804-8cad-4fde62da958b",
          "displayName": "Besitzer von <sitename>"
        }
      },
      "quota": {
        "deleted": 0,
        "remaining": 27487376431088,
        "state": "normal",
        "total": 27487790694400,
        "used": 414263312
      }
    },
    {
      "createdDateTime": "2022-09-15T09:41:10Z",
      "description": "sharepoint list",
      "id": "b!SMfBIuD3EE-
X9GS1FpSgypE4hAdzBPVAhzjowjy6CTC6kAXUUK7qSLG7oSTLYNsQ",
      "lastModifiedDateTime": "2022-09-15T09:41:10Z",
      "name": "Teams Wiki Data",
      "webUrl":
"https://<tenantname>.sharepoint.com/sites/<sitename>/Teams%20Wiki%20Data",
      "driveType": "documentLibrary",
```

```
        "createdBy": {
          "user": {
            "email": "<creatorupn>",
            "id": "9e5da9b4-7023-4d36-86f5-33768907270f",
            "displayName": "<creatordisplayname>"
          }
        },
        "lastModifiedBy": {
          "user": {
            "email": "<creatorupn>",
            "id": "9e5da9b4-7023-4d36-86f5-33768907270f",
            "displayName": "<creatordisplayname>"
          }
        },
        "owner": {
          "group": {
            "email": "<sitename>@<tenantname>.onmicrosoft.com",
            "id": "bab76dba-25aa-4804-8cad-4fde62da958b",
            "displayName": "Besitzer von <sitename>"
          }
        },
        "quota": {
          "deleted": 0,
          "remaining": 27487376431088,
          "state": "normal",
          "total": 27487790694400,
          "used": 414263312
        }
      }
    ]
}
```


Get Folder Documents

Get folder documents and content with graph api:

GET https://graph.microsoft.com/v1.0/sites/<SiteID>/drives/<FolderID>

Example: https://graph.microsoft.com/v1.0/sites/22c1c748-f7e0-4f10-97f4-64b51694a0ca/drives/b!SMfBIuD3EE-X9GS1FpSgypE4hAdzBPVAhzjowjy6CTAFMPrY5yGSRYYXW7Jhy97f

Example JSON Response in Graph Explorer:

Example JSON Response from Graph API


```
{
   "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#drives/$entity",
   "createdDateTime": "2021-04-03T23:11:15Z",
   "description": "",
   "id": "b!SMfBIuD3EE-X9GS1FpSgypE4hAdzBPVAhzjowjy6CTAFMPrY5yGSRYYXW7Jhy97f",
   "lastModifiedDateTime": "2022-05-06T07:25:15Z",
   "name": "Dokumente",
   "webUrl":
"https://<tenantname>.sharepoint.com/sites/<sitename>/Freigegebene%20Dokumente",
   "driveType": "documentLibrary",
   "createdBy": {
      "user": {
         "displayName": "Systemkonto"
      }
   },
   "lastModifiedBy": {
      "user": {
         "displayName": "Systemkonto"
      }
   },
   "owner": {
      "group": {
         "email": "<sitename>@<tenantname>.onmicrosoft.com",
         "id": "bab76dba-25aa-4804-8cad-4fde62da958b",
         "displayName": "Besitzer von <sitename>"
      }
   },
   "quota": {
      "deleted": 0,
      "remaining": 27487376431088,
      "state": "normal",
      "total": 27487790694400,
      "used": 414263312
   }
}
```

Download & read files via Graph API

> Requirements: App Registration with appropriate permissions, as described here: Read SharePoint files ... | LNC DOCS (lucanoahcaprez.ch)

With this tutorial, files can be read from SharePoint Online repositories automatically and without user interaction. This can be especially valuable when large accesses to files need to be made, but the budget for Azure Blob Storage, for example, is not available. It also allows very easy customization of the corresponding files in SharePoint Online, which can then be used directly in the automation.

Preparations

First of all, the app registration must be created with the appropriate permissions. This is described here: Read SharePoint files ... | LNC DOCS (lucanoahcaprez.ch) Then the app must be assigned the appropriate permissions on SharePoint Online once.

This can be achieved with the following PowerShell code. The two upper variables must be filled in accordingly with the SharePoint Site URL and the App Registration ID.

```
$siteUrl = "https://<yourtenantname>.sharepoint.com/sites/<yoursitename>"
$ClientId = "<yourappid>"

# Connect to SPO Site (PNPOnline Modul required)
Connect-PnPOnline -Url $siteUrl -Interactive #

# Grant write permission to the App Registration to the corresponding SPO site & files
Grant-PnPAzureADAppSitePermission -Permissions "Write" -Site $siteUrl -AppId $ClientId -
DisplayName "$ClientId - Write Access"
```

Getting the files via Graph API

The following code can be used in an automation for example. Here all files are written into an array without the execution needing any interaction.
It is also important here that the variables are correctly filled in at the beginning.

```
$SiteID = "<yoursharepointsiteid>"
$TenantId = "<yourtenantid>"
$ClientId = "<yourappregistrationid>"
$ClientSecret = "<yourclientsecret>"
$LibraryName = "Documents"
$OutPathRoot = "<youroutputfolderforfilecontent>"

if (-not $(Test-Path -LiteralPath $OutPathRoot)) {
    New-Item $OutPathRoot -Force -ItemType Directory | Out-Null
}

# 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/drive"
    "Headers"     = $Header
```

```
    "ContentType" = "application/json"
}
$DriveID = $(Invoke-RestMethod @Params | where { $_.name -eq $LibraryName }).id

# Get all files from the SharePoint folder
$Params = @{
    "Method"      = "Get"
    "Uri"         =
"https://graph.microsoft.com/v1.0/sites/$SiteID/drives/$DriveID/root/children"
    "Headers"     = $Header
    "ContentType" = "application/json"
}
$SPFiles = $(Invoke-RestMethod @Params).Value
```

Select the option how you want to work with the file(s)

Afterwards, the files can be either all downloaded, downloaded individually or only the content can be viewed. Accordingly, the appropriate code piece must be selected.

Download all files

```
# Download all the files
foreach ($File in $SPFiles) {
    if ($File.'@microsoft.graph.downloadUrl') { # Schliesst Ordner aus
        $Params = @{
            "Method"      = "Get"
            "Uri"         = $File.'@microsoft.graph.downloadUrl'
            "ContentType" = "application/json"
            "OutFile"     = "$OutPathRoot\$($File.name)"
        }
        Invoke-RestMethod @Params
    }
}
```

Download one file

```
# Download only one file
$File = $SPFiles[1]
$Params = @{
    "Method"      = "Get"
    "Uri"         = $File.'@microsoft.graph.downloadUrl'
    "ContentType" = "application/json"
    "OutFile"     = "$OutPathRoot\$($File.name)"
}
Invoke-RestMethod @Params
```

Get the content of one file

```
# Get the content of a file
$File = $SPFiles[1]
$Params = @{
    "Method"      = "Get"
    "Uri"         = $File.'@microsoft.graph.downloadUrl'
    "ContentType" = "application/json"
}
$Content = Invoke-RestMethod @Params
$Content
```

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++
        }
    }
}
```