tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

OAuth2.0 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.

Audience

This tutorial is designed for software programmers who would like to understand the concepts of OAuth. This tutorial will give you enough understanding on OAuth from where you can take yourself to higher levels of expertise.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of authorization and authentication of a basic client server application model.

Copyright & Disclaimer

Table of Contents

1. OAUTH 2.0 – OVERVIEW

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?

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

Features of OAuth 2.0

- OAuth 2.0 is a simple protocol that allows to access resources of the user without sharing passwords.
- It provides user agent flows for running clients application using a scripting language, such as JavaScript. Typically, a browser is a user agent.
- 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

- 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.
- OAuth 2.0 relies on SSL which is used to ensure cryptography industry protocols and are being used to keep the data safe.
- It allows limited access to the user's data and allows accessing when authorization tokens expire.
- It has ability to share data for users without having to release personal information.
- It is easier to implement and provides stronger authentication.

Disadvantages of OAuth 2.0

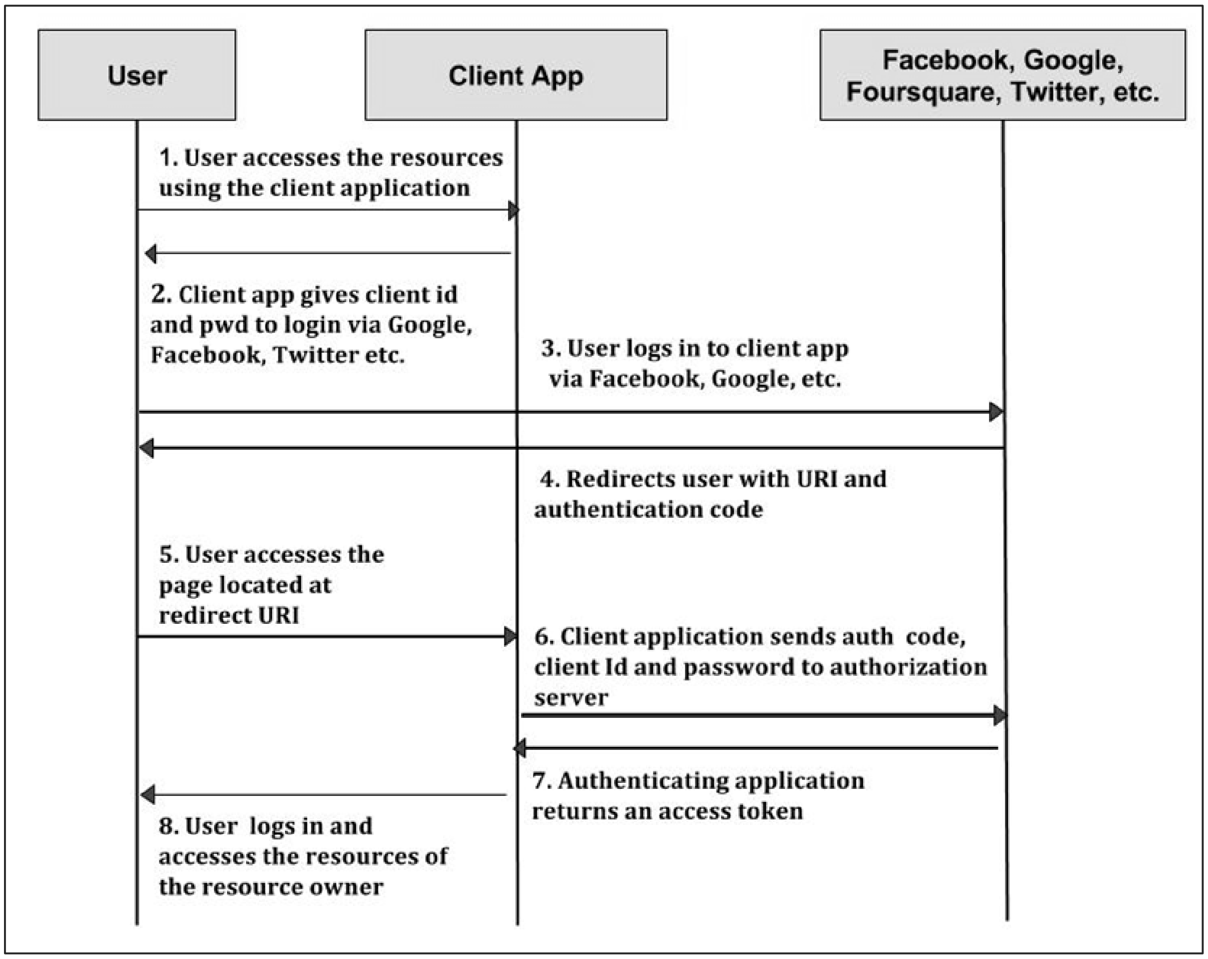
- If you are adding more extension at the ends in the specification, it will produce a wide range of non-interoperable implementations, which means you have to write separate pieces of code for Facebook, Google, etc.

