

# Programming with the Microsoft Graph API



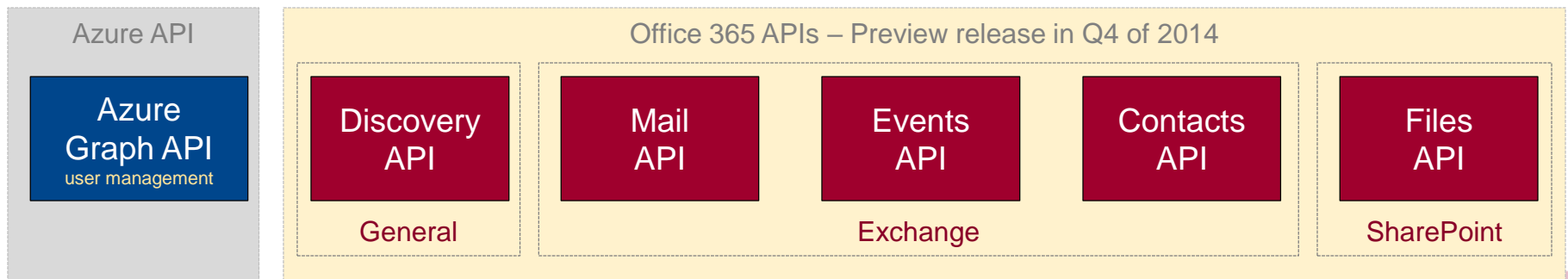
# Agenda

- Overview of Microsoft Graph API
- Constructing URLs for the Microsoft Graph API
- Developing Applications with the Microsoft Graph API
- Programming SPFx Webparts using MSGraphClient
- Granting Microsoft Graph API Permissions



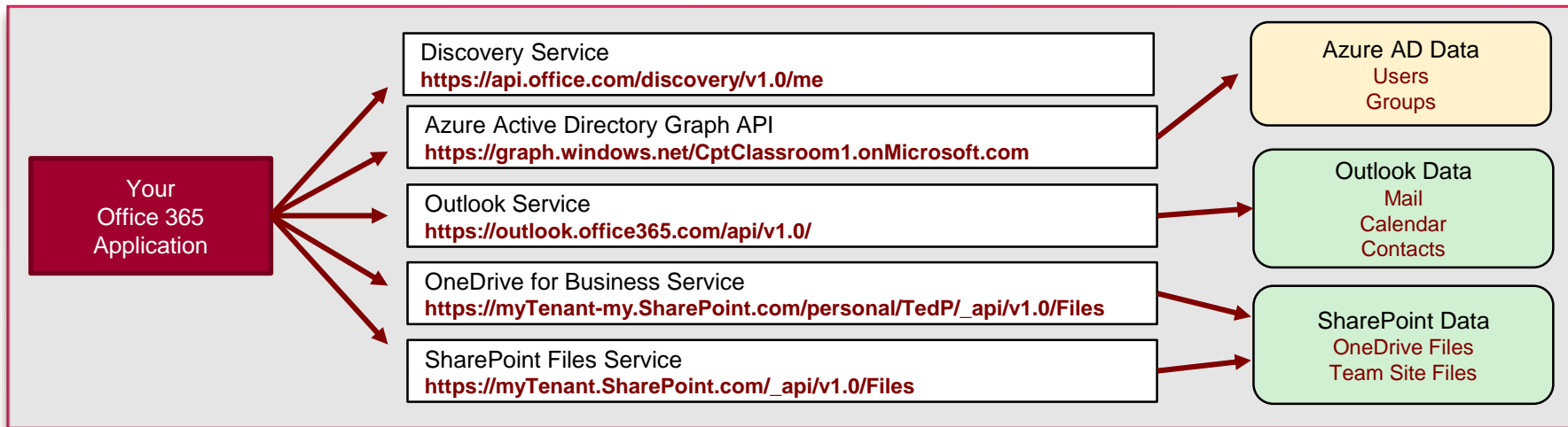
# Microsoft Released Office 365 APIs in 2014

