Integrating MS Flow and SPFx Extensions.

By John Benson

**About Me:**

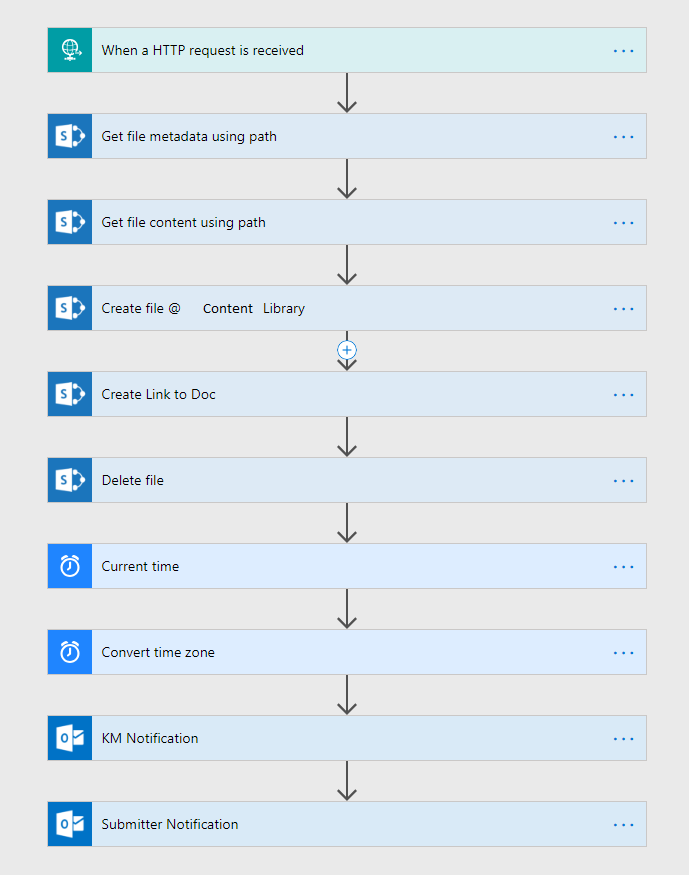
I am developer at Withum Digital. After a number years supporting legacy container freight systems I recently had the opportunity to move into the Practice’s main line of business. With it came learning new technology platforms such as PNP Powershell and SharePoint Framework. Below outlines some work I have done for a recent project that I thought maybe of interest.

**Case:**

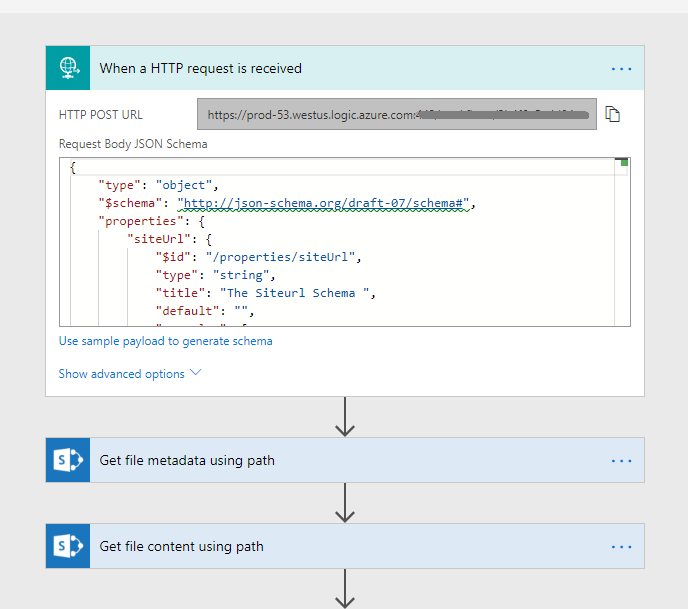
A client required a method for submitting documents from SharePoint sites to a central Curation Document Library. The solution we implemented involved utilizing a SPFx Extension to add an trigger in SharePoint that kicked off an MS Flow which in turn handled the document submission process. I will breakdown how this task was accomplished discussing first the MS Flow followed by the SPFx Extension.