- 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.

For information about OAuth 2.0 diagram and some various concepts, refer this link.

2. OAUTH 2.0 – ARCHITECTURE

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


User
Client App
Facebook, Google, Foursquare, Twitter, etc.
1. User accesses the resources using the client application
2. Client app gives client id and pwd to login via Google, Facebook, Twitter etc.
3. User logs in to client app via Facebook, Google, etc.
4. Redirects user with URI and authentication code
5. User accesses the page located at redirect URI
6. Client application sends auth code, client Id and password to authorization server
7. Authenticating application returns an access token
8. User logs in and accesses the resources of the resource owner

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.

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

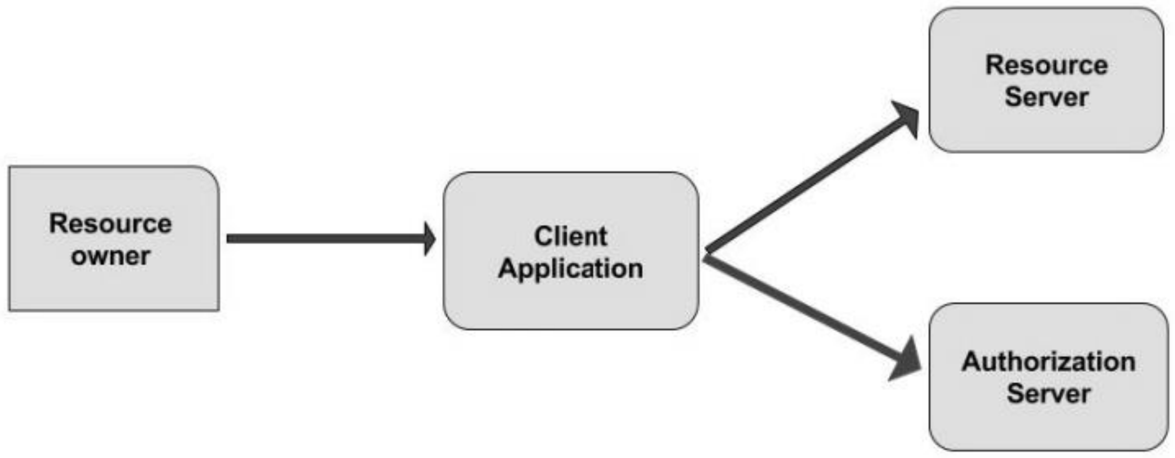
Delegated authorization is the process of giving one's credentials to other user to perform some actions on behalf of that user.

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.

