Connecting SharePoint Framework (SPFx) Projects to Data

Don Kirkham, Microsoft MVP, MCT

@DonKirkham

Microsoft MVP, M365 Development

DMI Enterprise Architect

https://donkirkham.com

@DonKirkham

/in/DonKirkham

DonKirkham

Overview Consume REST APIs in SPFx

- **Common requirement in SPFx project is to display or interact with data external to the web part**
  - Data in SharePoint lists & libraries
  - Data accessible via Microsoft Graph REST API
  - Data accessible in external 3rd Party APIs – anonymous & secured

- **SharePoint Framework provides APIs for all situations when you need to work with data sources external to the web part**
  - `SPHttpClient`: for calling the SharePoint REST APIs
  - `MSGraphClient`: for calling the Microsoft Graph in the same tenant as the SharePoint Online tenant
  - `HttpClient`: for calling 3rd party REST APIs
  - `AADHttpClient`: for 3rd party REST APIs secured with Azure Active Directory

- **Most scenarios require no extra clients / libraries are required**

CRUD with SharePoint Data in SPFx

CRUD Operations with SPFx & SharePoint REST API

Reading items

Creating items

Updating items

Deleting items

Accessing the SharePoint REST API

- **Must include an `Authorization` header containing an OAuth bearer token**

- **Other headers used to control how the REST API is used**
  - OData v3 or v4 (default = v4)
  - Amount & type of metadata returned
  - Type of operation to perform (in the case of updates: merge / update)
  - Match versions

OData Query Operators

```
https://sharepoint/sites/site/_api
/web/lists/getbytitle(‘Countries’)/items?
  $select=Id,Title, Continent
  &$filter= Continent eq ‘North America’
  &$orderby=Title desc
  &$top=10
```

SharePoint Framework & REST API

- **SPFx implements calls to SharePoint REST API via the `SPHttpClient`**

- **Available from the existing context:**
  - `this.context.spHttpClient.get()` & `this.context.spHttpClient.post()`

- **Based on the existing `HttpClient` API**

- **Handles the authentication & default config setting:**
  - Authorization HTTP header
  - OData v4
  - Minimal metadata returned

Reading List Items with the REST API & SPFx

```
private _getListItems(): Promise<ICountryListItem[]> {

  const endpoint: string = this.context.pageContext.web.absoluteUrl
    + `/_api/web/lists/getbytitle('Countries')/items?$select=Id,Title`;

  return this.context.spHttpClient.get(
      endpoint,
      SPHttpClient.configurations.v1
    )
    .then(response => {
      return response.json();
    })
    .then(jsonResponse => {
      return jsonResponse.value;
    }) as Promise<ICountryListItem[]>;
}
```

Write Operations with SPFx & REST API

- **Use the SharePoint Framework `SPHttpClient`'s `post()` method to write to the SharePoint REST API**

- **Some operations require additional HTTP headers:**
  - `X-HTTP-Method`: specify `MERGE` or `DELETE` in update & delete operations
  - `IF-MATCH`: specify version of the item on the server to be updated / deleted

- **Create operation require specific data in the payload body**
  - `@odata.type`: specify the type of data being written to the list when creating

Creating List Items with the REST API

- **Must specify the type of data as the `@odata.type` property in the payload that is being created**
  - Due to lists being able to support multiple content types

- **Pattern: request the type in a pre-request via the list's `ListItemEntityTypeFullName` property**

Get List Entity Type

```
private _getItemEntityType(): Promise<string> {

  const endpoint: string = this.context.pageContext.web.absoluteUrl
+ `/_api/web/lists/getbytitle('Countries')?$select=ListItemEntityTypeFullName`;

  return this.context.spHttpClient.get(
    endpoint,
    SPHttpClient.configurations.v1
  )
  .then(response => {
    return response.json();
  })
  .then(jsonResponse => {
    return jsonResponse.ListItemEntityTypeFullName;
  }) as Promise<string>;
}
```

