

Connecting SPFx Projects to Data Sources



JS PADOAN

MICROSOFT CERTIFIED TRAINER

@JsPadoan <https://www.linkedin.com/in/jspadoan>



Overview



Connecting and using Microsoft Graph

- Graph APIs
- Graph Explorer
- Graph Toolkit

Using third party APIs

Establishing connections between SPFx Webparts

- Provider
- Consumer



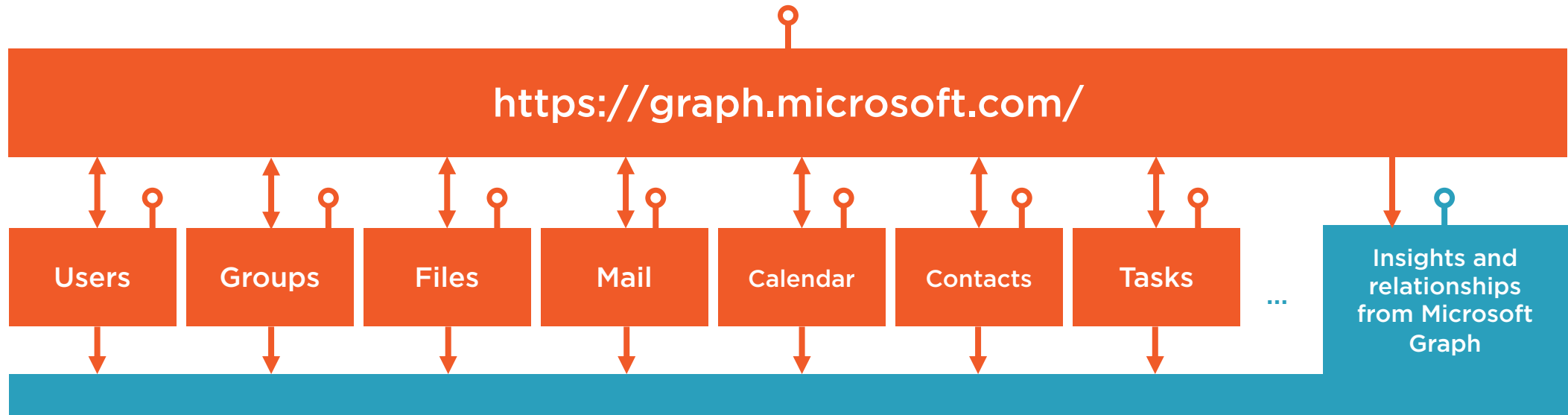
Connecting and Using Microsoft Graph



Overview of Microsoft Graph



Overview of Graph APIs



`https://graph.microsoft.com/v1.0/me/planner/tasks`

`https://graph.microsoft.com/v1.0/me/drive/recent`



Making Live Testing with Graph Explorer

The screenshot shows the Microsoft Graph Explorer interface. On the left sidebar, annotation 1 points to the user profile section (JS PADOAN, admin@m365x026310.on...), and annotation 2 points to the 'Sample queries' section. The main area has a top bar with a GET method dropdown, a version selector (v1.0), a URL input field (https://graph.microsoft.com/v1.0/me/), and a 'Run query' button (annotation 4). Below this is the 'Request body' section, which is currently empty. A status bar shows a successful response: 'OK - 200 - 368ms'. Below the status bar is the 'Response preview' section, which displays the JSON response (annotation 5). The response includes the user's profile information, such as 'displayName': 'JS PADOAN' and 'givenName': 'Jean-Sébastien'.

1

2

3

4

5

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



Making Live Testing with Graph Explorer

The screenshot shows the Microsoft Graph Explorer interface. On the left is a sidebar with the user profile (JS PADOAN) and a list of sample queries under 'Microsoft To Do (4)'. The main area is divided into two tabs: 'Modify permissions' (active) and 'Response preview'. The 'Modify permissions' tab shows a table of required permissions for the 'Tasks.ReadWrite' scope. The 'Response preview' tab shows an error response from the API.

2 points to the 'beta' dropdown menu in the top bar.

3 points to the 'Consent' button in the 'Permissions (1)' section.