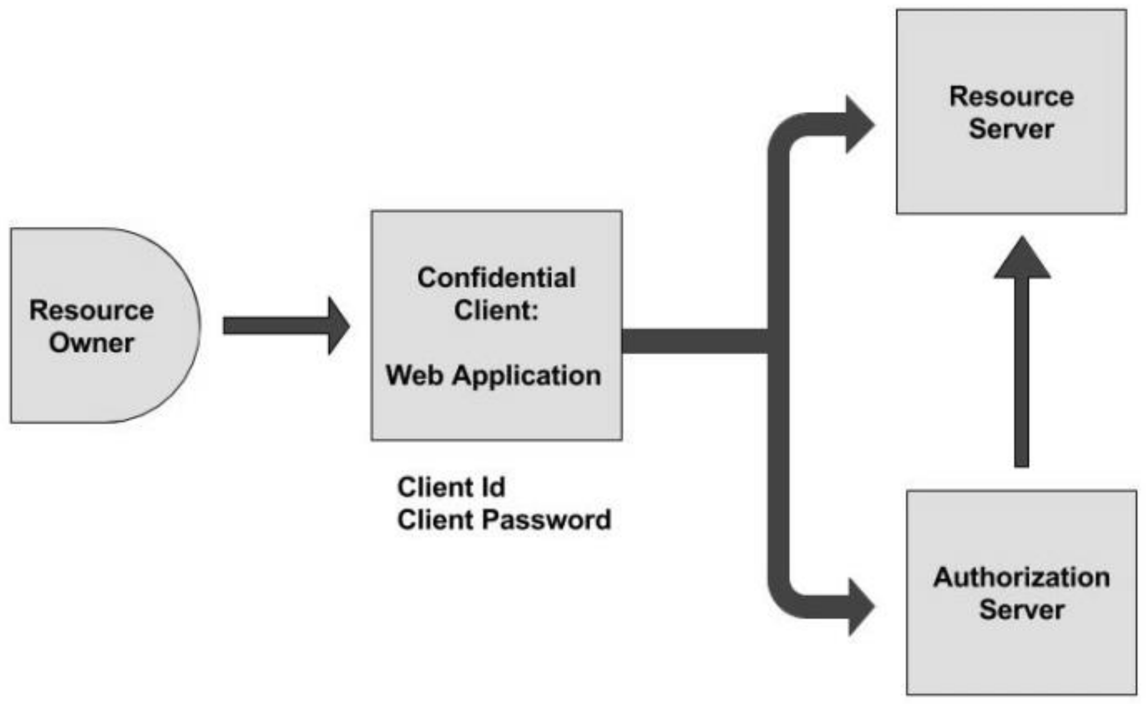
- **Resource Server**: Resource server is API server that can be used to access the user's information. It has the capability of accepting and responding to protected resource requests with the help of access tokens.

- **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.

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.

