

SumUp

OAuth (Open Authorization) Authorization Code Flow

by: Waquas Saeed

Overview

To use SumUp APIs your application must authenticate itself with one of the following flow:

1. Authorization Code Flow + Refresh Token
 - a. Used when merchant data is required along with payment scope. Such as transaction history, subaccount. (depending on scopes enabled)
 - b. Is more secure.
2. Client Credentials Flow
 - a. Merchant data like (transaction history) cannot be done through this. This only allows merchants to pay online.

Audience:

This document will help the technical team on the partner/merchant side for detailed information on Authorization Code Flow for SumUp.

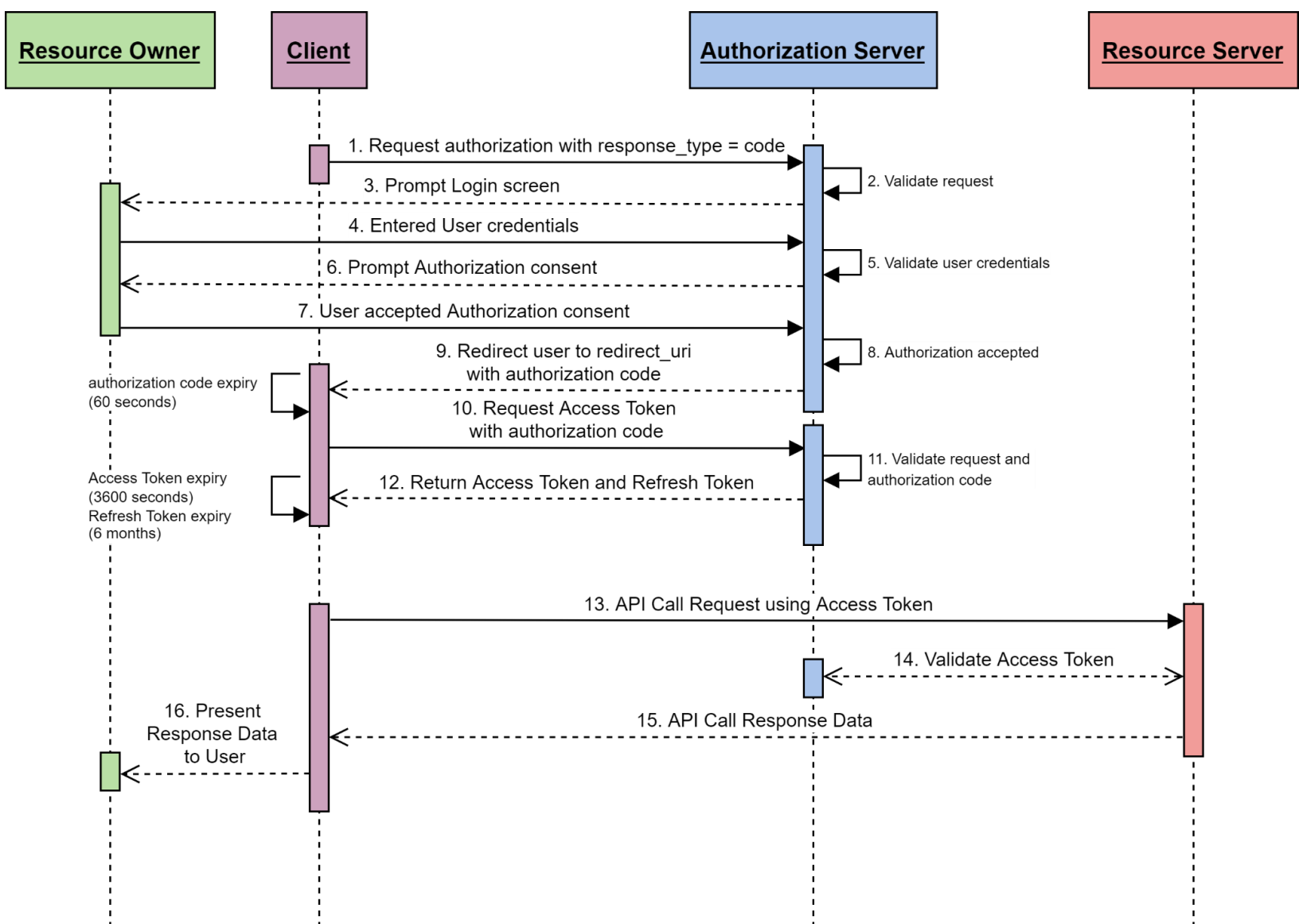
It covers the following details:

- Step by step flow with diagrams
- Sample requests and responses
- Technical details for implementation

1. Authorization Code Flow:

Following diagram shows the step by step process of authorization code flow and each step is explained in more detail below.