1 points to the 'Response preview' tab, which displays the following error response:

```
{
  "error": {
    "code": "notAllowed",
    "message": "Access is denied to the requested resource. The user might not have enough permission.",
    "innerError": {
      "code": "ErrorAccessDenied",
      "message": "Access is denied. Check credentials and try again.",
      "date": "2020-10-29T13:33:30",
      "request-id": "7648d104-5d8d-4faf-b20f-a8936f2547c8",
      "client-request-id": "75f4438a-fe51-cf20-dcde-ed0ac3cf07b2"
    }
  }
}
```



Making Live Testing with Graph Explorer

Graph Explorer

JS PADOAN
admin@m365x026310.onmicrosoft.com

Sample queries History

Search sample queries

See more queries in the [Microsoft Graph API Reference docs](#).

- Excel (7)
- Extensions (7)
- Groups (8)
- Insights (4)
- Microsoft Teams (9)
- Microsoft Teams (beta) (4)
- Microsoft To Do (4)
 - GET** get To Do task lists
 - POST** create To Do task list

GET beta <https://graph.microsoft.com/beta/me/todo/lists> **Run query**

Request body Request headers **Modify permissions** Access token

Permissions (1)

The following permissions are required to run the query. To consent to the permissions, click Consent.

Permission	Display string	Description	Admin consent requir...	Status
Tasks.ReadWrite	Create, read, update, and delete your t...	Allows the app to create, read, update, and delete your tasks and task lists, in	x	Consented

OK - 200 - 2095ms

When you use Microsoft Graph APIs, you agree to the [Microsoft APIs Terms of Use](#). View the [Microsoft Privacy Statement](#).

Response preview Response headers Code snippets Toolkit component Adaptive cards

```
{
  "@odata.context": "https://graph.microsoft.com/beta/$metadata#users('8fbff396-ba2b-43c8-a236-c7855bdbfabe')/todo/lists",
  "value": [
    {
      "@odata.etag": "W/\"NmyisKOCQkiZYUmhWiIvJQAABNMxog==\"",
      "displayName": "Tasks",
      "isOwner": true,
      "isShared": false,
      "wellknownListName": "defaultList",
      "id": "AAMKADQ2Y2E0NGNhLTi4ZjMtNGFlYi05MmQ1LTht5Yjk5MmFkYTk2OAAUAAAAACuentGZpvfSZpnU0ag9YyHAQBabKKwo4JCSj1hSaFaIi81AAAAAESAAA="
    }
  ]
}
```



Using Microsoft Graph Toolkit

The screenshot displays the Microsoft Graph Toolkit Playground interface. On the left, a sidebar contains a search bar and two sections: 'COMPONENTS' and 'SAMPLES'. The 'COMPONENTS' section lists various components like 'mgt-agenda', 'mgt-get', 'mgt-login', 'mgt-people', 'mgt-people-picker', 'mgt-person', 'mgt-person-card', 'mgt-tasks', and 'mgt-teams-channel-picker'. The 'SAMPLES' section includes 'General', 'Login To Show Agenda' (highlighted with a blue bar), and 'Templating'. An orange arrow labeled '1' points to the 'COMPONENTS' section, and an orange arrow labeled '2' points to the 'Login To Show Agenda' sample. In the center, a 'Canvas' area shows a preview of the 'Login To Show Agenda' sample, featuring a user profile for 'Megan Bowen' and a list of meetings: 'Project Team Meeting' (4:00 PM - 5:30 PM), 'Tailspin Project Discussion' (6:00 PM - 7:00 PM), and another meeting (6:30 PM - 8:30 PM). An orange arrow labeled '4' points to the 'mgt-people-picker' component in the 'COMPONENTS' list. On the right, a code editor shows the HTML code for the sample, with tabs for 'html', 'js', and 'css'. The 'html' tab is active, displaying the following code:

```
1 <mgt-login></mgt-login>
2 <mgt-agenda></mgt-agenda>
```

 An orange arrow labeled '3' points to the code editor. At the bottom, a 'Sign In' button and 'Actions' section are visible, along with a note: 'All components are using mock data - sign in function will be available in a future release'.

