Create user access token & authorization header

This article explains how to authenticate to Microsoft Graph in **delegated user context**. Use this approach when an action must run on behalf of a signed-in user.

> Important Avoid Resource Owner Password Credentials (ROPC) whenever possible. Microsoft does not recommend it for modern environments because it is incompatible with many MFA, Conditional Access, passwordless, and passkey-based sign-in scenarios.

When to use delegated authentication

Use delegated authentication when:

- a signed-in user triggers the action
- the API call must respect the user's permissions
- the action should be auditable as that user
- interactive consent or MFA may be required

Recommended options

Preferred approaches for delegated Graph access:

1. **Device code flow** for scripts and terminals
2. **Interactive browser sign-in** for desktop tools
3. **Authorization code flow with PKCE** for web apps and SPAs
4. **MSAL** instead of building raw OAuth requests manually

Example with MSAL.PS

```
Install-Module MSAL.PS -Scope CurrentUser

$TenantId = "<tenant-id>"
$ClientId = "<app-client-id>"
$Scopes   = @("User.Read", "Mail.Read")

Token=Get-MsalToken-TenantIdTenantId -ClientId ClientId-ScopesScopes

$Header = @{
    Authorization = "Bearer (Token.AccessToken)"
    "Content-Type" = "application/json"
}
```

Best practices

- prefer **MSAL** over custom username/password token requests
- request only the scopes you really need
- do not store user passwords in scripts
- expect MFA and Conditional Access challenges in enterprise tenants
- use app-only access only when there is no real user context

Summary

For modern Microsoft Graph delegated access, use **MSAL** plus interactive or device-based sign-in methods. Avoid ROPC unless you have a specific legacy exception and fully understand the risk.

Revision #13
Created 2022-12-12 08:47:49 UTC
Updated 2026-04-15 21:31:39 UTC by Luca Noah Caprez