- Office 365 APIs created for accessing data in Office 365
  - Implemented as RESTful services based on ODATA version 4.0
  - Provides authentication and authorization based on OAuth 2.0
  - Provides extra authentication support for OpenID Connect
- Open standards provide wide range of accessibility
  - Many choices for tools, languages and development platforms
  - Microsoft has created Office 365 SDKs for specific platforms
- Office 365 APIs set the stage for the Microsoft Graph API
  - Microsoft Graph API created to simplify accessing Office 365



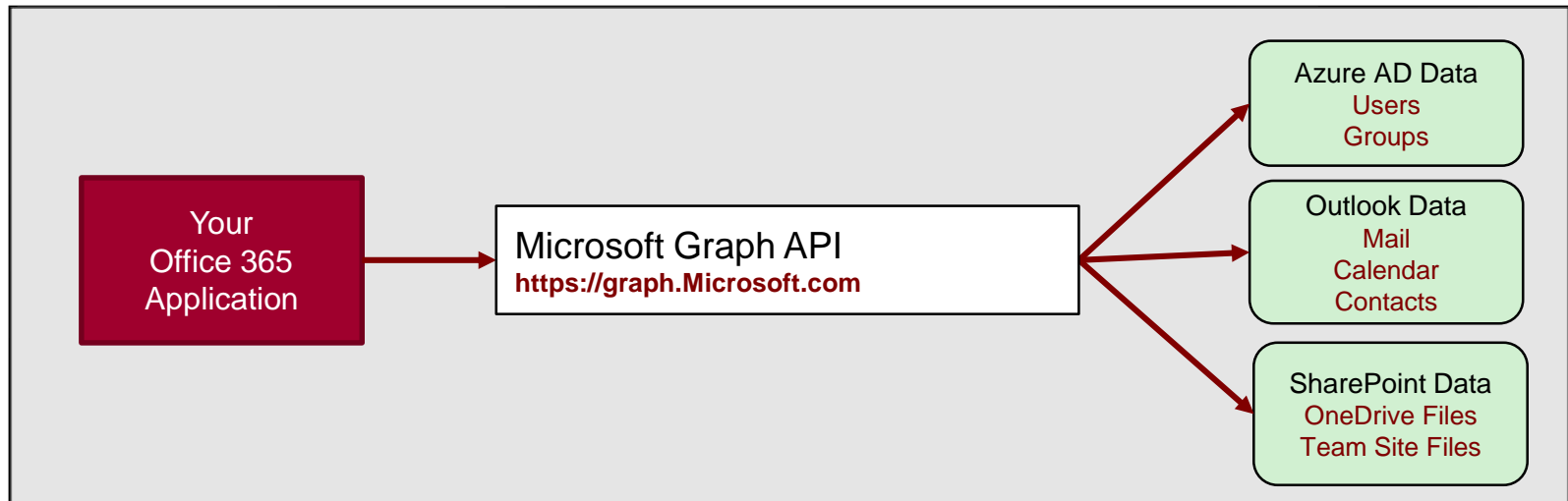
# Office 365 API Endpoint in Initial Release

- Azure Endpoints
  - Azure Graph API
- Office 365 API Endpoints
  - Outlook service
  - OneDrive for Business Service
  - SharePoint Files Service
  - Discovery Services



# The Microsoft Graph API

- Designed as a single, more-comprehensive service
  - Abstracts away divisions between AD, Exchange and SharePoint
  - No need to discover endpoints using the Discovery Service
  - You can acquire and cache a single access token per user



# What Does It Do?

- Single resource that proxies multiple Microsoft services
- Allows for easy traversal of objects and relationships
- Simplifies token acquisition and management
- Eliminates the need to traditional discovery
  - It does this by using "me" and "myorganization"



# Universal Access

- Direct REST API
  - Any platform
  - Any language
  - Any framework
- Native SDKs
  - Utilize framework & platform specific implementations
  - Abstracts the details of building & processing requests over HTTP
  - .NET, iOS, Android, PHP, Ruby, JavaScript, etc.