Creating List Items with the REST API & SPFx

```
private _addListItem(): Promise<SPHttpClientResponse> {

  const endpoint: string = this.context.pageContext.web.absoluteUrl
        + `/_api/web/lists/getbytitle('Countries')/items`;

  return this._getItemEntityType()
    .then(spEntityType => {
      const request: any = {};
      request.body = JSON.stringify({
        '@odata.type': spEntityType,
        Title: new Date().toUTCString(),
      });

      return this.context.spHttpClient.post(
        endpoint, SPHttpClient.configurations.v1, request);
    });
}
```

Updating List Items with the REST API

- **Should specify the type of operation to perform**
  - Default behavior is to set properties to supplied values, BUT omitted properties are set to `null`
  - Override behavior using the `MERGE` method
  - Set using the `X-HTTP-Method` header

- **Specify the version of the item to update**
  - When updating items, can specify “only update the item on the server if it is version X”
  - Ensures you aren’t overwriting someone else’s changes unknowingly
  - Enforced with the `IF-MATCH` header & `etag`’s

Updating List Items with REST API & SPFx

```
private _updateListItem(): Promise<SPHttpClientResponse> {
  // get the first item
  return this.context.spHttpClient.get(
      // .. code to get item from SP REST API
    })
    .then((listItem: ICountryListItem) => {
      // update item
      listItem.Title = 'USA';
      // save it
      const request: any = {};
      request.headers = {
        'X-HTTP-Method': 'MERGE',
        'IF-MATCH': (listItem as any)['@odata.etag']
      };
      request.body = JSON.stringify(listItem);

      const endpoint: string = this.context.pageContext.web.absoluteUrl
                               + `/_api/web/lists/getbytitle('Countries')/items(${listItem.Id})`

      return this.context.spHttpClient.post(endpoint, SPHttpClient.configurations.v1, request);
    });
}
```

Deleting List Items with the REST API

- **Should specify the type of operation to perform**
  - Underlying `fetch` API only contains `post()` method; not `delete()`
  - Override behavior using the `DELETE` method
  - Set using the `X-HTTP-Method` header

- **Specify the version of the item to delete**
  - When updating items, can specify "only update the item on the server if it is version X"
  - Enforced with the `IF-MATCH` header & `etag`'s
  - Decide: does it matter if the version is different?
  - If not, use `IF-MATCH = '*'`

Deleting List Items with REST API & SPFx

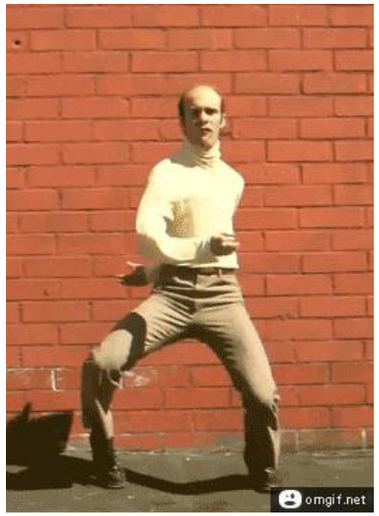
```
private _deleteListItem(): Promise<SPHttpClientResponse> {
  // get the first item
  return this.context.spHttpClient.get(
      // .. code to get item from SP REST API
    })
    .then((listItem: ICountryListItem) => {
      // delete it
      const request: any = {};
      request.headers = {
        'X-HTTP-Method': 'DELETE',
        'IF-MATCH': '*'
      };

      const endpoint: string = this.context.pageContext.web.absoluteUrl
                               + `/_api/web/lists/getbytitle('Countries')/items(${listItem.Id})`

      return this.context.spHttpClient.post(endpoint, SPHttpClient.configurations.v1, request);
    });
}
```

DEMO TIME!

Put on your dancing shoes and let’s have some fun!