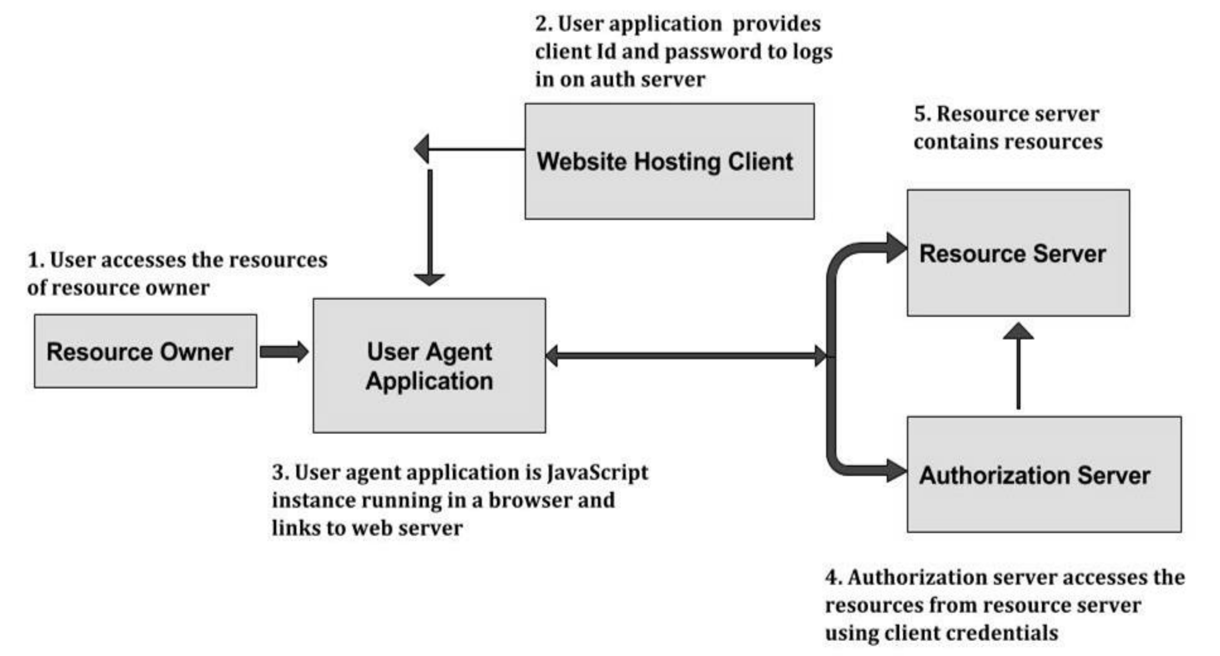
- The resource server is a server, which hosts the resources such as Facebook, Twitter, Google, etc. These resources are stored on the resource server and are accessed by the client application and the resource owner owns these resources.

- The resources of the resource owner are then accessed by the authorization server using confidential client web application.

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.


2. User application provides client Id and password to logs in on auth server
5. Resource server contains resources
Website Hosting Client
1. User accesses the resources of resource owner
Resource Server
Resource Owner
User Agent Application
Authorization Server
3. User agent application is JavaScript instance running in a browser and links to web server
4. Authorization server accesses the resources from resource server using client credentials

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.

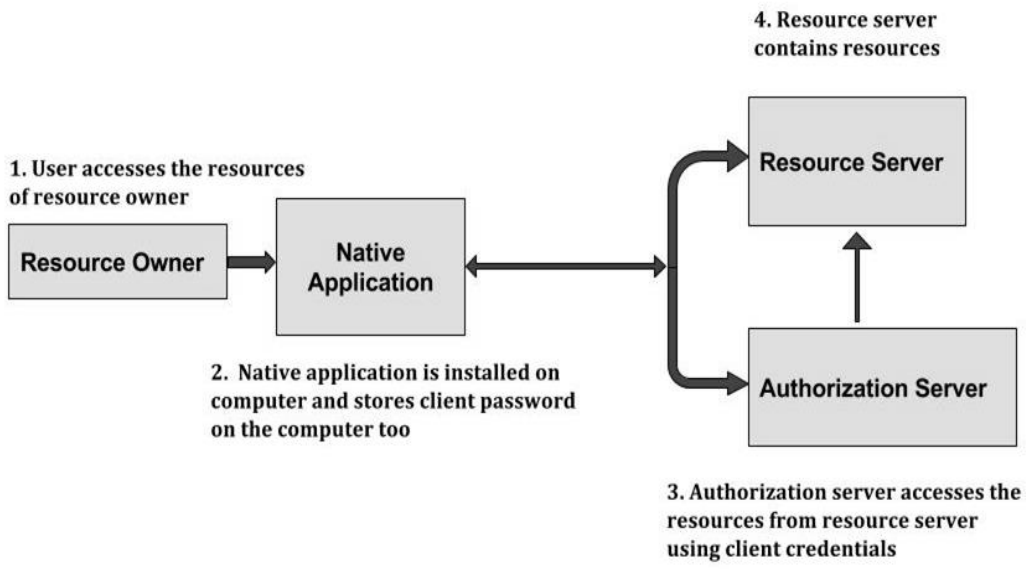
Step 5: The resource server contains the resources, which are owned by the resource owner.

