What is 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.

OAuth 2.0 is developed by the IETF OAuth Working Group, published in October 2012.

Why Use OAuth 2.0?

1. You can use OAuth 2.0 to read data of a user from another application.
2. It supplies the authorization workflow for web, desktop applications, and mobile devices.
3. It is a server side web app that uses authorization code and does not interact with user credentials.

Features of OAuth 2.0

1. OAuth 2.0 is a simple protocol that allows to access resources of the user without sharing passwords.
2. It provides user agent flows for running clients application using a scripting language, such as JavaScript. Typically, a browser is a user agent.
3. It accesses the data using tokens instead of using their credentials and stores data in online file system of the user such as Google Docs or Dropbox account.

Advantages of OAuth 2.0

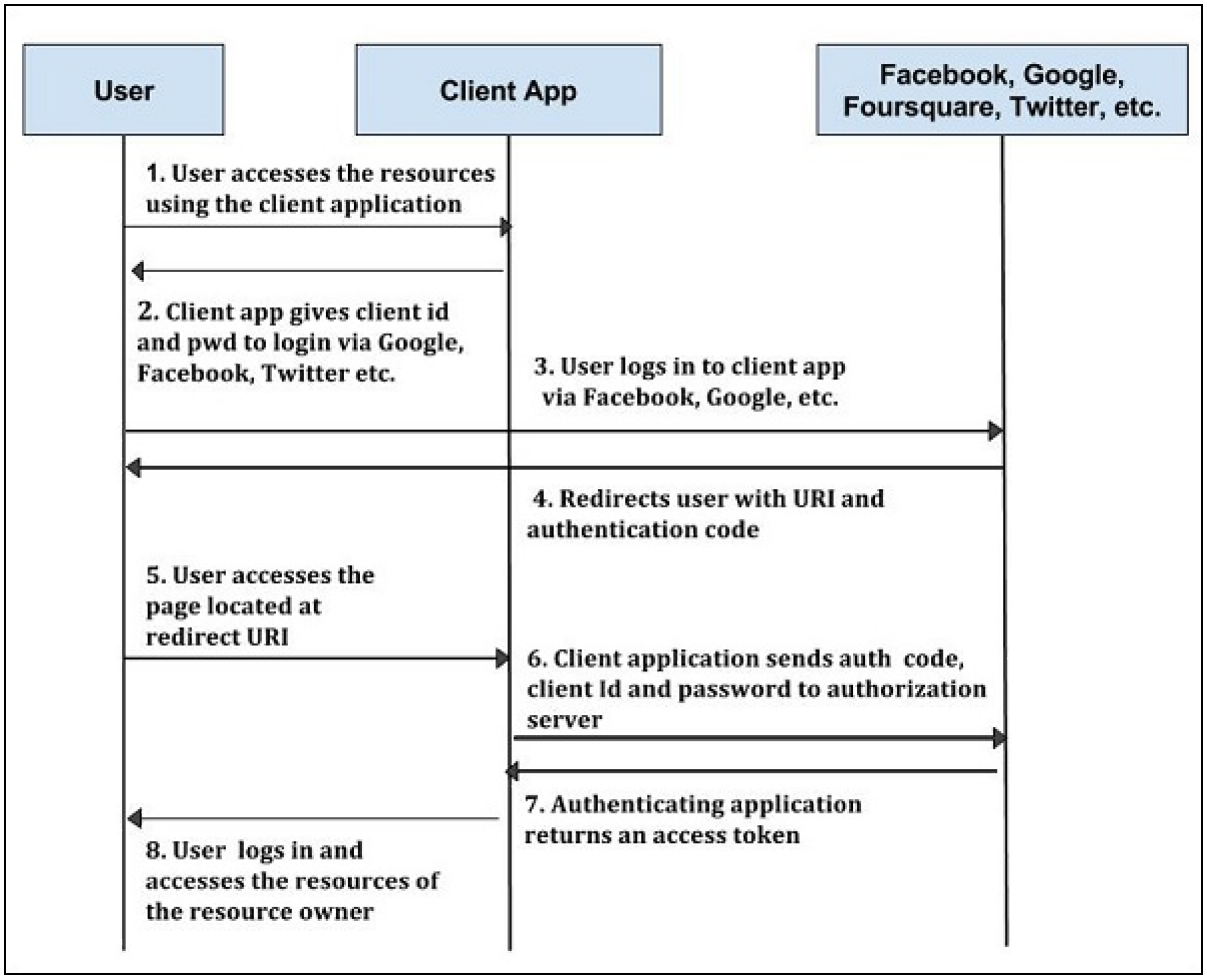
1. OAuth 2.0 is a very flexible protocol that relies on SSL (Secure Sockets Layer that ensures data between the web server and browsers remain private) to save user access token.
2. OAuth 2.0 relies on SSL which is used to ensure cryptography industry protocols and are being used to keep the data safe.
3. It allows limited access to the user's data and allows accessing when authorization tokens expire.
4. It has ability to share data for users without having to release personal information.
5. It is easier to implement and provides stronger authentication.

