Which OAuth 2.0 Flow Should I Use?

The OAuth 2.0 Authorization Framework supports several different flows (or grants). Flows are ways of retrieving an Access Token. Deciding which one is suited for your use case depends mostly on your application type, but other parameters weigh in as well, like the level of trust for the client, or the experience you want your users to have.

OAuth 2.0 terminology

- **Resource Owner**: Entity that can grant access to a protected resource. Typically, this is the end-user.
- **Client**: Application requesting access to a protected resource on behalf of the Resource Owner.
- **Resource Server**: Server hosting the protected resources. This is the API you want to access.

- **Authorization Server**: Server that authenticates the Resource Owner and issues Access Tokens after getting proper authorization. In this case, Auth0.

- **User Agent**: Agent used by the Resource Owner to interact with the Client (for example, a browser or a native application).

Is the Client the Resource Owner?

The first decision point is about whether the party that requires access to resources is a machine. In the case of machine-to-machine authorization, the Client is also the Resource Owner, so no end-user authorization is needed. An example is a cron job that uses an API to import information to a database. In this example, the cron job is the Client and the Resource Owner since it holds the Client ID and Client Secret and uses them to get an Access Token from the Authorization Server.

If this case matches your needs, then to learn how this flow works and how to implement it, see Client Credentials Flow.

Is the Client a web app executing on the server?

If the Client is a regular web app executing on a server, then the Authorization Code Flow is the flow you should use. Using this the Client can retrieve an Access Token and, optionally, a Refresh Token. It's considered the safest choice since the Access Token is passed directly to the web server hosting the Client, without going through the user's web browser and risking exposure.

If this case matches your needs, then to learn how this flow works and how to implement it, see Authorization Code Flow.

Is the Client absolutely trusted with user credentials?

This decision point may result in the Resource Owner Password Credentials Grant. In this flow, the end-user is asked to fill in credentials (username/password), typically using an interactive form. This information is sent to the backend and from there to Auth0. It is therefore imperative that the Client is absolutely trusted with this information.

This grant should only be used when redirect-based flows (like the Authorization Code Flow) are not possible. If this is your case, then to learn about how this flow works and how to implement it, see Resource Owner Password Flow.

Is the Client a Single-Page App?

If the Client is a Single-Page App (SPA), an application running in a browser using a scripting language like JavaScript, there are two grant options: the Authorization Code Flow with Proof Key for Code Exchange (PKCE) and the Implicit Flow with Form Post. For most cases, we recommend using the Authorization Code Flow with PKCE because the Access Token is not exposed on the client side, and this flow can return Refresh Tokens.

To learn more about how this flow works and how to implement it, see Authorization Code Flow with Proof Key for Code Exchange (PKCE). The Auth0 Single-Page App SDK provides high-level API for implementing Authorization Code Flow with PKCE in SPAs.

If your SPA doesn't need an Access Token, you can use the Implicit Flow with Form Post. To learn more about how this flow works and how to implement it, see Implicit Flow with Form Post.

Is the Client a Native/Mobile App?

If the Application is a native app, then use the **Authorization Code Flow with Proof Key for Code Exchange (PKCE)**.

To learn more about how this flow works and how to implement it, see Authorization Code Flow with Proof Key for Code Exchange (PKCE).

I have an application that needs to talk to different resource servers

If a single application needs access tokens for different resource servers, then multiple calls to `/authorize` (that is, multiple executions of the same or different Authorization Flow) needs to be performed. Each authorization will use a different value for `audience`, which will result in a different access token at the end of the flow. For more information, see the OAuth 2.0: Audience Information Specification.

Can I try the endpoints before I implement my application?

Sure! You have two options:

- Download our Postman collection. To learn more about how to use our Postman collections, see Use Auth0 APIs with Postman Collections.
- Use our Authentication API Debugger Extension. You can find detailed instructions per `/grant` endpoint at our Authentication API Reference.
    - For the Authorize endpoint, go to Authorize Application and read the "Test this endpoint" paragraph for the grant you want to test.

- For the Token endpoint, go to Get Token and read the "Test this endpoint" section for the grant you want to test.

Keep reading

- Authentication and Authorization Flows

Lock

COMPANY

About Us

Blog

Jobs

Press

LEARN

Availability & Trust

Security

White Hat

API Explorer

MORE

Help & Support

Professional Services

Documentation

Open Source

WordPress

CONTACT

10800 NE 8th Street
Suite 600
Bellevue, WA 98004

+1 (888) 235-2699
+1 (425) 312-6521
+44 (0) 33-3234-1966