API

An API is a set of commands, [functions](https://techterms.com/definition/function), [protocols](https://techterms.com/definition/protocol), and objects that programmers can **use to create**[**software**](https://techterms.com/definition/software)**or interact with an external system**. It provides [developers](https://techterms.com/definition/developer) with standard commands for performing common operations so they do not have to write the code from scratch. Additionally, **APIs** are used when programming graphical user interface (GUI) components.

OAuth/OpenID

[OpenID](http://openid.net/) is about authentication (ie. proving who you are), [OAuth](http://oauth.net/) is about authorization (ex. I grant your application access to my Google drive account. When your application is accessing my drive account there is no guarantee that it is me preforming the action.).

OAuth is often used in external sites to allow access to protected data without them having to re-authenticate a user.

If so why cannot I use OAuth for authentication ?

Because Oauth knows nothing about who is preforming the task. It could be an application running in the background.

## OAuth Roles

OAuth defines four roles:

* Resource Owner:  is the user who authorizes an application to access their account. The application's access to the user's account is limited to the "scope" of the authorization granted (e.g. read or write access).
* Client: The client is the application that wants to access the user's account. Before it may do so, it must be authorized by the user, and the authorization must be validated by the API.
* Resource/Authorization Server(API): The resource server hosts the protected user accounts, and the authorization server verifies the identity of the user then issues access tokens to the application. From an application developer's point of view, a service's **API** fulfills both the resource and authorization server roles. We will refer to both of these roles combined, as the Service or API role.

How Oauth2 work?

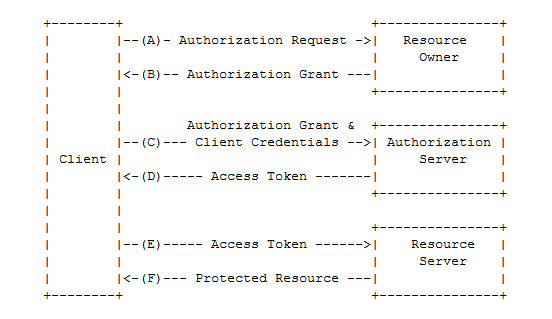


Figure 1Abstract protocol flow

1. Client send authorization request(AR) to resourse owner(RO). AR can be proceeded at resourse owner or better at a middle man such as authorization server(AS).
2. Client will receive a resourse grant from RO. Resourse grant depends on operation which client used and supported by AS.
3. Client request an access token by verify with AS
4. AS confirmed then issues the access token.
5. Client requests resourses which are protected by RO and verify by giving access token
6. RO confirm the access token. If it valid, server will proceed the request.

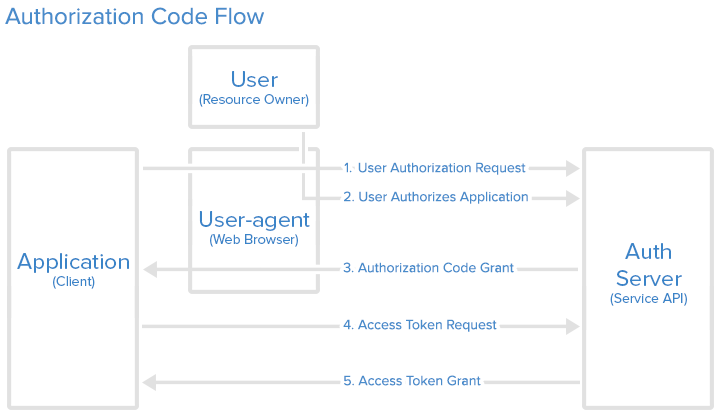
## Authorization Grant

In the Abstract Protocol Flow above, the first four steps cover obtaining an authorization grant and access token. The authorization grant type depends on the method used by the application to request authorization, and the grant types supported by the API. OAuth 2 defines four grant types, each of which is useful in different cases:

* **Authorization Code**: used with server-side Applications
* **Implicit**: used with Mobile Apps or Web Applications (applications that run on the user's device)
* **Resource Owner Password Credentials**: used with trusted Applications, such as those owned by the service itself
* **Client Credentials**: used with Applications API access

## Grant Type: Authorization Code (API dùng cho authorize trainee và sup)

The **authorization code** grant type is the most commonly used because it is optimized for server-side applications, where source code is not publicly exposed, and Client Secret confidentiality can be maintained. This is a redirection-based flow, which means that the application must be capable of interacting with the user-agent (i.e. the user's web browser) and receiving API authorization codes that are routed through the user-agent.