Disadvantages of OAuth 2.0

1. If you are adding more extension at the ends in the specification, it will produce a wide range of non-interoperable implementations, which means you have to write separate pieces of code for Facebook, Google, etc.
2. If your favorite sites are connected to the central hub and the central account is hacked, then it will lead to serious effects across several sites instead of just one.

OAuth 2.0 - Architecture

In this chapter, we will discuss the architectural style of OAuth 2.0.



Step 1 − First, the user accesses resources using the client application such as Google, Facebook, Twitter, etc.
Step 2 − Next, the client application will be provided with the client id and client password during registering the redirect URI (Uniform Resource Identifier).
Step 3 − The user logs in using the authenticating application. The client ID and client password is unique to the client application on the authorization server.
Step 4 − The authenticating server redirects the user to a redirect Uniform Resource Identifier (URI) using authorization code.
Step 5 − The user accesses the page located at redirect URI in the client application.
Step 6 − The client application will be provided with the authentication code, client id and client password, and send them to the authorization server.
Step 7 − The authenticating application returns an access token to the client application.
Step 8 − Once the client application gets an access token, the user starts accessing the resources of the resource owner using the client application.

OAuth 2.0 has various concepts, which are briefly explained in the following table.

Sr.No.	Concept & Description
1	Terminology OAuth provides some additional terms to understand the concepts of authorization.
2	Web Server Web server delivers the web pages and uses HTTP to serve the files that forms the web pages to the users.
3	User-Agent The user agent application is used by client applications in the user's device, which acts as the scripting language instance.
4	Native Application Native application can be used as an instance of desktop or mobile phone application, which uses the resource owner password credentials.

OAuth 2.0 - Terminology
Following is the explanation of OAuth 2.0 terms −

Authentication
Authentication is a process of identifying an individual, usually based on a username and password. It is about knowing that the user is the owner of the account on the web and desktop computers.

Federated Authentication
Many applications have their own username and passwords. Some applications depend on other services for verification of the user's identity. A federated identity management system provides a single access to multiple systems. This is known as federated authentication.

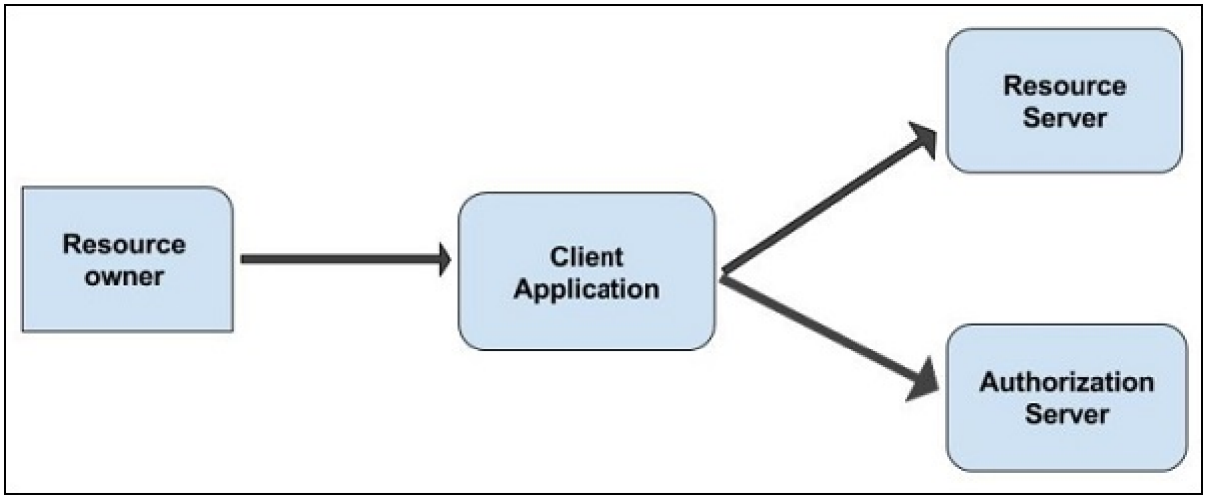
Authorization
Authorization is the process of giving someone the permission to do something. It needs the valid user's identification to check whether that user is authorized or not.

