OAuth 2.0 and OpenID Connect protocols - Microsoft identity platform

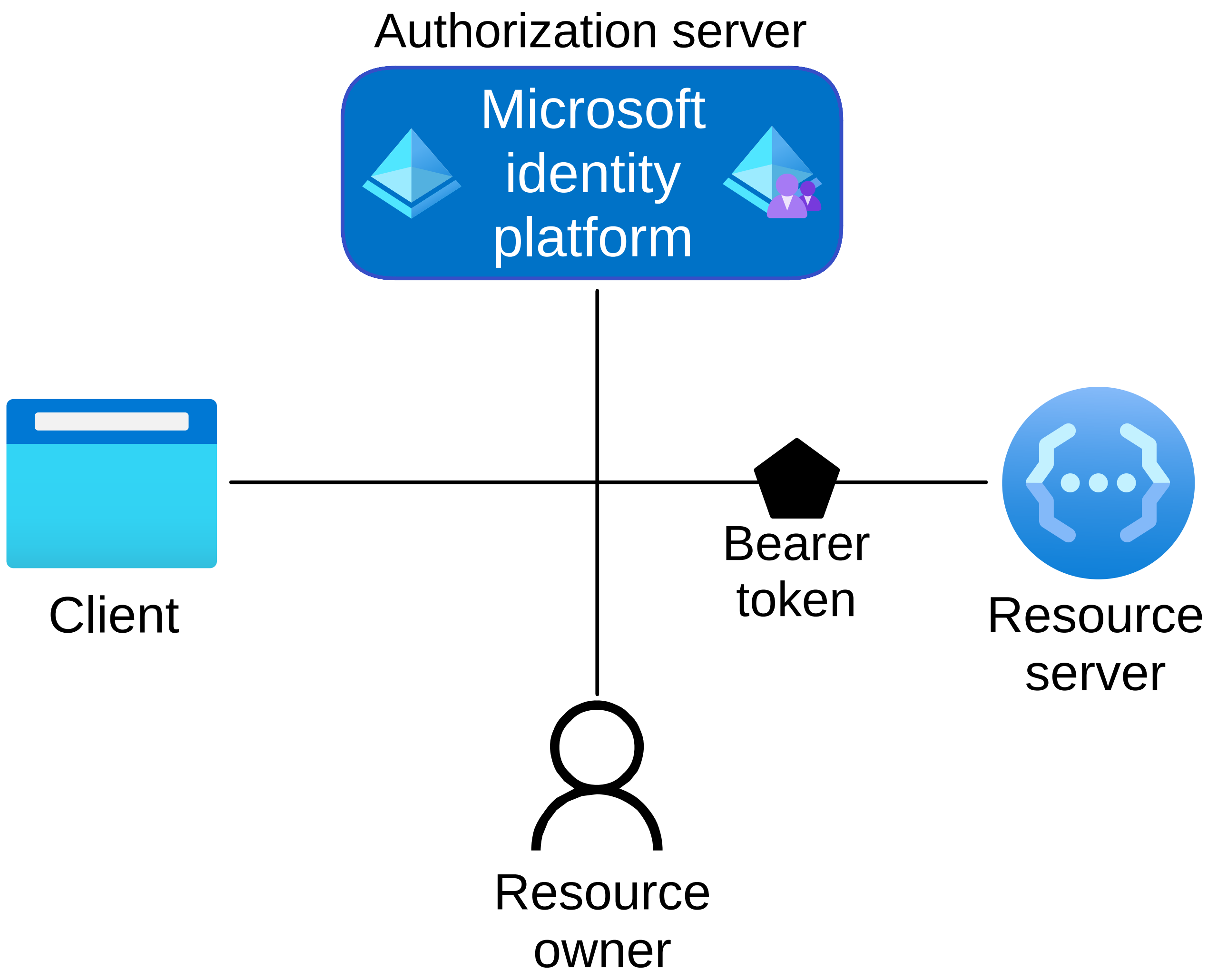
By henrymbuguakiarie

Archived: 2026-04-05 16:15:01 UTC

Knowing about OAuth or OpenID Connect (OIDC) at the protocol level isn't required to use the Microsoft identity platform. However, you'll encounter protocol terms and concepts as you use the identity platform to add authentication to your apps. As you work with the Microsoft Entra admin center, our documentation, and authentication libraries, knowing some fundamentals can assist your integration and overall experience.

Roles in OAuth 2.0

Four parties are generally involved in an OAuth 2.0 and OpenID Connect authentication and authorization exchange. These exchanges are often called *authentication flows* or *auth flows*.



- **Authorization server -** The Microsoft identity platform is the authorization server. Also called an *identity provider* or *IdP*, it securely handles the end-user's information, their access, and the trust relationships between the parties in the auth flow. The authorization server issues the security tokens your apps and APIs use for granting, denying, or revoking access to resources (authorization) after the user has signed in (authenticated).

