Microsoft identity platform

Developer community call

Deep dive on using MSAL Node to integrate Electron desktop applications with Azure AD

October 20, 2022 | 9:00AM PST

Emily Lauber

Product Manager

Dogan Erisen

Software Engineer

Salman Salem

Software Engineer

Introduction

- First things first
  - Please note: **We are recording this call** so those unable to attend can benefit from the recording.
  - This call is designed for developers who implement or are interested in implementing Microsoft identity platform solutions.
- What kind of topics will we discuss?
  - We will address development related topics submitted to us by the community for discussion.
  - We build a pipeline of topics for the next few weeks, please submit your feedback and topic suggestions - https://aka.ms/IDDevCommunityCallSurvey
  - View recordings on the Microsoft 365 Developer YouTube channel - **https://aka.ms/M365PnP/videos**
  - Follow us on Twitter **@Microsoft365Dev** and **@azuread**
  - This is NOT a support channel. Please use Stack Overflow to ask your immediate support related questions.
- When is the next session?
  - Community Calls: Monthly – 3rd Thursday of every month

Microsoft identity platform Community call

20th of October 2022

Microsoft identity platform

Developer community call

Deep dive on using MSAL Node to integrate Electron desktop applications with Azure AD

October 20, 2022 | 9:00AM PST

Emily Lauber
Product Manager

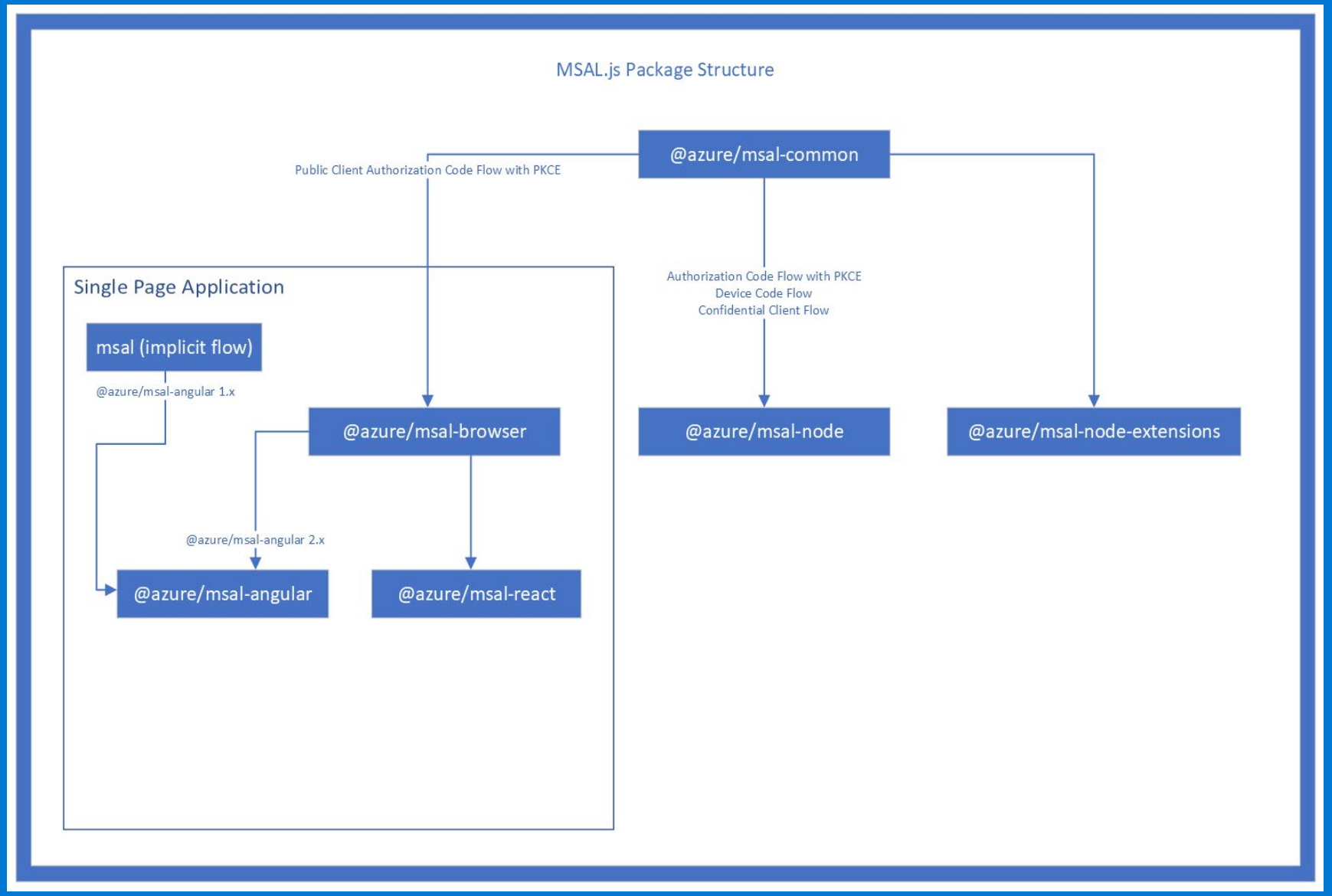
Dogan Erisen
Software Engineer

Salman Salem
Software Engineer

Agenda

- Overview of MSAL Node
- OAuth 2.0 native app best practices
- Enabling single sign-on (SSO)
- Acquiring a token for Microsoft Graph
- Calling Microsoft Graph
- Q & A

Microsoft Authentication Library for JavaScript



- Client-side apps (public client)
- Server-side apps (confidential client)

- Authorization Code Grant with PKCE
- Client Credentials Grant
- Device Code Grant
- On-behalf-of Grant
- ROPC Grant

- OpenID-compliant

MSAL Node

- Supports public and confidential client applications
- Follows support schedule of Node.js project for even-numbered stable releases

Use MSAL Node in your applications

- Step 1: Download a code sample or two (aka.ms/aadcodesamples)
- Step 2: Register your app on Azure AD (portal.azure.com)
- Step 3: Start coding!