Diagram 1.1: Step by Step flow:



The diagram (1.1) shows the complete flow of authorization code flow and the entities involved in the process. The main entities with responsibilities are:

Entity	Responsibility
Resource Owner (Merchant User)	An entity capable of granting access to a protected resource. A merchant or end-user having SumUp credentials.
Client (Client Application)	An application making protected resource requests on behalf of the resource owner and with its authorization. An application can be executed on a server, desktop or other devices.
Authorization Server - AS (SumUp)	The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization. Tokens can be issued by the Authorization Server.
Resource Server - RS (SumUp)	The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens. API endpoints hosted by SumUp for accessing the resources.

The authorization code flow initiates in the following sequence:

[Seq 1] - the client application redirects the merchant user to the request authorization endpoint. The request authorization endpoint parameters are:

Request Authorization endpoint URL: <https://api.sumup.com/authorize>

Parameter Name	Parameter Value	Parameter Type
response_type (mandatory)	The value must be code to indicate that you expect to receive an authorization code	string
client_id (mandatory)	The Client ID of your application generated from SumUp Dashboard (OAuth - Create Client Credentials JSON file)	string
redirect_uri (mandatory)	The URI to which the merchant user is redirected after authorizing your application to access their user's account data and to which the authorization code is sent. The value must match exactly one of the registered URIs for your application	string
scope	A space-separated list of scopes for which you request authorization. If you don't specify any scopes	string

	in the request, your application will be granted authorization for the default scopes	
state	A unique local state that can be used for correlating requests and responses and for preventing cross-site request forgery	string

2. Sample Request Authorization URL:

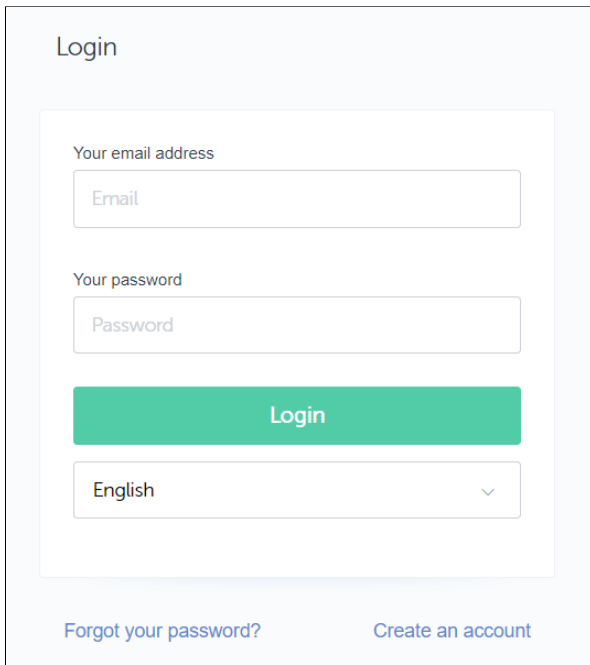
*https://api.sumup.com/authorize?
response_type=code&
client_id=fOcmczrYtYMJ7Li5GjMLLcUeC9dN&
redirect_uri=https://sample-app.example.com/callback&
scope=payments user.app-settings transactions.history user.profile_readonly&
state=2cFCsY36y95lFHk4*

[Seq 2] - the Authorization Server (AS) validates the request authorization and verifies the URL parameters sent by the client application.

The Authorization Server (AS) shall validate:

- response_type value should be equal to code. ***If code value is not defined then the client application will redirect to invalid request error by using the redirect_uri.***
- client_id should associate with merchant's user OAuth Client Credentials. ***If no client_id or incorrect client_id is defined then an empty page is shown with no further processing.***
- redirect_uri should match against the client_id being used. ***If no redirect_uri or incorrect redirect_uri is defined then an empty page is shown with no further processing.***
- scope is optional. If any authorization scopes are to be requested in the URL then the same should be enabled at SumUp end. ***If additional authorization scopes are sent in the URL which are not activated at SumUp end then the client application will redirect to the invalid scope error by using the redirect_uri with no further processing.***
- state is optional.