Delegated Authorization
Delegated authorization is the process of giving one’s credentials to other user to perform some actions on behalf of that user.

Roles
OAuth defines the following roles −

- Resource Owner
- Client Application
- Resource Server
- Authentication Server

The roles are illustrated in the following figure −

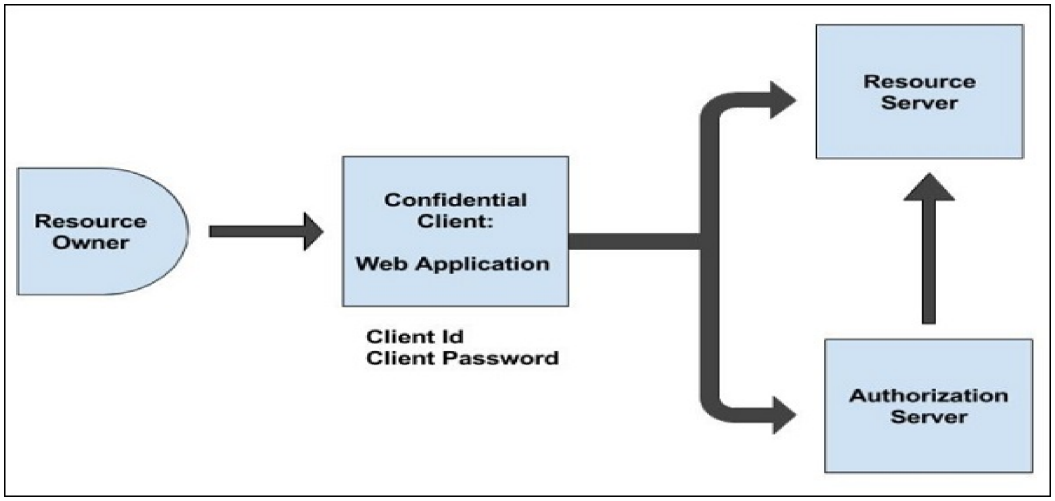


- **Resource Owner** − Resource owner is defined as an entity having the ability to grant access to their own data hosted on the resource server. When the resource owner is a person, it is called the end-user.
- **Client Application** − Client is an application making protected resource requests to perform actions on behalf of the resource owner.
- **Resource Server** − Resource server is API server that can be used to access the user's information. It has the capability of accepting and responding to protected resource requests with the help of access tokens.
- **Authentication Server** − The authentication server gets permission from the resource owner and distributes the access tokens to clients, to access protected resource hosted by the resource server.

OAuth 2.0 - Web Server

The web server is a computer system that delivers the web pages to the users by using HTTP. The client ID and password is stored on the web application server, whenever the application wants to access the resource server. The client ID and password which is stored on the web application server is intended to be kept secret.

