Microsoft identity platform

Developer community call

Deep dive on using MSAL.js to integrate Angular single-page applications with Azure AD

August 18, 2022 | 9:00AM PST

Doğan Erişen

Software Engineer

Salman Salem

Software Engineer

Kalyan Krishna

Product Manager

@kalyankrishna1

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

AGENDA

Microsoft Identity dev community call 18th of August

Download recurrent invite from
https://aka.ms/IDDevCommunityCalendar

Deep dive on using MSAL.js to integrate Angular single-page applications with Azure Active Directory

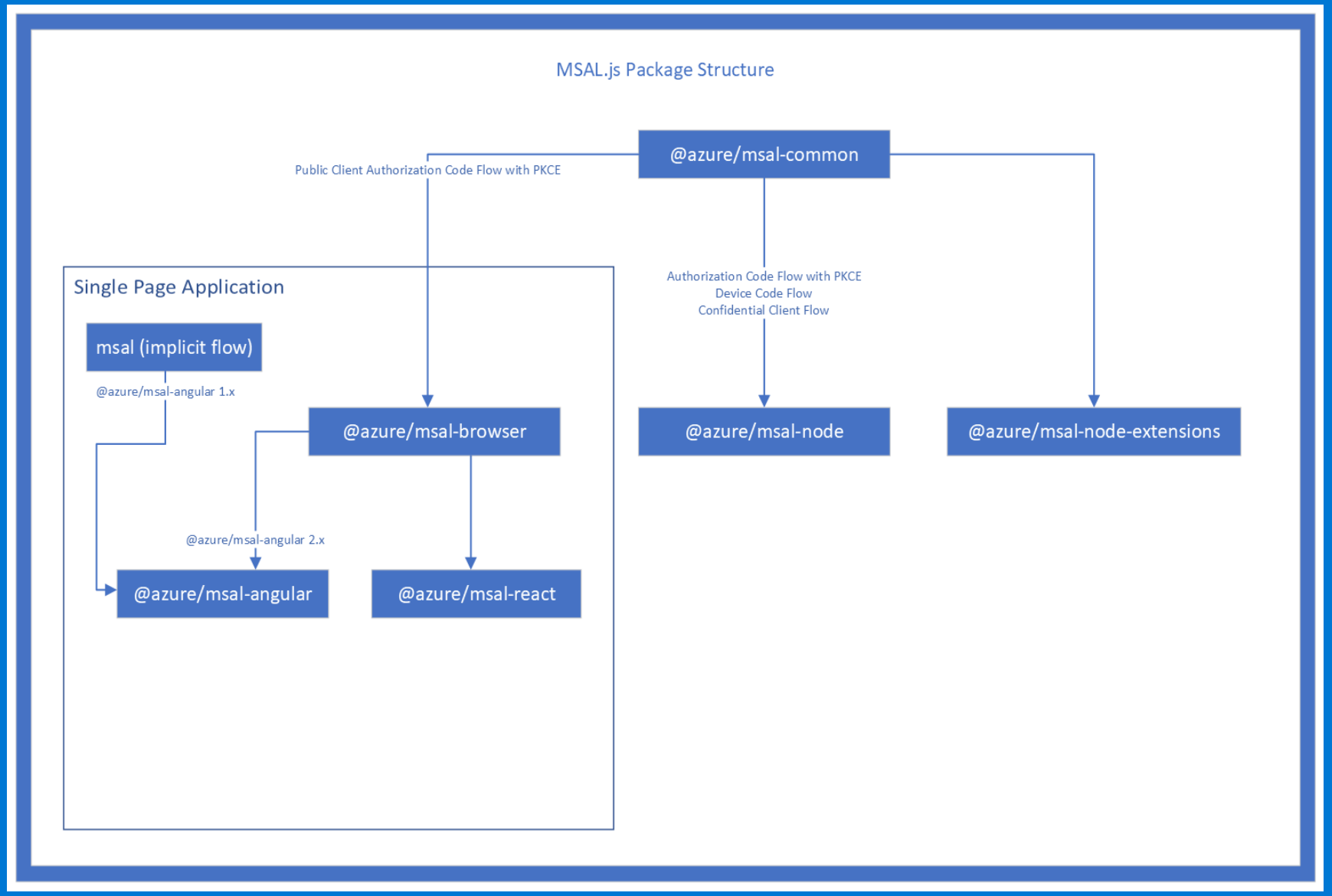
– Doğan Erişen (Microsoft)
– Salman Salem (Microsoft)
– Kalyan Krishna (Microsoft)

Microsoft

Agenda

- Overview of MSAL.js and MSAL Angular v2
- Using MSAL Angular v2 in an Angular SPA
- Enabling Single Sign-on
- Acquiring a token for Microsoft Graph
- Continuous Access Evaluation (CAE) Events
- Deployment to Azure Static Web Apps
- Next steps and references