Calling Anonymous REST APIs

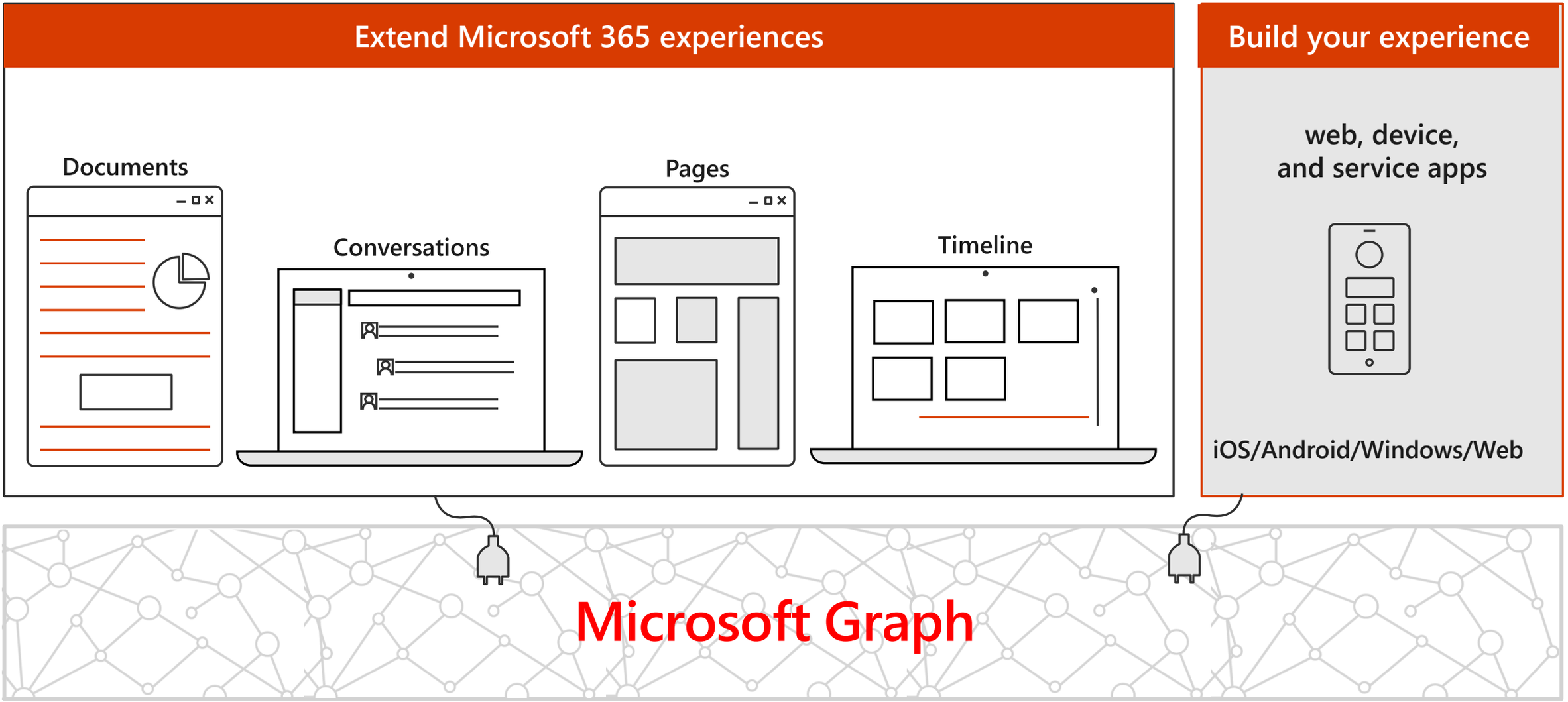
```
private _getSomething: Promise<any> {

  const endpoint: string = `http://[rest-endpoint]`,

  return this.context.httpClient.get(
  endpoint,
  HttpClient.configurations.v1
  )
  .then((response: HttpClientResponse) => {
    return response.json();
  })
  .then(jsonResponse => {
  return jsonResponse;
  }) as Promise<any>;
}
```

Calling the Microsoft Graph

Overview of the Microsoft Graph

SPFx's MSGraphClient

Microsoft 365 Platform



Microsoft Graph
Gateway to your data in the Microsoft cloud

https://graph.microsoft.com

Office 365

Users, Groups, Organizations
Outlook
SharePoint
OneDrive
Teams
Planner
Excel
OneNote