# The Graph Explorer

- Graph Explorer used to execute calls into Microsoft Graph API
  - Great place to discover what the API can do
  - <https://developer.microsoft.com/en-us/graph/graph-explorer>

The screenshot displays the Microsoft Graph Explorer web application. The browser address bar shows the URL: `https://developer.microsoft.com/en-us/graph/graph-explorer#access_token=eyJ0eXAiOiJKV1QiLCJub25jZSI6IkFRQUJBQUFBQUFEWHpaM2lmc1H`. The page header includes the Microsoft logo and navigation links: Microsoft Graph, Solutions, Graph Explorer, Quick Starts, Docs, API Reference, Samples & SDKs, Blog, Support, and My Apps.

The left sidebar contains the following sections:

- Authentication:** Shows the user "Ted Pattison" with email "student@msd0910.onmicrosoft.com" and a "sign out" link.
- Sample Queries:** A list of queries for SharePoint Sites and Microsoft Teams (beta), each with a "GET" button and a document icon.
  - SharePoint Sites:
    - my organization's default SharePoint site
    - Enumerate the document libraries under the root site
    - SharePoint site based on relative path of the site
    - Search for a SharePoint site by keyword
    - Enumerate subsites of the root site
    - Enumerate site columns of the root site
    - Enumerate site content types of the root site
  - Microsoft Teams (beta):
    - my joined teams
    - members of a team
    - channels of a team which I am member of
    - channel info

The main content area shows a request configuration with the method "GET", version "v1.0", and the URL `https://graph.microsoft.com/v1.0/sites/root/drives`. Below the configuration, the "Request Body" tab is active, showing an empty field. The "Response Preview" tab is also active, displaying a JSON response:

```
{
  "@odata.context": "https://graph.microsoft.com/v1.0/$metadata#drives",
  "value": [
    {
      "createdDateTime": "2018-08-31T20:01:13Z",
      "description": "",
      "id": "b!_TPjP3rNE2wzU5_dTud11E14Yk5H49OveTp-rjXyUuWpVhKpWdNR6nG3_U8xa9W",
      "lastModifiedDateTime": "2018-08-31T20:01:13Z",
      "name": "Documents",
      "webUrl": "https://msd0910.sharepoint.com/Shared%20Documents",
      "driveType": "documentLibrary",
      "createdBy": {

```

A green status bar indicates "Success - Status Code 200, 720ms".





# Agenda

- ✓ Overview of Microsoft Graph API
- Constructing URLs for the Microsoft Graph API
  - Developing Applications with the Microsoft Graph API
  - Programming SPFx Webparts using MSGraphClient
  - Granting Microsoft Graph API Permissions



# Calling the Graph API

```
https://graph.microsoft.com  
/{version}/{resource}/{id}/{property} ?{query-parameters}
```

HTTP verbs dictate the request intent: GET | POST | PATCH | PUT | DELETE

- Version: /v1.0 or /beta
- Resource: /users, /groups, /sites, /drives, /devices, more...
- Member from collection: /users/AAA
- Property: /users/AAA/department
- Traverse to related resources via navigations: /users/AAA/events
- Query parameters: /users/AAA/events?\$top=5

Format results: \$select | \$orderby

Control results: \$filter | \$expand

Paging: \$top | \$skip | \$skiptoken



# Pagination

---

Graph uses server-side page size limits

---

---

When querying collections, Graph may return the results in many pages

---

---

Always expect an `@odata.nextLink` property in the response

---

Contains the URL to the next page

---



## Request

```
GET https://graph.microsoft.com/v1.0/me/messages?$select=subject,from
```

## Response

```
{
  "@odata.nextLink": "https://graph.microsoft.com/v1.0/me/messages?$select=subject%2cfrom&$skip=16",
  "value": [
    {
      "subject": "Your Azure AD Identity Protection Weekly Digest",
```

1.

Always handle the possibility that the responses are paged in nature

2.

Follow the @odata.nextLink to obtain the next page of results

3.

Final page will not contain an @odata.nextLink property

4.

