Migrate Markdown files to OneNote Notebook

> **Requirements:** You need an Authorization header and Notes.ReadWrite.All permissions in the Bearer token. Eighter Delegated or Application permissions needed.

The Microsoft Graph API enables the creation of content, structure, and notes in Microsoft OneNote. In the backend it uses plain html as the basic content structure. Which means we can apply the widespread syntax for images, lists and more.

Explanation of Microsoft OneNote elements

- **Notebooks**: Top-level containers that store all content in OneNote. They organize related notes into a cohesive structure, similar to physical notebooks.
- **Section Groups**: Hierarchical containers within notebooks that organize multiple sections together, enabling complex categorization. Section Groups can be part of other Section Groups.
- **Sections**: Dividers within a notebook that group related notes. Each section can contain multiple pages. Sections can be part of Section Groups but can not be part of Sections.
- **Pages**: The core content units in OneNote where notes, images, and other data are created and stored. Pages reside within sections.

Microsoft Graph URLs for OneNote

Replace <userupn> with user principle name that the access token is issued to. Alternatively you can use the /me endpoint (https://graph.microsoft.com/v1.0/me/onenote/*) to get the data of the user from the access token dynamically.

The Microsoft Graph API for OneNote supports a hierarchical URL structure, allowing you to drill down through related elements in a single query. You can filter sequentially from notebooks to section groups, then sections, pages, and finally retrieve the content of a specific page. This structure enables precise access to data without making multiple API calls. Example query:

```
https://graph.microsoft.com/v1.0/users/<userid>/onenote/notebooks/<notebookid>/sectionGroups/<sectiongroupid>/sections/<sectionid>/pages/<pageid>/content
```

Recommendation: Filter the results on the server side using Graph data preparation in URL parameters. More on this topic here: Serverside data preper... | LNC DOCS All of these endpoints support common query parameters to filter, sort, or paginate the results, making it a versatile tool for managing and interacting with OneNote content programmatically.

Get OneNote notebooks

The Microsoft Graph API endpoint for OneNote notebooks retrieves a list of OneNote notebooks associated with a specific user.

```
GET https://graph.microsoft.com/v1.0/users/<userupn>/onenote/notebooks
```

Get OneNote section groups

This endpoint retrieves all section groups in the users OneNote account, spanning across all notebooks. Section groups are hierarchical containers for organizing sections within OneNote.

```
GET https://graph.microsoft.com/v1.0/users/<userupn>/onenote/sectionGroups
```

This second endpoint retrieves all section groups within a specific OneNote notebook identified by <notebookid>. It allows targeted access to the section groups of a particular notebook

```
GET https://graph.microsoft.com/v1.0/users/<userupn>/onenote/notebooks/<notebookid>/sectionGroups
```

Get OneNote sections

This endpoint retrieves all sections in the users OneNote account, spanning across all available notebooks. It provides a comprehensive view of the users sections

```
GET https://graph.microsoft.com/v1.0/users/<userupn>/onenote/sections
```

The second endpoint retrieves all sections within a specific OneNote notebook identified by <notebookid>. It enables precise access to the sections of a targeted notebook.

```
GET https://graph.microsoft.com/v1.0/users/<userupn>/onenote/notebooks/<notebookid>/sections
```

This third endpoint retrieves all sections within a specific OneNote section group, identified by its unique section group ID. It allows focused access to sections organized within a particular section group.

```
GET
https://graph.microsoft.com/v1.0/users/<userupn>/onenote/sectionGroups/<sectiongroupid>/sectio
ns
```

Get OneNote pages

This endpoint provides a complete list of all pages within the users OneNote account, regardless of their notebook or section. It’s a useful way to access all notes in one request. **Attention:** Since this list can grow with time, it may be required to use pagination (make multiple queries because large data needs to be splittet into manageable chunks).

```
GET https://graph.microsoft.com/v1.0/users/<userupn>/onenote/pages
```

This endpoint retrieves all pages contained within a specific OneNote section, identified by <sectionid>. It enables targeted access to notes under a particular organizational context.

```
GET https://graph.microsoft.com/v1.0/users/<userupn>/onenote/sections/<sectionid>/pages
```

Get OneNote page content

This endpoint retrieves the full content of a specific OneNote page, identified by <pageid>. The content is returned in **HTML format**, which includes all text, images, and other elements structured in a web-friendly format. This allows for easy rendering or further processing in web-based or other HTML-compatible applications.

```
GET https://graph.microsoft.com/v1.0/users/<userupn>/onenote/pages/<pageid>/content
```

This endpoint is ideal for applications that need to display, edit, or analyze OneNote page data in its full context. Appropriate permissions, such as *Notes.Read* is required for access.

