**SharePoint Framework (SPFx) - Graph API - ReactJS Get Calendar Events from Office 365.**

In this article, we will see how we can get calendar events from Office 365 into SharePoint framework – SPFx ReactJS web part using Graph API.

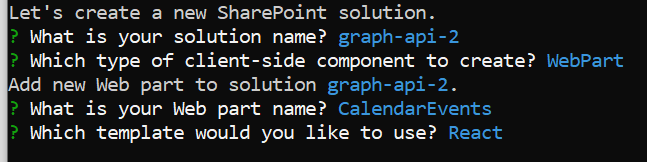
**Step 1:**Create a folder with the name GraphAPI-2 on your local drive.

C:\Users\ABC\Documents\SPFx\**GraphAPI-2**

**Step 2:**Open the location in the command prompt using cd command. Scaffold the SPFx solution using Yeoman Generator

*yo @microsoft/sharepoint*

**Step 3:**Give the webpart name and other details as shown below



**Step 4:**Install MS Graph types node package by using command:

*npm install @microsoft/microsoft-graph-types --save-dev*

**Step 5:** Open the solution in VS Code. You can use the "Code ." command in the command prompt directly to open the solution in VS Code. Update React component properties file **ICalendarEventsProps.ts.** Add property for **WebPartContext**

import { WebPartContext } from '@microsoft/sp-webpart-base';

export interface ICalendarEventsProps {

  context: WebPartContext;

}

**Step 6:** Add a new Type script file **ICalendarEventsState.ts** in components folder for component state to store data from Graph API. It holds all the events returned by Graph API.

import \* as MicrosoftGraph from '@microsoft/microsoft-graph-types';

export interface ICalendarEventsState {

    events: MicrosoftGraph.Event[];

}

**Step 7:** Update web part render() method and add context property.

context: this.context

**Step 8:** Update component tsx file **CalendarEvents.tsx**

Import Graph and State

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

import \* as MicrosoftGraph from '@microsoft/microsoft-graph-types';

import { ICalendarEventsState } from './ICalendarEventsState';

Add state in component

export default class CalendarEvents extends React.Component<ICalendarEventsProps, ICalendarEventsState>

Define constructor and initialize state:

  constructor(props: ICalendarEventsProps) {

    super(props);

    this.state = {

      events: []

    };

  }

Add **componentDidMount()** method which is called when everything is rendered.

  public componentDidMount(): void {

    this.props.context.msGraphClientFactory.getClient().then((client: MSGraphClient): void => {

      client

        .api('/me/calendar/events')

        .version("v1.0")

        .select("\*")

        .get((error: any, eventsResponse, rawResponse?: any) => {

          if (error) {

            console.error("Message is : " + error);

            return;

          }

          const calendarEvents: MicrosoftGraph.Event[] = eventsResponse.value;

          this.setState({ events: calendarEvents });

        });

    });

  }

**Step 9:** Update **render** method of component tsx file **CalendarEvents.tsx**

<div>

         <ul>

            {

              this.state.events.map((item, key) =>

                <li key={item.id}>

                  {item.subject},{item.organizer.emailAddress.name},

                  {item.start.dateTime.substring(0, 10)},

                  {item.start.dateTime.substring(12, 5)},

                  {item.end.dateTime.substring(0, 10)},

                  {item.end.dateTime.substring(12, 5)}

                </li>)

            }

          </ul>

   </div>

Once **render()** method gets over **componentDidMount()** method is called and it uses setState to add events data in array. As setState is called, it calls render method again and display all the events data in web part.

**Step 10:** Update **package-solution.json** file inside **config** folder. Add **webApiPermissionRequests** inside **solutions** tag. This is for read permission of calendar events.

