<https://medium.com/@shoaib.alam/part-3-oauth-2-0-client-credentials-grant-with-azure-ad-36ac0f9ab2b7>

<https://learn.microsoft.com/en-us/entra/identity-platform/reference-app-manifest>

In Azure AD, the Client Credentials and Authorization Code flows are distinct OAuth 2.0 grant types used for application authentication. The Client Credentials flow is suitable for applications running on servers and needing to access their own resources, while the Authorization Code flow is used for applications that require user interaction (e.g., web apps, mobile apps) to access user-specific resources.

Client Credentials Flow:

* **Purpose:**

Used when an application (e.g., a backend service) needs to access its own resources without user involvement.

* **Process:**

The application presents its client ID and secret to the authorization server, and in return, receives an access token.

* **Suitable for:**

Server-side applications, background services, or applications that don't require user interaction to obtain resources.

Authorization Code Flow:

* **Purpose:**

Enables a client application to obtain authorized access to protected resources (like web APIs) on behalf of a user.

* **Process:**

The client redirects the user to the authorization server (Azure AD), where the user logs in and grants consent. The server returns an authorization code, which the client then exchanges for an access token.

* **Suitable for:**

Web applications, mobile apps, or any application that requires a user to sign in and grant consent for accessing their data.

Key Differences:

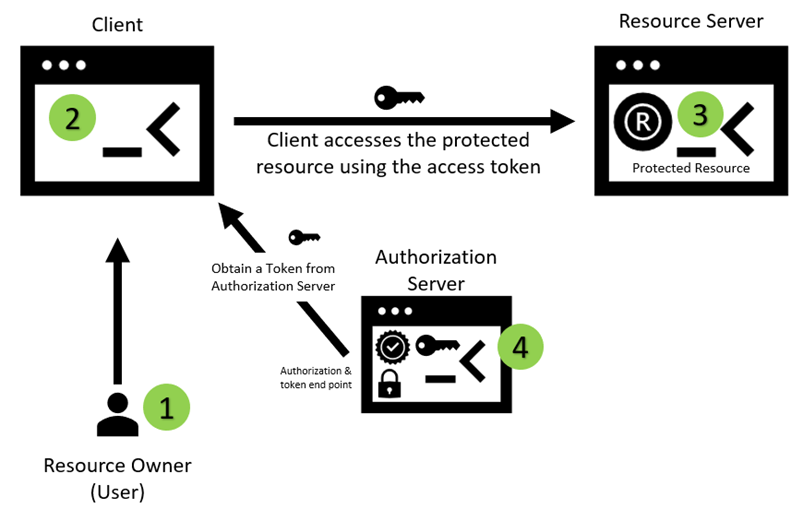
|  |  |  |
| --- | --- | --- |
| **Feature** | **Client Credentials Flow** | **Authorization Code Flow** |
| User Interaction | None | User required to sign in |
| Application Type | Server-side | Web/Mobile/Native |
| Token Exchange | Client ID and Secret | Authorization Code |
| Resource Access | Own Resources | User's Resources |
| Security | Basic (Client secret) | More secure (Authorization code, redirection) |

<https://www.getpostman.com/oauth2/callback.>

<https://medium.com/@shoaib.alam/part-2b-oauth-2-0-authorization-code-grant-with-azure-ad-d97b213ecc9>

<https://medium.com/@shoaib.alam/what-is-oauth-2-0-and-how-it-works-90a3ee752de1>

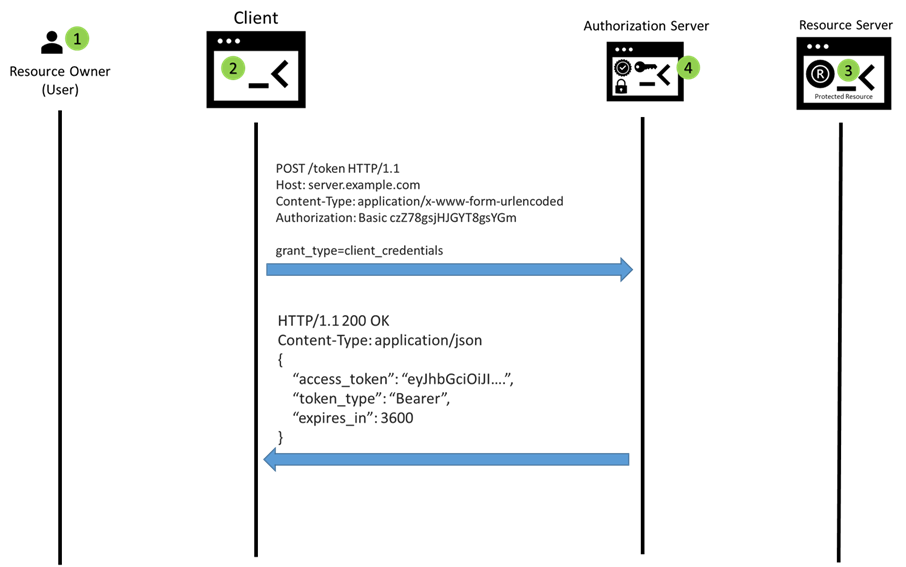
<https://medium.com/@shoaib.alam/part-3-oauth-2-0-client-credentials-grant-with-azure-ad-36ac0f9ab2b7>



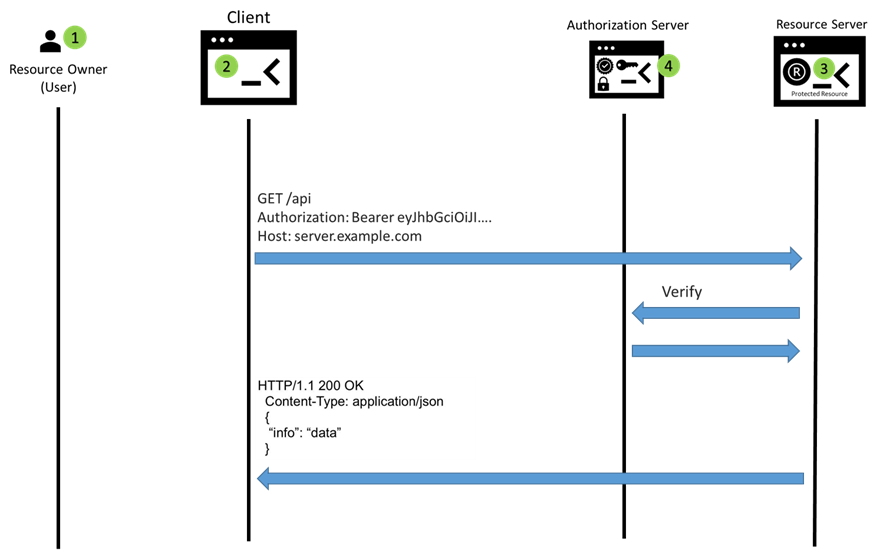
The response from the authorization server is an access token. The client credentials flow does not issue a refresh token because the client is assumed to be in the position of being able to request a new token for itself at any time without involving a separate resource owner and it makes refresh token unnecessary in this context.