github.com/Azure-Samples/ms-identity-javascript-nodejs-desktop

Prerequisites

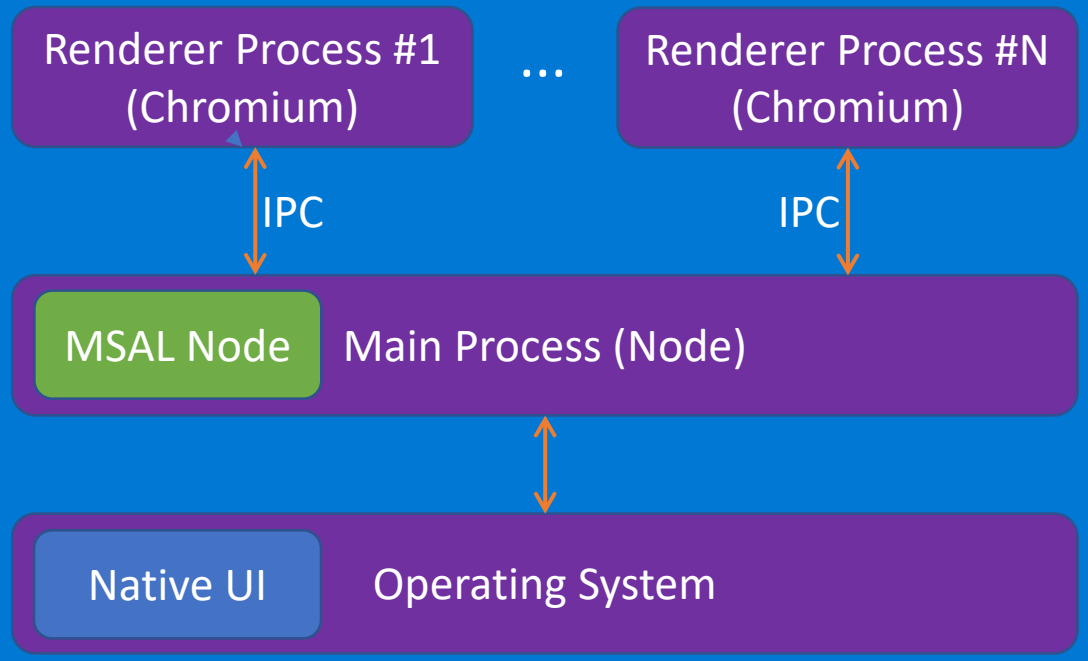
- Node.js v14+
- An Azure AD tenant

github.com/Azure-Samples/ms-identity-javascript-nodejs-desktop

OAuth 2.0, Electron and MSAL

Electron.js

- Build cross-platform native desktop applications
- Main process vs Renderer process(es)
- Inter-process communication (IPC)



OAuth 2.0 Desktop App Best practices (RFC 8252)

- MUST implement Proof Key for Code Exchange (PKCE)
- RECOMMENDED to use external user-agent for auth requests
- Redirect URI options:
  - Private URI Scheme (e.g. com.example.app:/oauth2redirect/example-provider)
  - Loopback Redirect (e.g. http://127.0.0.1 or localhost)

Enabling single sign-on

Single sign-on between multiple apps

- If the user has an active session with Azure AD and tries to sign-in to an app using MSAL, they may automatically sign-in without entering their credentials again.

- If the user has multiple active sessions with Azure AD, then the user is prompted to pick an account to sign in with
  - To bypass account selection screen, use a **loginHint**:
    - Session Id (sid)
    - login_hint (e.g. upn or preferred_username)
    - MSAL account object