"webApiPermissionRequests": [

      {

        "resource": "Microsoft Graph",

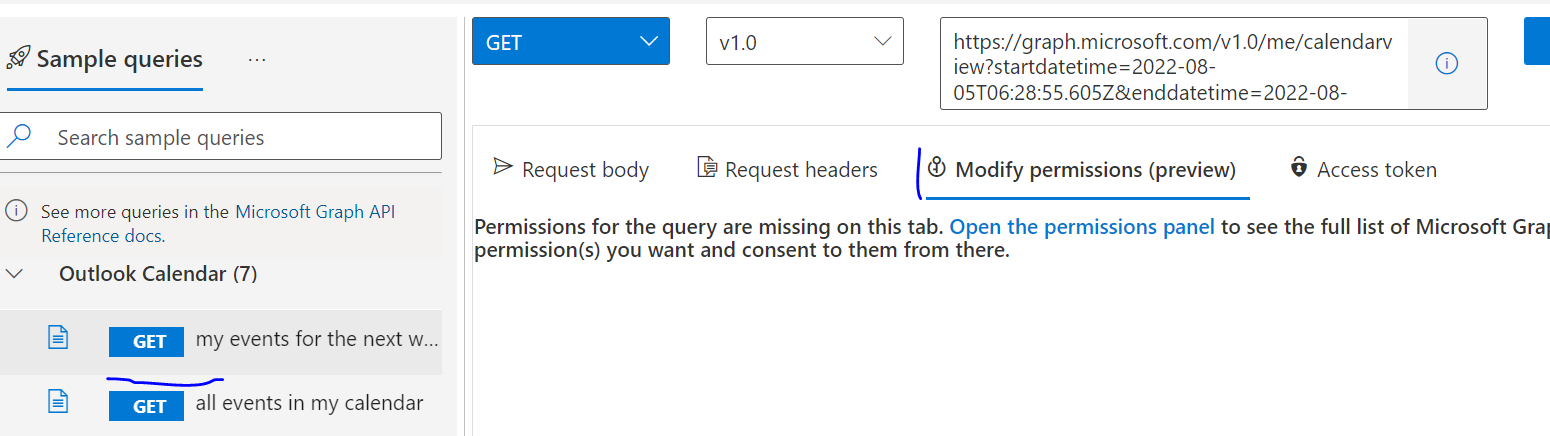
        "scope": "Calendars.Read"

      }

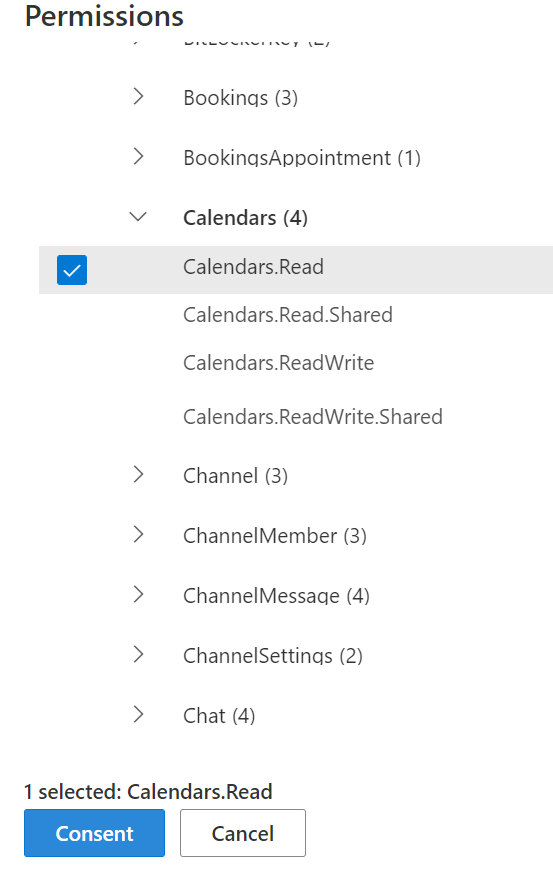
    ]

**Step 11:** To test this web part in workbench, go to Graph explorer. <https://developer.microsoft.com/en-us/graph/graph-explorer>

Login with your credentials. Open Outlook Calendar queries and click on ‘Get my events for the next week’. Go to **Modify permissions** and click on ‘**Open the permissions panel**’