Treat the entire URL as an opaque string



## Querying data | Use filters

---

Choose the **records** your app really needs and no more

---

---

Don't send unnecessary data over the wire

---

---

**Tip**  
Use **\$filter**

---

**GET** [https://graph.microsoft.com/v1.0/users?\\$filter=department eq 'Sales' & \\$select=givenName,mail](https://graph.microsoft.com/v1.0/users?$filter=department eq 'Sales' & $select=givenName,mail)



## POST/PATCH/PUT | no response required

---

If your code doesn't need to get a response, then opt out

---

---

Don't send unnecessary data over the wire

---

---

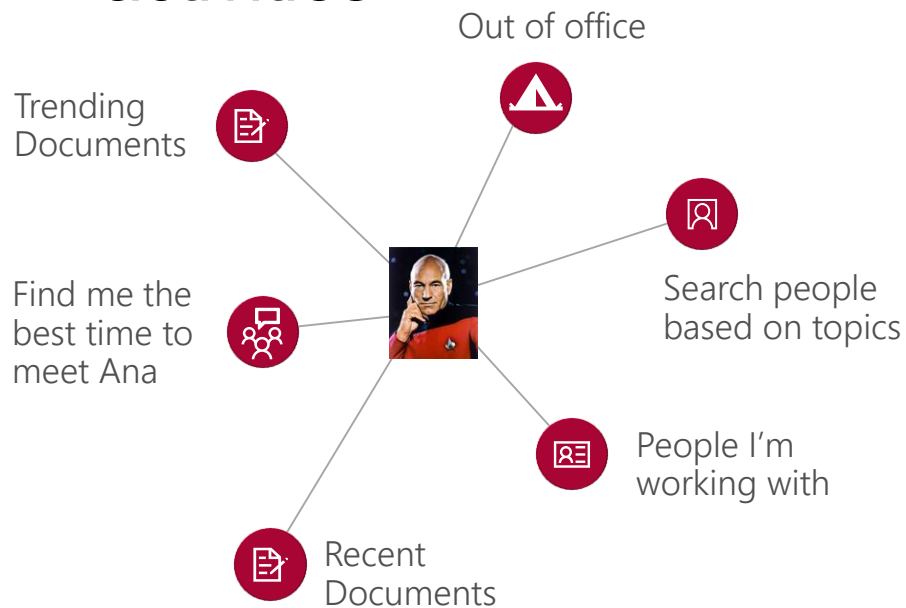
**Tip**  
Use HTTP  
**Prefer return=minimal**  
request header

---



# With Microsoft Graph

## Get **insights** based on activities



```
GET /me/insights/trending
"value" : [
  {"name": "presentation.pptx", ...},
  {"name": "forecast.xlsx", ...}
]
```

```
GET /me/drive/recent
"value" : [
  {"name": "guidelines.pptx", ...},
  {"name": "budget.xlsx", ...}
]
```

```
GET people/?$search="topic: planning"
"value" : [
  {"displayName": "Data", ...},
  {"displayName": "Worf", ...}
]
```

```
POST: /me/findMeetingTimes
{
  "attendees": [
    {
      "type": "required",
      "emailAddress": {
        "address": "riker@contoso.com"
      }
    }
  ],
  "meetingDuration": "2h"
}
```



# Read data from existing SharePoint site

GET <https://graph.microsoft.com/beta/sites/{siteId}/lists/{listId}/items?expand=columnSet>

```
{ "value": [ {  
  "createdBy": { "user" : { "id": "d54e4cdd-d2ca-4c39-bfa5-35895bca12f0", "displayName": "John" } },  
  "createdDateTime": "2017-09-20T08:16:21Z",  
  "eTag": "1610ac6a-24f6-4458-9733-1e5977c63caa,1",  
  "id": "1610ac6a-24f6-4458-9733-1e5977c63caa",  
  "lastModifiedBy": { "user": { "id": "d54e4cdd-d2ca-4c39-bfa5-35895bca12f0", "displayName": "John" } },  
  "lastModifiedDateTime": "2017-09-0T08:16:21Z",  
  "webUrl": "https://site.sharepoint.com/sites/mysite/Lists/mytasks/1_.000",  
  "listItemId": 1,  
  "columnSet": {  
    "Title": "Project Upgrade: Use the Microsoft Graph",  
    "Description": "Set up group for new technologies.",  
    "id": "1",  
    ...  
  }  
} ] }
```