Create OneNote section group

This Graph endpoint creates a new section group within the specified notebook identified by <notebookid>. Provide a JSON payload with the "displayname" field to define the group’s name.

```
POST
https://graph.microsoft.com/v1.0/users/<userupn>/onenote/notebooks/<notebookid>/sectionGroup
{
  "displayname":"<yoursectiongroupname>"
}
```

Create OneNote sections

Adds a new section to a specific notebook identified by <notebookid>. Supply a JSON payload containing the "displayname" field for the section name.

```
POST https://graph.microsoft.com/v1.0/users/<userupn>/onenote/notebooks/<notebookid>/sections
{
  "displayname":"<yoursectiongroupname>"
}
```

The followin endpoint creates a new section within an existing section group identified by <sectiongroupid>. Include a "displayname" in the JSON payload to specify the section name.

```
POST
https://graph.microsoft.com/v1.0/users/<userupn>/onenote/sectionGroups/<sectiongroupid>/sectio
ns
{
  "displayname":"<yoursectiongroupname>"
}
```

Upload OneNote page content

The POST request to the following endpoint creates a new OneNote page in a specific section, identified by <sectionid>. The page content must be provided in HTML format within the request body.

Key Steps for Page Creation:

1. **Prepare the Request:**
   - Set the target section ID in the URL.
   - Include an HTML body for the page content, specifying the title and desired structure.
2. **Set Metadata:**
   - Use meta tags (e.g., "created" date) to provide additional details for the page.
3. **Send the Request:**
   - Use an HTTP client to send a POST request with a valid access token for authorization.
   - Ensure the token has Notes.ReadWrite permission.
4. **Handle the Response:**
   - A successful response includes the newly created pages details, including its ID and URL for further operations.

This method efficiently uploads custom page content to a specified section, making it ideal for programmatically managing OneNote.

```
POST https://graph.microsoft.com/v1.0/users/<userupn>/onenote/sections/<sectionid>/pages
```

This is an example usage of a PowerShell request to create a new OneNote page.

```
$Global:MicrosoftEntraIDAccessToken = "<yourbeareraccesstoken>"

$YourSiteTitle = "<yoursitetitle"
$ParentSectionId = "<yourparentsectionid"
$UserUPN = "<youruserupn>"
$NotesContent = "<yourhtmlsitecontent>"
$CreationDate = Get-Date

$PageCreationUrl =
"https://graph.microsoft.com/v1.0/users/$UserUPN/onenote/sections/$($ParentSectionId)/pages"
$PageCreationBody = @"
<!DOCTYPE html>
<html>
<head>
<title>$YourSiteTitle</title>
<meta name="created" content="$($YourCreationDate.ToString("yyyy-MM-ddTHH:mm:sszzz"))" />
</head>
<body>
$($NotesContent)
</body>
</html>
"@
$PageCreationResponse = Invoke-RestMethod -Uri $PageCreationUrl -Header @{Authorization =
"Bearer $Global:MicrosoftEntraIDAccessToken" } -Method POST -Body $PageCreationBody -
ContentType "application/xhtml+xml"
```