Windows 10

Activities
Device Relay
Commands
Notifications

Enterprise Mobility + Security

Azure AD
Intune
Identity Manager
Advanced Threat Analytics
Advanced Threat Protection

Mail, Calendar,
Contacts and Tasks
Sites and Lists
Drives and Files

Channels, Messages
Tasks and Plans
Spreadsheets
Notes, and more…

Identity Management
Access Control
Synchronization
Domains

Administrative Units
Applications and Devices
Advanced Threat Analytics
Advanced Threat Protection

Alerts
Policies
and more…

SharePoint Framework Includes a Microsoft Graph Client

- **`MSGraphClient`**: SharePoint Framework's Microsoft Graph Client
- Abstracts the token acquisition from the SharePoint Framework's support for Azure AD
- Wraps the Microsoft Graph JavaScript SDK and initializes it with one line that returns a promise

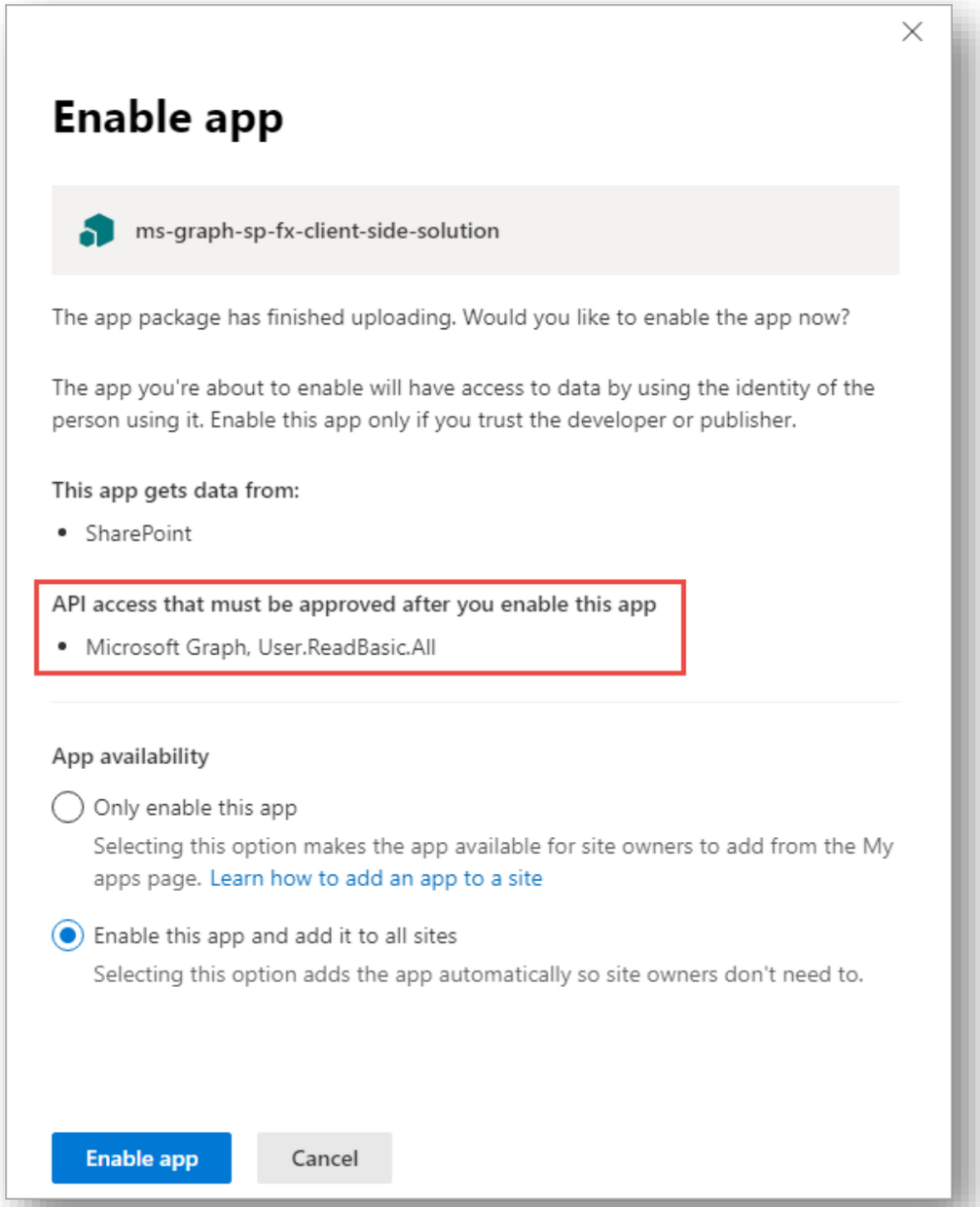
```
import { MSGraphClientV3 } from '@microsoft/sp-http’;
// . . .
const client: MSGraphClientV3 = await this.context.msGraphClientFactory.getClient('3');

// use client here
const myEmail: string = await client.api('/me')
        .get((error: GraphError, user: MicrosoftGraph.User) => {
                return user.mail
        });
```

