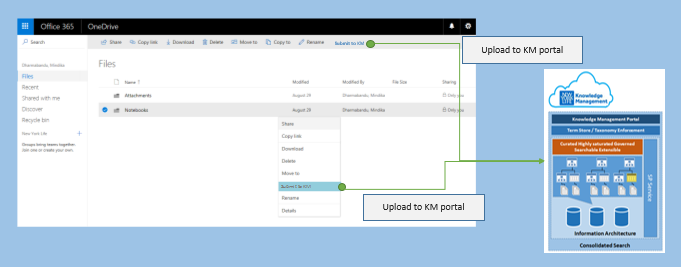
OneDrive file upload to KM Portal

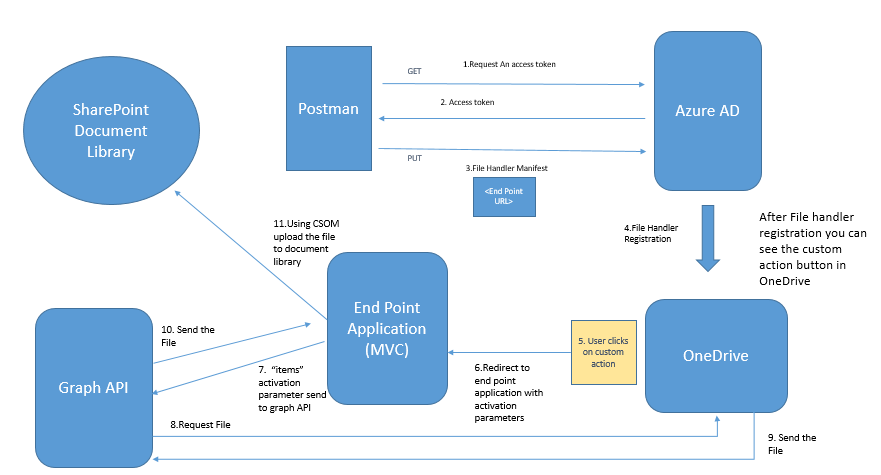
**Introduction**

This documentation describes how to upload OneDrive documents to KM portal. We are developing a custom action for fulfilling this requirement.

POC Task :



Overview Diagram :



**Tools used**

1. Visual studio 2017 – Building Endpoint application
2. Postman (Chrome Extension) - Deployment Purpose

**Templates used**

1. Downloaded the ASP.NET MVC sample

<https://developer.microsoft.com/en-us/graph/quick-start>

**Steps**

1. Register a File Handler in Azure AD
2. Endpoint application & service development

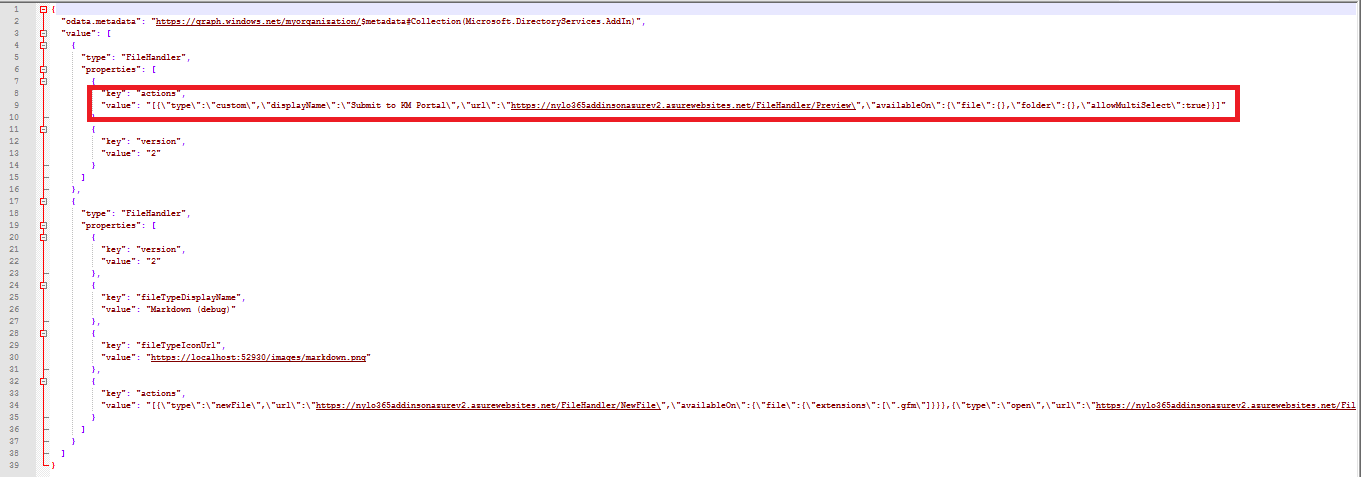
# Register a File Handler in Azure AD

**File Handler includes two parts**

1. File Handler Manifest
2. File Handler Endpoint

**Register File Handler Manifest**

Manifest file defines the interaction between OneDrive and the File handler endpoint. The manifest should be registered with the Azure AD.



In this Solution we are considering “custom” type section only. There you need to change the following parameters according to your requirement.

* type : This should be ‘**custom’**
* displayName : This is the text that will show in the custom action
* url : URL to the end point application
* availableOn : What type of files or folders this custom action should be available. You can provide extensions also.
* allowMultiSelect : Allow on multi select or not

**Required Permissions**

* Tenant Admin user

**Obtain an access token**

To obtain an access token, you can use a tool like Postman.