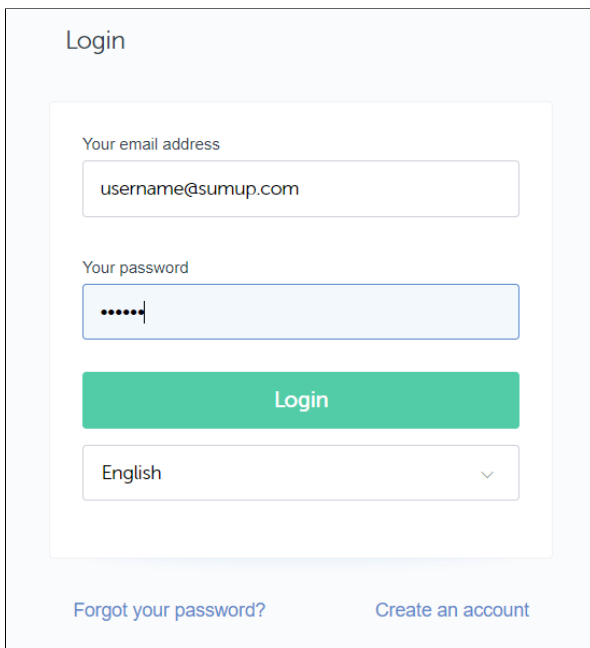
[Seq 3] - the Authorization Server (AS) displays the login screen to the merchant user if the previous request in Seq 2 is validated.



The screenshot shows a login form titled "Login". It contains the following elements:

- A label "Your email address" above a text input field containing the placeholder text "Email".
- A label "Your password" above a text input field containing the placeholder text "Password".
- A prominent green button labeled "Login".
- A dropdown menu showing "English" with a downward arrow.
- At the bottom, two links: "Forgot your password?" and "Create an account".

[Seq 4] - the merchant user enters the SumUp Credentials for authentication.

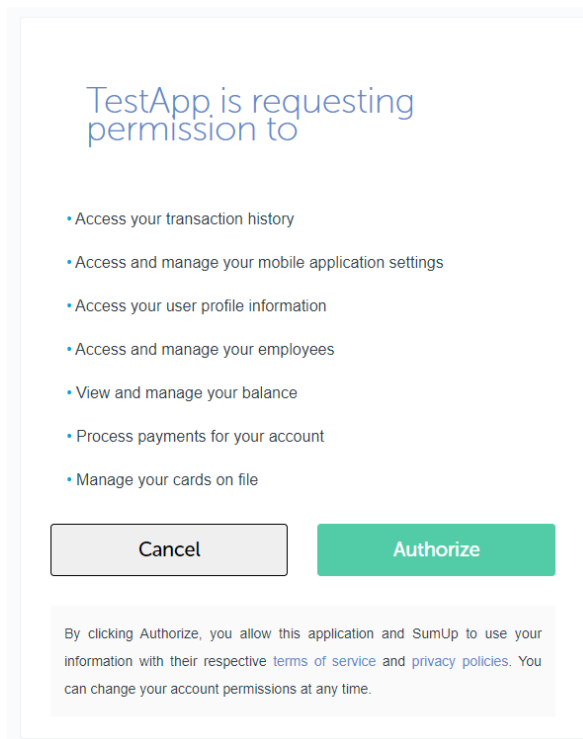


The screenshot shows the same login form as in Seq 3, but with user credentials entered:

- The email input field now contains "username@sumup.com".
- The password input field contains six dots, indicating a masked password.
- The "Login" button remains green and is still visible.
- The dropdown menu still shows "English".
- The links "Forgot your password?" and "Create an account" are still present at the bottom.

[Seq 5] - the Authorization Server (AS) validates the SumUp credentials and further processing is not being made if no credentials or incorrect credentials are provided.

[Seq 6] - the Authorization Server (AS) displays the authorization consent screen to the merchant user after validating the SumUp credentials in the Seq 5. This triggers an authorization prompt describing which application is requesting authorization from the merchant user.



[Seq 7] - the merchant user accepts the authorization consent by clicking on the Authorize button. If the merchant user denies and clicks on the Cancel button the Authorization Server (AS) will return an error parameter with an access_denied value to the redirect_uri with no further processing.