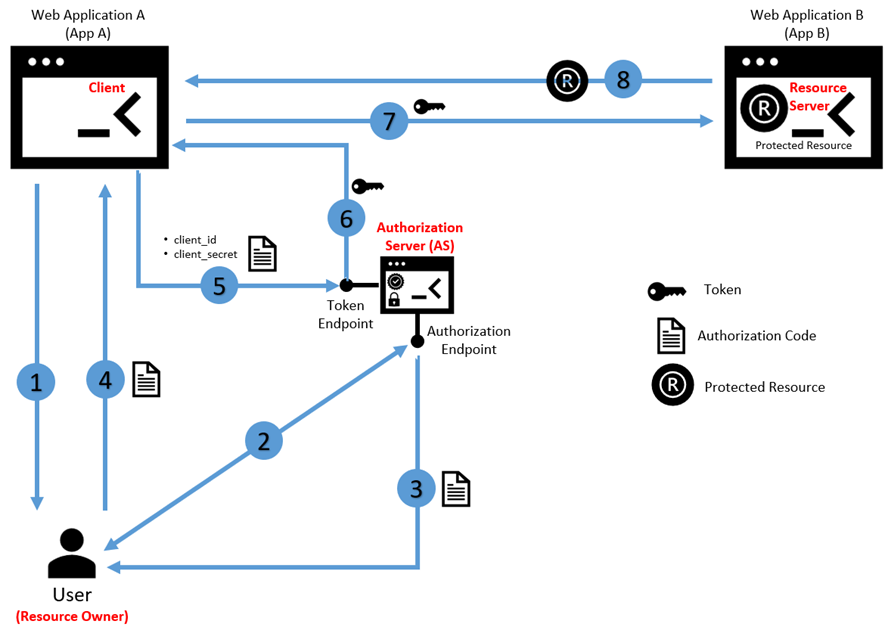
**Sequence Diagram**

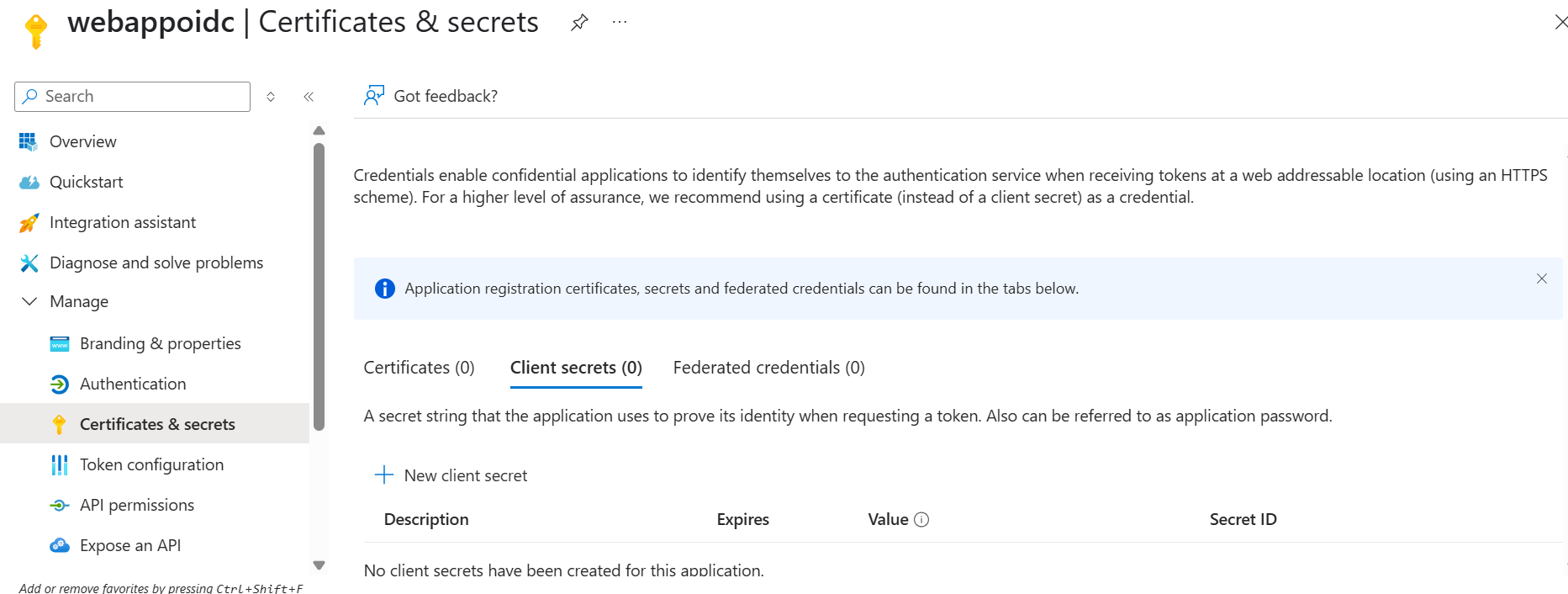
**Token End Point**

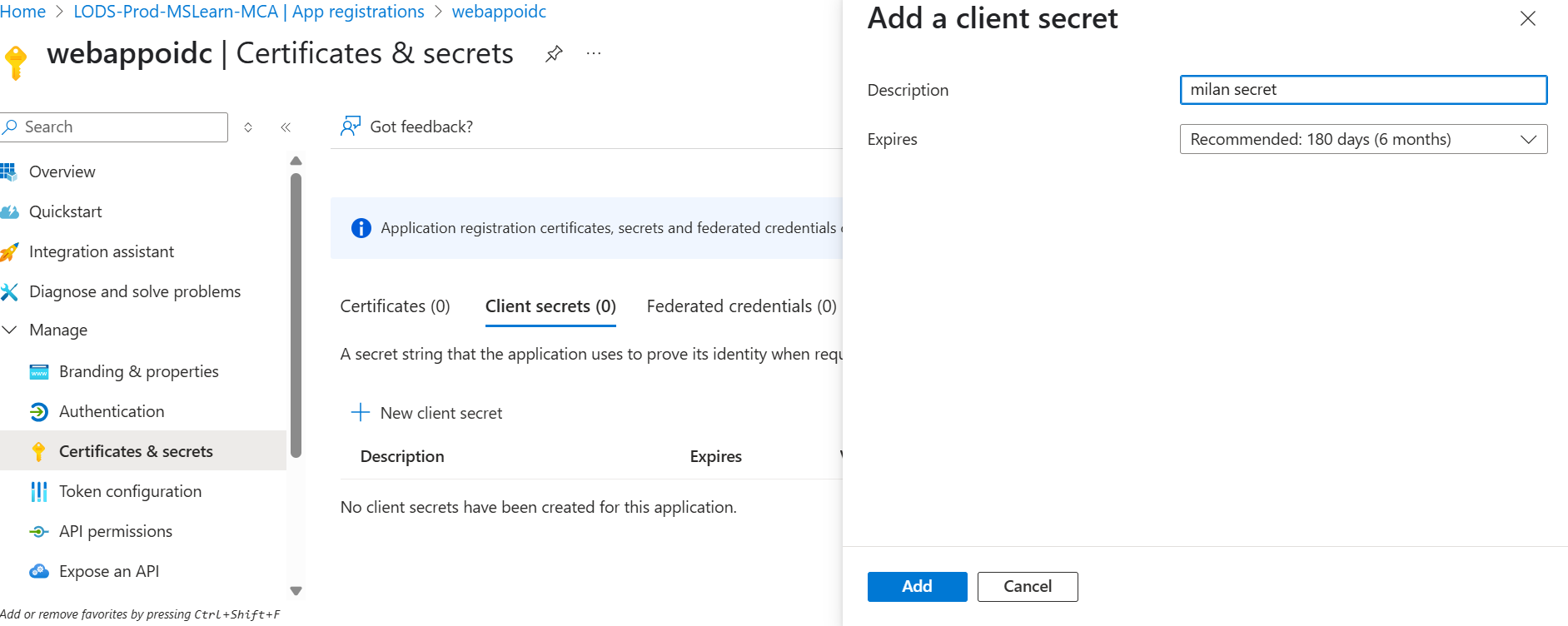


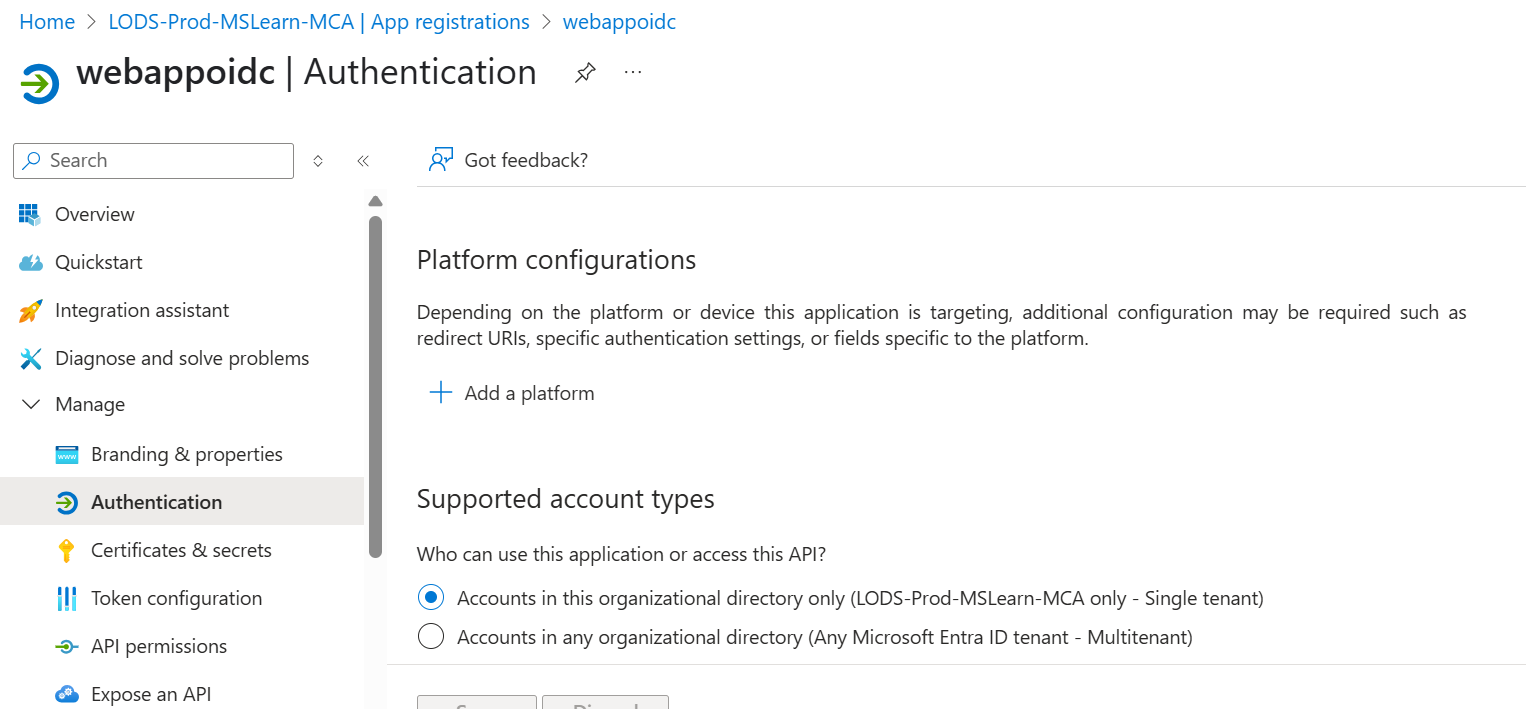