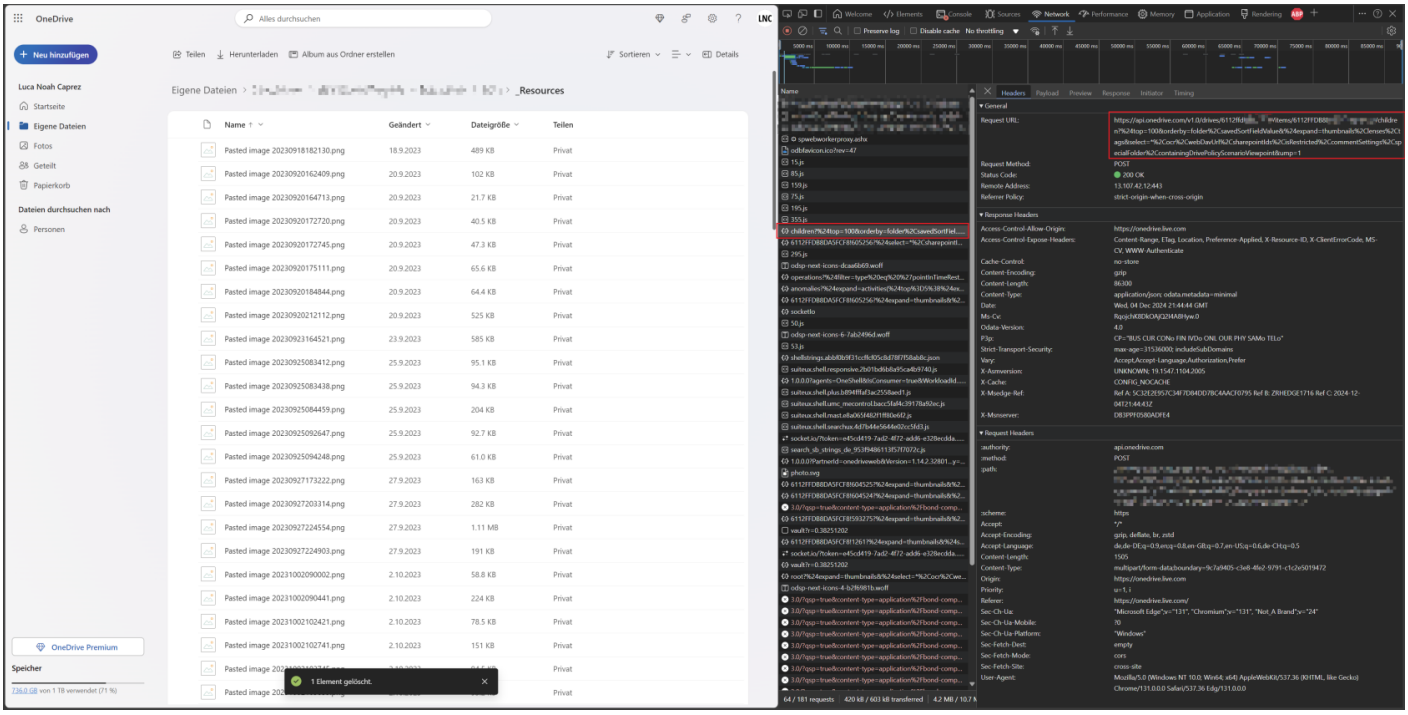
Migration script for Obsidian Markdown

This chapter provides a detailed guide on how to use this PowerShell script to migrate an entire folder structure of Markdown files from Obsidian into Microsoft OneNote while preserving the original hierarchy. The script processes folders as section groups, subfolders as sections or section groups, and Markdown files as pages. Using this migration script you can seamlessly transferring content into OneNote.

