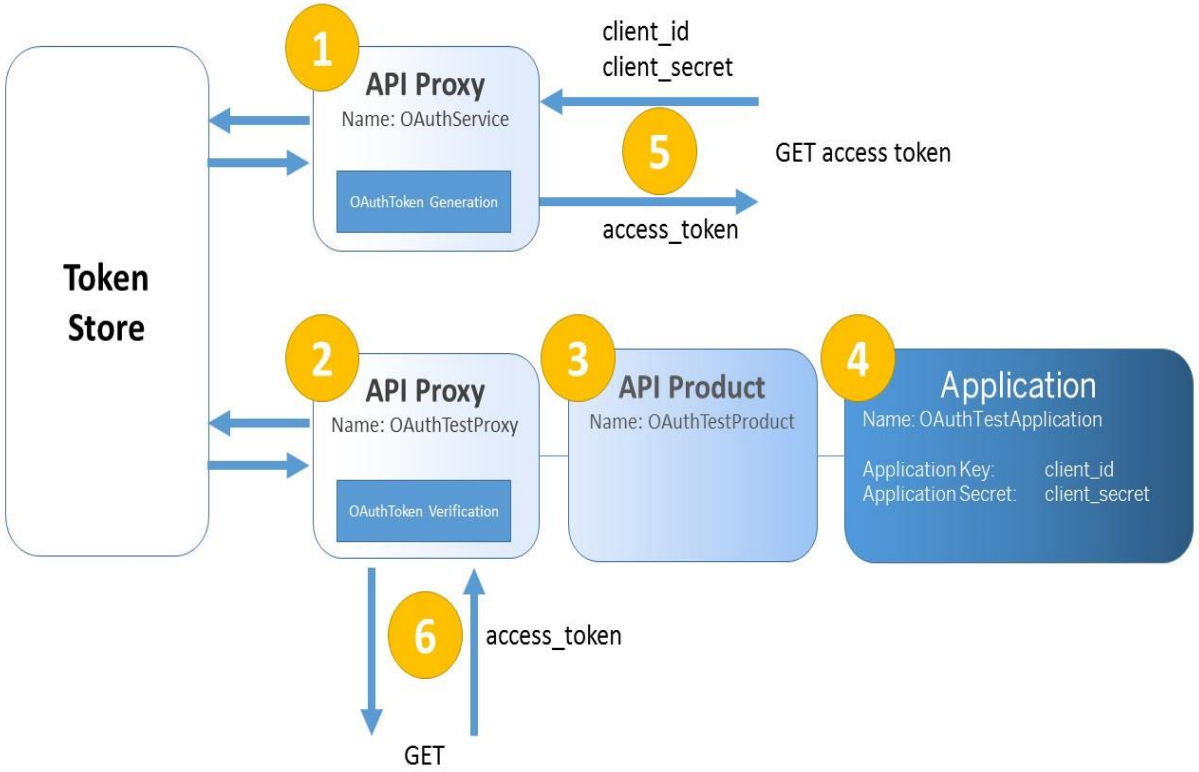
OAuth 2.0

OAuth is an open authorization protocol, which allows accessing the resources of the resource owner by enabling the client applications on HTTP services such as Facebook, GitHub, etc. It allows sharing of resources stored on one site to another site without using their credentials. It uses username and password tokens instead.

The most common OAuth 2.0 grant types are listed below.