**MS Flow:**

MS Flow was leveraged to create a process that moved documents to the central Document Library, created a URL link to the new document on the source library, then sent email notifications to the submitting user and library curation team. Below is the complete MS Flow Process:

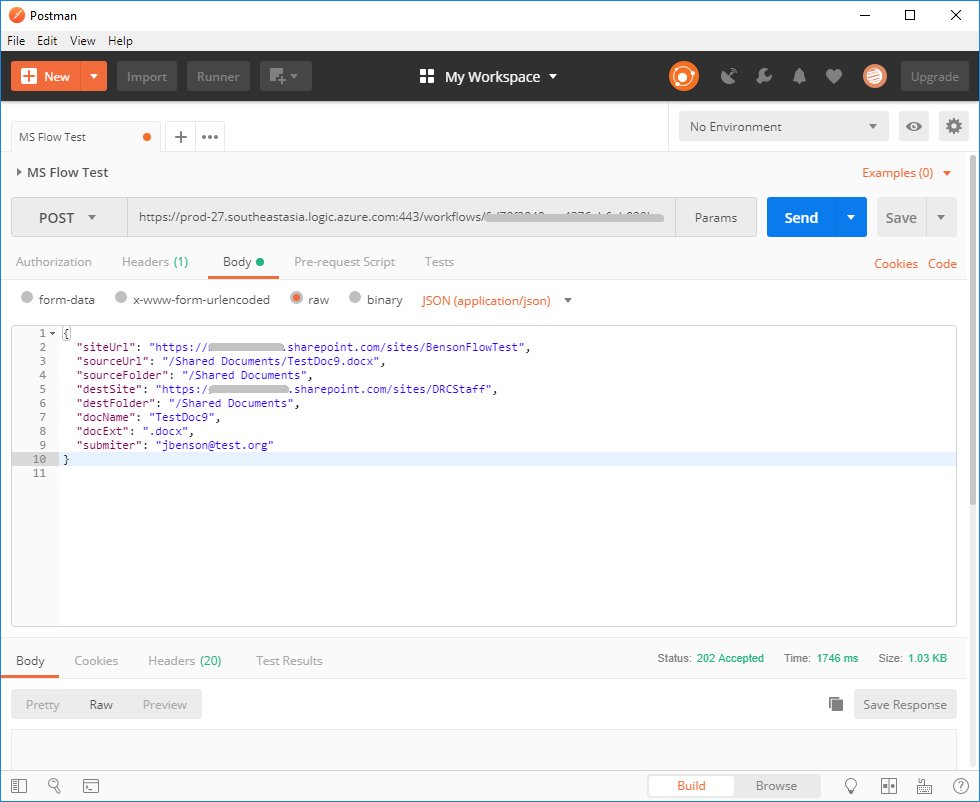


* The first step was creating the entry point for the Flow, in this case we utilized the Flow HTTP Request action as the trigger. I used <https://jsonschema.net/> to create a JSON schema to Post the document metadata we needed to pass into the Flow.

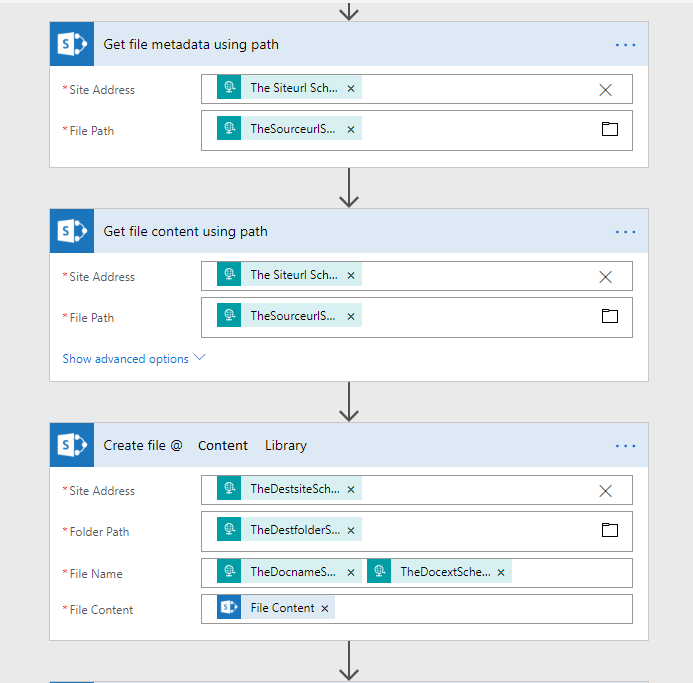


For this process I ended up passing the following variables via JSON:

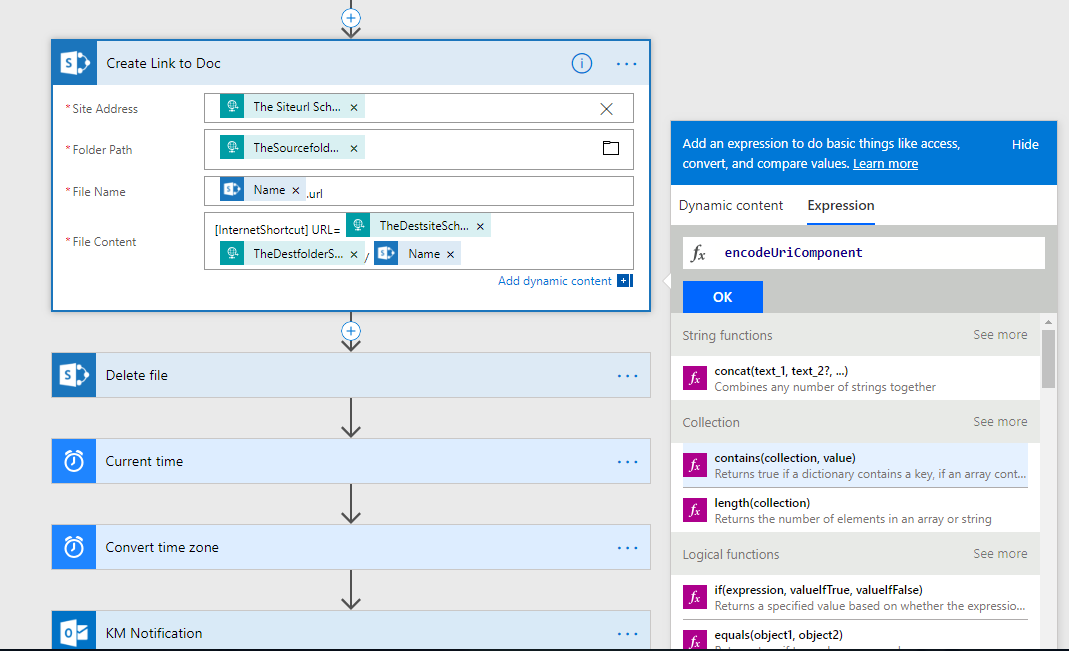
* + siteUrl: The absolute url of the source SharePoint site  
    *https://mySite.sharepoint.com/sites/SP-Source*
  + sourceUrl: The relative url of the document  
     */Shared Documents/Curation Test Document.docx*
  + sourceFolder: the document Source folder  
    */Shared Documents*
  + destSite: The url of the SharePoint Site where the Curation Document Library resides  
    *https://mySite.sharepoint.com/sites/curation-library*
  + destFolder: The library folder   
    */Shared Documents*
  + docName: The submitted document filename.  
    *Curation Test Document*
  + docExt: The file extension for the document.  
    *.docx*
  + submitter: The email address of the user submitting the document  
    *joeUser@mySite.com*
* Once the HTTP Request trigger was initialized in Flow it creates a HTTP POST URL for the user to target. I used a tool know as [Postman](https://www.getpostman.com/) in order to simulate the JSON Post from my extension into the MS Flow. This is a pretty nifty tool that allowed me first validate my JSON Schema and then exercise the Flow as I built out the process.



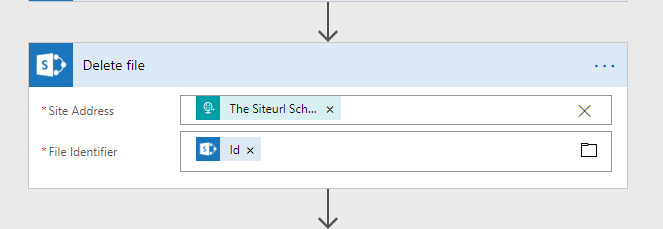
* At the time of writing there is no Cut/Move Document action available in Flow. Internet guidance recommended the Copy File step followed by a Delete step from the Flow SharePoint actions. Unfortunately testing revealed that the Copy action only worked intra-site, while the process needed to move documents site to site. Further research revealed the way forward involved the File Create action. First I added some Get Metadata and Get File Content actions to the Flow, the former was for the Delete step while the later was needed for the File Create. When creating the Create File Step the key part was the File Content parameter which allowed us to create a new document with the content of the source document:



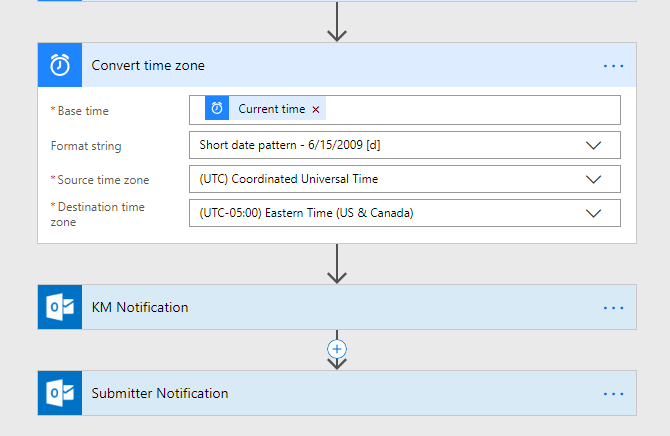
* The next step was to create a link to the new document in the source library. The keys to this are specifying the .url extension in the File Name and the formatting of the File Content. Specifying the *“[InternetShortcut] URL =”* allows you to build the link content. I would also recommend using the Expression section to encode the Name variable.



* With the new document created on the Curation site and a URL link added to the source library it was time to delete the original copy. Using the SharePoint Delete File Action we use the Id variable created from the Get Meta Data Step:



* The Final steps involve sending email notifications to both the curation team and the submitting user. I used some Date actions to get and format the timestamp for the emails. For a more complex system you could integrate some of the Approval Process actions offered by Flow but in this caste the client opted to forgo those options.



With the notification steps complete the MS Flow process for moving Documents has finished. Using Postman you can fully exercise the Flow and confirm that it is ready to go. Now let’s look at a way to trigger it with an SPFx Extension.

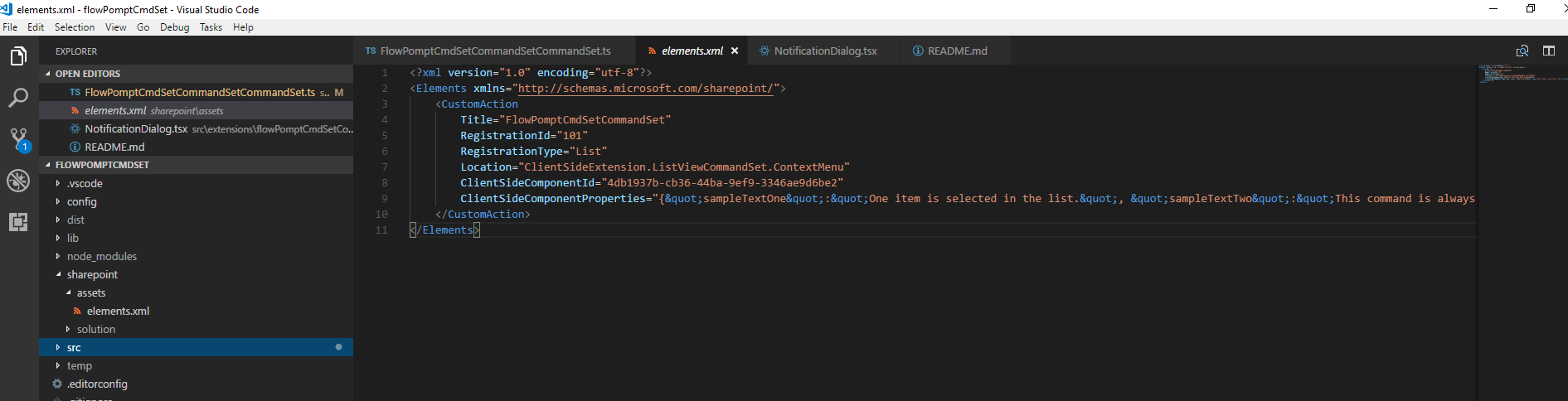
**SPFx Extension:**

As part of the extensions offered in Yeoman’s @microsoft/sharepoint package is the ListView Command Set. This base extension already allows us to inject new commands into the menus for Lists and Libraries, [check](https://docs.microsoft.com/en-us/sharepoint/dev/spfx/extensions/get-started/building-simple-cmdset-with-dialog-api) out MSFT’s Demo for setting one up. Below I will cover the adjustments I made specifically for the this Curation project:

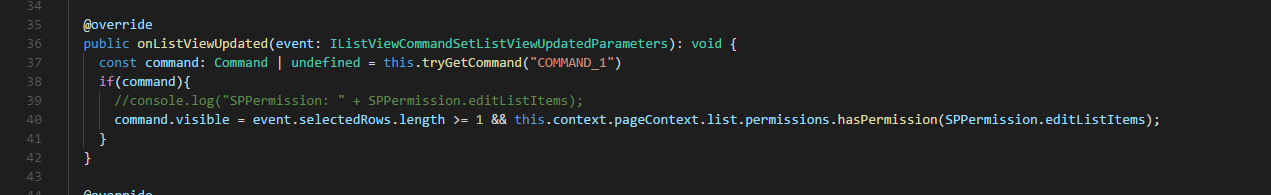
* The OOB Extension includes pretty much all you need to get going, the only modules I added where for creating a simple Dialog box and more importantly the MSFT SP-Http module for making the JSON Post. Here is a look at my package.json file for reference:



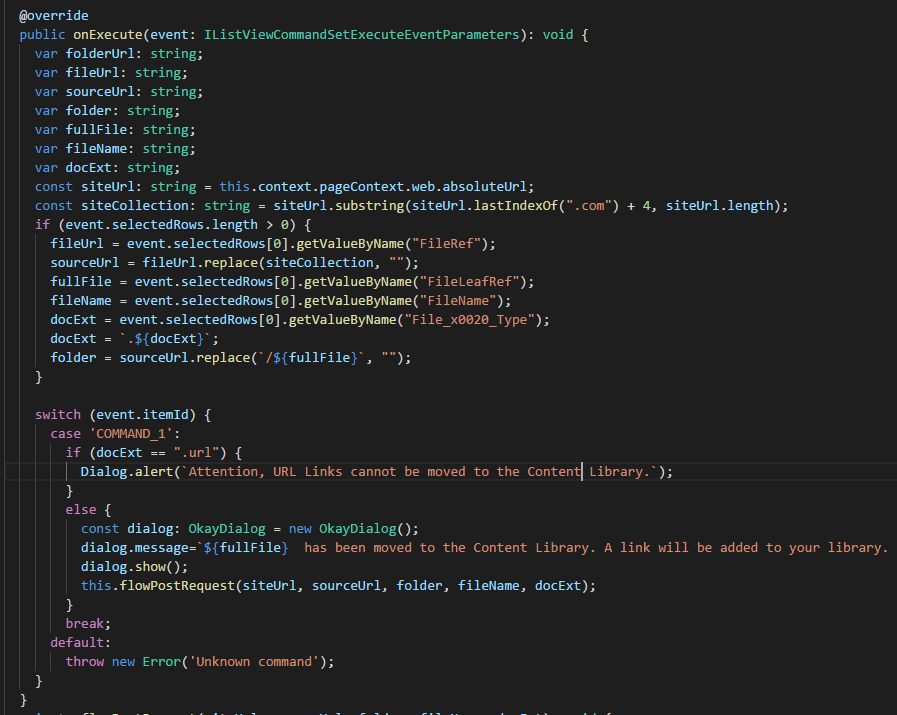
* By default the List View Command Set targets Lists and inserts Commands into the CommandBar, in this case, however, we wanted the extension to appear in Document Libraries and trigger from the Document Context Menu. Note that while testing the extension will appear on both Lists and Doc Libraries but if you deploy the Extension without making these adjustments you may end up wondering where your Commands went. To accomplish this we need to update the sharepoint/assets/elements.xml file:
  + RegistrationId needs to be set to “101” to display the new commands in Document Libraries.
  + Location needs to specify "ClientSideExtension.ListViewCommandSet.ContextMenu"