SPFx Solutions Declare Permission Requests

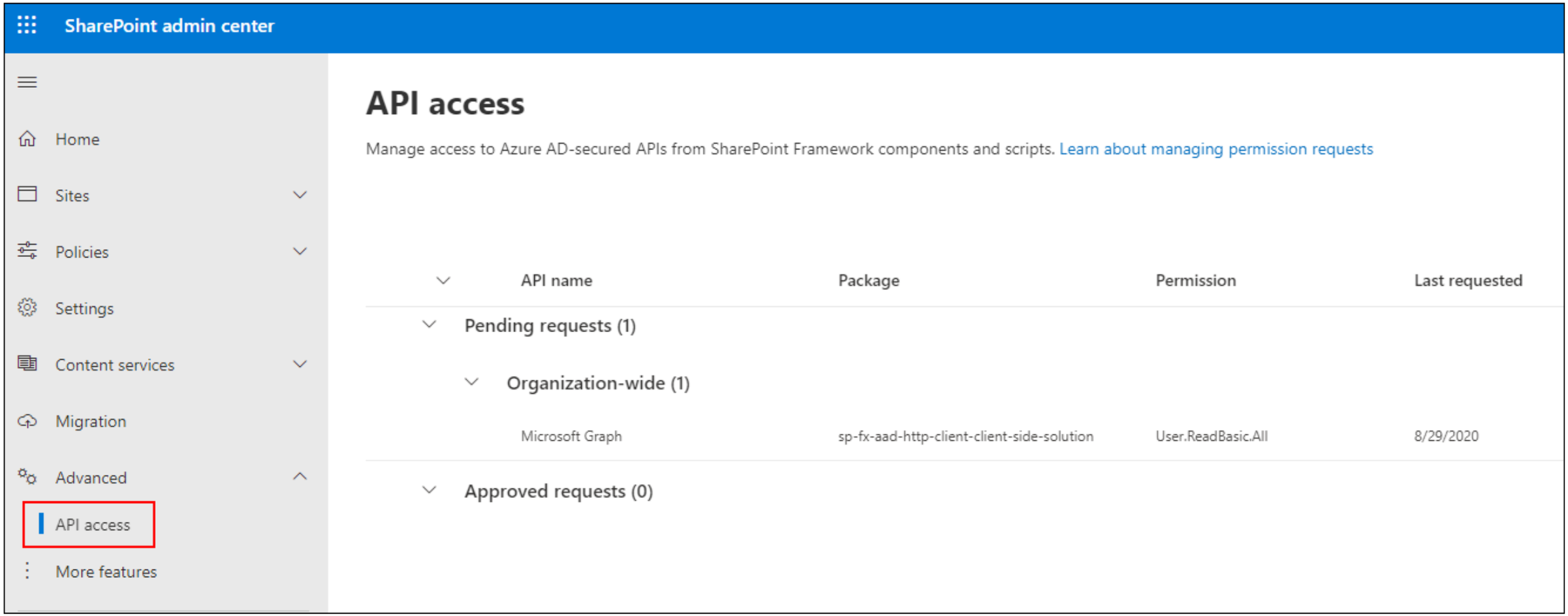
```
// package-solution.json
{
  "solution": {
    "name": "msgraph-sp-fx-client-side-solution",
    "id": "dfb230b7-4f61-431f-9b65-a34e83922663",
    "version": "1.0.0.0",
    "includeClientSideAssets": true,
    "isDomainIsolated": false,
    "webApiPermissionRequests": [
      { "resource": "Microsoft Graph", "scope": "User.ReadBasic.All" },
      { "resource": "Microsoft Graph", "scope": "Calendars.Read" },
      { "resource": "Microsoft Graph", "scope": "Tasks.Read" }
    ]
  },
  "paths": {
    "zippedPackage": "solution/msgraph-sp-fx.sppkg"
  }
}
```