The following figure depicts the Confidential Client Web Application Server −

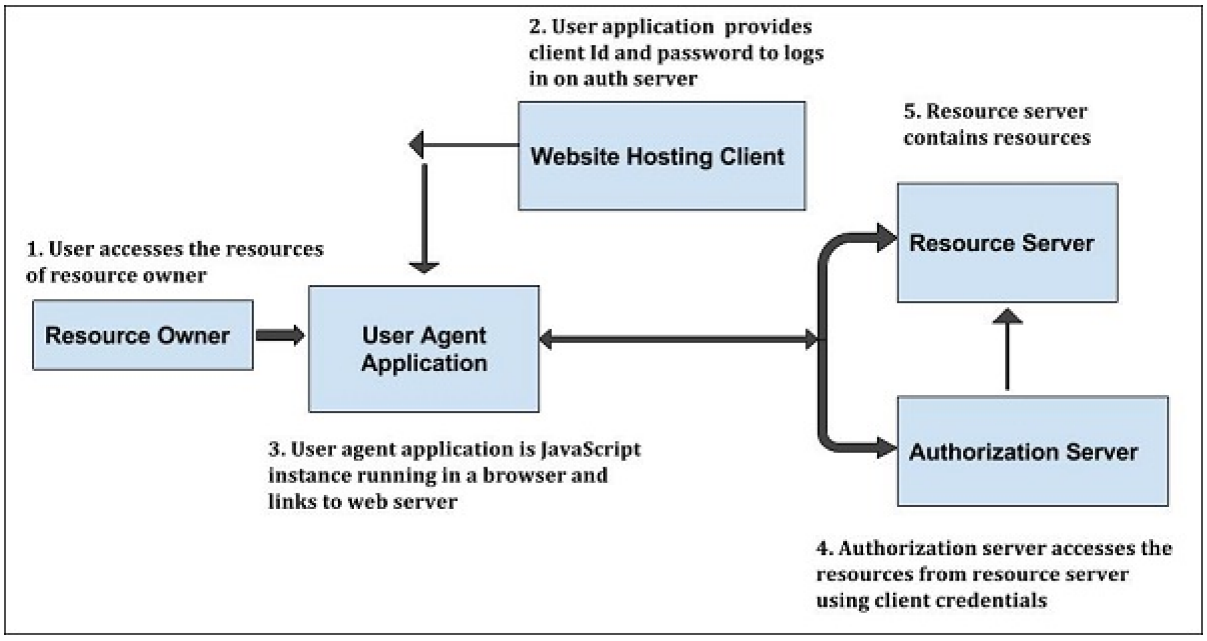


- In the above figure, the resource owner allows the confidential client to access the data that is hosted on the resource server, where client ID and password are kept confidential on the server.
- The client ID and password is unique to the client application on the authorization server.
- The resource server is a server, which hosts the resources such as Facebook, Twitter, Google, etc. These resources are stored on the resource server and are accessed by the client application and the resource owner owns these resources.
- The resources of the resource owner are then accessed by the authorization server using confidential client web application.

OAuth 2.0 - User Agent

The user agent application is used by the client applications in the user's device, which acts as the scripting language instance such as JavaScript running in a browser. You can store the user agent application on a web server.

The following diagram shows the architecture of the client user agent application.



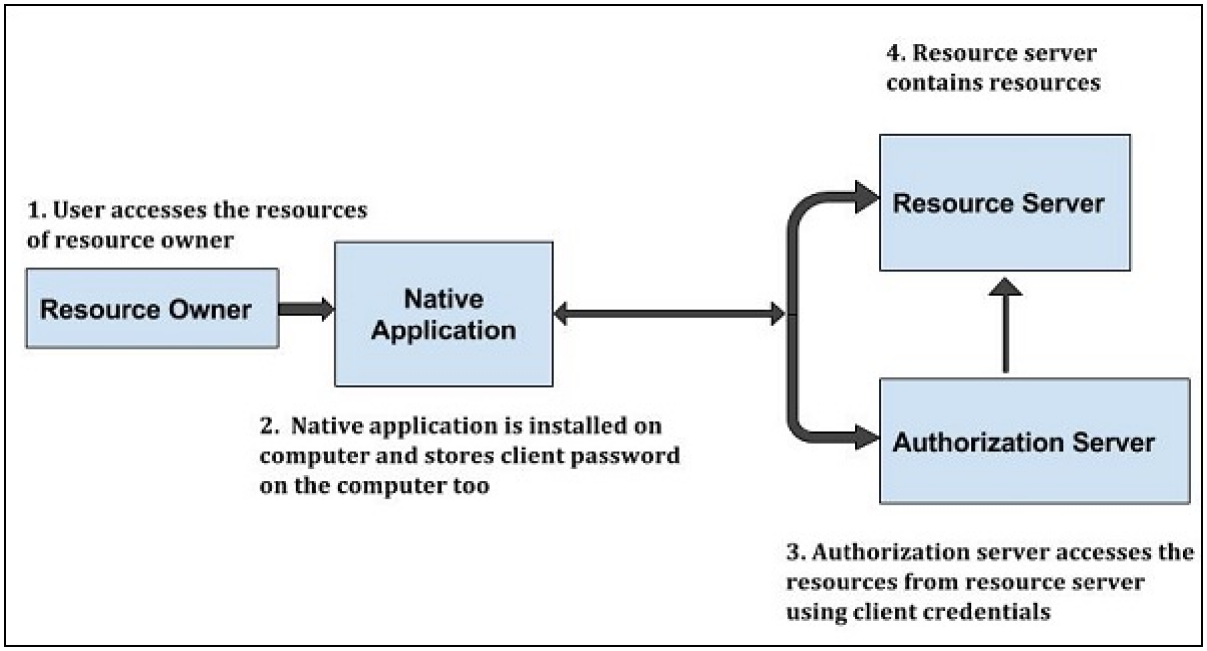
Step 1 − First, the user accesses the resources of the resource owner by using authenticating application such as Google, Facebook, Twitter, etc.
Step 2 − Next, the user application provides the client Id and client password to log on to the authorization server.
Step 3 − Then, the user agent application provides an instance of a JavaScript application running in a browser and links to the web server.
Step 4 − The authorization server allows access to the resources from the resource server using the client credentials.
Step 5 − The resource server contains the resources, which are owned by the resource owner.