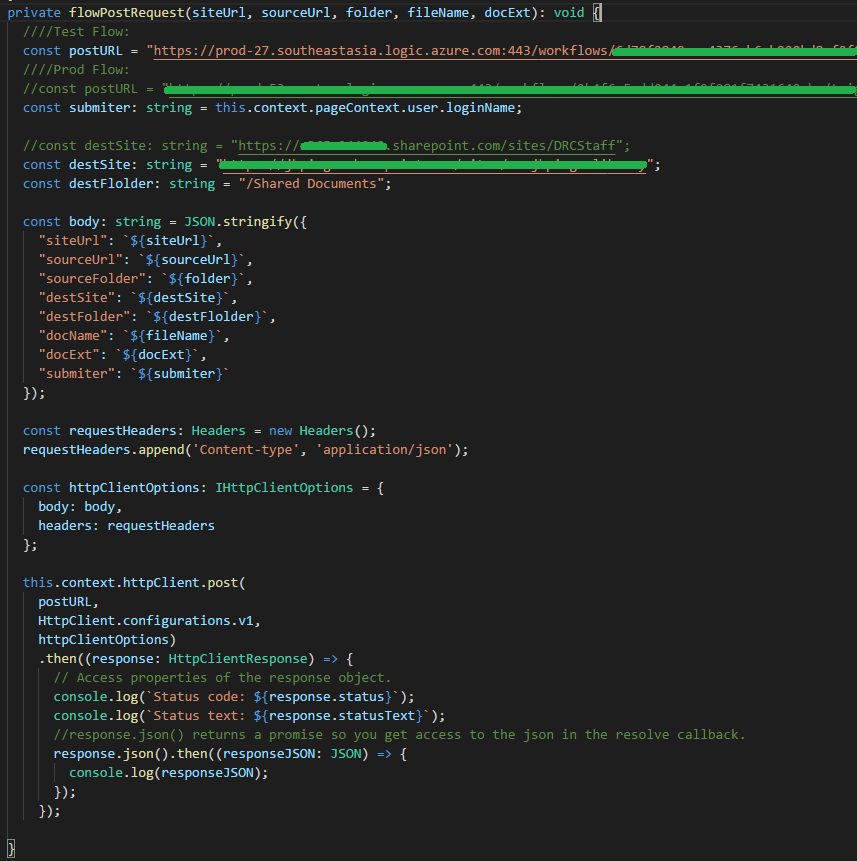
* With the configuration out of the way let’s move on to the main body, first we want to make sure our extension only appears for users with high enough permissions. You will want to *import { SPPermission } from "@microsoft/sp-page-context";* in order to interrogate the user’s permission level. We then modify the onListViewUpdated to display the command if the user has permissions to edit the document:



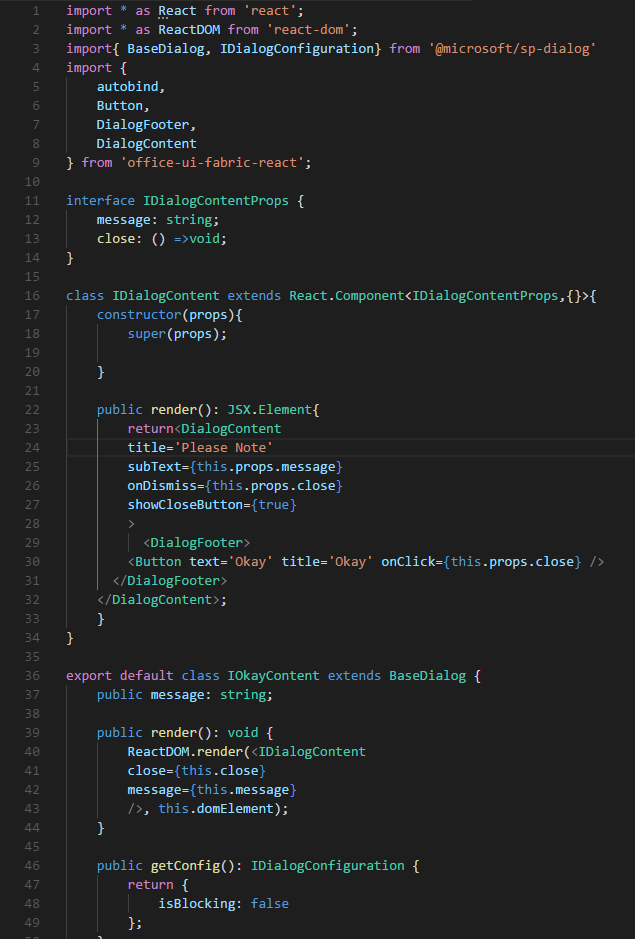
* The next step is the code for when our command is executed. We start by compiling the document metadata for out JSON request, filter out attempts to move url documents, call a basic notification dialog and finally call the function to send the Post request to out Flow.