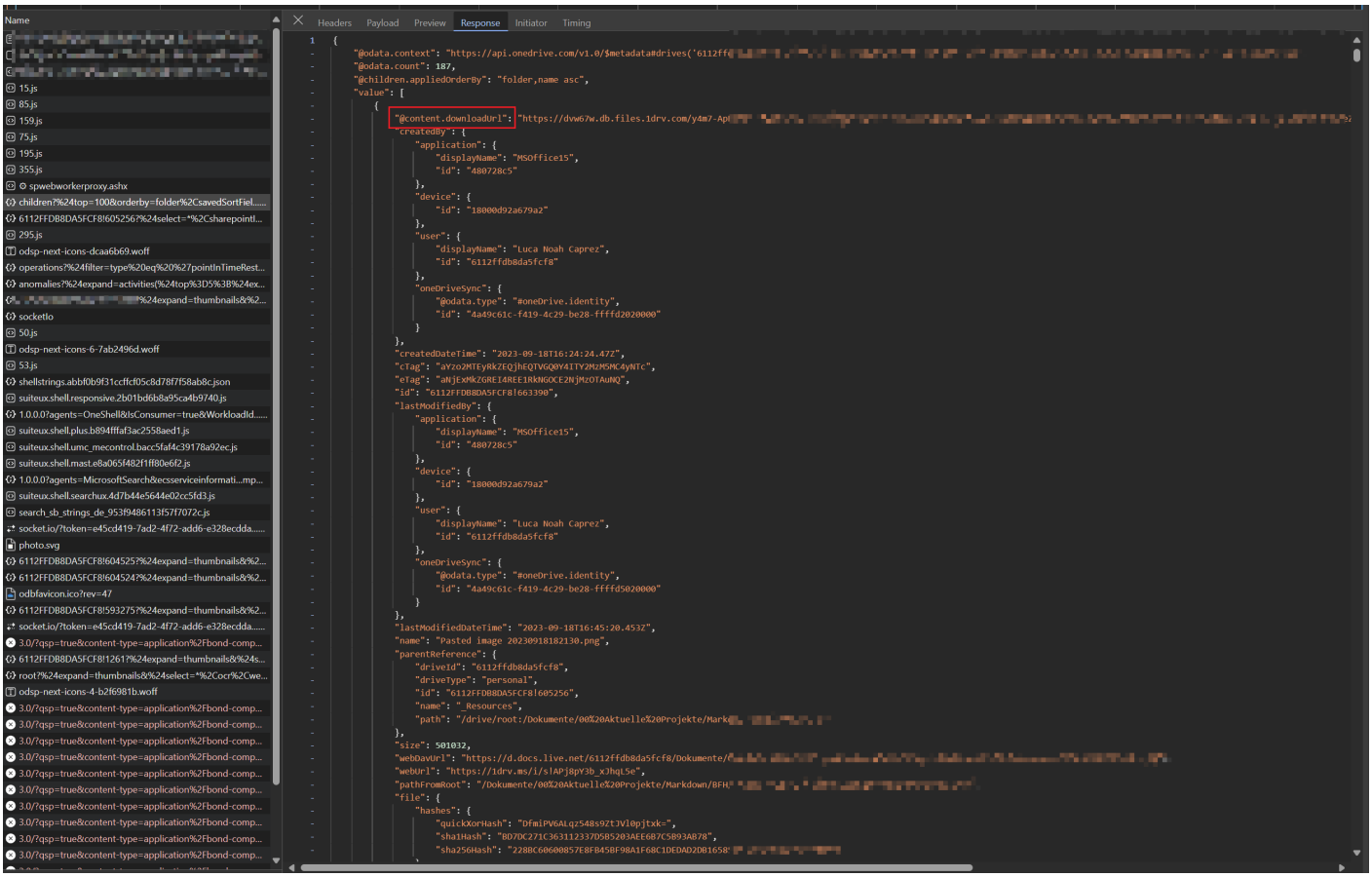
Import images from OneDrive (optional)

To integrate pictures into the OneNote page, we need a static link to the binary source of the image. If you are storing your Markdown structure including the images in OneDrive or SharePoint you can retrieve the direct download URL from their storage.

Open the desired folder in the browser and open the DevTools (F12) of your browser. Go to the corresponding resource folder and reload the page. Search for the query to https://api.onedrive.com and review the link on the right side.



On this entry switch to "Response" and copy the whole JSON content into a file on your local computer. This file will then be imported in the next script. We want the direct URL of each image (@content.downloadUrl) as this is static per file and will not change. It also returns the binary element of the image which we want on the OneNote side.



To comply with the script below, name the file "AllMarkdownPictures.json" and put it into the same folder as the script. Now you are ready to start the migration of your Markdown files to OneNote.