Add SharePoint Package to SharePoint App Catalog

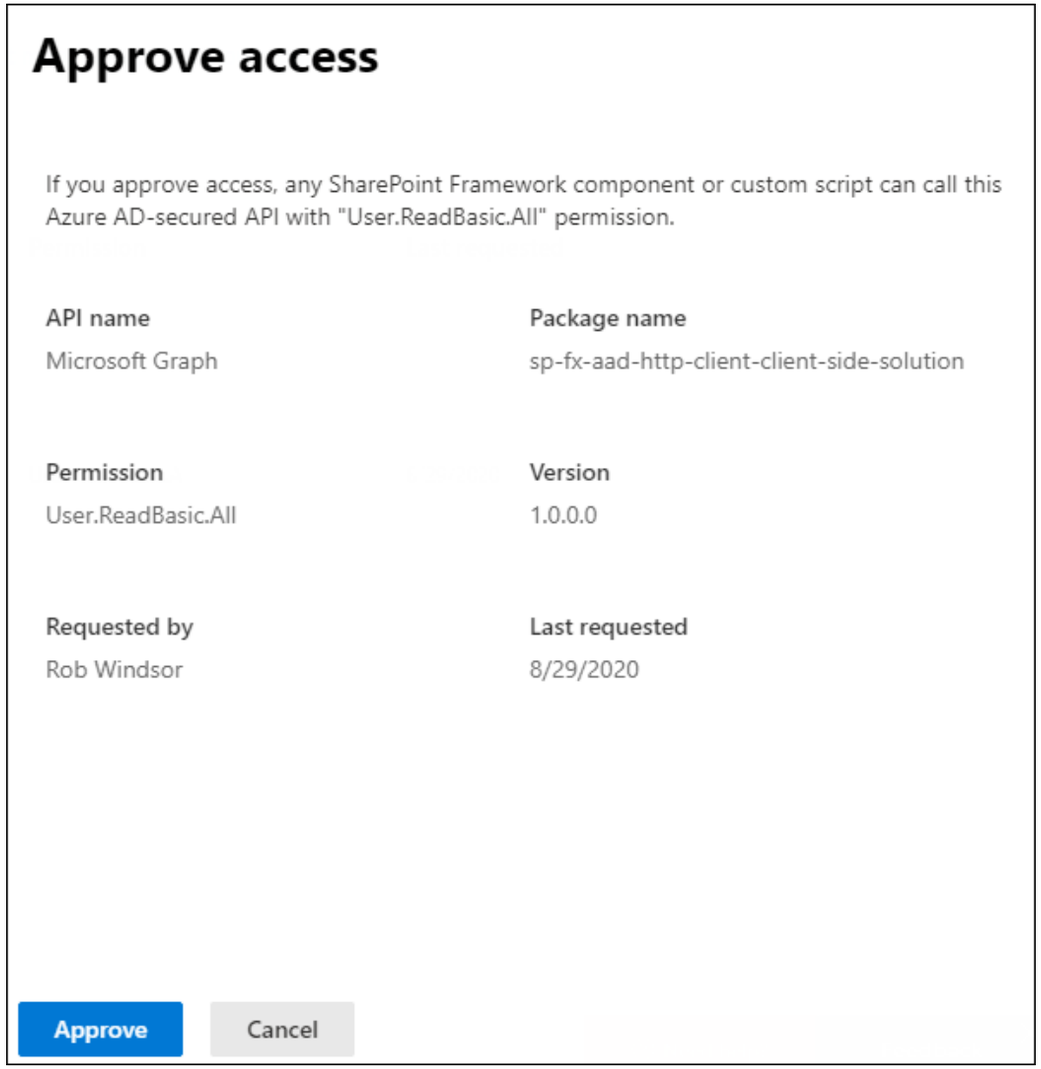
- Trust dialog
  - Extra note in dialog notifies of additional step required
  - While application can be installed in SharePoint sites, it does not have the permissions granted that it needs to access Azure AD protected resources



