

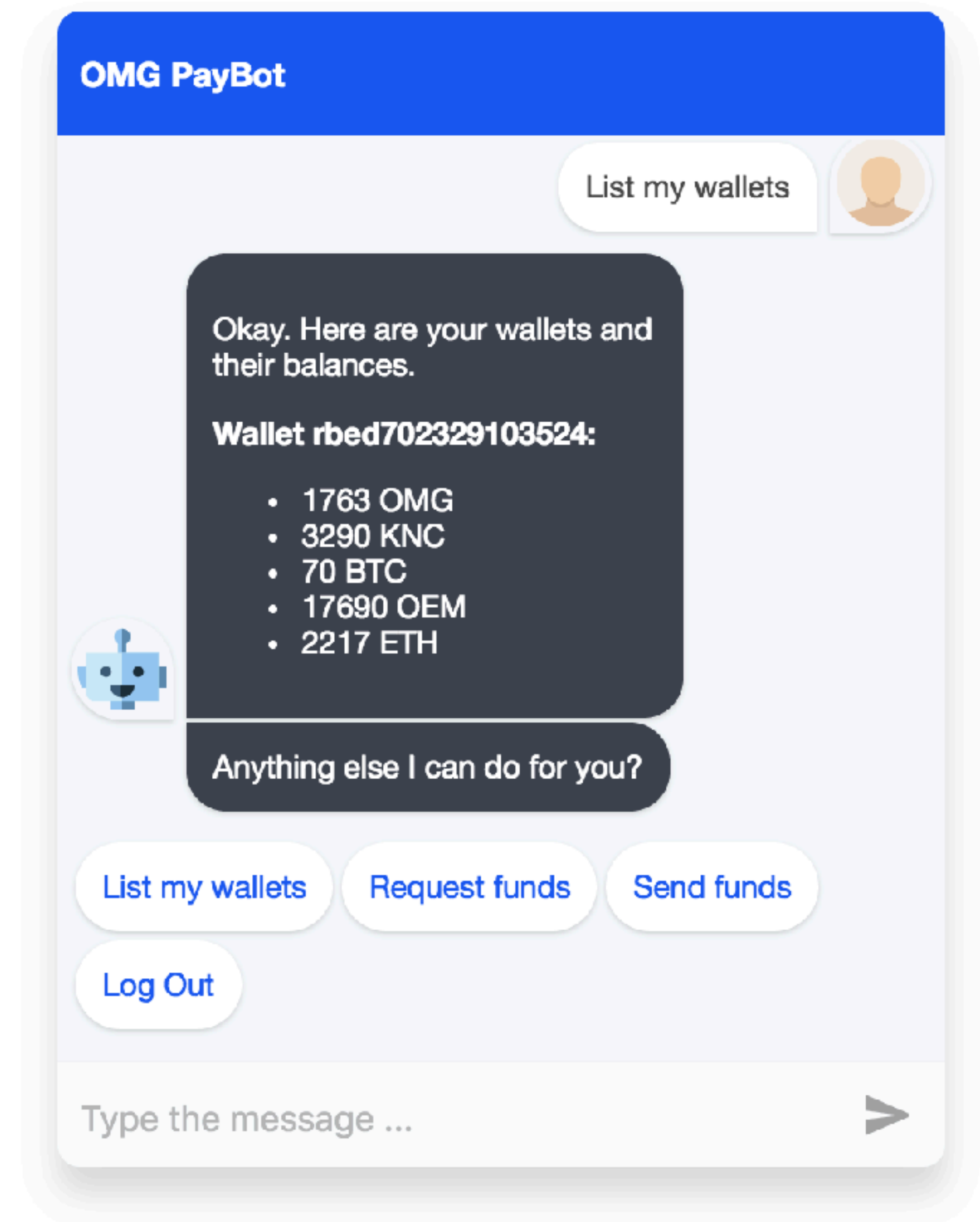


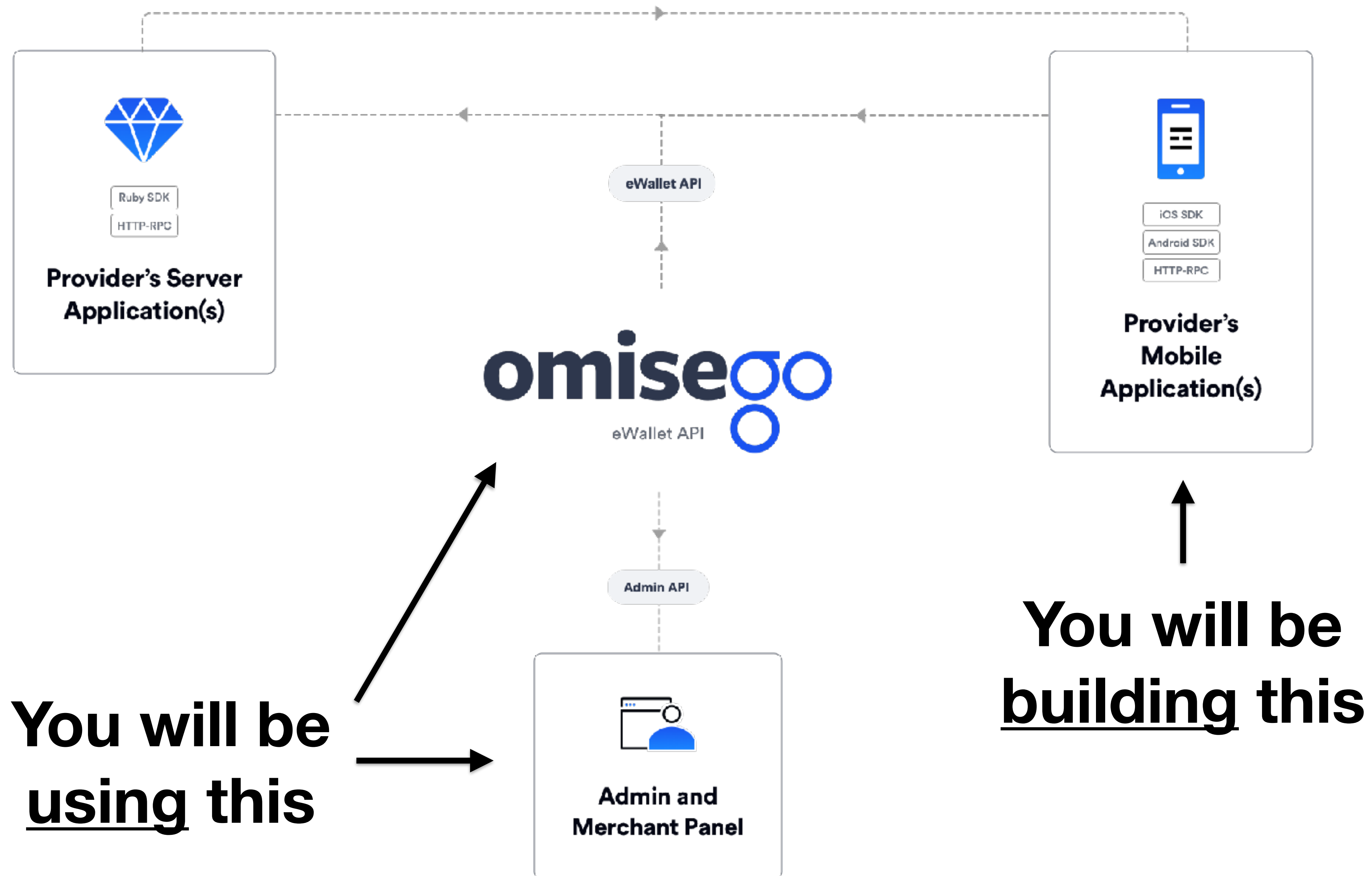
# OmiseGO eWallet Workshop

# Meet OMG PayBot

Today we'll explore...