PowerShell script

1. **Configure Parameters**:
   Update variables such as the access token, user principal name (UPN), parent section group ID, and the local path to the Obsidian Markdown folder.Specify whether you want to import the images directly from OneDrive (previous chapter).
2. **Run the Script**:
   Execute the script in PowerShell. The script uses Microsoft Graph API endpoints to create section groups, sections, and pages dynamically.
3. **Output**:
   The script outputs the progress of folder, section, and page creation, and highlights any issues encountered during the migration. Make sure to check the structure in OneNote for any errors or wrong content!

```
$Global:MicrosoftEntraIDAccessToken = "<yourentraidaccesstoken>"
$UserUPN = "<youruserupn>"
$ParentSectionGroup = "<parentsectiongroupid>" # Find via Microsoft Graph Explorer
$ParentFolder = "<yourlocalpathforobsidian structure>"
$ImportImages = $true # Set to false if you don't want to import images -> Importing images is advanced and requires more indepth knowledge of OneDrive API
$AllNotebooksFolder = Get-ChildItem -Path $ParentFolder -Directory
```

```
# Got Image JSON via Browser DevTools
(https://api.onedrive.com/v1.0/drives/6112ffdb8da5fcf8/items/6112ffdb8da5fcf8!605256/children?
%24top=1000)
$AllFolders = Get-Content ".\AllMarkdownPictures.json"
$AllPictures = $AllFolders | ConvertFrom-Json -Depth 10

foreach ($Folder in $AllNotebooksFolder) {
  $FolderName = $Folder.Name
  if ($FolderName.Length -gt 50) {
    $FolderName = $FolderName.Substring(0, 50)
  }
  if ($FolderName -like ".*" -or $FolderName -like "_*" ) {
    Write-Output "Skipping $Folder"
    continue
  }

  $SectionUrl =
"https://graph.microsoft.com/v1.0/users/$UserUPN/onenote/sections?`$filter=displayName eq
'$($FolderName)'"
  $SectionResponse = (Invoke-RestMethod -Uri $SectionUrl -Header @{Authorization = "Bearer
$Global:MicrosoftEntraIDAccessToken" } -Method GET).value
  if ($SectionResponse.Count -eq 0) {
    $SectionCreationUrl =
"https://graph.microsoft.com/v1.0/users/$UserUPN/onenote/sectionGroups/$ParentSectionGroup/sec
tions"
    $SectionCreationBody = @{
      displayName = "$($FolderName)"
    } | ConvertTo-Json
    $SectionResponse = Invoke-RestMethod -Uri $SectionCreationUrl -Body $SectionCreationBody -
Header @{Authorization = "Bearer $Global:MicrosoftEntraIDAccessToken" } -Method POST -
ContentType "application/json"
    Write-Output "Created section $($FolderName)"
  }

  $MarkdownNotes = Get-ChildItem -Path $Folder.FullName
  foreach ($FileName in $MarkdownNotes) {
    Write-Output "Processing $($FileName.BaseName)"
    $PageUrl = "https://graph.microsoft.com/v1.0/users/$UserUPN/onenote/pages?`$filter=title
eq '$($FileName.BaseName)'"
    $PageResponse = (Invoke-RestMethod -Uri $PageUrl -Header @{Authorization = "Bearer
```

```
$Global:MicrosoftEntraIDAccessToken" } -Method GET).value

    $NotesContent = (Get-Content $FileName.FullName -Raw | ConvertFrom-Markdown).html
    if ($ImportImages) {
      # Add Pictures from OneDrive Source
      $PictureMatches = ($NotesContent | Select-String -Pattern '!\[\[(.*?)\]\]' -
AllMatches).Matches
      if ($PictureMatches.Count -gt 0) {
        $Pictures = $PictureMatches | ForEach-Object { $_.Groups[1].Value }
        foreach ($Picture in $Pictures) {
          $PicturePath = ($AllPictures.value | Where-Object { $_.name -eq $Picture
})."@content.downloadUrl"
          if ($PicturePath) {
            $NotesContent = $NotesContent -replace "!\[\[$Picture\]\]", "<img
src='$($PicturePath)' alt='$Picture' />"
          }
        }
      }
    }

    # Add New Line Characters
    $Lines = $NotesContent -split "`n"
    $ModifiedLines = $Lines | ForEach-Object {
      if (-not $_.EndsWith(">")) {
        $_ + "<br>"
      }
      else {
        $_
      }
    }
    $NotesContent = $ModifiedLines -join "`n"

    # if (!$PageResponse) {
    try {

      $PageCreationUrl =
"https://graph.microsoft.com/v1.0/users/$UserUPN/onenote/sections/$($SectionResponse.id)/pages
"
      $PageCreationBody = @"
<!DOCTYPE html>
<html>
```

```
<head>
<title>$($FileName.BaseName)</title>
<meta name="created" content="$($FileName.CreationTime.ToString("yyyy-MM-ddTHH:mm:sszzz"))" />
</head>
<body>
$($NotesContent)
</body>
</html>
"@
      $PageCreationResponse = Invoke-RestMethod -Uri $PageCreationUrl -Header @{Authorization
= "Bearer $Global:MicrosoftEntraIDAccessToken" } -Method POST -Body $PageCreationBody -
ContentType "application/xhtml+xml"
      Write-Output "Created $($FileName.BaseName)"
    }
    catch {
      Write-Output "Error creating page $($FileName.BaseName)"
      Write-Output $_
    }
    # }
    # else {
    #   Write-Output "Page $($FileName.BaseName) already exists"
    #   # TODO Update the pages content
    # }
  }
}
```

Revision #3
Created 2024-12-04 18:52:19 UTC by Luca Noah Caprez
Updated 2024-12-04 22:01:12 UTC by Luca Noah Caprez