OAuth 2.0 - Native Application

Native application can be used as instance of desktop or mobile phone application, which uses the resource owner credentials. It is a public client installed that executes on the resource's owner device.

The authentication credentials used by the application are included in the application code. Hence, do not use the native application that runs in the external user agents.

The following diagram shows the architecture of the client native application −



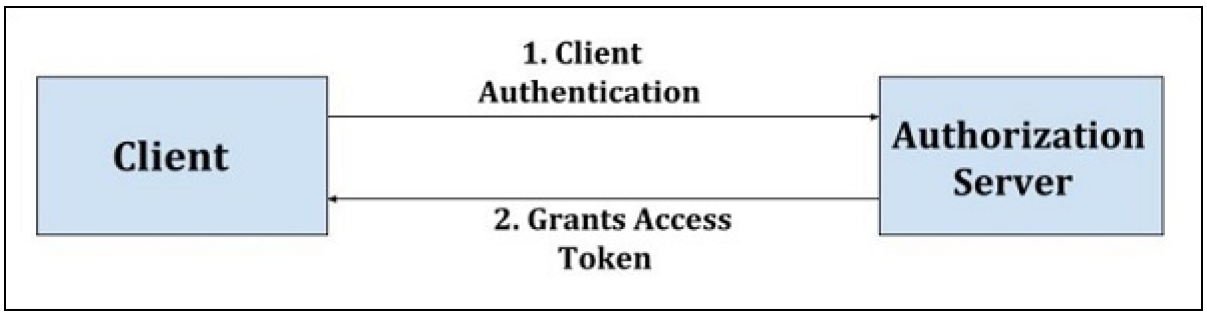
Step 1 − First, the user accesses the resources of the resource owner by using authenticating application such as Google, Facebook, Twitter, etc.
Step 2 − Next, the native application uses client Id and client password to log on to the authorization server. The native application is an instance of desktop or mobile phone application, which is installed on the user computer and stores the client password on the computer or device.
Step 3 − The authorization server allows accessing the resources from the resource server using the client credentials.
Step 4 − The resource server contains the resources, which are owned by the resource owner.

OAuth 2.0 - Client Credentials

The client credentials can be used as an authorization grant when the client is the resource owner, or when the authorization scope is limited to protected resources under the control of the client.

- The client requests an access token only with the help of client credentials.
- The client credentials authorization flow is used to acquire access token to authorize API requests.
- Using client credentials authorization, access token which is acquired, only grants permission for your client application to search and get catalog documents.

The following figure depicts the Client Credentials Flow.



The flow illustrated in the above figure consists of the following steps −
Step 1 − The client authenticates with the authorization server and makes a request for access token from the token endpoint.
Step 2 − The authorization server authenticates the client and provides access token if it's valid and authorized.

The following table lists the concepts of Client Credentials.