- How to set up an eWallet server
- How to authenticate with the eWallet
  - Sign up
  - Log in
- How to interact with the eWallet
  - List wallets
  - Request funds
  - Send funds

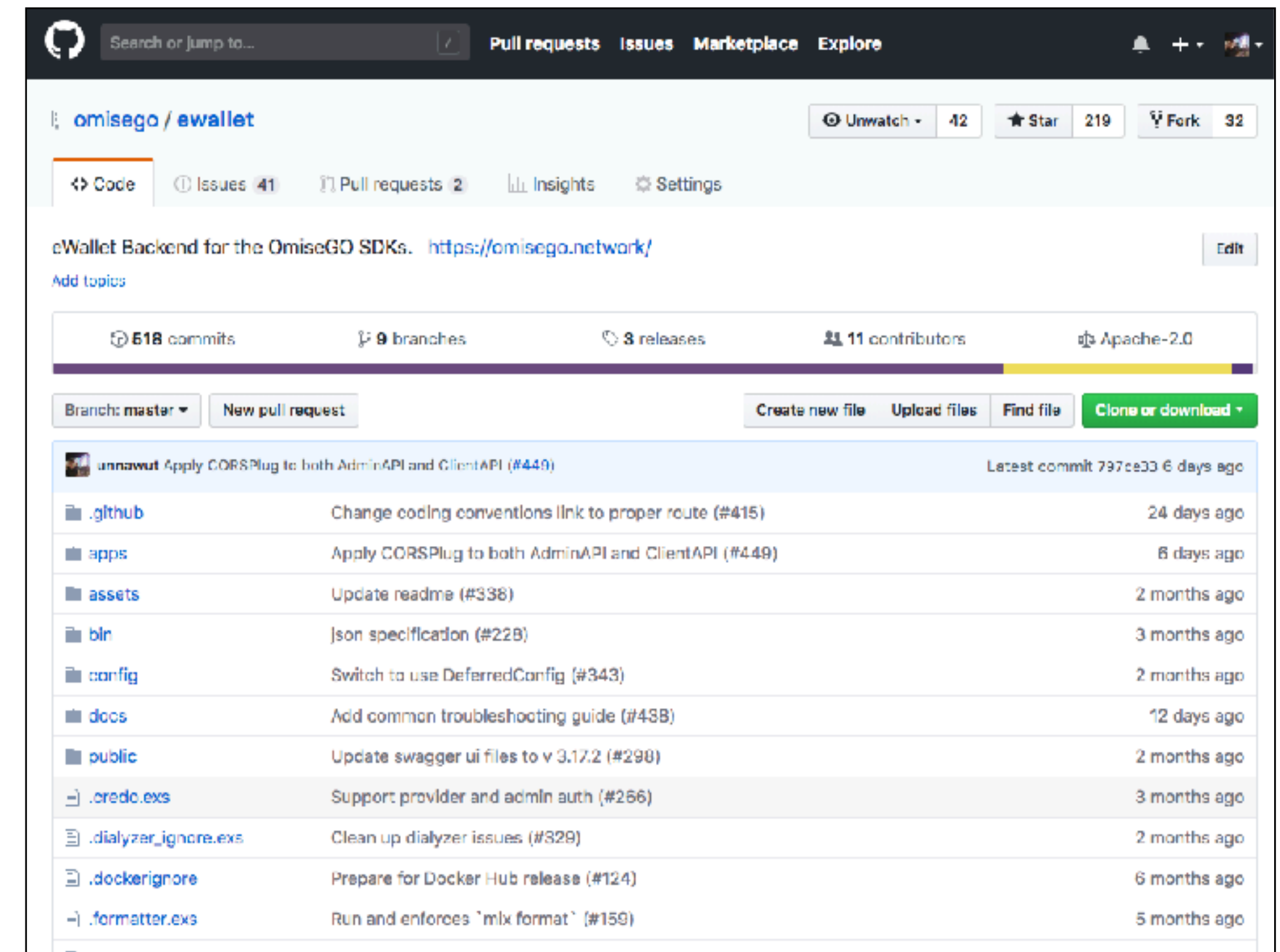




# Set up the eWallet

Grab the code: [omisego/ewallet](https://github.com/omisego/ewallet)

1. Follow the readme and...
2. Set up the app directly (install Elixir, etc.)
3. Or use our docker-compose script



# Start the eWallet

Command:

```
$ ENABLE_STANDALONE=true \  
  BASE_URL=http://localhost:4000 \  
  REDIRECT_URL_PREFIXES=http://localhost:4000 \  
  CORS_ORIGIN=http://localhost:3000 \  
  mix omg.server
```

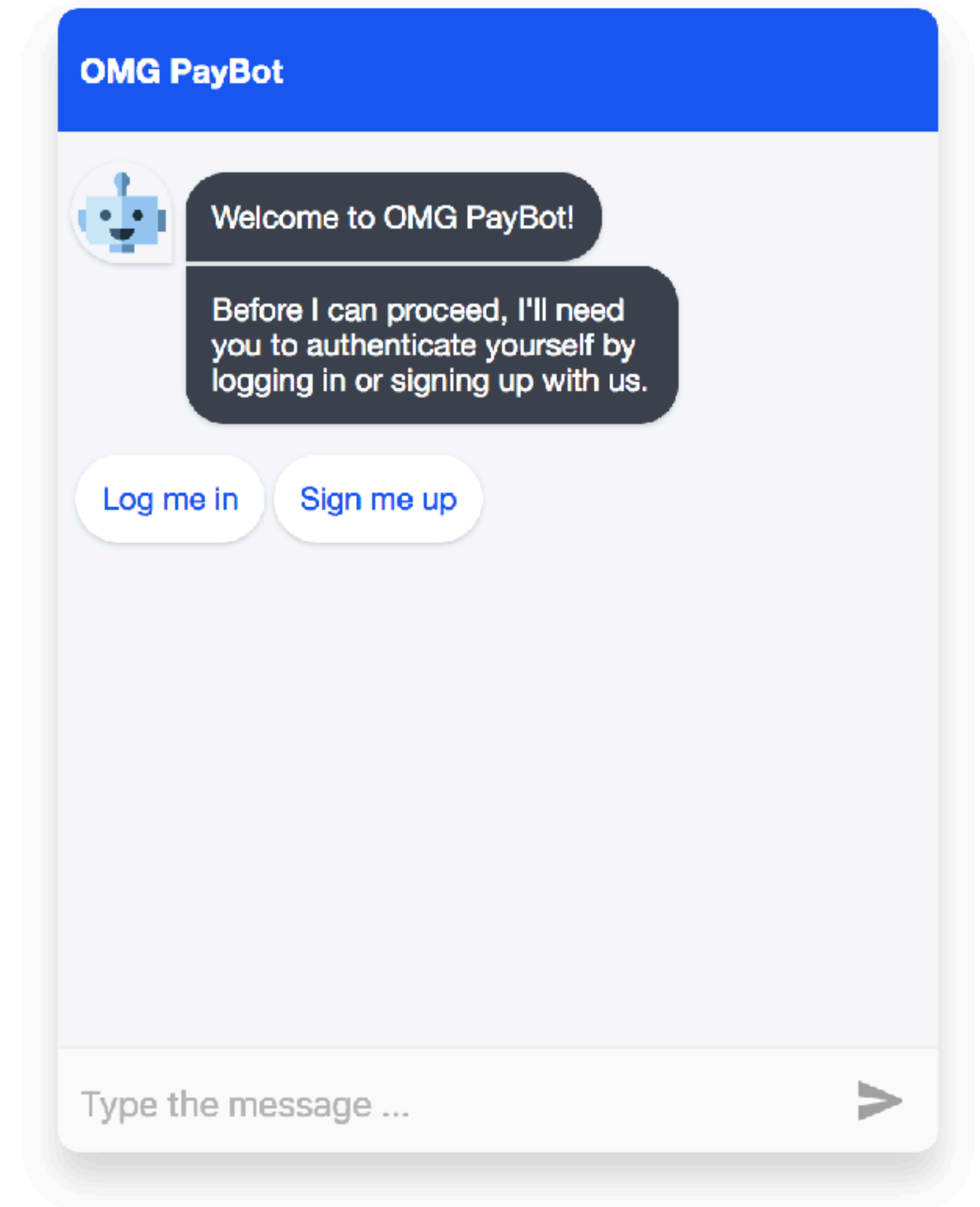
The standalone mode allows you to sign up and login via the Client API.

```
vagrant@ewallet:/vagrant$ ENABLE_STANDALONE=1 \  
> BASE_URL=http://localhost:3000 \  
> CORS_ORIGIN=http://localhost:3000 \  
> mix omg.server --no-watch  
[info] Setting up websockets dispatchers...  
[info] Running UrlDispatcher.Plug with Cowboy http on port 4000  
█
```