**Resource Access**

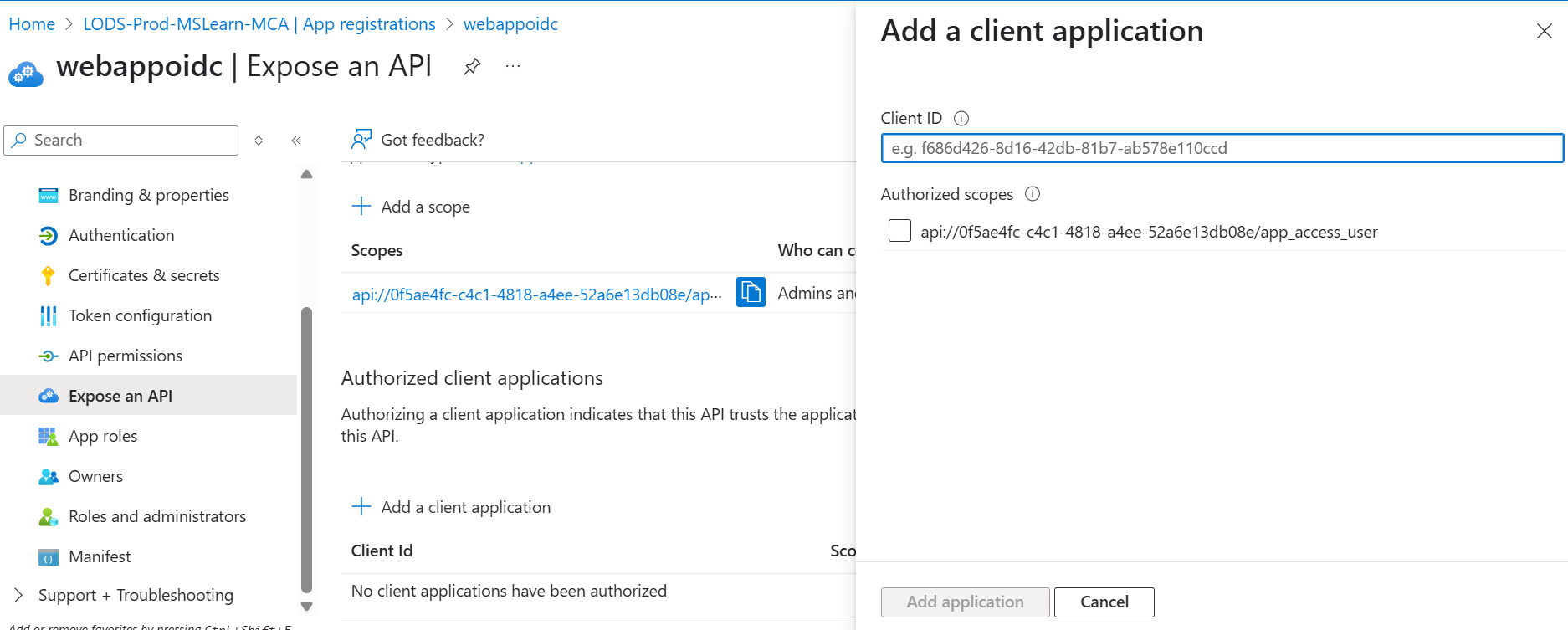


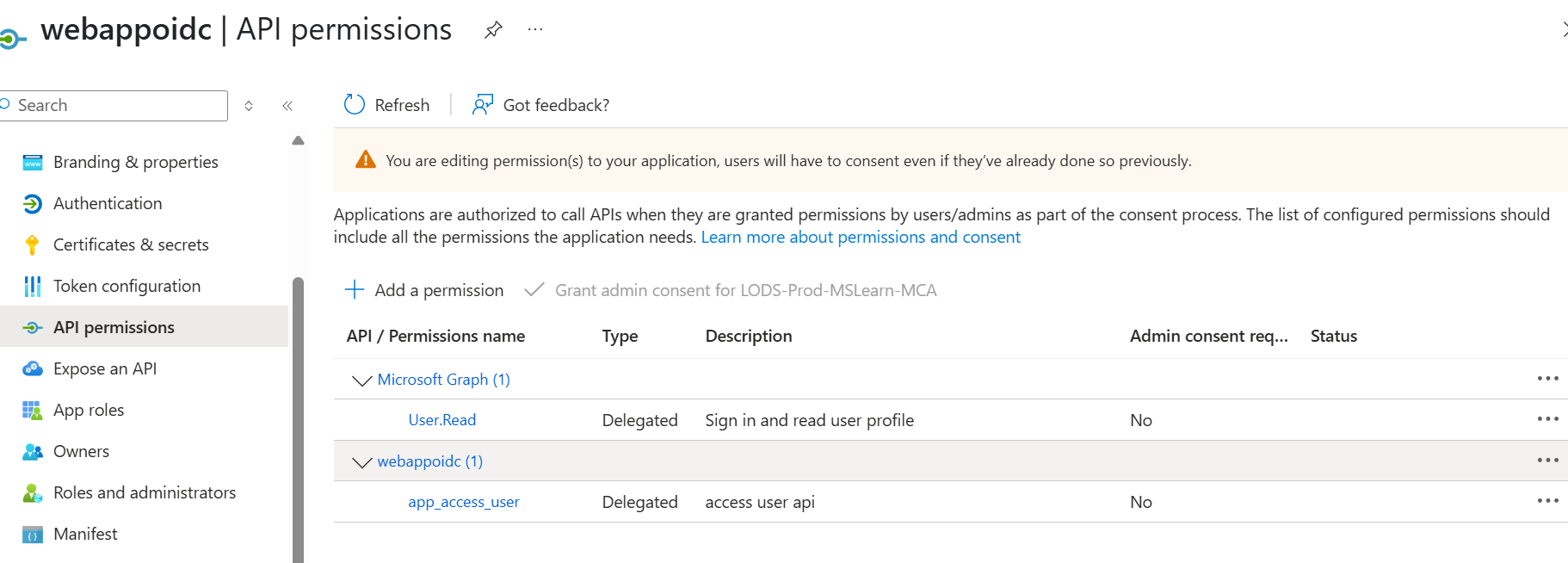


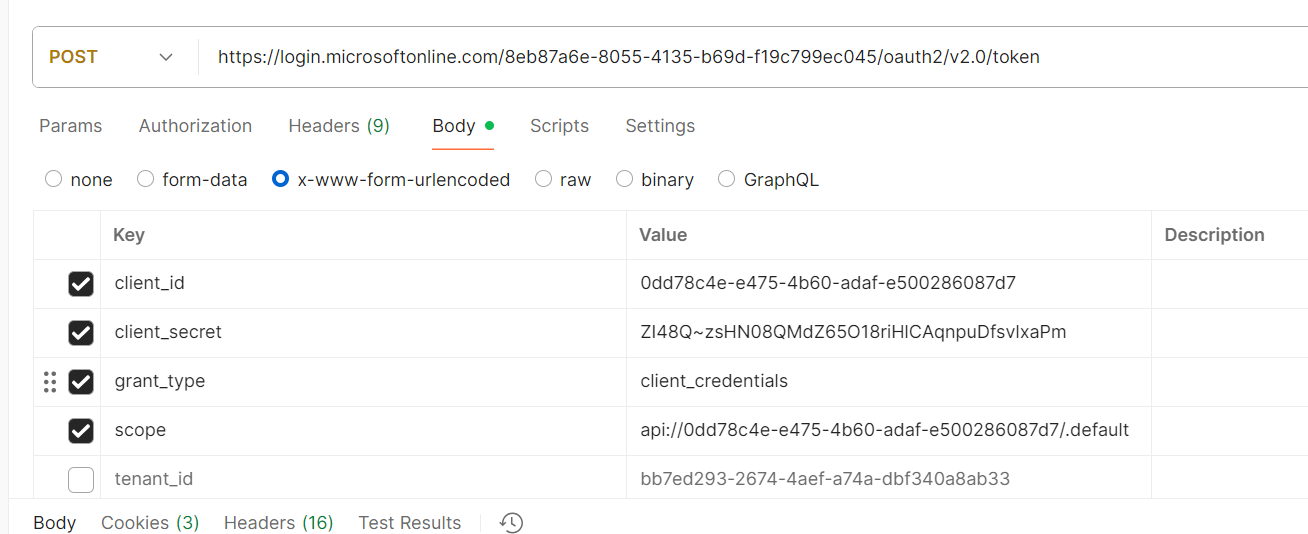








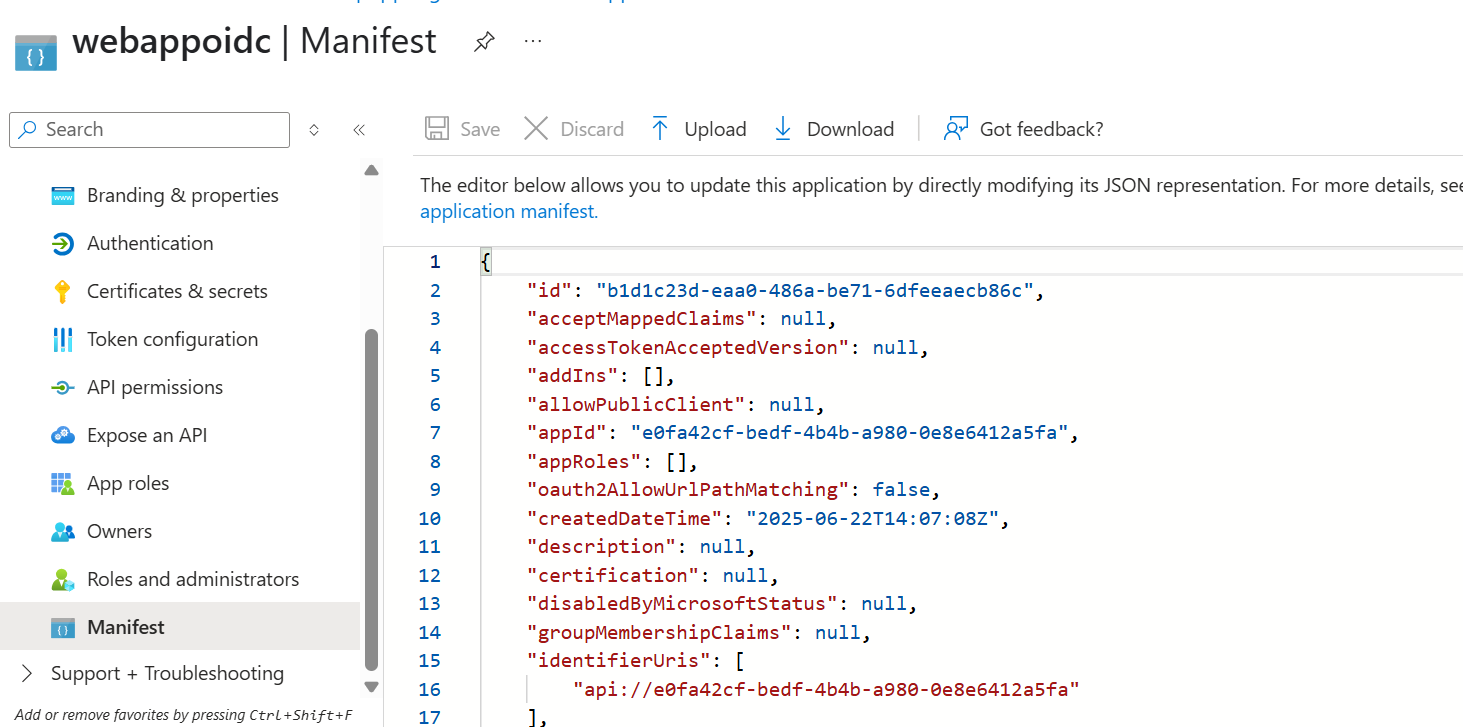




**3. Check Your App Registration**

Ensure your app is registered to support v2.0:

* Go to **Azure Portal > App registrations > Your App**
* Under **Authentication**, ensure ID tokens and Access tokens are enabled
* Check the **manifest**, make sure "accessTokenAcceptedVersion": 2



Version 2 payload

{

"aud": "e0fa42cf-bedf-4b4b-a980-0e8e6412a5fa",

"iss": "https://login.microsoftonline.com/bb7ed293-2674-4aef-a74a-dbf340a8ab33/v2.0",