Approve / Reject with SharePoint Online API Management Page



Approve / Reject with SharePoint Online API Management Page

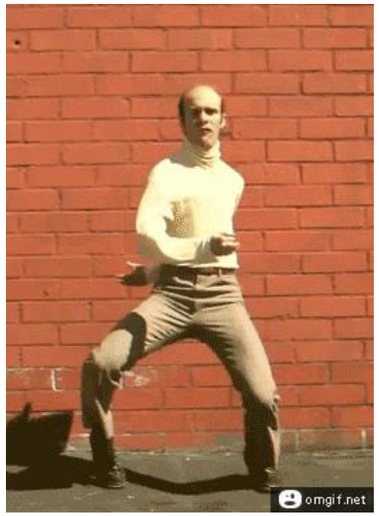


DEMO TIME!

Put on your dancing shoes and let’s have some fun!







PNP Js library for data interaction

Overview of the PnPJs

Getting started

Getting Started with PnPJs

- PnPjs is a collection of fluent libraries for consuming SharePoint, Graph, and Office 365 REST APIs in a type-safe way.
- https://aka.ms/pnpjs
- Ideal for SPFx
- Great documentation
- Easy to set up and use

```
npm install @pnp/sp @pnp/graph --save
```

Getting Started with PnPJs

- Initialize by passing the context

```
import { spfi, SPFx } from "@pnp/sp";

//...

protected async onInit(): Promise<void> {
  await super.onInit();
  const sp = spfi().using(SPFx(this.context));
}

//...
```

Getting Started with PnPJs

- Import additional components vs All
  - Keep the bundle smaller
  - SharePoint

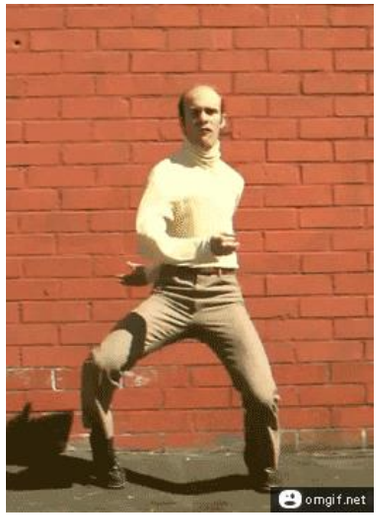
```
import { spfi, SPFx } from "@pnp/sp";
import "@pnp/sp/webs";
import "@pnp/sp/lists";
import "@pnp/sp/items";

// initialize
const sp = spfi (...)

// get all the items from a list
const items: any[] = await sp.web.lists.getByTitle("My List").items();
```

DEMO TIME!

Put on your dancing shoes and let’s have some fun!







Microsoft 365 &
Power Platform Community

BLOG COMMUNITY CALLS GUIDANCE SAMPLES & SOLUTIONS SDKS TOOLS

Microsoft 365 & Power Platform Community

Learn from others how to build apps on Microsoft 365 & Power Platform.

Don't reinvent the wheel. Focus on what truly matters for your organization.

Changing the world one contribution at a time!

SEE INITIATIVES →

aka.ms/m365/community

Thanks!

Do you have any questions?

https://donkirkham.com
@DonKirkham
/in/DonKirkham
DonKirkham