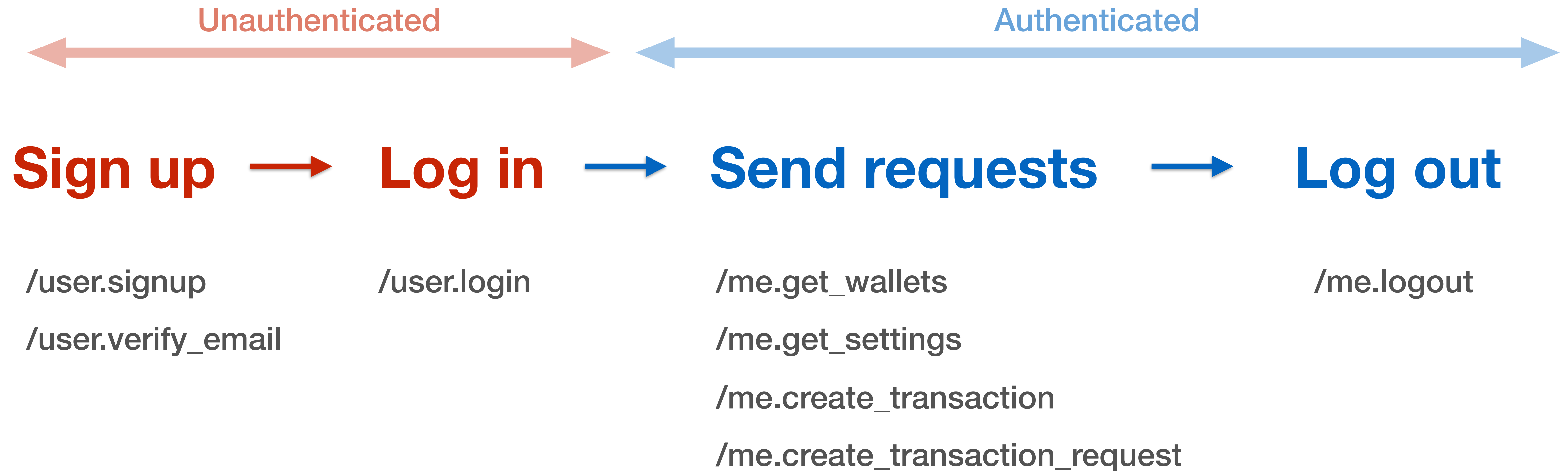
# Set up the OMG PayBot

Grab the code: [omisego/omg-paybot-exercise](https://github.com/omisego/omg-paybot-exercise)

1. Install nodejs & yarn
2. Install dependencies: `yarn install`
3. Run `yarn start`



# User flow





# Behind an unauthenticated request ...

## ▼ General

Request URL: http://10.5.10.10:4000/api/client/user.login

Request Method: POST

Status Code: 🟢 200 OK

Remote Address: 10.5.10.10:4000

Referrer Policy: no-referrer-when-downgrade

## ► Response Headers (9)

## ▼ Request Headers

⚠ Provisional headers are shown

Accept: application/vnd.omisego.v1+json

Content-Type: application/json; charset=utf-8

Origin: http://localhost:3000

Referer: http://localhost:3000/

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_13\_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36

## ▼ Request Payload [view source](#)

```
▼ {email: "email@example.com", password: "password"}  
  email: "email@example.com"  
  password: "password"
```

The following are required to make an unauthenticated request:

- **Accept**
- **Content-Type**

**Request data must be in json**



# Behind an authenticated request ...

## ▼ General

**Request URL:** http://10.5.10.10:4000/api/client/me.get\_wallets

**Request Method:** POST

**Status Code:** 🟢 200 OK

**Remote Address:** 10.5.10.10:4000

**Referrer Policy:** no-referrer-when-downgrade

## ▶ Response Headers (9)

## ▼ Request Headers

⚠ Provisional headers are shown

**Accept:** application/vnd.omisego.v1+json

**Authorization:** OMGClient RkxtTG1iRnd4bTFPbVZrWkZNUFMwakFnM1Z6NlZpdXhQWllWMloxWUs2dzppWlR0RUJGWnhLMnFrM3AzaW1xU3NSeHdMbGxVMEZDNnF1Sk1ZZGhKc29z

**Content-Type:** application/json; charset=utf-8

**Origin:** http://localhost:3000

**Referer:** http://localhost:3000/

**User-Agent:** Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_13\_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36

The following are required to make an authenticated request:

- **Accept**
- **Authorization**
- **Content-Type**



# Behind an authenticated request ...

## ▼ General

**Request URL:** http://10.5.10.10:4000/api/client/me.get\_wallets

**Request Method:** POST

**Status Code:** 🟢 200 OK

**Remote Address:** 10.5.10.10:4000

**Referrer Policy:** no-referrer-when-downgrade

## ▶ Response Headers (9)

## ▼ Request Headers

⚠ Provisional headers are shown

**Accept:** application/vnd.omisego.v1+json

**Authorization:** OMGClient RkxtTG1iRnd4bTFPbVZrWkZNUFMwakFnM1Z6NlZpdXhQWllWMloxWUs2dzppWlR0RUJGWnhLMnFrM3AzaW1xU3NSeHdMbGxVMEZDNnF1Sk1ZZGhKc29z