Sr.No.	Concept & Description
1	**Obtaining End-User Authorization** The authorization end point is typically URI on the authorization server in which the resource owner logs in and permits to access the data to the client application.
2	**Authorization Response** The authorization response can be used to get the access token for accessing the owner resources in the system using the authorization code.
3	**Error Response and Codes** The authorization server responds with a HTTP 400 or 401 (bad request) status codes, if an error occurs during authorization.

OAuth 2.0 - Obtaining an Access Token
An access token is a string that identifies a user, an application, or a page. The token includes information such as when the token will expire and which app created that token.

- First, it is necessary to acquire OAuth 2.0 client credentials from API console.
- Then, the access token is requested from the authorization server by the client.
- It gets an access token from the response and sends the token to the API that you wish to access.

You must send the user to the authorization endpoint at the beginning. Following is an example of a dummy request

https://publicapi.example.com/oauth2/authorize?client_id=your_client_id&redirect_uri=your_url&response_type=code

Following are the parameters and their descriptions.

- **client_id** − It should be set to the client id of your application.
- **redirect_uri** − It should be set to the URL. After the request is authorized, the user will be redirected back.
- **response_type** − It can either be a code or a token. The code must be used for server side

applications, whereas the token must be used for client side applications. In server side applications, you can make sure that the secrets are saved safely.

Following table lists the concepts of Client Credentials.

Sr.No.	Concept & Description
1	Authorization Code The authorization code allows accessing the authorization request and grants access to the client application to fetch the owner resources.
2	Resource Owner Password Credentials The resource owner password credentials include only one request and one response, and is useful where the resource owner has a good relationship with the client.
3	Assertion Assertion is a package of information that makes the sharing of identity and security information across various security domains possible.
4	Refresh Token The refresh tokens are used to acquire a new access tokens, which carries the information necessary to get a new access token.
5	Access Token Response Access token is a type of token that is assigned by the authorization server.
6	Access Token Error Response Codes If the token access request, which is issued by the authorization server is invalid or unauthorized, then the authorization server returns an error response.

OAuth 2.0 - Accessing a Protected Resource

The client provides an access token to the resource server to access protected resources. The resource server must validate and verify that the access token is valid and has not expired.

There are two standard ways of sending credentials −

- Bearer Token − The access token can only be placed in POST request body or GET URL parameter as a fallback option in the authorization HTTP header.

They are included in the authorization header as follows −

Authorization: Bearer [token-value]

For Example −

```
GET/resource/1 HTTP /1.1
Host: example.com
Authorization: Bearer abc...
```

- MAC − A cryptographic Message Authentication Code (MAC) is computed using the elements of the request and is sent to the authorization header. Upon receiving the request, the MAC is then compared and computed by the resource owner.

The following table shows the concepts of accessing protected resource.

Sr.No.	Concept & Description
1	Authenticated Requests It is used to get the authorization code token for accessing the owner resources in the system.
2	The WWW-Authenticate Response Header Field The resource server includes the "WWW-Authenticate" response header field, if the protected resource request contains an invalid access token.

OAuth 2.0 - Extensibility

here are two ways in which the access token types can be defined −

- By registering in the access token type's registry.
- By using a unique absolute URI (Uniform Resource Identifier) as its name.

Defining New Endpoint Parameters
Parameter names must obey the param-name ABNF (Augmented Backus-Naur Form is a metalanguage based on Backus-Naur Form consisting of its own syntax and derivation rules) and the syntax of parameter values must be well-defined.

```
param-name = 1* name-char
name-char = "-" / "." / "_" / DIGIT / ALPHA
```

Defining New Authorization Grant Types
New authorization grant types can be assigned a distinct absolute URI for use, with the help of "grant_type" parameter. The extension grant type must be registered in the OAuth parameters registry, if it requires additional token endpoint parameters.
Defining New Authorization Endpoint Response Types

```
response-type = response-name *(SP response-name)
response-name = 1* response-char
response-char = "_" / DIGIT / ALPHA
```

The response type is compared as space-delimited list of values, if it has one or more space characters where the order of the values does not matter and only one order of value can be registered.
Defining Additional Error Codes

The extension error codes must be registered, if the extensions they use are either a registered access token, or a registered endpoint parameter. The error code must obey the error ABNF (Augmented Backus-Naur Form) and when possible it should be prefixed by a name identifying it.

```
error = 1 * error_char
error-char =  %x20-21 / %x23-5B / 5D-7E
```

OAuth 2.0 - IANA Considerations

IANA stands for Internet Assigned Numbers Authority which provides the information about the registration values related to the Remote Authentication Dial In User Service (RADIUS).