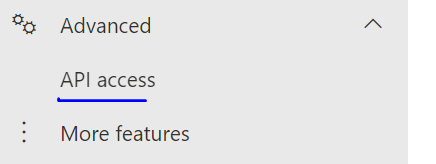


**Step 12:** Click on ‘**Calendars.Read**’ and provide the consent by clicking on Consent button.

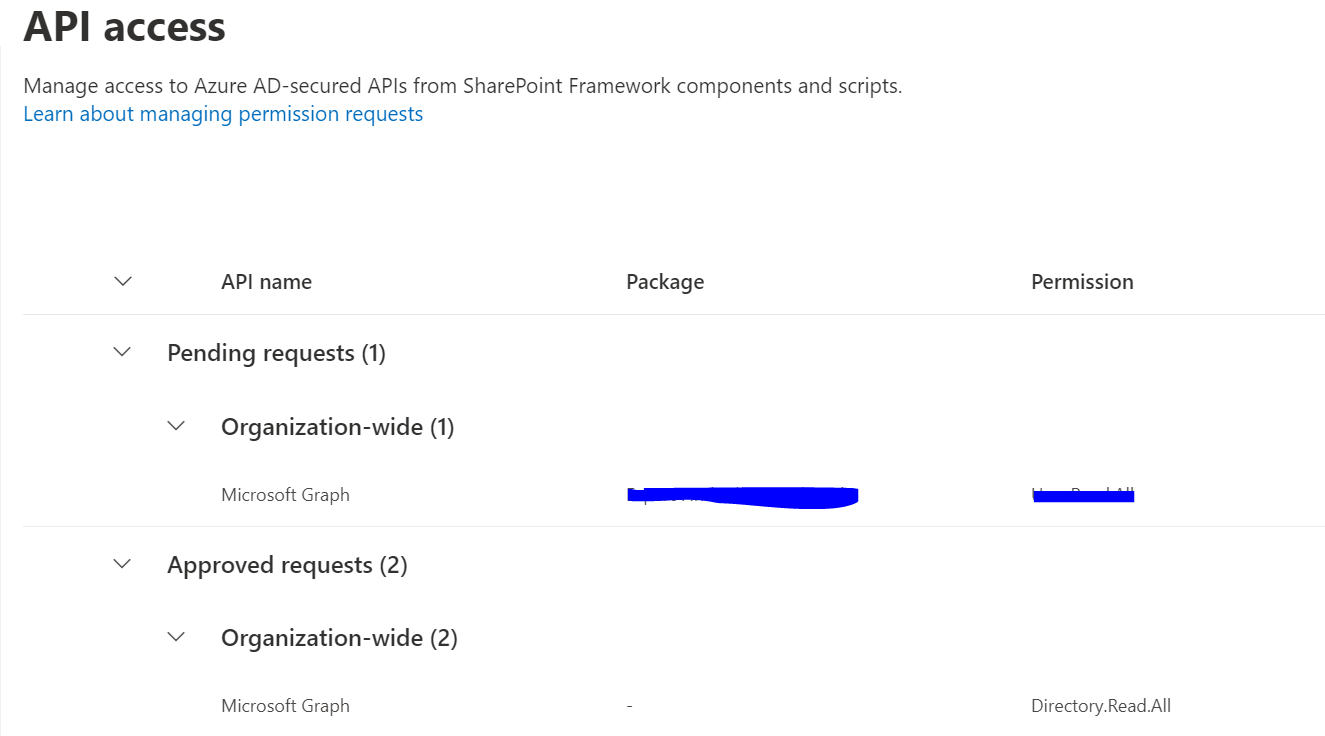


**Step 13:** If you are not able to build test web part in workbench then deploy your web part in App Catalog.

**Step 14:** After deployment in App Catalog. Go SharePoint admin center. Go to Advanced 🡪 **API Access**



You will see Pending API access requests. In this case it will be **Package**: Your package deployed in App Catalog. **Permission**: Calendar.Read (what we specified in **package-solution.json** file of the SPFx solution.



Once you approve the request, your web part has permissions to read the calendar.

Now you can add your web part to page and it will show entries from your calendar.