AuthZ via External User-Agent

https://www.rfc-editor.org/rfc/rfc8252

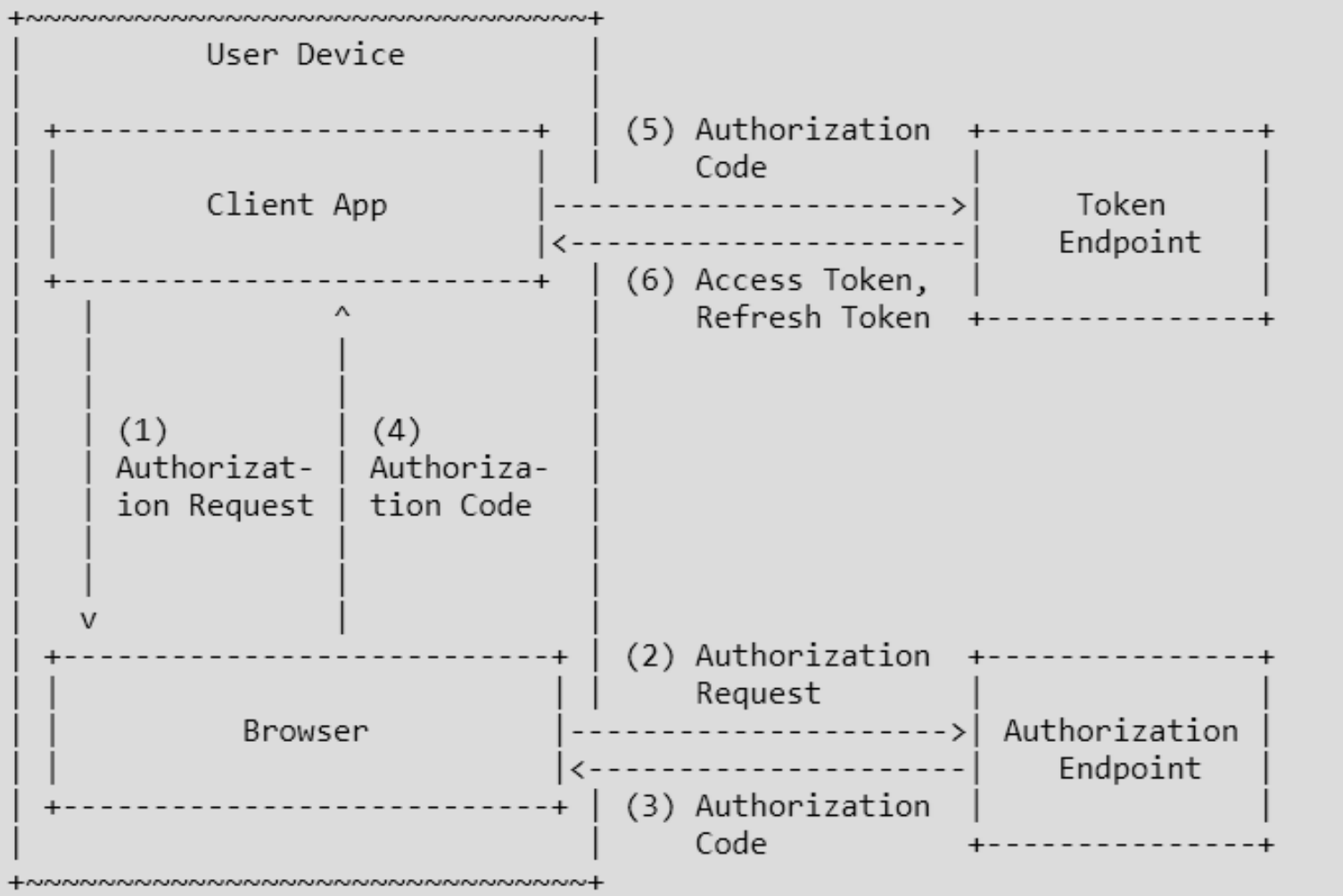


Figure 1: Native App Authorization via an External User-Agent

(1)Client app opens browser tab with authorization request.

(2)Authorization endpoint receives the authorization request, authenticates the user, and obtains authorization.

(3)Authorization server issues an authorization code to the redirect URI.

(4)Client receives the authorization code from the redirect URI.