IANA includes the following considerations −

OAuth Access Token Types Registry
OAuth access tokens are registered by experts with required specification. If they are satisfied with the registration, only then they will publish the specification. The registration request will be sent to the @ietf.org for reviewing with the subject ("Request for access token type: example"). Experts will either reject or accept the request within 14 days of the request.

Registration Template
The registration template contains the following specifications −

- **Type Name −** It is the name of the request.
- **Token Endpoint Response Parameters −** The additional access token response parameter will be registered separately in OAuth parameters registry.
- **HTTP Authentication Scheme −** The HTTP authentication scheme can be used to authenticate the resources by using the access token.
- **Change Controller −** Give the state name as "IETF" for standard track RFCs, and for others, use the name of the responsible party.
- **Specification Document −** The specification document contains the parameter that can be used to retrieve a copy of the document.

OAuth Parameters Registry
OAuth parameters registry contains registration of authorization endpoint request or response, token endpoint request or response by the experts with the required specification. The registration request will be sent to the experts and if they are satisfied with registration, then they will publish the specification.

Registration Template
The registration template contains specifications such as Type Name, Change Controller and Specification Document as defined in the above OAuth Access Token Types Registry section, except the following specification −

Parameter Usage Location − It specifies the location of the parameter such as authorization request or response, token request or response.

Initial Registry Contents
The following table shows OAuth parameters registry containing the initial contents −

Sr.No.	Parameter Name & Usage Location	Change Controller	Specification Document
1	**client_id** authorization request, token request	IETF	RFC 6749
2	**client_secret** token request	IETF	RFC 6749
3	**response_type** authorization_request	IETF	RFC 6749
4	**redirect_uri** authorization request, authorization	IETF	RFC 6749

5	**scope** authorization request or response, token request or response	IETF	RFC 6749
6	**state** authorization request or response	IETF	RFC 6749
7	**code** token request, authorization response	IETF	RFC 6749
8	**error_description** authorization response, token response	IETF	RFC 6749
9	**error_uri** authorization response, token response	IETF	RFC 6749
10	**grant_type** token request	IETF	RFC 6749
11	**access_token** authorization response, token response	IETF	RFC 6749
12	**token_type** authorization response, token response	IETF	RFC 6749
13	**expires_in** authorization response, token response	IETF	RFC 6749
14	**username** token request	IETF	RFC 6749
15	**password** token request	IETF	RFC 6749
16	**refresh_token** token request, token response	IETF	RFC 6749

OAuth Authorization Endpoint Response Type Registry

This can be used to define OAuth Authorization Endpoint Response Type Registry. The response types are registered by experts with the required specification and if they are satisfied with the registration, only then they will publish the specification. The registration request will be sent to the @ietf.org for reviewing. The experts will either reject or accept the request within 14 days of the request.

Registration Template

The registration template contains specifications such as Type Name, Change Controller and Specification Document as defined in the above OAuth Access Token Types Registry section.

Initial Registry Contents

The following table shows the authorization endpoint response type registry containing the initial contents.

Sr.No.	Parameter Name	Change Controller	Specification Document
1	code	IETF	RFC 6749
2	token	IETF	RFC 6749

OAuth Extensions Error Registry

This can be used to define OAuth Extensions Error Registry. The error codes along with protocol extensions such as grant types, token types, etc. are registered by experts with the required specification. If they are satisfied with the registration, then they will publish the specification. The registration request will be sent to the @ietf.org for reviewing with subject ("Request for error code: example"). Experts will either reject or accept the request within 14 days of the request.

Registration Template

The registration template contains specifications such as Change Controller and Specification Document as defined in the above OAuth Access Token Types Registry section, except the following specifications −

1. **Error Name** − It is the name of the request.
2. **Error Usage Location** − It specifies the location of the error such as authorization code grant error response, implicit grant response or token error response, etc, which specifies where the error can be used.
3. **Related Protocol Extension** − You can use protocol extensions such as extension grant type, access token type, extension parameter, etc.