**Content-Type:** application/json; charset=utf-8

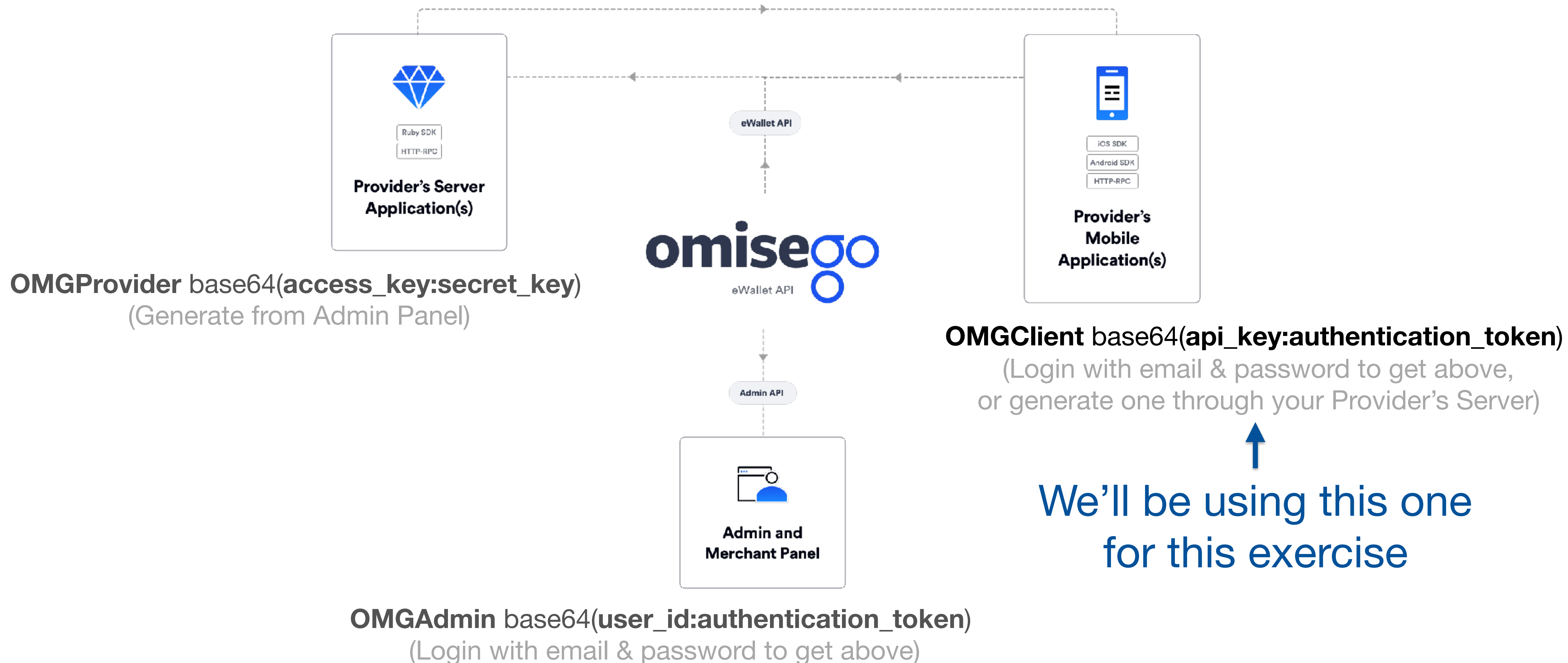
**Origin:** http://localhost:3000

**Referer:** http://localhost:3000/

**User-Agent:** Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_13\_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/68.0.3440.106 Safari/537.36

But how do you get this encoded  
Authorization header?

# What goes in the Authorization header?



# PayBot Demo!

# Sign up

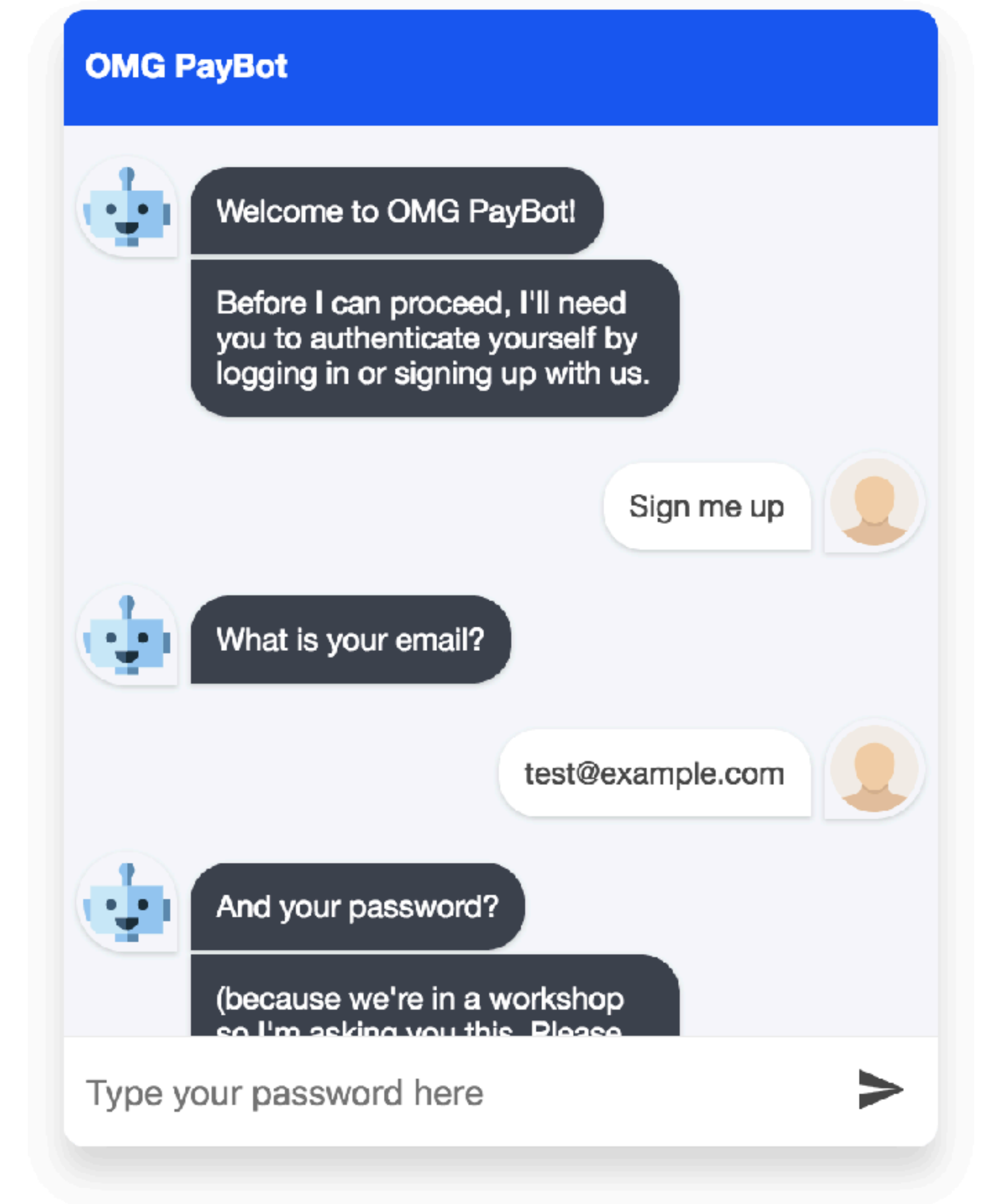
**POST /api/client/user.sign\_up**

## Header:

- **Accept:** application/vnd.omisego.v1+json
- **Content-Type:** application/json; charset=utf-8

## Body:

- **email** (string)
- **password** (string)
- **password\_confirmation** (string)