[Seq 8] - the Authorization Server (AS) prepares the redirect_uri after accepting the authorization consent from the merchant user.

[Seq 9] - the Authorization Server(AS) redirects the client application to the redirect_uri specified in the Seq 1 with additional parameters such as code and state (if included in the Seq 1). The example shows the response received after a successful authorization:

3. Sample Response from Authorization Server:

[https://sample-app.example.com/callback?
code=be366ce9fccd0c337d1da29b31d06dd1135ab95401562883&
state=2cFCsY36y95IFHk4](https://sample-app.example.com/callback?code=be366ce9fccd0c337d1da29b31d06dd1135ab95401562883&state=2cFCsY36y95IFHk4)

The code received from Authorization Server (AS) is:

- valid for only 60 seconds.
- mandatory for retrieving an access token using authorization code flow.
- invalid after 60 seconds. **The authorization code flow should be initiated from the Seq 1 when the code becomes invalid or expired.**

[Seq 10] - the client application makes a request to generate a token endpoint for retrieving tokens. The generate a token endpoint parameters are:

Token endpoint URL: <https://api.sumup.com/token>

Header should have **Content-Type: application/x-www-form-urlencoded**

Parameter Name	Parameter Value	Parameter Type
grant_type (mandatory)	The value must be authorization_code to indicate that you expect to receive tokens	string
client_id (mandatory)	The Client ID of your application generated from SumUp Dashboard (OAuth - Create Client Credentials JSON file)	string
client_secret (mandatory)	The Client Secret of your application generated from SumUp Dashboard (OAuth - Create Client Credentials JSON file)	string
code (mandatory)	The authorization code received from requesting an authorization code	string

4. Sample Generate a token Request:

```
curl -X POST \
  https://api.sumup.com/token \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  -d 'grant_type=authorization_code' \
  -d 'client_id=fOcmczrYtYMJ7Li5GjMLLcUeC9dN' \
  -d 'client_secret=717bd571b54297494cd7a79b491e8f2c1da6189c4cc2d3481380e8366eef539c' \
  -d 'code=be366ce9fccd0c337d1da29b31d06dd1135ab95401562883'
```

[Seq 11] - the Authorization Server (AS) validates the token generation request along with the authorization code being sent in the request. The authorization code has a validity of 60 seconds after which it becomes invalid or expired and no tokens can be retrieved using the token generation endpoint. ***If this condition is met, the authorization code flow should be initiated from the Seq 1.***

[Seq 12] - the Authorization Server (AS) sends a response back to the client application with tokens along with additional information returned in a JSON structure in the body of the response. The JSON structure has following data:

Parameter Name	Parameter Value	Parameter Type
access_token	The access token that you need to use in your requests to the SumUp API.	string
token_type	The type of the token. The value is always <i>Bearer</i> .	string
expires_in	The validity of the access token in seconds.	integer
refresh_token	The refresh token provided in the request call	string

5. Example JSON structure:

```
{  
  "access_token":  
  "565e2d19cef68203170ddadb952141326d14e03f4ccbd46daa079c26c910a864",  
  "token_type": "Bearer",  
  "expires_in": 3600,  
  "refresh_token":  
  "d180031bfe9bac36c336e5746637810272546865e9c9586012f462a56f3fe9af"  
}
```

The access token received from Authorization Server (AS) is:

- valid for only 3600 seconds (1 hour). The expiry describes in *expires_in* parameter of the response.
- mandatory to access SumUp REST API endpoints.
- invalid/expired after 3600 seconds and can not be used.

The refresh token received from Authorization Server (AS) is:

- valid for a period of 6 months.
- invalid/expired after 6 months and can not be used.

Grant Types:

The term grant type refers to the way an application gets an access token. The grant type also determines the method and exact sequence that are involved in the OAuth process.

Grant Type	Use Case
Authorization Code (<i>authorization_code</i>)	Used for most web and mobile application scenarios that want to call the REST web services. Uses the client application to transport the intermediate code (authorization code) with the login screen, which is then exchanged for the tokens.
Refresh Token (<i>refresh_token</i>)	Used to extend the Authorization Code grant type without the login screen. Exchanges the refresh token to issue tokens.
Client Credentials (<i>client_credentials</i>)	When a user is not involved and is being used for a service to call REST web services. Exchanges the client credentials for a token.