"iat": 1750602147,

"nbf": 1750602147,

"exp": 1750606047,

"aio": "k2RgYAg/5yrnwxXMZbwg4PQXruc3zl66Y6Crq72e75SedhFjwEcA",

"azp": "e0fa42cf-bedf-4b4b-a980-0e8e6412a5fa",

"azpacr": "1",

"oid": "fa7e3444-4326-4475-bd46-3409b9f9841c",

"rh": "1.AVAAk9J-u3Qm70qnStvzQKirM89C-uDfvktLqYAOjmQSpfpQAABQAA.",

"sub": "fa7e3444-4326-4475-bd46-3409b9f9841c",

"tid": "bb7ed293-2674-4aef-a74a-dbf340a8ab33",

"uti": "KS-6CAlaF0OPqqKdi\_1kAA",

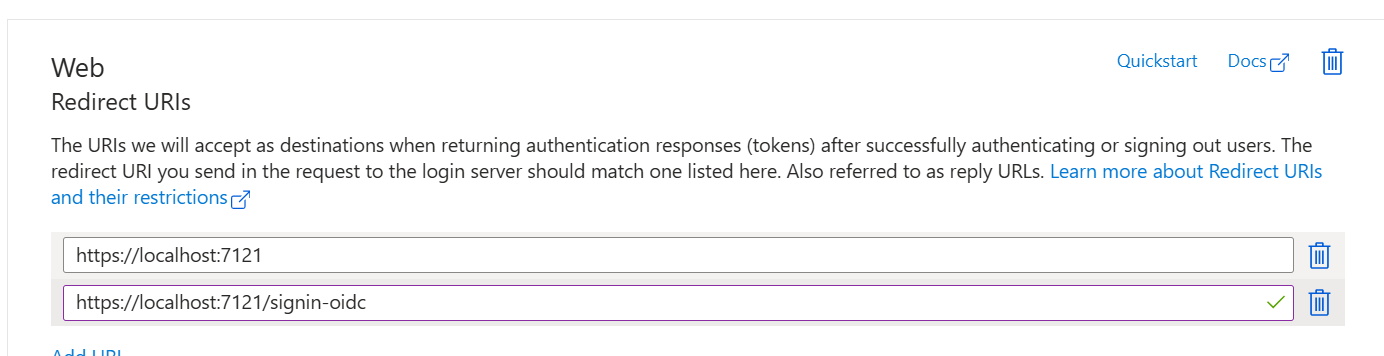
"ver": "2.0",

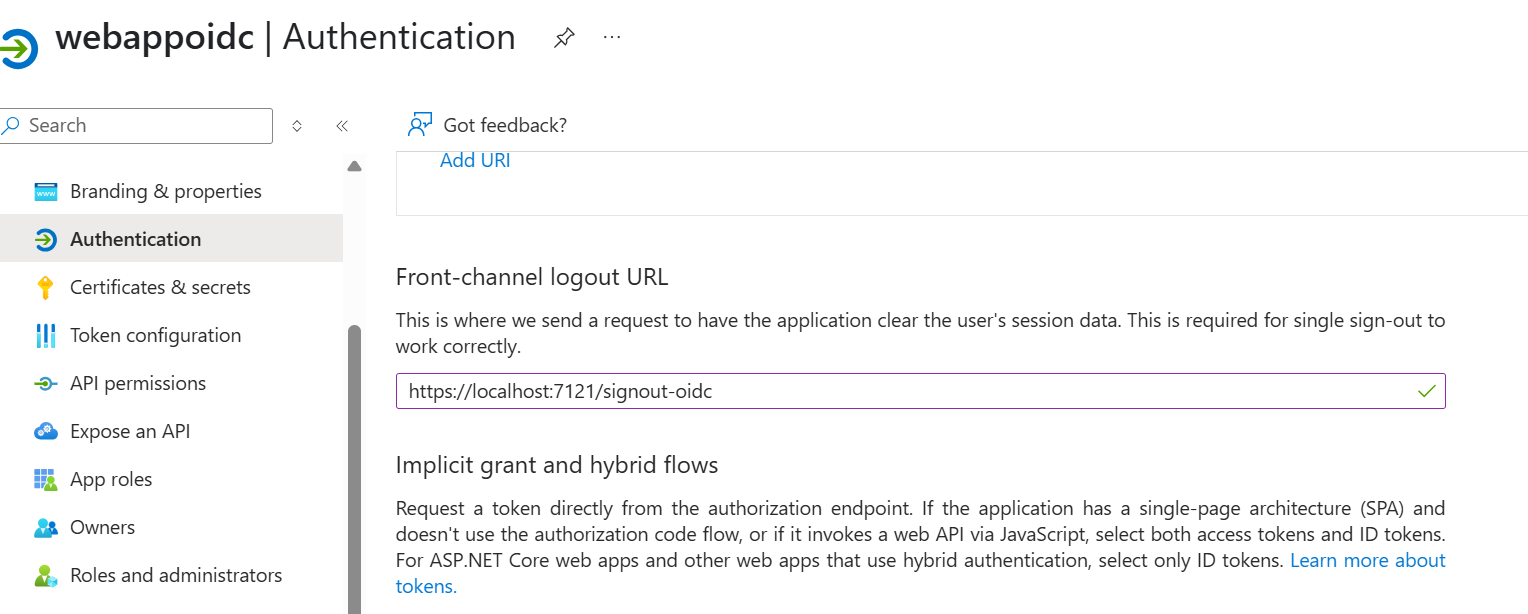
"xms\_ftd": "KlsJ2vh7AeyHYRrRh1Zlvyx5Ze30KFMpKJyptZlaRscBdXNlYXN0LWRzbXM"