# Login

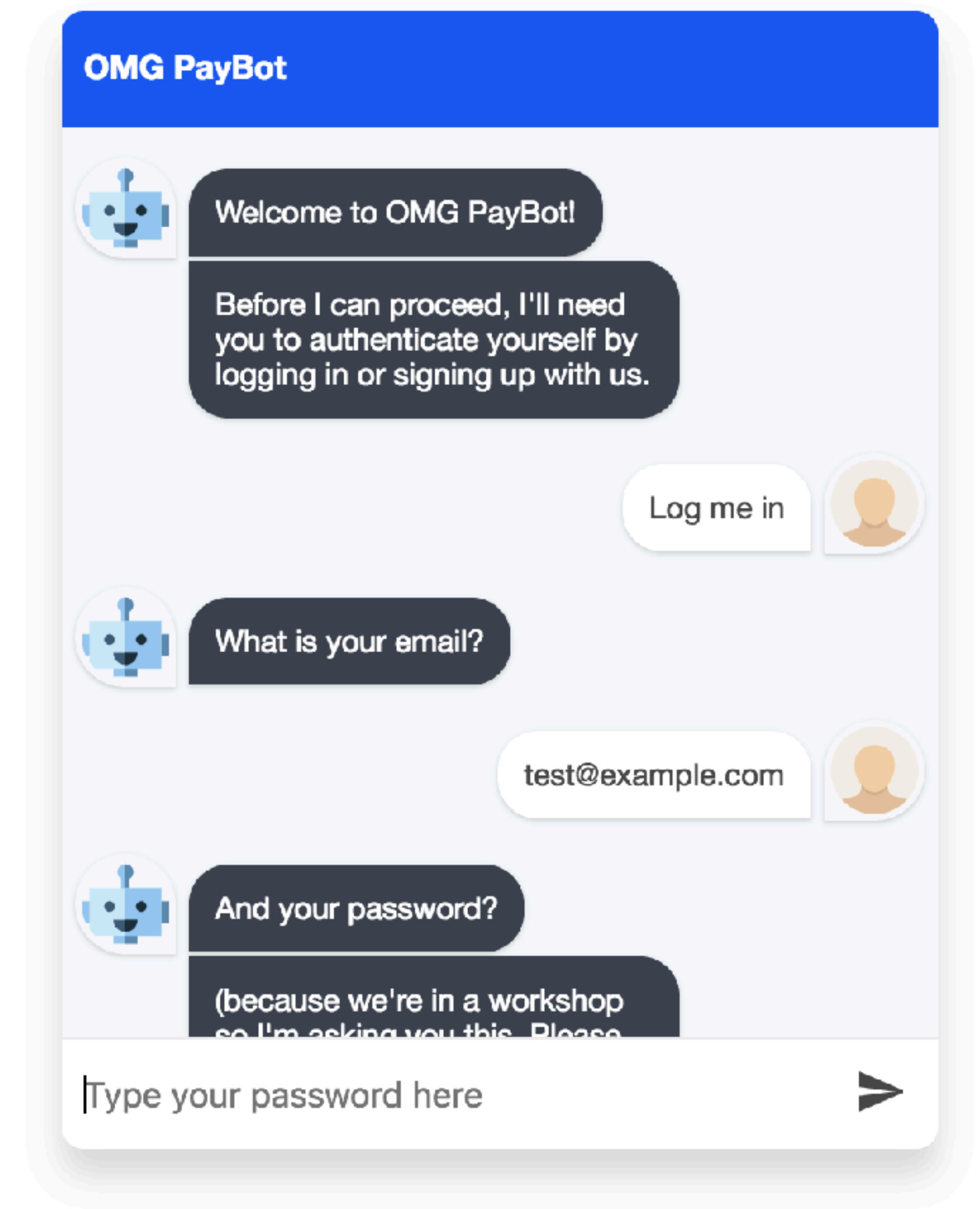
**POST /api/client/user.login**

## Header:

- **Accept:** application/vnd.omisego.v1+json
- **Content-Type:** application/json; charset=utf-8

## Body:

- **email** (string)
- **password** (string)





# List the user's wallets

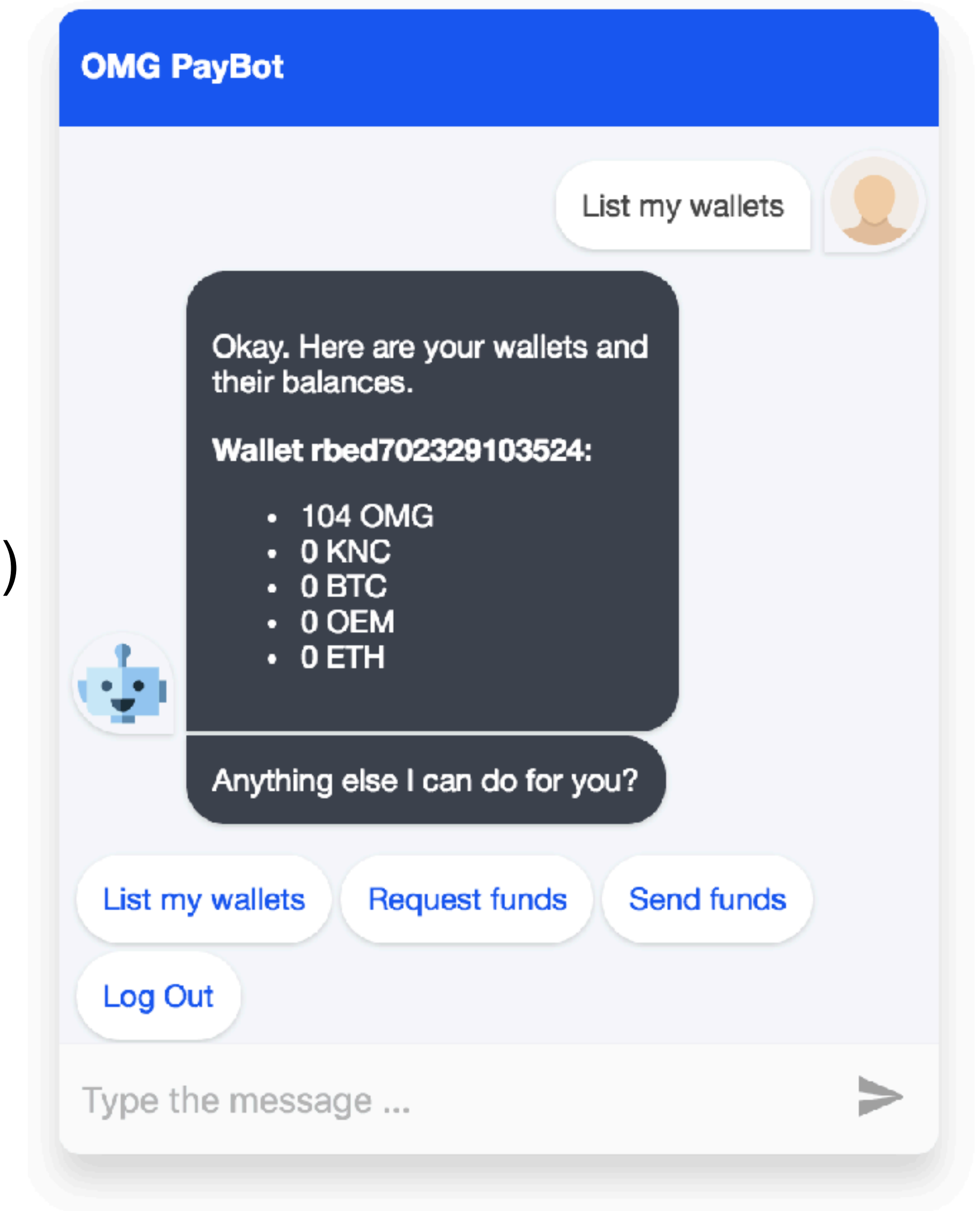
**POST /api/client/me.get\_wallets**

## Header:

- **Accept:** application/vnd.omisego.v1+json
- **Authorization:** OMGClient base64(api\_key:auth\_token)
- **Content-Type:** application/json; charset=utf-8