**Steps to Follow:**

1. Install the Postman chrome extension or desktop application and launch the app.
2. On the Authorization tab, change Type to OAuth 2.0.
3. Click Get New Access Token. A dialog will appear asking for a number of fields to be completed.
4. In your browser, navigate to the Azure Portal and sign-in.
5. Select Azure Active Directory, then click on App registrations.
6. Click Add to create a new application, which will be used to manage file handler manifests
7. Name the application File handler registration tool (FileHandler Display Name) and set the sign-on URL to the value Postman tells you to use: https://www.getpostman.com/oauth2/callback.
8. Scroll to the end of the application list and click on File handler registration tool.
9. Copy the application ID value (guid) and paste it into Postman next to Client ID.
10. Click on Required permissions, then Windows Azure Active Directory.
11. Click on Required permissions, then Windows Azure Active Directory.
12. Under Delegated permissions find and check the following permissions:

* Sign in and read user profile
* Read and write directory data
* Access the directory as the signed in user

1. Click Save.
2. Click on Keys, then create a new key by changing the Expires drop down to Never expires and clicking Save.
3. Copy the value of the new key, and paste it into Postman next to Client Secret.
4. In Postman, complete the access token form, by providing the following values:

* Token name: File handler registration tool
* Auth URL: https://login.microsoftonline.com/common/oauth2/authorize?resource=https://graph.windows.net
* Access Token URL: https://login.microsoftonline.com/common/oauth2/token
* Grant Type: Authorization code

1. Click Request Token and sign in as a **tenant administrator**.
2. After consenting to the permissions, Postman will show you a token for File handler registration tool. Select that entry, and change Add token to **Header** and then click **Use** **Token** to append the new access token.
3. Make a request to find your file handler application manifest, by entering the URL: https://graph.windows.net/myorganization/applications/?api-version=1.6.
4. Change the URL to https://graph.windows.net/myorganization/applications/{applicationObjectId}/addIns?api-version=1.6. Replace {applicationObjectId} (In AAD) with the value obtained previously.
5. Change the request type to PUT.
6. Click on the Body tab, and then select Raw and change the Text drop down to JSON.
7. Copy and paste the file handler manifest into the body window. The body request needs to be wrapped inside a JSON object, with a value array containing the file handler manifest: {"value": [ { /\* manifest here \*/} ] }. Important: this request will overwrite any existing file handlers or addIns registered for this application. If there are other addIns registered, you need to submit all of the addIns in the same request. (Please find the attached Sample Manifest)
8. Click **Send** to update the application.
9. If you see the **response status as 204 No Content** then the update was **successful**.

Now that your File handler manifest has been registered with your application in Azure AD.

**Note: Changes to the add-in manifest may not be immediately applied. The file handlers manifest is cached for performance. Changes to the file handler manifest can take up to 24 hours to reflect.**

**Force the cache to be cleared for Development purposes**:

**https://docs.microsoft.com/en-us/onedrive/developer/file-handlers/reset-cache**

# Endpoint application development

Endpoint application includes the logic of the File handler action. In File handler manifest, we are registering an endpoint Url in Azure AD. This Url should point to this application. When the user clicks on OneDrive Custom action, it sends the **POST** Request to End point Application. This POST Request includes,

* cultureName : The locale identifier for the user's current display language.
* client: The Office 365 application from which the file handler was invoked; for example "SharePoint" or "OneDrive".
* userId: The UPN/login email for the user who invoked the file handler.
* domainHint: A domain hint string that indicates either organizations or consumers.
* items: A collection of Microsoft Graph URLs to the selected item(s).

For this application, we are using “items” Activation parameter. We can retrieve them in C# using:

var itemsJson = Request.Form["items"];

var itemUrls = JsonConvert.DeserializeObject<string[]>(itemsJson);

itemUrls are Graph API urls. We need to develop graph authenticated application.

To do that we can make use of “**Microsoft Application Registration portal**” (<https://developer.microsoft.com/en-us/graph/quick-start>). After filling required fields, you can download a graph authenticated sample application.

This is a basically C# MVC application. You can pass the **itemurl** to graph api and retrieve the selected file as a memory stream using following code.

GraphServiceClient graphClient = SDKHelper.GetAuthenticatedClient();

var request = graphClient.Shares[itemUrls];

var foundFile = await request.Request().GetAsync();

MemoryStream stream = (MemoryStream)await graphClient.Me.Drive.Items[foundFile.Id].Content.Request().GetAsync();

After retrieving the memory stream, we can upload the document to SharePoint Document library using CSOM code.

# Issues

1. In current solution, activation parameters getting lost in first attempt.

Reason to this is the application redirects to O365 sing-in URL. Within this request, parameters are got lost. Second time onwards it’s working correctly.

1. Custom action displayed only for Filehandler registered users.
2. Filehandler Manifest registration takes 24 hours in Azure AD, but o365 gives by-force alternative for development :

See Refresh file handler cache:

<https://docs.microsoft.com/en-us/onedrive/developer/file-handlers/reset-cache>

# References

**File handler tutorial**

<https://docs.microsoft.com/en-us/onedrive/developer/file-handlers/>

**Microsoft graph explorer**

By using this you can test “items” sent from OneDrive with activation parameter

<https://developer.microsoft.com/en-us/graph/graph-explorer>

**Microsoft Application Registration portal**

You can manage the registered application

<https://apps.dev.microsoft.com/>

# Appendix

Sample Manifest File