### Step 1: Authorization Code Link( trainee ra yêu cầu tạo QR code và nhận đc link)

First, the user is given an authorization code link that looks like the following:

https://cloud.digitalocean.com/v1/oauth/authorize?response\_type=code&client\_id=CLIENT\_ID&redirect\_uri=CALLBACK\_URL&scope=read

Here is an explanation of the link components:

* [**https://cloud.digitalocean.com/v1/oauth/authorize**](https://cloud.digitalocean.com/v1/oauth/authorize): the API authorization endpoint
* **client\_id=client\_id**: the application's client ID (how the API identifies the application)
* **redirect\_uri=CALLBACK\_URL**: where the service redirects the user-agent after an authorization code is granted
* **response\_type=code**: specifies that your application is requesting an authorization code grant
* **scope=read**: specifies the level of access that the application is requesting

### Step 2: User Authorizes Application( nếu trainee cấp quyền cho service tạo QR)

When the user clicks the link, they must first log in to the service, to authenticate their identity (unless they are already logged in). Then they will be prompted by the service to authorize or deny the application access to their account.

### Step 3: Application Receives Authorization Code( nhận QR)

If the user clicks "Authorize Application", the service redirects the user-agent to the application redirect URI, which was specified during the client registration, along with an authorization code. The redirect would look something like this (assuming the application is "dropletbook.com"):

<https://dropletbook.com/callback?code=AUTHORIZATION_CODE>

### Step 4: Application Requests Access Token( đưa QR cho sup quét)

The application requests an access token from the API, by passing the authorization code along with authentication details, including the client secret, to the API token endpoint. Here is an example POST request to DigitalOcean's token endpoint:

https://cloud.digitalocean.com/v1/oauth/token?client\_id=CLIENT\_ID&client\_secret=CLIENT\_SECRET&grant\_type=authorization\_code&code=AUTHORIZATION\_CODE&redirect\_uri=CALLBACK\_URL

### Step 5: Application Receives Access Token( sup nhận đc quyền truy cập[token] vào profile của trainee)

If the authorization is valid, the API will send a response containing the access token (and optionally, a refresh token) to the application. The entire response will look something like this:

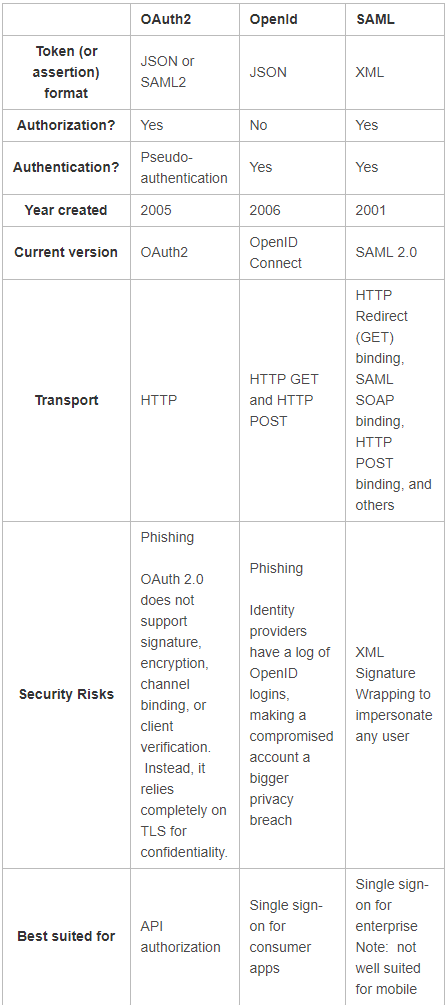
{"access\_token":"ACCESS\_TOKEN","token\_type":"bearer","expires\_in":2592000,"refresh\_token":"REFRESH\_TOKEN","scope":"read","uid":100101,"info":{"name":"Mark E. Mark","email":"mark@thefunkybunch.com"}}

Ý tưởng sẽ sử dụng email làm việc của bệnh viện cấp cho trainee và sup, email đc dùng để sử dụng google api được tích hợp Oauth2

References:

<https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2>

<https://developers.google.com/identity/protocols/OAuth2InstalledApp> (install google api)



 It’s important for API traffic to be protected by encrypting sensitive data, implementing [data masking](https://docs.apigee.com/api-services/content/data-masking) for trace/logging, and using tokenization for card information.