## Body:

- (none)



# Transaction Request Flow

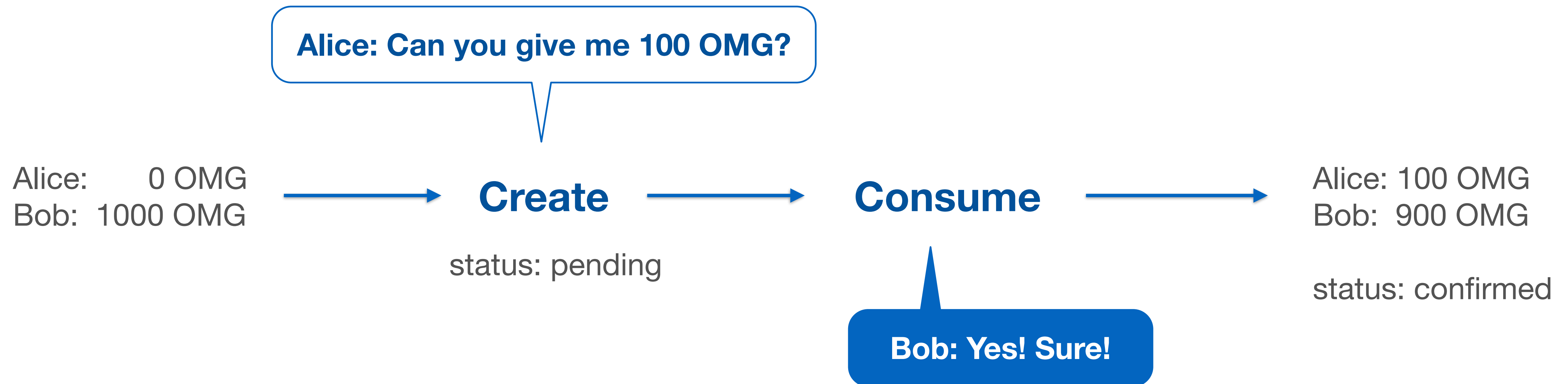
- You don't have any funds in your wallets yet
- Funds cannot be made out of thin air
- (well... the provider can, but not your user)
- So you have to request funds from the provider
- This is done through Transaction Request Flow

# Transaction Request Flow



# Transaction Request Flow

Create & consume



# Transaction Request Flow

Create & consume

Alice: Can you give me 100 OMG?

Alice: 0 OMG  
Bob: 1000 OMG

**Create**

status: pending

**Consume**

Alice: 50 OMG  
Bob: 950 OMG

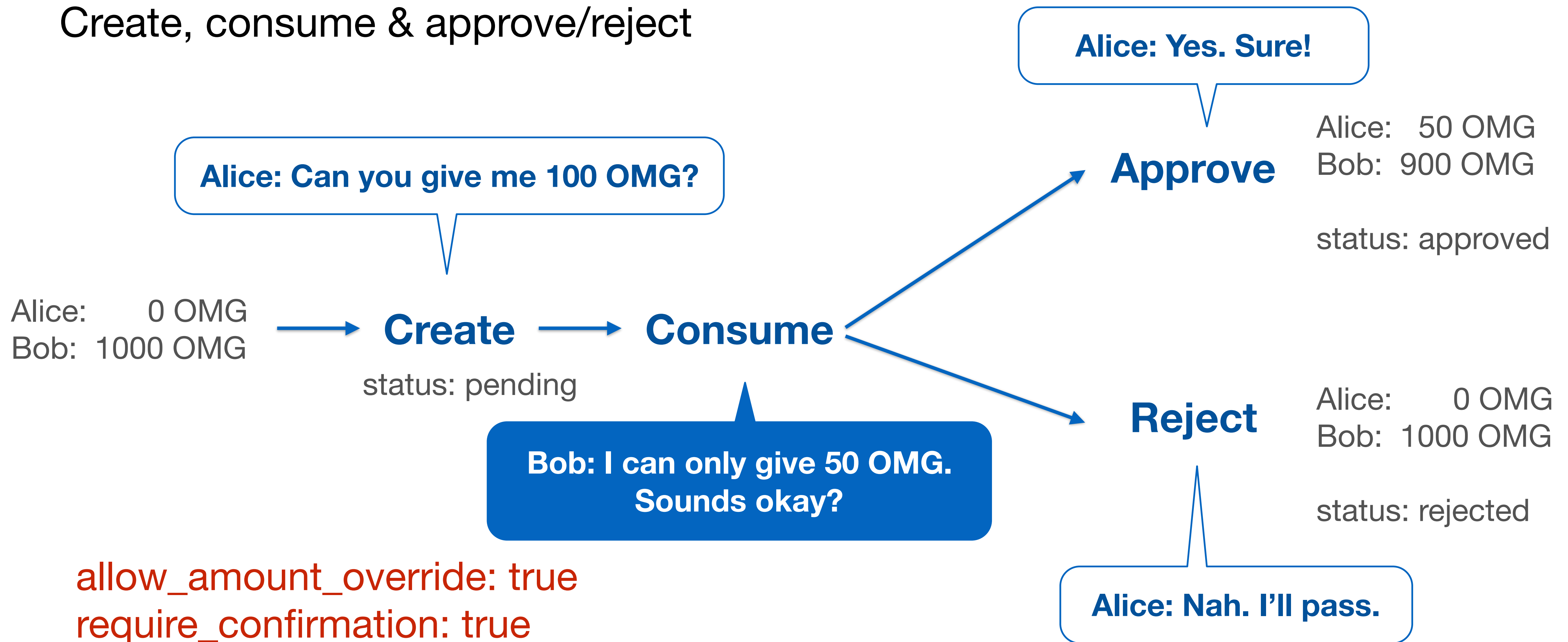
status: confirmed

Bob: I'm giving you  
only 50 OMG.