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:


4. Resource server contains resources
Resource Server
1. User accesses the resources of resource owner
Resource Owner
Native Application
Authorization Server
2. Native application is installed on computer and stores client password on the computer too
3. Authorization server accesses the resources from resource server using client credentials

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.

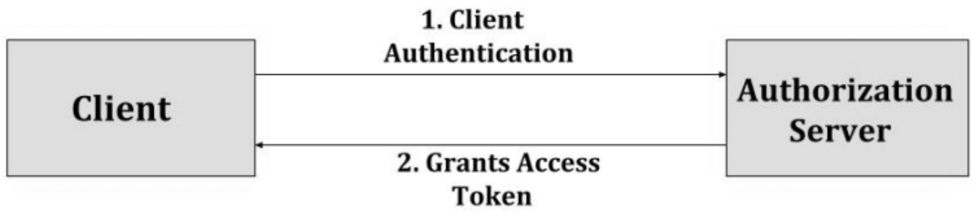
Step 4: The resource server contains the resources, which are owned by the resource owner.

3. 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.

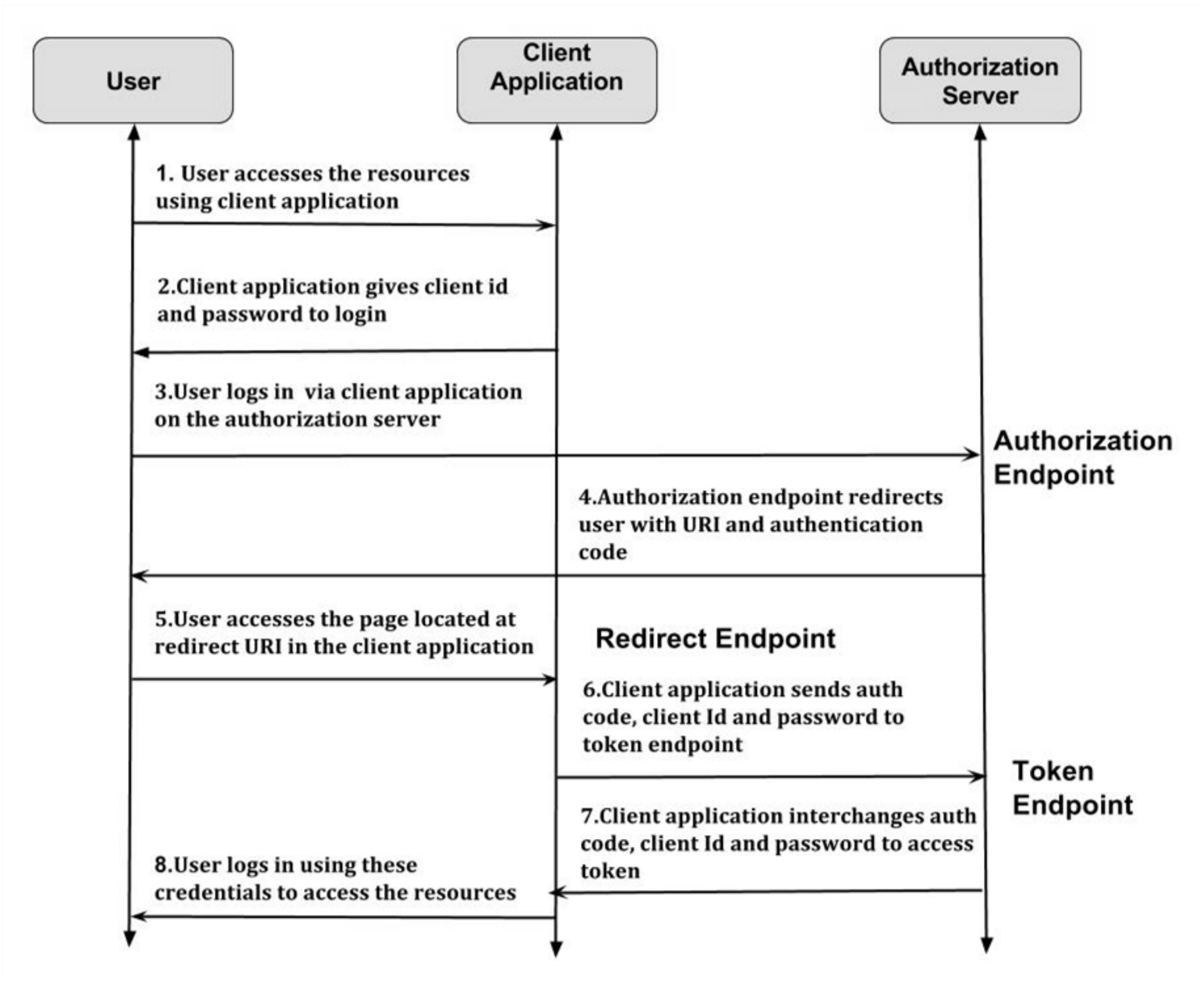
Sr. No.	Concept & Description
1	**Obtaining End-User Authorization** The authorization end point is typically URI on the authorization server in which the resource owner logs in and permits to access the data to the client application.
2	**Authorization Response** The authorization response can be used 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.