- Authorization Code
- Implicit
- Password
- Client Credentials
- Refresh Token

Client Credential Grant Type

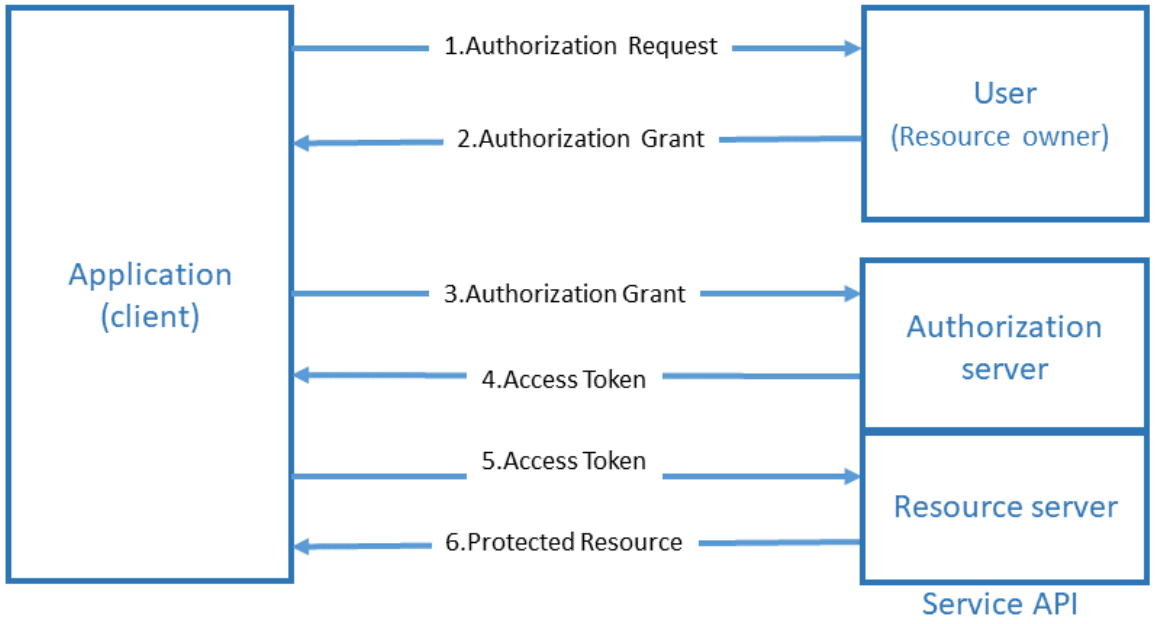


- The client can request an access token using only its client credentials (or other supported means of authentication) when the client is requesting access to the protected resources under its control, or those of another resource owner that have been previously arranged with the authorization server (the method of which is beyond the scope of this specification).

Client Credential Flow:

- The client requests for the client credentials from developer portal by consuming the application.
- The client authenticates with the authorization server and requests an access token from the token endpoint.
- The authorization server authenticates the client, and if valid, issues an access token.
- Using the access token client will be able to access the API Proxy.