allow\_amount\_override: true  
require\_confirmation: false

# Transaction Request Flow

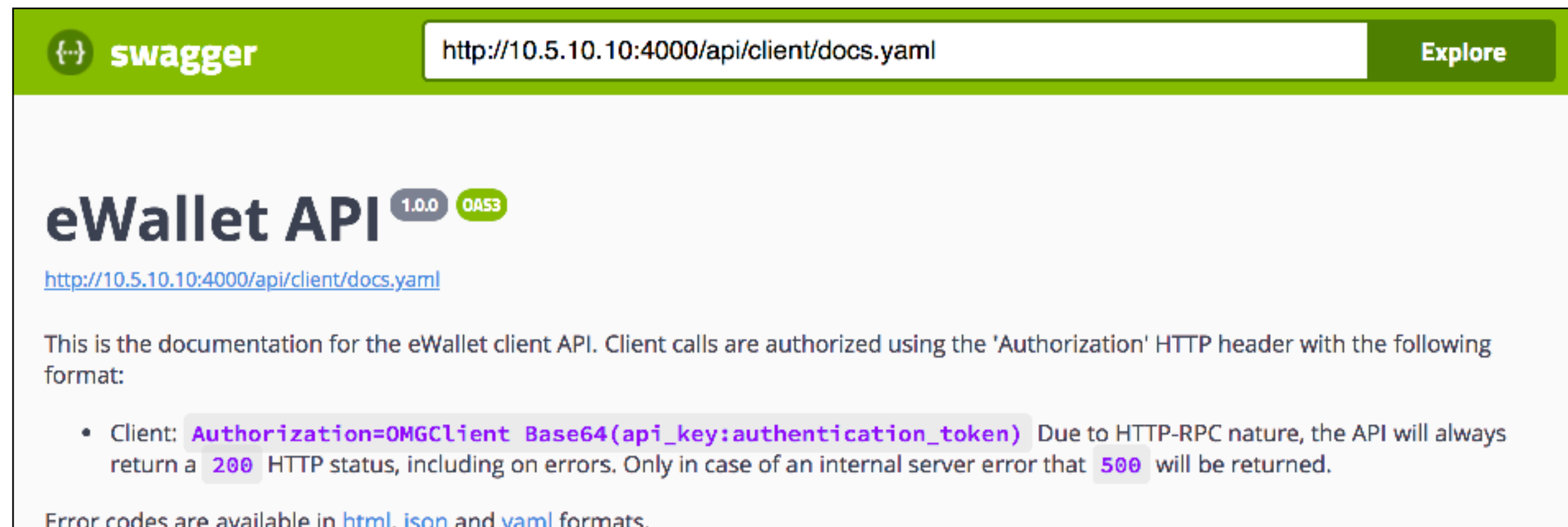
Create, consume & approve/reject





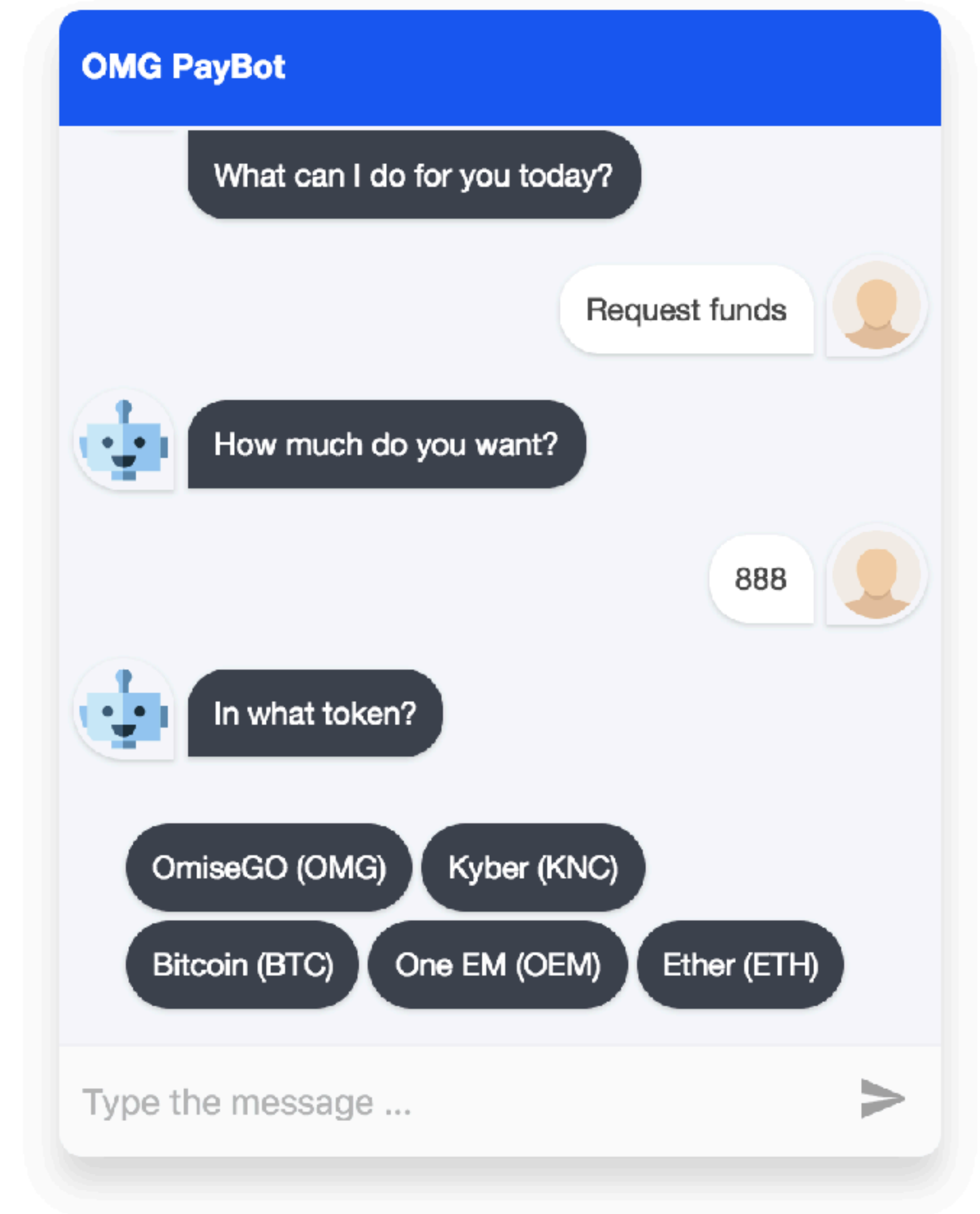
# Request funds

Can you find out how to request funds with our API docs?



The image shows the Swagger UI for the eWallet API. The top bar is green with the Swagger logo and the URL `http://10.5.10.10:4000/api/client/docs.yaml`. Below the bar, the title "eWallet API" is displayed with version "1.0.0" and a "0A53" tag. The description states: "This is the documentation for the eWallet client API. Client calls are authorized using the 'Authorization' HTTP header with the following format:" followed by a list item: "Client: `Authorization=OMGClient Base64(api_key:authentication_token)` Due to HTTP-RPC nature, the API will always return a `200` HTTP status, including on errors. Only in case of an internal server error that `500` will be returned." At the bottom, it says "Error codes are available in [html](#), [json](#) and [yaml](#) formats."

<http://localhost:4000/api/client/docs.ui>



Approve your request via the Admin Panel at <http://localhost:4000/admin>

Approve your request via the Admin Panel at <http://localhost:4000/admin>

The screenshot displays the 'Transaction Requests' interface. On the left is a dark sidebar with navigation links: MA master\_account, Switch Account >, DASHBOARD, ACCOUNT, TOKEN, WALLET, TRANSACTION, REQUEST, CONSUMPTION, and API. The main area shows a table of transaction requests with columns for REQUEST ID and TYPE. The requests are as follows:

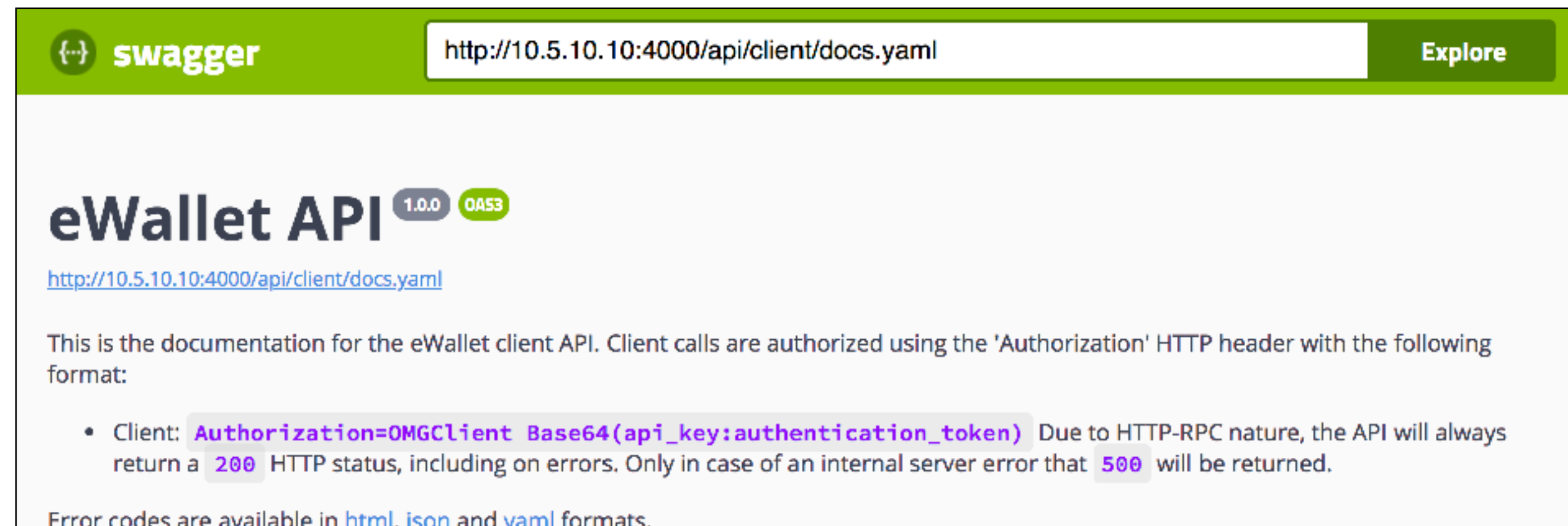
REQUEST ID	TYPE
txr_01cpz57ebqyfh63d9w5h79fdqq	Receive
txr_01cph5cm6pcej7x6jt0w5acbs	Send
txr_01cph5ck68wt5s73hw5vnb4k4	Send
txr_01cph5ck0ewtx941gjjhnt7e2	Receive
txr_01cph5cj060ca02sm216wj5ta	Receive
txr_01cph5chssksx8yzdc37276yh	Send
txr_01cph5chmk74e2edazd4xx5fq	Receive
txr_01cph5chdww8krdg3jrkehqas	Send
txr_01cph5ch6qrzp4k8etn5jkiv3	Receive
txr_01cpz57ebqyfh63d9w5h79fdqq	Receive

A blue arrow points from the table to a detailed view of a 'Request to receive OMG'. This view includes a QR code, a 'Wallet' dropdown set to 'acc\_0x0000000000000000', and fields for 'Type : receive', 'Amount : 5', and 'Token : OmiseGO (OMG)'. A 'Consume' button is present. Below this, 'ADDITIONAL REQUEST DETAILS' are listed: 'Type : receive', 'Token: OmiseGO', 'Amount : 5 OMG', 'Requester Address : rbed702329103524', 'Account ID : -', and 'Account Name : -'.

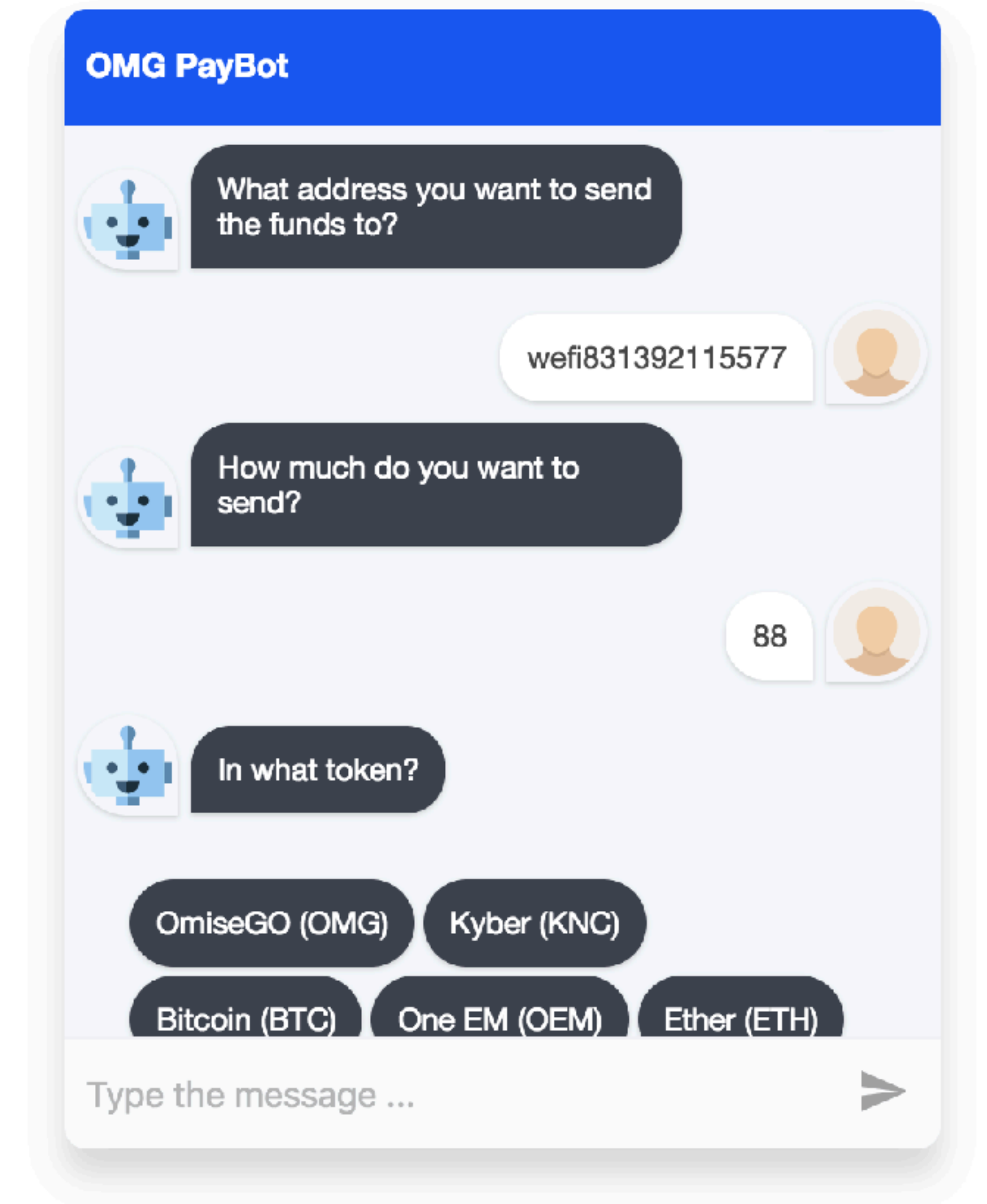
Check your wallet balances again. The tokens you requested have arrived!

# Send funds

Can you find out how to send funds with our API docs?



<http://localhost:4000/api/client/docs.ui>



# Let's get started!

# Time's up!



# The OmiseGO eWallet in Summary

- An easy way to build a ledger system for loyalty points, gaming points, or any other quantifiable values.
- Once available, it will allow a seamless transition to the blockchain world.
- Can be used along side your existing systems, or as a standalone solution.





# Full API documentations

**Admin API:** `/api/admin/docs`

Example: <https://ewallet.staging.omise.go.io/api/admin/docs>

**Client API:** `/api/client/docs`

Example: <https://ewallet.staging.omise.go.io/api/client/docs>

# Available SDKs...

Server (Ruby): <https://github.com/omisego/ruby-sdk>

Client (iOS): <https://github.com/omisego/ios-sdk>

Client (Android): <https://github.com/omisego/android-sdk>

# The eWallet Team and where to find them...

- Report issues: <https://github.com/omisego/ewallet/issues>
- Chat: <https://gitter.im/omisego/ewallet>
- StackOverflow: <https://stackoverflow.com/questions/tagged/omisego>





**Thank You!**