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

- OpenID-compliant

MSAL Angular v2

- OAuth 2.0 Authorization Code Grant with Proof Key for Code Exchange (PKCE)
- Supports Angular v9 and later
- Limited support for server-side rendering (Angular Universal SSR)
- Same public API with msal-browser
- Adds components, services and observables on top

Use MSAL Angular in your applications

Previous session on React

Using MSAL.js to integrate React Single-page applications with Azure Active Directory – July 2022

Use MSAL Angular in your applications

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

Today's demo
github.com/derisen/msal-angular-demo

Full tutorial
https://aka.ms/msalangulartutorial

Prerequisites

- Node.js v14+
- An Azure Subscription
- A GitHub account

Today's demo
github.com/derisen/msal-angular-demo

Full tutorial
https://aka.ms/msalangulartutorial

Configuring MSAL

Option	Description	Format	Default Value
clientId	App ID of your application. Can be found in your portal registration.	UUID/GUID	None. This parameter is required in order for MSAL to perform any actions.
authority	URI of the tenant to authenticate and authorize with. Usually takes the form of https://{uri}/{tenantid}	String in URI format with tenant - https://{uri}/{tenantid}	https://login.microsoftonline.com/common
redirectUri	URI where the authorization code response is sent back to. Whatever location is specified here must have the MSAL library available to handle the response.	String in absolute or relative URI format	Login request page (window.location.href of page which made auth request)
postLogoutRedirectUri	URI that is redirected to after a logout() call is made.	String in absolute or relative URI format. Pass null to disable post logout redirect.	Login request page (window.location.href of page which made auth request)
navigateToLoginRequestUrl	If true, will navigate back to the original request location before processing the authorization code response. If the redirectUri is the same as the original request location, this flag should be set to false.	boolean	true
clientCapabilities	Array of capabilities to be added to all network requests as part of the xms_cc claims request	Array of strings	[]

MSAL Events

Event Type	Description	Interaction Type	Payload	Error
LOGIN_SUCCESS	Successfully logged in	Popup or Redirect	AuthenticationResult	
LOGIN_FAILURE	Error when logging in	Popup or Redirect		AuthError or Error
ACQUIRE_TOKEN_SUCCESS	Successfully acquired token from cache or network	Popup or Redirect or Silent	AuthenticationResult	
ACQUIRE_TOKEN_FAILURE	Error when acquiring token	Popup or Redirect or Silent		AuthError or Error
LOGOUT_SUCCESS	Logout success	Redirect or Popup	EndSessionRequest or EndSessionPopupRequest	
LOGOUT_FAILURE	Logout failed	Redirect or Popup		AuthError or Error
ACCOUNT_ADDED	Account logged-in in a different tab or window	N/A	AccountInfo	N/A
ACCOUNT_REMOVED	Account logged-out in a different tab or window	N/A	AccountInfo	N/A