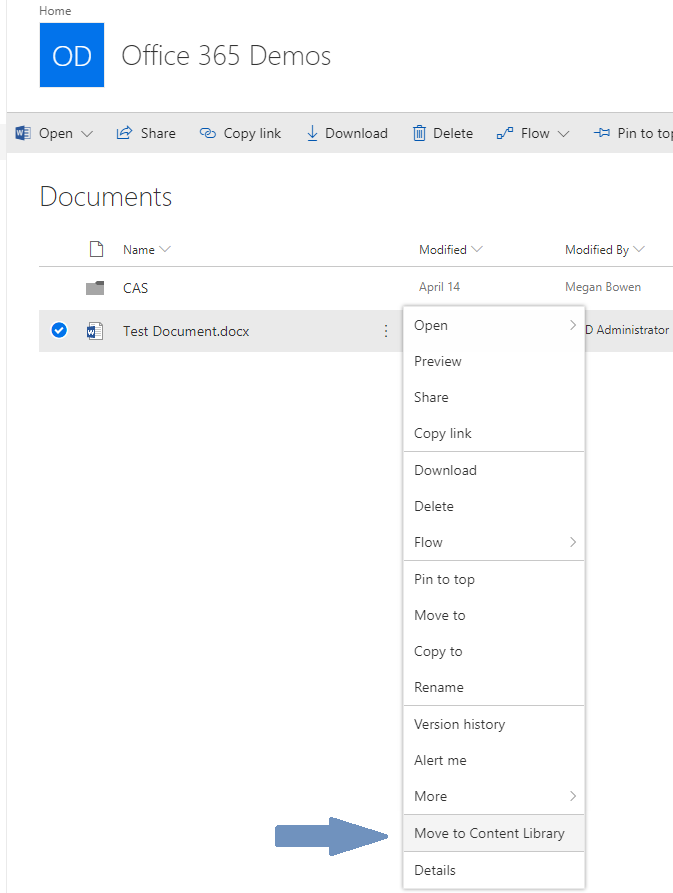
* Let’s look at the JSON Post function.
  + First you will need to import *SPHttpClientResponse, { HttpClient, IHttpClientOptions, HttpClientResponse } from '@microsoft/sp-http';*
  + In you Post function you will want to set the postURL using the HTTP POST URL created in your MS Flow, grab the login Name for the submitting user and set the destination variables for the Curation site:
  + With all the variables set it is time to put together the JSON payload and post to the site



* Since the MS Flow operates separately from the SharePoint site there is not a good way to notify the Document Library Page that the Flow has completed. Therefore I created a simple Dialog Box with instructions to wait for the Notification Email before refreshing to see the changes:



* Note that the Extension be can tested from SPFx refer to the [instructions offered by MSFT](https://docs.microsoft.com/en-us/sharepoint/dev/spfx/extensions/get-started/building-simple-cmdset-with-dialog-api#debug-your-listview-command-set). When you are satisfied that the Extension has passed your testing process you can then package the Web Part and deploy it following [MSFT’s guidance](https://docs.microsoft.com/en-us/sharepoint/dev/spfx/extensions/get-started/building-simple-cmdset-with-dialog-api#deploy-the-extension-to-sharepoint-online-and-host-javascript-from-local-host).
* Once the extension has been built and deployed to a SharePoint site, we can now activate our Flow from a Document Library. By expanding the Document ellipse we can now see our command available.



**Conclusion:**

As has hopefully been illustrated, the basic ListView Command Set package offered by the SharePoint Framework provides a great foundation for integrating custom commands into lists and libraries. Coupled with the HTTP Trigger option in MS Flow we can unlock a considerable amount of flexibility in calling Workflows that can manipulate your SharePoint content.

[**Extension Code Available on GitHub**](https://github.com/john88benson/flowPomptCmdSet)