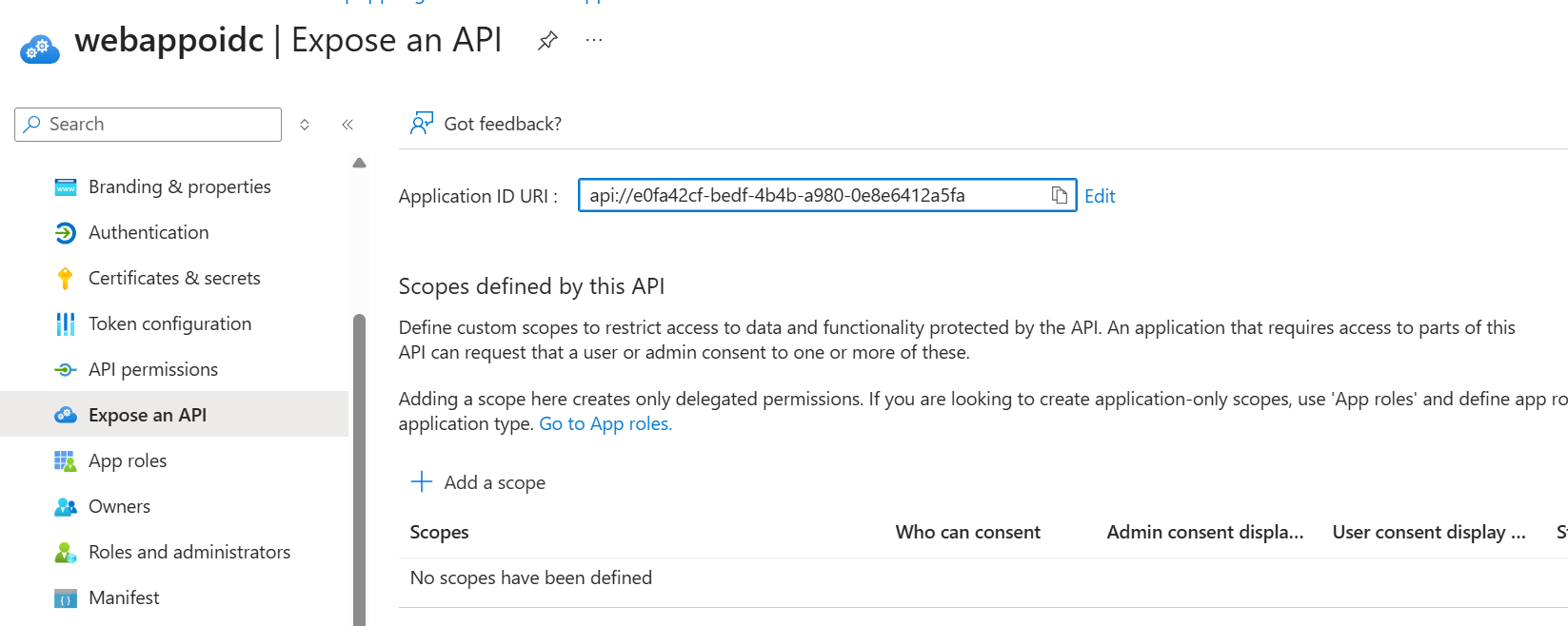
}

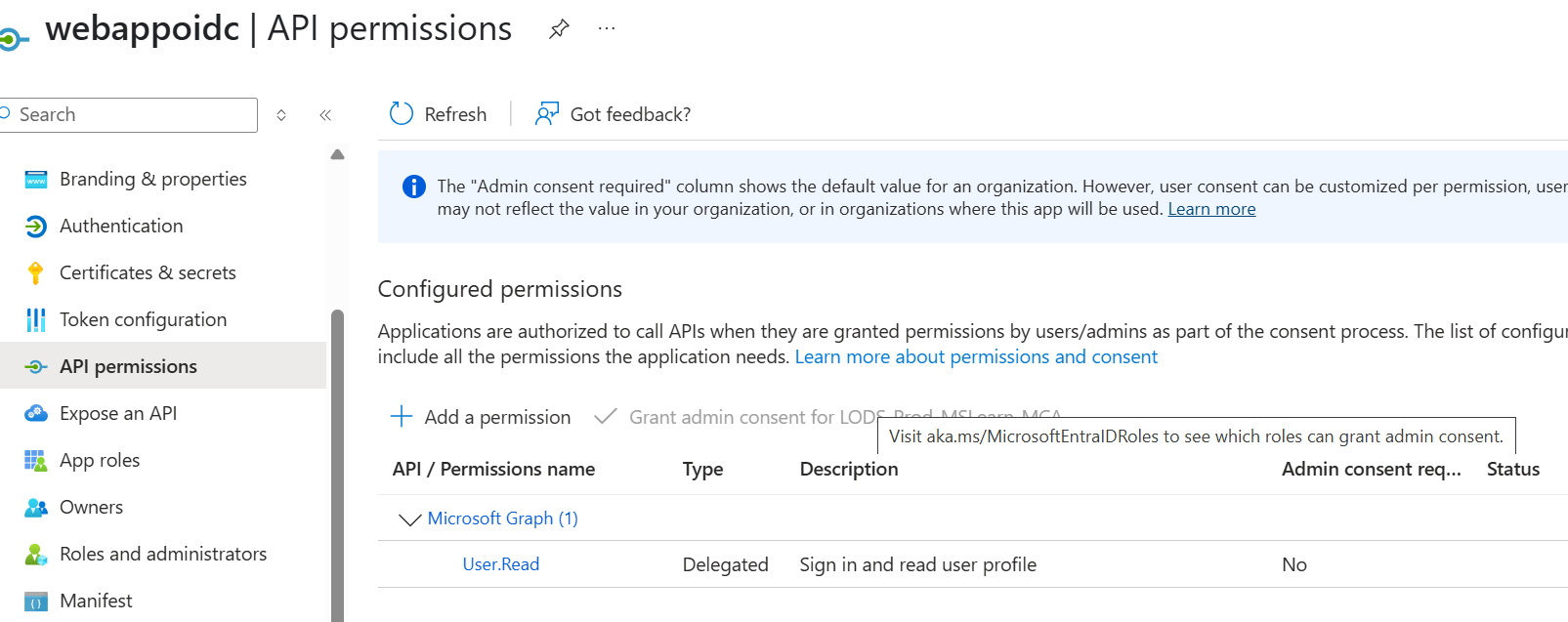
Version 1 payload("accessTokenAcceptedVersion": 1/null)

{ "typ": "JWT", "alg": "RS256", "x5t": "CNv0OI3RwqlHFEVnaoMAshCH2XE", "kid": "CNv0OI3RwqlHFEVnaoMAshCH2XE" }.{ "aud": "api://0dd78c4e-e475-4b60-adaf-e500286087d7", "iss": "https://sts.windows.net/8eb87a6e-8055-4135-b69d-f19c799ec045/", "iat": 1750586350, "nbf": 1750586350, "exp": 1750590250, "aio": "k2RgYPDdFNRptyu6aWWo7Unb89/0AA==", "appid": "0dd78c4e-e475-4b60-adaf-e500286087d7", "appidacr": "1", "idp": "https://sts.windows.net/8eb87a6e-8055-4135-b69d-f19c799ec045/", "oid": "5c22ba2e-a217-4529-9ed4-cad9f58d0cd2", "rh": "1.ASkAbnq4jlWANUG2nfGceZ7ARU6M1w115GBLra\_lAChgh9fOAQApAA.", "sub": "5c22ba2e-a217-4529-9ed4-cad9f58d0cd2", "tid": "8eb87a6e-8055-4135-b69d-f19c799ec045", "uti": "6Jxt5zB-P0ug5VooRxqLAA", "ver": "1.0", "xms\_ftd": "3MC0rS4UxaOH6OVQVlcxxaakWeMzlMN11hmPIk1F--wBdXNub3J0aC1kc21z" }.[Signature]