- **Client -** The client in an OAuth exchange is the application requesting access to a protected resource. The client could be a web app running on a server, a single-page web app running in a user's web browser, or a web API that calls another web API. You'll often see the client referred to as *client application*, *application*, or *app*.

- **Resource owner -** The resource owner in an auth flow is usually the application user, or *end-user* in OAuth terminology. The end-user "owns" the protected resource (their data) which your app accesses on their behalf. The resources owners can grant or deny your app (the client) access to the resources they own. For example, your app might call an external system's API to get a user's email address from their profile on that system. Their profile data is a resource the end-user owns on the external system, and the end-user can consent to or deny your app's request to access their data.

- **Resource server -** The resource server hosts or provides access to a resource owner's data. Most often, the resource server is a web API fronting a data store. The resource server relies on the authorization server to perform authentication and uses information in bearer tokens issued by the authorization server to grant or deny access to resources.

Tokens

The parties in an authentication flow use **bearer tokens** to assure, verify, and authenticate a principal (user, host, or service) and to grant or deny access to protected resources (authorization). Bearer tokens in the Microsoft identity platform are formatted as JSON Web Tokens (JWT).

Three types of bearer tokens are used by the identity platform as *security tokens*:

- Access tokens **-** Access tokens are issued by the authorization server to the client application. The client passes access tokens to the resource server. Access tokens contain the permissions the client has been granted by the authorization server.

- ID tokens **-** ID tokens are issued by the authorization server to the client application. Clients use ID tokens when signing in users and to get basic information about them.

- Refresh tokens **-** The client uses a refresh token, or *RT*, to request new access and ID tokens from the authorization server. Your code should treat refresh tokens and their string content as sensitive data because they're intended for use only by authorization server.

App registration

Your client app needs a way to trust the security tokens issued to it by the Microsoft identity platform. The first step in establishing trust is by registering your app. When you register your app, the identity platform automatically assigns it some values, while others you configure based on the application's type.

Two of the most commonly referenced app registration settings are:

- **Application (client) ID -** Also called *application ID* and *client ID*, this value is assigned to your app by the identity platform. The client ID uniquely identifies your app in the identity platform and is included in the security tokens the platform issues.
- **Redirect URI -** The authorization server uses a redirect URI to direct the resource owner's *user-agent* (web browser, mobile app) to another destination after completing their interaction. For example, after the end-user authenticates with the authorization server. Not all client types use redirect URIs.

Your app's registration also holds information about the authentication and authorization *endpoints* you'll use in your code to get ID and access tokens.

Endpoints

The Microsoft identity platform offers authentication and authorization services using standards-compliant implementations of OAuth 2.0 and OpenID Connect (OIDC) 1.0. Standards-compliant authorization servers like the identity platform provide a set of HTTP endpoints for use by the parties in an auth flow to execute the flow.

The endpoint URIs for your app are generated automatically when you register or configure your app. The endpoints you use in your app's code depend on the application's type and the identities (account types) it should support.

Two commonly used endpoints are the [authorization endpoint](#) and [token endpoint](#). Here are examples of the `authorize` and `token` endpoints:

```
 # Authorization endpoint - used by client to obtain authorization from the resource owner.
https://login.microsoftonline.com/<issuer>/oauth2/v2.0/authorize
# Token endpoint - used by client to exchange an authorization grant or refresh token for an access token.
https://login.microsoftonline.com/<issuer>/oauth2/v2.0/token

# NOTE: These are examples. Endpoint URI format may vary based on application type,
#       sign-in audience, and Azure cloud instance (global or national cloud).

#       The {issuer} value in the path of the request can be used to control who can sign into the application.
#       The allowed values are **common** for both Microsoft accounts and work or school accounts,
#       **organizations** for work or school accounts only, **consumers** for Microsoft accounts only,
#       and **tenant identifiers** such as the tenant ID or domain name.
```

To find the endpoints for an application you've registered, in the [Microsoft Entra admin center](#) navigate to:

Entra ID > **App registrations** > <YOUR-APPLICATION> > **Endpoints**

Next steps

Next, learn about the OAuth 2.0 authentication flows used by each application type and the libraries you can use in your apps to perform them:

- Authentication flows and application scenarios
- Microsoft Authentication Library (MSAL)

We strongly advise against crafting your own library or raw HTTP calls to execute authentication flows. A Microsoft Authentication Library is safer and easier. However, if your scenario prevents you from using our libraries or you'd just like to learn more about the Microsoft identity platform's implementation, we have protocol reference:

- Authorization code grant flow - Single-page apps (SPA), mobile apps, native (desktop) applications
- Client credentials flow - Server-side processes, scripts, daemons
- On-behalf-of (OBO) flow - Web APIs that call another web API on a user's behalf
- OpenID Connect - User sign-in, sign out, and single sign-on (SSO)