<https://mgt.dev/>



Using Graph APIs into SPFx Components

```
import { MSGraphClient } from '@microsoft/sp-http'; ← 1

...

context.msGraphClientFactory ← 2
    .getClient() ← 3
    .then((c: MSGraphClient): void => {
        c.api('/me/joinedTeams').get((error, response: any, rawResponse?: any) => {
            ← 4
            this.setState({ nbTeams : Object.keys(response).length});
            ← 5
        });
    });
});
```



Using Graph APIs into SPFx Components

```
import { MSGraphClient } from '@microsoft/sp-http';  
  
import * as MicrosoftGraph from '@microsoft/microsoft-graph-types'; ← 1  
  
...  
  
context.msGraphClientFactory  
  .getClient()  
  .then((c: MSGraphClient): void => {  
    c.api('/me').get((error, user: MicrosoftGraph.User, rawResponse?: any) => { ← 2  
      this.setState({ profile : user}); ← 3  
    });  
  });
```



Demo



Create a SPFx Webpart to display user details:

- Use Graph API
- Retrieve user profile details
- Retrieve the number of Teams the currently connected user belongs to



Using Third Party APIs



Using Third Party APIs in SPFx Components

```
import {  
    HttpClient,  
    HttpClientResponse  
} from "@microsoft/sp-http";
```



...

```
private _prepareHeaders(): Headers {  
    const requestHeaders: Headers = new Headers();  
    requestHeaders.append("Accept", "application/json");  
    requestHeaders.append("Content-Type", "application/json");  
    requestHeaders.append("Cache-Control", "no-cache");  
    requestHeaders.append("Ocp-Apim-Subscription-Key", "THEKEY");  
    return requestHeaders;  
}
```



...



Using Third Party APIs in SPFx Components

```
const response: HttpClientResponse = await context.httpClient.post(  
  "URL",  
  HttpClient.configurations.v1,  
  {  
    body: JSON.stringify(body),  
    headers: this._prepareHeaders()  
  }  
);
```

```
const responseJSON: any = await response.json();
```



Demo



Extract and display language and sentiment of comments

Modify previously created Webpart (ArticleComments)

- Connect to SharePoint List API
- Retrieve comments on an article page

Call Microsoft Sentiment Analysis Cognitive API

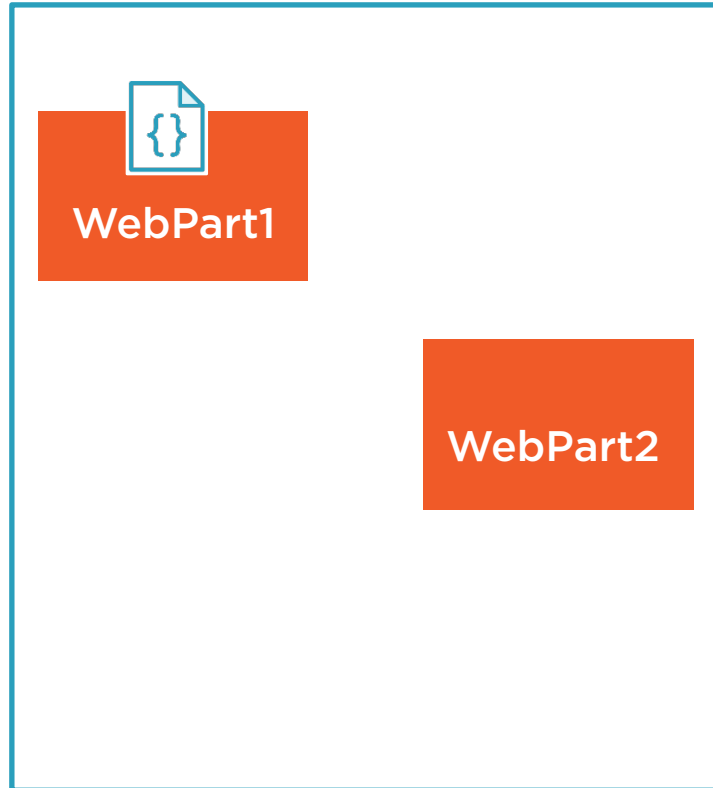
- Test Cognitive API calls with Postman
- Modify the Webpart to include on-the-fly sentiment analysis