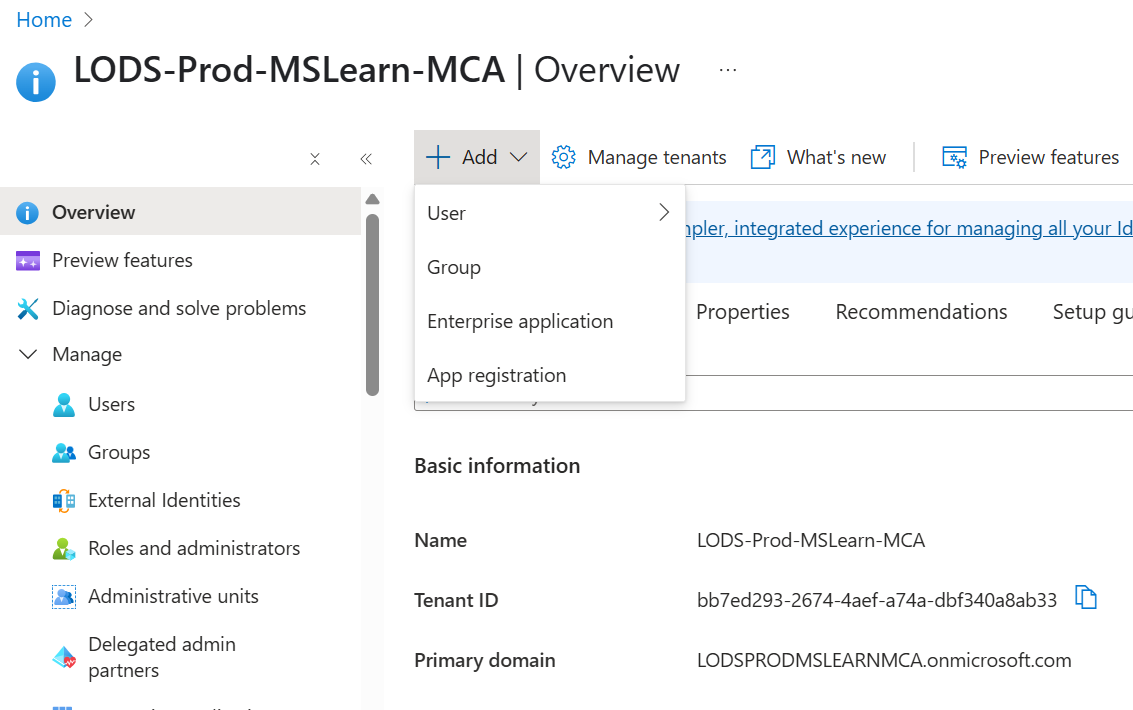


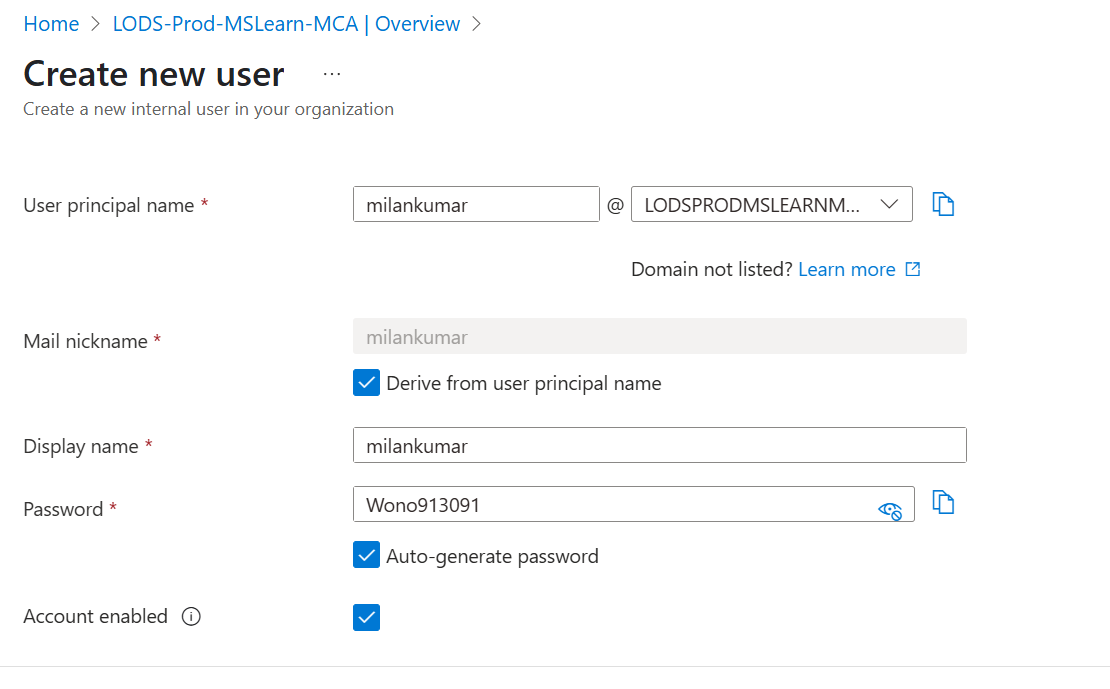






Microsoft Entra Id





**oauth2AllowIdTokenImplicitFlow attribute**

Specifies whether this web app can request OAuth2.0 implicit flow ID tokens. The default is false. This flag is used for browser-based apps, like JavaScript single-page apps. We, however, discourage the use of implicit grant even in SPAs and recommend using the [authorization code flow](https://learn.microsoft.com/en-us/entra/identity-platform/v2-oauth2-auth-code-flow) with PKCE.

Example:

JSONCopy

"oauth2AllowIdTokenImplicitFlow": false

The oauth2AllowImplicitFlow setting in Azure AD, when set to true, enables the implicit flow for an application registration. This allows the application to request access tokens directly from Azure AD without going through the authorization code flow. This is particularly useful for single-page applications (SPAs) built with JavaScript, where a server-side component is not readily available to handle the authorization code exchange.

The oauth2AllowImplicitFlow setting in Azure AD, when set to true, enables the implicit flow for an application registration. This allows the application to request access tokens directly from Azure AD without going through the authorization code flow. This is particularly useful for single-page applications (SPAs) built with JavaScript, where a server-side component is not readily available to handle the authorization code exchange.

Key points about the implicit flow and oauth2AllowImplicitFlow:

* **Purpose:**

The implicit flow is designed for client-side applications, like SPAs, where it's not feasible to securely store a client secret.

* **Tokens:**

When enabled, it allows the application to receive ID tokens and access tokens directly from the authorization endpoint.

* **Refresh tokens:**

The implicit flow does not return refresh tokens.

* **Security Considerations:**

The implicit flow is considered less secure than the authorization code flow due to the direct exposure of tokens in the browser's address bar. It's generally recommended to use the authorization code flow with PKCE (Proof Key for Code Exchange) for enhanced security, especially when handling sensitive data.

* **Manifest setting:**

The oauth2AllowImplicitFlow property is found within the application manifest, accessible in the Azure portal under App registrations, and then selecting the application, and clicking on "Manifest".

* **Enabling the flow:**

To enable the implicit flow, you need to set the oauth2AllowImplicitFlow property to true in the application manifest.

* **Example:**

A JavaScript SPA using the implicit flow might request an ID token and access token for accessing a protected API.