Authorization Code Grant Type

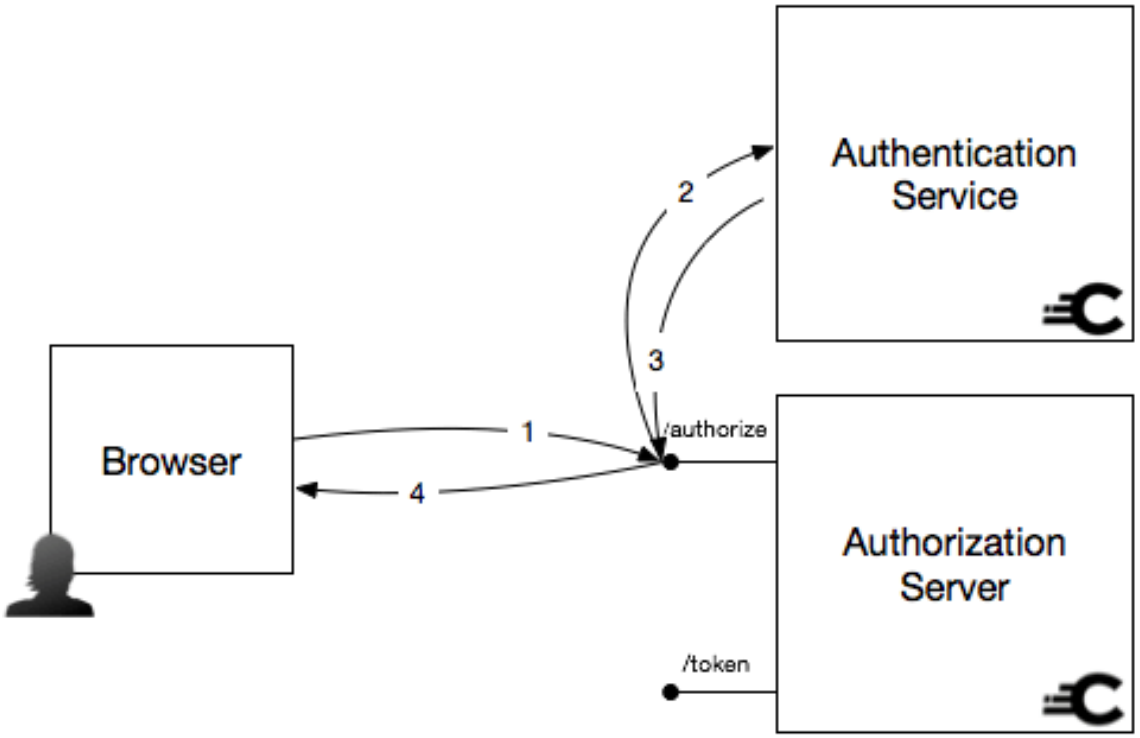


- The authorization code grant type is used to obtain both access tokens and refresh tokens and is optimized for confidential clients. Since this is a redirection-based flow, the client must be capable of interacting with the resource owner's user-agent (typically a web browser) and capable of receiving incoming requests (via redirection) from the authorization server

Authorization Code

- The client initiates the flow by directing the resource owner's user-agent to the authorization endpoint. The client includes its client identifier, requested scope, local state, and a redirection URI to which the authorization server will send the user-agent back once access is granted (or denied).
- The authorization server authenticates the resource owner (via the user-agent) and establishes whether the resource owner grants or denies the client's access request.
- Assuming the resource owner grants access, the authorization server redirects the user-agent back to the client using the redirection URI provided earlier (in the request or during client registration). The redirection URI includes an authorization code and any local state provided by the client earlier.
- The client requests an access token from the authorization server's token endpoint by including the authorization code received in the previous step. When making the request, the client authenticates with the authorization server. The client includes the redirection URI used to obtain the authorization code for verification.
- The authorization server authenticates the client, validates the authorization code, and ensures that the redirection URI received matches the URI used to redirect the client in step (C). If valid, the authorization server responds back with an access token and, optionally, a refresh token.

Implicit Grant Type

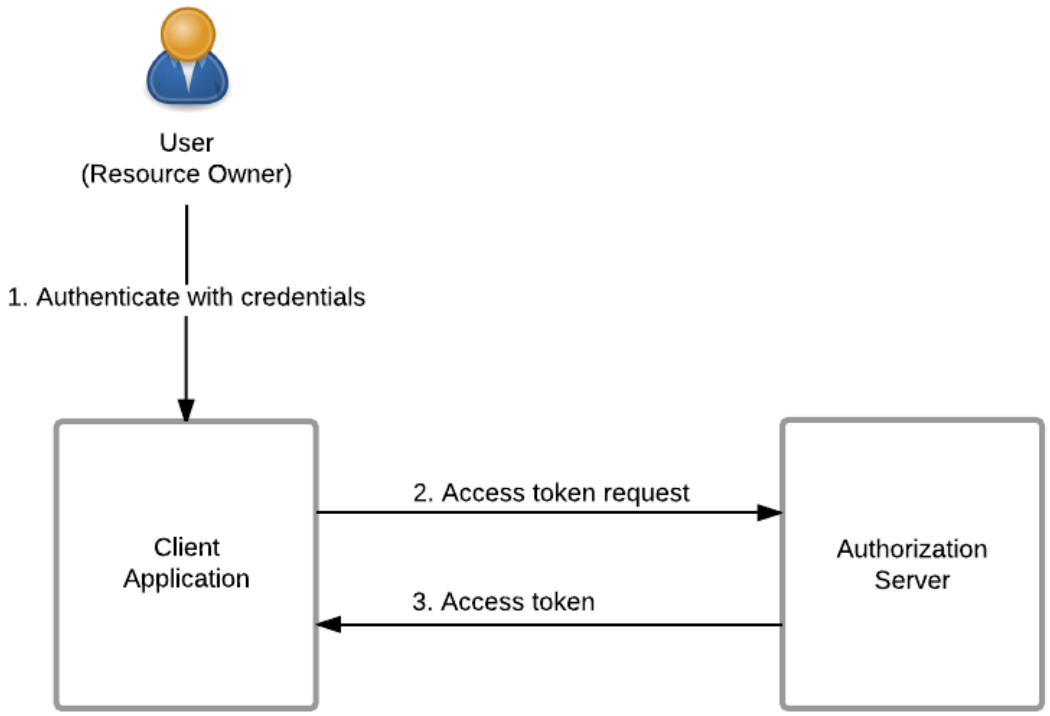


- The implicit grant type is used to obtain access tokens (it does not support the issuance of refresh tokens) and is optimized for public clients known to operate a particular redirection URI. These clients are typically implemented in a browser using a scripting language such as JavaScript.