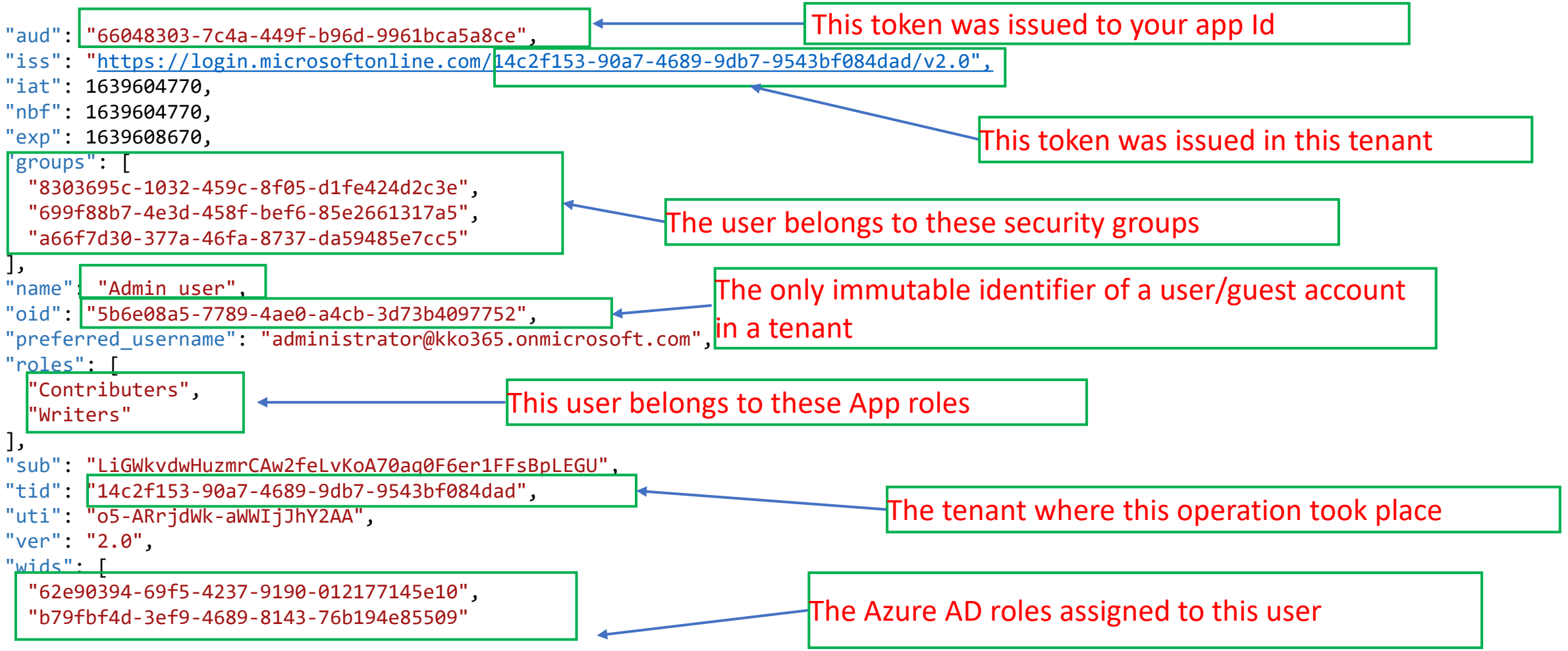
(5)Client app presents the authorization code at the token endpoint.

(6)Token endpoint validates the authorization code and issues the tokens requested.

Questions?

ID Token

Learn and use claims provided in tokens



aka.ms/id-tokens

Questions?

Get an access token for Microsoft Graph

Microsoft Graph SDK

- Why use Graph SDK?
  - Simplify building high-quality, efficient, and resilient applications that access Microsoft Graph.
    - support for retry handling
    - payload compression
    - paging through collections
    - creating batch requests

- Ways of using Graph SDK

Create With ClientOptions	Create With Options

`let clientOptions = {`
`  authProvider: new AuthProviderClass()`
`};`
 	
`const client =`
`Client.initWithMiddleware(clientOptions);`
	`Let authProvider = (callback) => {`
	`  callback(error, accessToken);`
	`};`
	
	`let options = { authProvider };`
	`const client = Client.init(options);`

Calling Microsoft Graph

1. Instantiate the Graph SDK Client
   - Obtain an access token with MSAL and pass it to the SDK
2. Call the */me* endpoint to access user's profile
   - Required permissions: *User.Read*

Q & A

Learn More

- MSAL.js on GitHub
- MSAL Node docs
- MSAL Node samples
- Microsoft identity platform:
  - OAuth 2.0 Authorization Code Grant
  - Continuous Access Evaluation
  - Code samples
- Graph SDK for JavaScript on GitHub

Zero Trust Guide

- Zero Trust Developer Guide: aka.ms/ztdev
- Zero Trust Code Sample: aka.ms/identity-zerotrust-sample
- Extended Token Validation: aka.ms/extendtokenvalidation
- Use App Roles in your app: aka.ms/approles
- Use Groups in your app: aka.ms/groupssample
- Zero Trust Blog Series: aka.ms/ZTRappsblog-part1

Thank you

Recording will be available soon on the
Microsoft 365 Community (PnP) YouTube channel
https://aka.ms/M365PnP/videos

(subscribe today)

Follow us on Twitter
@Microsoft365Dev and @azuread

Survey - https://aka.ms/IDDevCommunityCallSurvey

Next call: **November 17th at 9:00am PST**

https://aka.ms/IDDevCommunityCalendar