Enabling Single sign-on

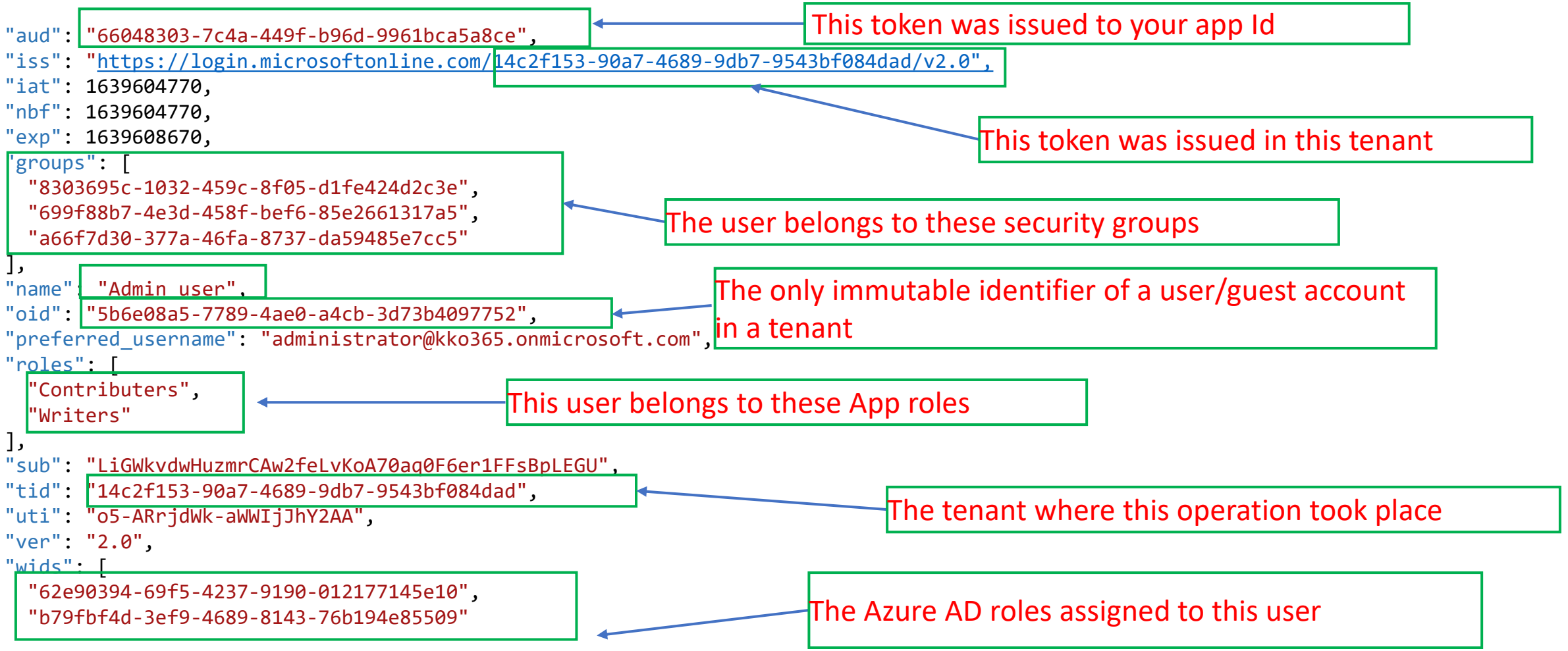
Single sign-on between multiple apps

- SSO between different apps
  - If the user has an active session with Azure AD and tries to sign-in to an app using MSAL, they will 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
  - **ssoSilent** method allows sign-in the user and obtain tokens without an interaction
  - If third-party cookies are blocked (e.g. Safari), **ssoSilent** will fail with:
    - **InteractionRequiredAuthError**: *login_required: AADSTS50058: A silent sign-in request was sent but no user is signed in. The cookies used to represent the user's session were not sent in the request to Azure AD*

Questions?

ID Token

Learn about and use claims provided in tokens



aka.ms/id-tokens

Questions?

Caching

- MSAL Browser (and MSAL Angular) caches authentication artifacts in either:
  - Session storage (default)
  - Local storage
  - Memory storage
- **local storage** and **session storage** are considered secure if applications do not have XSS (or similar) vulnerabilities (OWASP XSS Prevention Cheat Sheet)
  - Furthermore:
    - Short-lived tokens (1 hr for access tokens, 24 hr for refresh and id tokens)
    - Proof of Possession (currently only for access tokens, work-in-progress for refresh tokens)
    - Continuous Access Evaluation (CAE) events

Authentication artifacts MSAL obtains from AAD

- To faciliate a authN and authZ while maintaining a good UX, MSAL caches various artifacts:
  - Persistent auth artifacts
    - access tokens
    - id tokens
    - refresh tokens
    - accounts
  - Temporary auth artifacts
    - request metadata e.g. state, nonce, authority,
    - errors
    - interaction status
  - Telemetry
    - previous failed request
    - performance data

Logging

- MSAL provides the ability to log authN/authZ activity at various levels of detail
  - **Error**
  - **Warning**
  - **Info**
  - **Verbose**
  - **Trace**
- You can choose to log **Personally Identifiable Information** (PII) or not
- MSAL also provides the ability to collect telemetry for your application

Questions?

MSAL Angular Components

MSAL Service	MSAL Broadcast Service	MSAL Guard	MSAL Interceptor
A class that provides access to MSAL application instance and the APIs/methods it epxoses.	A class that emits MSAL events, allowing you to take action when certain events occur. MsalBroadcastService exposes 2 observables using RxJs: msalSubject$ for accessing events, and inProgress$ for accessing current interaction status.	A class you can use to protect routes and require authentication before accessing the protected route. MSAL Guard should not be relied upon for security. Attackers can potentially get around client-side guards, and you should ensure that the server does not return any data the user should not access.	A class that automatically acquires tokens for outgoing requests that use the Angular http client to known protected resources. MSAL Interceptor is designed to acquire tokens silently. In the event that a silent request fails, it will fall back to acquiring tokens interactively.

Get an access token for Microsoft Graph

Today's demo
github.com/derisen/msal-angular-demo

Full tutorial
https://aka.ms/msalangulartutorial

Questions?