Implicit Grant Flow:

- The client initiates the flow by directing the resource owner's user-agent to the authorization endpoint. The client includes its client identifier, requested scope, local state, and a redirection URI to which the authorization server will send the user-agent back once access is granted (or denied).
- The authorization server authenticates the resource owner (via the user-agent) and establishes whether the resource owner grants or denies the client's access request.
- Assuming the resource owner grants access, the authorization server redirects the user-agent back to the client using the redirection URI provided earlier. The redirection URI includes the access token in the URI fragment.
- The user-agent follows the redirection instructions by making a request to the web-hosted client resource (which does not include the fragment per [RFC2616]). The user-agent retains the fragment information locally.
- The web-hosted client resource returns a web page (typically an HTML document with an embedded script) capable of accessing the full redirection URI including the fragment retained by the user-agent, and extracting the access token (and other parameters) contained in the fragment.
- The user-agent executes the script provided by the web-hosted client resource locally, which extracts the access token.
- The user-agent passes the access token to the client.

Password Grant Type

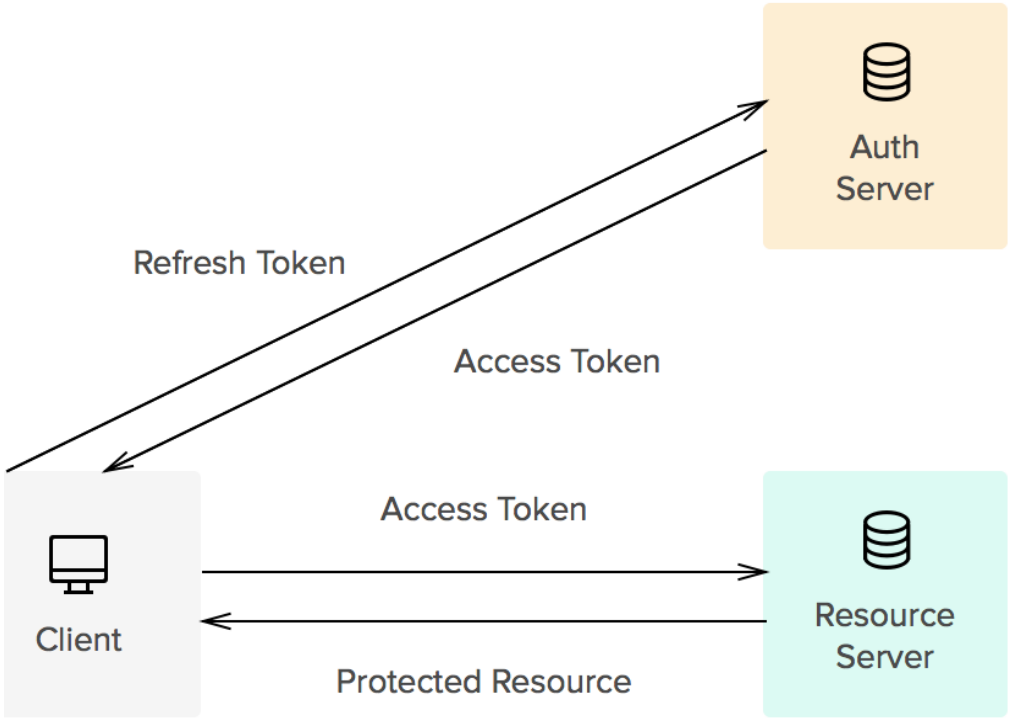


- The resource owner password credentials grant type is suitable in cases where the resource owner has a trust relationship with the client, such as the device operating system or a highly privileged application. The authorization server should take special care when enabling this grant type and only allow it when other flows are not viable.

Password Grant Flow:

- The resource owner provides the client with its username and password.
- The client requests an access token from the authorization server’s token endpoint by including the credentials received from the resource owner.
- When making the request, the client authenticates with the authorization server.
- The authorization server authenticates the client and validates the resource owner credentials, and if valid, issues an access token.

Refresh Token Grant Type



- The Refresh Token grant type is used by clients to exchange a refresh token for an access token when the access token has expired. This allows clients to continue to have a valid access token without further interaction with the user.

Refresh Token Flow

- When a access token expires client requests authorization server for another access token through a refresh token.
- Authorization server grants the access token.
- This token is sent to the Resource server in order to gain the API Proxies access.
- Resource server validates the access token and approves the access for the API Proxy.