# OneDrive + Excel Services

```
GET https://graph.microsoft.com/v1.0/me/drive/  
    root/search(q='.xlsx')?select=name,id,webUrl
```

```
GET https://graph.microsoft.com/1.0/me/drive/  
    items/<id>/workbook/worksheets
```

```
GET https://graph.microsoft.com/beta/me/drive/  
    items/{itemId}/workbook/worksheets('Time')/  
    range(address='a2:d4')
```



# Update Excel Timesheet Data

```
PATCH https://graph.microsoft.com/beta/me/drive/  
      items/{itemId}/workbook/worksheets('Sheet1')/  
      range(address='a2:b2')  
  
{  
  "values": [ ["September", "200.0"] ],  
  "valueTypes": [ ["String", "Double"] ],  
}
```



# New Capabilities

- Traversal of relationships
- Query parameters
- Batching - preview
- Notifications - users & groups - preview
- Track changes - GA
- Extensions - GA



# Agenda

- ✓ Overview of Microsoft Graph API
- ✓ Constructing URLs for the Microsoft Graph API
- Developing Applications with the Microsoft Graph API
  - Programming SPFx Webparts using MSGraphClient
  - Granting Microsoft Graph API Permissions



# Microsoft Graph JavaScript SDK

- Microsoft Graph is accessible via REST API & various SDKs
- Client-side solutions can leverage the JavaScript SDK
  - <https://github.com/microsoftgraph/msgraph-sdk-javascript>
  - Requires initialization with an Azure AD provided OAuth2 access token to create the client



# Initializing the Microsoft Graph JS SDK

```
var client = MicrosoftGraph.Client.init({  
  authProvider: (done) => {  
    done(null, access_token);  
  }  
});
```

```
client  
  .api('/me')  
  .get((err, res) => {  
    console.log(res);  
  });
```





# Microsoft Graph TypeScript Type Declarations

- Use the Microsoft Graph JavaScript SDK in TypeScript applications
- TypeScript type declarations introduce strong types and documentation to client-side projects
  - <https://github.com/microsoftgraph/msgraph-typescript-typings>



# Microsoft Graph TypeScript Type Declarations

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

// init Microsoft Graph client

client
  .api('/me')
  .get((error: any, user: MicrosoftGraph.User, rawResponse?: any) => {
    console.log('name: ', user.displayName);
    console.log('email: ', user.displayName);
    console.log('phone: ', user.businessPhones[0]);
  });
});
```



# Agenda

- ✓ Overview of Microsoft Graph API
- ✓ Constructing URLs for the Microsoft Graph API
- ✓ Developing Applications with the Microsoft Graph API
- Programming SPFx Webparts using MSGraphClient
  - Granting Microsoft Graph API Permissions



# SharePoint Online User Already Authenticated

- Users in SharePoint Online & Office 365 are already authenticated
  - SharePoint Online has same Azure AD dependency as Microsoft Graph
  - Users login to Office 365 with their Work & School account (Azure AD)
- Calls to the Microsoft Graph are proxied through SharePoint Online
  - Eliminates the need for creation of a separate Azure AD application
  - Does not bypass any permission / scope requirements
  - Can only access business entities, not consumer entities



# Calling Services from Client-side Code

- Calling a secured service requires the acquisition of token
  - Authenticate and get authorized in Azure AD application
  - Interactive login & programmatic acquisition of access token
- Client-side components cannot do the same
  - Components cannot securely do this across domains seamlessly
  - Can't store application ID's and secrets client side
  - Require an authentication prompt with a popup or redirection



# SPFx Includes Microsoft Graph Client

- MSGraphClient: Microsoft Graph Client for SPFx
  - Abstracts the token acquisition from the SPFx development
  - Wraps the Microsoft Graph JavaScript SDK line

```
let graphClient: MSGraphClient =  
    this.context.serviceScope.consume(MSGraphClient.serviceKey);
```



# SPFx Proxy Calls through Existing Application

- SharePoint Online already has an Azure AD application
  - Client-side solutions in SharePoint Online call the SharePoint REST API in the same domain
  - No extra authentication required
  - Provided these tenants have granted the necessary scopes, SharePoint will call the Microsoft Graph
  - Responses from the Microsoft Graph are returned back to the client-side application
- Permission requests to Azure AD applications
  - Only SharePoint Online tenant administrators can [grant|reject] permission requests
  - Approved permissions are available to all client-side solutions in a tenant





# SPFx Solutions Declare Permission Requests