Continuous Access Evaluation (CAE)

Continuous Access Evaluation (CAE)

CAE introduces real time enforcement of account lifecycle events and policies:

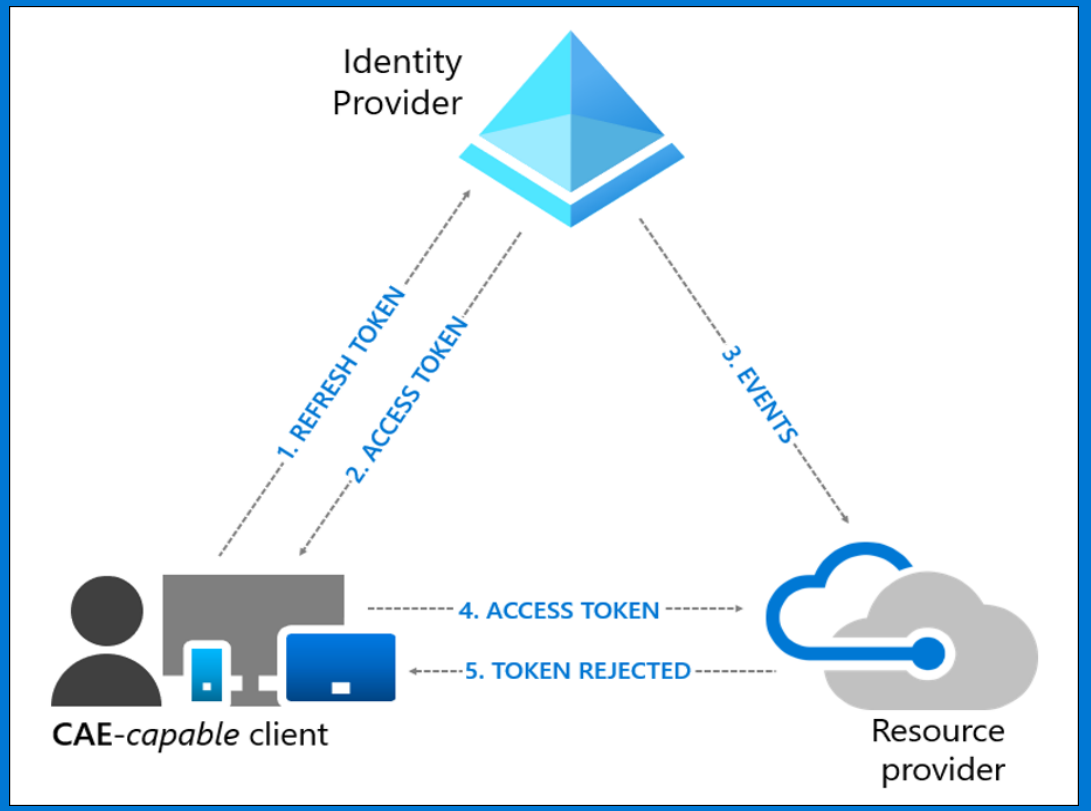
- Account revocation
- Account disablement/deletion
- Password change
- User Location change
- User risk increase

Tenant admins are being encouraged to enable CAE in their tenants
Developers need to explicitly add support for CAE in their apps.

Learn about claims challenge and client capabilities - **aka.ms/ClaimsChallenge**

Handling CAE Challenges

Continuous Access Evaluation (CAE) is an Azure AD feature that allows access tokens to be revoked based on critical events and policy evaluation rather than relying on token expiry based on lifetime.



HTTP 401;

Unauthorized WWW-Authenticate=Bearer authorization_uri="https://login.windows.net/common/oauth2/authorize", error="insufficient_claims", claims="eyJhY2Nlc3NfdG9rZW4iOnsibmJmIjp7ImVzc2VudGlhbCI6dHJ1ZSwgInZhbHVlIjoiMTYwNDEwNjY1MSJ9fX0=”

Steps to handle CAE challenges

- Step 1: Configure your MSAL app to declare that it's capable of handling claim challenges
- Step 2: Grab the **www-authenticate** header from the 401 response
- Step 3: Process the header
- Step 4: Pass the claims challenge to MSAL as a request parameter

- CAE Code example - >

  https://github.com/Azure-Samples/ms-identity-javascript-angular-tutorial/blob/main/2-Authorization-I/1-call-graph

Questions?

Learn More

- MSAL.js on GitHub
- MSAL Browser docs
- MSAL Angular docs
- MSAL Angular samples:
  - Testing samples
  - Chapter wise tutorials – aka.ms/msalangulartutorial
- Microsoft identity platform:
  - OAuth 2.0 Authorization Code Grant
  - Single Sign-on with MSAL.js
  - Continuous Access Evaluation
  - Code samples – aka.ms/aadcodesamples
- OWASP XSS Prevention Cheat Sheet

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: **September 15th at 9:00am PST**
https://aka.ms/IDDevCommunityCalendar