Obtaining End-User Authorization

The authorization end points are the URL's which makes an authentication request on the authorization server, in which the resource owner logs in and permits to access the data to the client application. For instance, address of JSP page, PHP page, etc.

The authorization end user can be described as shown in the following diagram.


User
Client Application
Authorization Server
1. User accesses the resources using client application
2.Client application gives client id and password to login
3.User logs in via client application on the authorization server
Authorization Endpoint
4.Authorization endpoint redirects user with URI and authentication code
5.User accesses the page located at redirect URI in the client application
Redirect Endpoint
6.Client application sends auth code, client Id and password to token endpoint
Token Endpoint
7.Client application interchanges auth code, client Id and password to access token
8.User logs in using these credentials to access the resources

The authorization endpoint can be defined in three ways:

- Authorization Endpoint
- Redirect Endpoint
- Token Endpoint

Authorization Endpoint

- Authorization endpoint can be used to interact with the resource owner who permits the authorization to access the resource of the resource owner.

- First, the user accesses the resources of the resource owner by using the client application. The client application will be provided with the client id and client password during registering the redirect URI (Uniform Resource Identifier).

- Next, the user can login via client application on the authorization server. which contains Authorization Endpoint.

- Authorization endpoint redirects the user with URI (Uniform Resource Identifier) and authentication code to the user.

Redirect Endpoint

- The user accesses the page located at redirect URI (Uniform Resource Identifier) in the client application.

- The client application provides client id, client password and authentication code to the authorization server.

Token Endpoint

- At this point, the client application interchanges the client id, client password and authorization code to obtain an access token.

- The client application sends these credentials to the user along with the token. Once the user receives the token, it can be sent to the access resources such as Facebook, Google, etc. to access the resources in the system, related to the logged in users.

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. The access token is given by the authorization server when it accepts the client ID, client password and authorization code sent by the client application.

The authorization code will be issued by the authorization server, which allows accessing the request by using the following parameters:

- **Code**: It is a required parameter that specifies the authorization code produced by the authorization server. The lifetime of the authorization code is maximum 10 minutes and the authorization code cannot be used more than once. The authorization server rejects the request and cancels all tokens that are issued previously based on the authorization code, if the client application uses the authorization code more than once.

- **State**: It is a required parameter, if the authorization code is available in the authorization request.

The authorization server provides authorization code and grants access to the client application by using the following format:

```
"application/x-www-form-urlencoded"
```

It is the default MIME (Multipurpose Internet Mail Extensions) type of your request, which must be encoded in a such way that control names and values are escaped, space characters are replaced by the '+' sign, name/value pairs are separated from each other by '&', etc.