```
// package-solution.json
{
  "$schema": "https://developer.microsoft.com/json-schemas/spfx-build/package-solution.schema.json",
  "solution": {
    "name": "ms-graph-sp-fx-client-side-solution",
    "id": "dfb230b7-4f61-431f-9b65-a34e83922663",
    "version": "1.0.0.0",
    "includeClientSideAssets": true,
    "webApiPermissionRequests": [
      { "resource": "Microsoft Graph", "scope": "User.ReadBasic.All" },
      { "resource": "Microsoft Graph", "scope": "Calendars.Read" },
      { "resource": "Microsoft Graph", "scope": "Tasks.Read" }
    ]
  },
  "paths": {
    "zippedPackage": "solution/ms-graph-sp-fx.sppkg"
  }
}
```



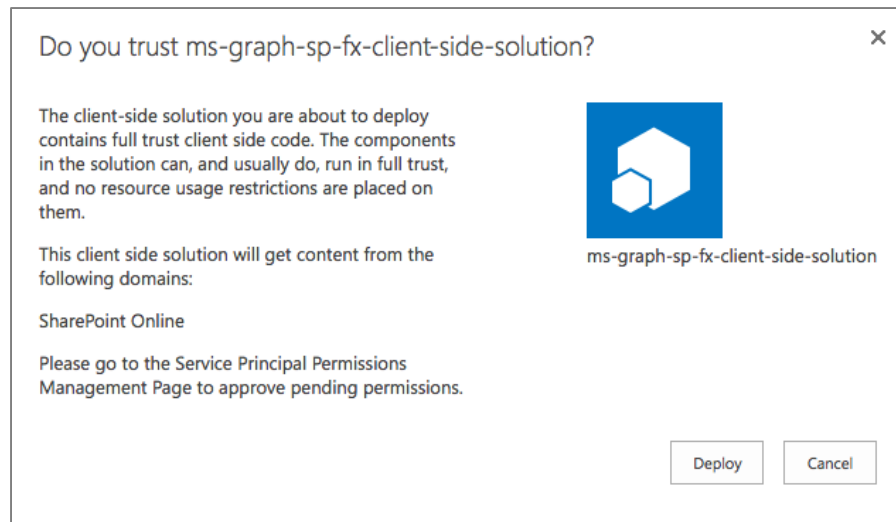
# Agenda

- ✓ Overview of Microsoft Graph API
- ✓ Constructing URLs for the Microsoft Graph API
- ✓ Developing Applications with the Microsoft Graph API
- ✓ Programming SPFx Webparts using MSGraphClient
- Granting Microsoft Graph API Permissions

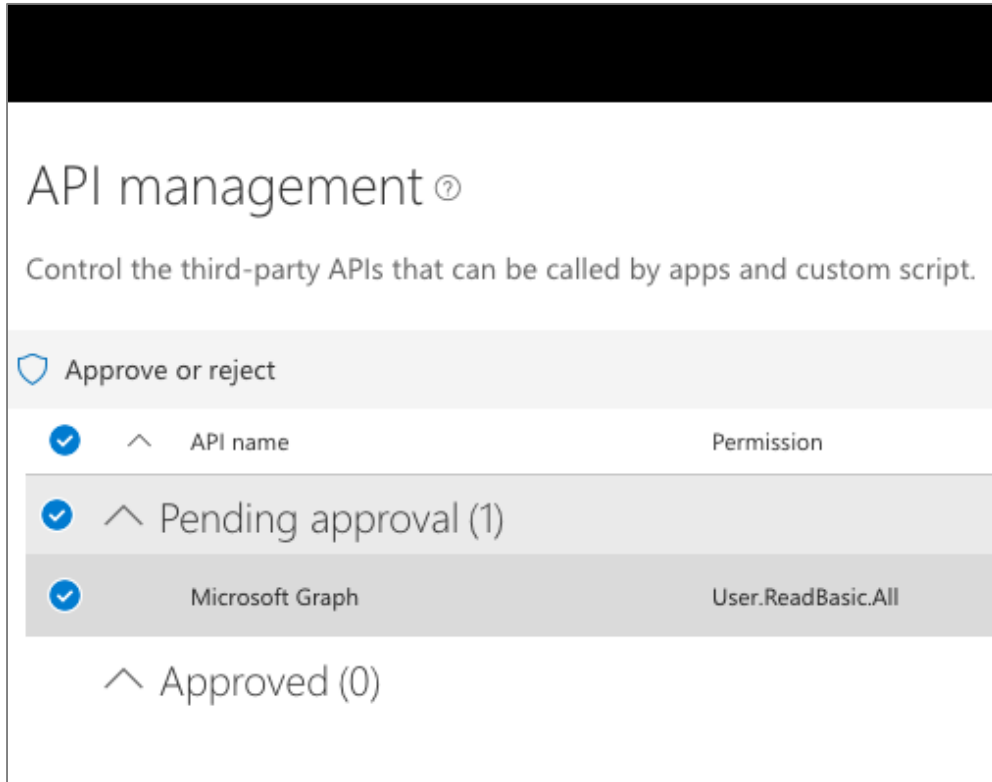


# Add Package to SharePoint App Catalog

- 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







The screenshot shows the 'API management' page in SharePoint Online. The page has a dark header bar. Below the header, the title 'API management' is followed by a help icon. A subtitle reads 'Control the third-party APIs that can be called by apps and custom script.' A section titled 'Approve or reject' with a shield icon contains a table. The table has two columns: 'API name' and 'Permission'. It shows one pending approval for 'Microsoft Graph' with the permission 'User.ReadBasic.All'. Below the table, there is a link to view 'Approved (0)' items.