Establishing Connections between SPFx WebParts



Overview of Connecting Web Parts



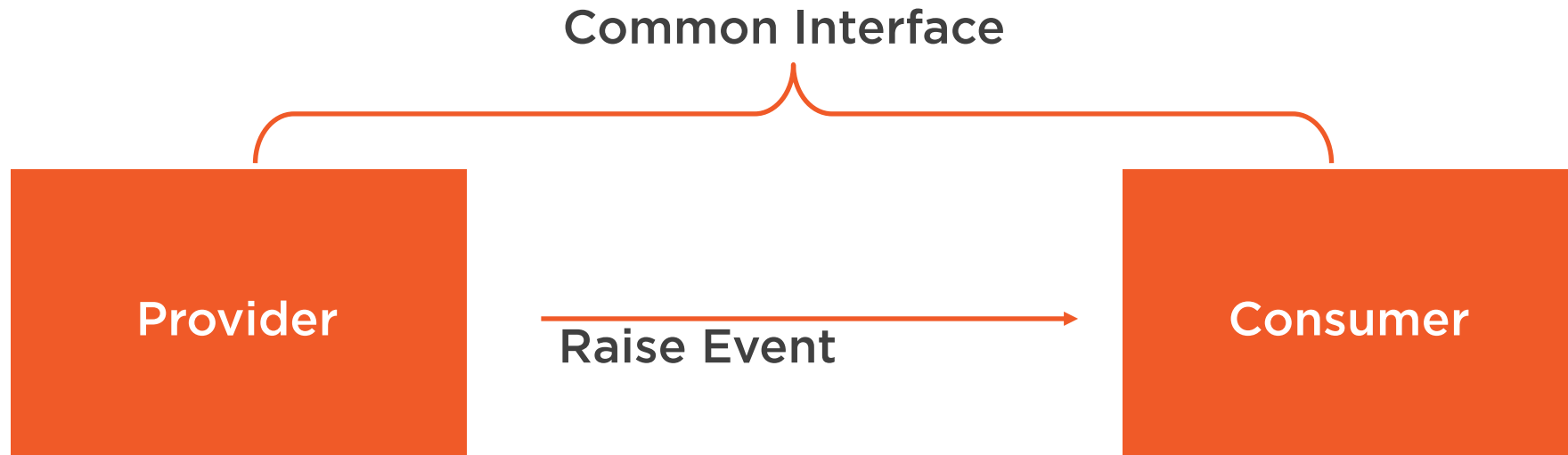
Establish link between web parts on the same page

A link connects 1 origin to 1 destination

Multiple links can be made to the same origin

At a given time, data will be transmitted through established links

Overview of Connecting Web Parts



Creating Endpoints (Provider)

```
export interface IData { ... } ← 1

...

import { ← 3
  IDynamicDataPropertyDefinition,
  IDynamicDataCallables, IDynamicDataAnnotatedPropertyValue
} from '@microsoft/sp-dynamic-data';
...
export default class myWebPart ... implements IDynamicDataCallables { ← 2

  public getPropertyDefinitions(): ReadonlyArray<IDynamicDataPropertyDefinition> { } ← 4

  public getPropertyValue(propertyId: string) : IData { } ← 5

  public getAnnotatedPropertyValue?(propertyId: string): IDynamicDataAnnotatedPropertyValue { } ← 6
  ...

  // notify subscribers that 'TheProperty' has changed
  WebPartContext.dynamicDataSourceManager.notifyPropertyChanged('TheProperty'); ← 7

}
```



Consuming Endpoints

```
export interface IWebPartProps {  
myProperty: DynamicProperty<IData>;
```

← 1

```
...  
}
```

```
...
```

```
protected getPropertyPaneConfiguration(): IPropertyPaneConfiguration {
```

← 2

```
...  
  PropertyPaneDynamicFieldSet({  
    label: 'Select event source',  
    fields: [ PropertyPaneDynamicField('myProperty', {label: '...'}) ]  
  })  
...  
}
```

```
public async componentDidUpdate?(...): Promise<void> {  
  const data: IData = this.props.myProperty.tryGetValue();
```

← 3

```
...  
}
```



Demo



Create Webparts and make them compliant to be connected:

- Create a Webpart (FilterNews) that could expose category of news
- Create a Webpart (SentimentNews) that could retrieve the category of news to search with Bing Search API
- Narrow the search with Sentiment Analysis

Connect the Webparts on a page to see the connection working



Summary



Connecting and using Microsoft Graph

- Graph APIs
- Graph Explorer
- Graph Toolkit

Using third party APIs

Establishing connections between SPFx web parts

- Provider
- Consumer