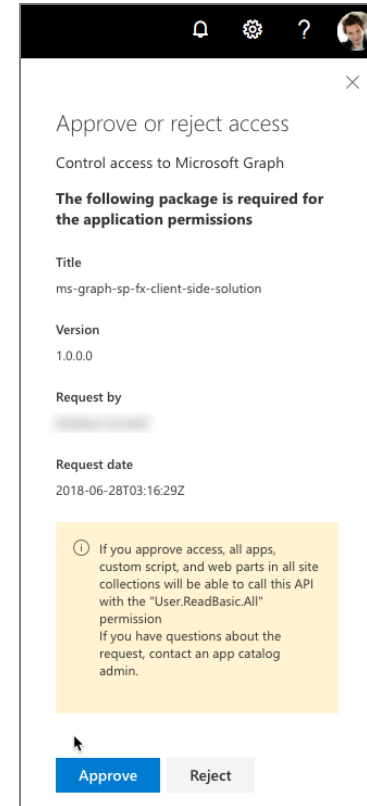
## API management ?

Control the third-party APIs that can be called by apps and custom script.

 Approve or reject

 ^ API name	Permission
 ^ Pending approval (1)	
 Microsoft Graph	User.ReadBasic.All

 ^ Approved (0)



The screenshot shows a dialog box titled 'Approve or reject access'. It contains information about a request to control access to Microsoft Graph. The request is for the package 'ms-graph-sp-fx-client-side-solution' with version '1.0.0.0'. The request was made by a user (represented by a blurred name) on 2018-06-28T03:16:29Z. A warning message states that approving access will allow all apps, custom script, and web parts to call the API with the 'User.ReadBasic.All' permission. At the bottom, there are 'Approve' and 'Reject' buttons.


Approve or reject access

Control access to Microsoft Graph


**The following package is required for the application permissions**


Title  
ms-graph-sp-fx-client-side-solution

Version  
1.0.0.0

Request by  


Request date  
2018-06-28T03:16:29Z

 If you approve access, all apps, custom script, and web parts in all site collections will be able to call this API with the "User.ReadBasic.All" permission. If you have questions about the request, contact an app catalog admin.





# Summary

- ✓ Overview of Microsoft Graph API
- ✓ Constructing URLs for the Microsoft Graph API
- ✓ Developing Applications with the Microsoft Graph API
- ✓ Programming SPFx Webparts using MSGraphClient
- ✓ Granting Microsoft Graph API Permissions

