FLOWE OPEN BANKING INTERFACE

Developer’s guide v2.0.0

Contents

1. Introduction

Flowe exposes Berlin Group v1.3 compliant Open Banking APIs for Account Information Services (AIS) and Payment Initiation Services (PIS) TPP in two environments.

Table 1 lists API and IDP URIs as well as scopes for SANDBOX and PRODUCTION environments. Note that scopes contain both AISP and PISP scopes. Remember to remove AISP or PISP scope in case the role requested is PISP or AISP only.

Environment		
Sandbox	Open Banking API URI	https://tp-pp.flowe.com
Sandbox	IDP URI	https://bm0p0tier4.b2clogin.com/bm0p0tier4.onmicrosoft.com
Sandbox	Developer Portal URI	https://tp-portal-pp.flowe.com/
Sandbox	Scopes string	offline_access https://bm0p0tier4.onmicrosoft.com/9d957d7c-9550-487d-b127-4d146e618846/user_impersonation https://bm0p0tier4.onmicrosoft.com/9d957d7c-9550-487d-b127-4d146e618846/aisp https://bm0p0tier4.onmicrosoft.com/9d957d7c-9550-487d-b127-4d146e618846/pisp
Production	Open Banking API URI	https://tp.flowe.com/
Production	IDP URI	https://floweidp.b2clogin.com/2e7da2ff-0e8f-41bc-83c3-b5933ad2445c
Production	Developer Portal URI	https://tp-portal.flowe.com/
Production	Scopes String	offline_access https://floweidp.onmicrosoft.com/8cf3af01-3882-4ad4-b92a-759078c0bedd/user_impersonation https://floweidp.onmicrosoft.com/8cf3af01-3882-4ad4-b92a-759078c0bedd/aisp https://floweidp.onmicrosoft.com/8cf3af01-3882-4ad4-b92a-759078c0bedd/pisp

Table 1

2. TPP Onboarding

The onboarding procedure description is available at https://www.flowe.com/open-banking-api.

As the result of the onboarding procedure the TPP will receive at least the following parameters:

- <SUBSCRIPTION_KEY>: you need to click the link in the registration email received to access the API portal. There, in the Profile section, you can find the subscription keys
- <CLIENT_ID>
- <CLIENT_SECRET>
- <CLIENT_SECRET_EXPIRATION_DATE>: TPPs are strongly recommended to request a new client secret long before its expiration. For this purpose, contact openbanking at flowe dot com using the contact email provided during the onboarding process.

3. TPP Authentication and Authorization

The authentication and authorization approach applied by Flowe to the TPPs is the **decoupled SCA approach** with **OAuth2 as a pre-step**.

That basically means that the TPP client should obtain an access token from the Flowe IdP, either by redirecting PSU User-Agent to the Flowe IdP login page or redeeming a new one given the previously received refresh token. Contextually the TPP may submit a consent request or a payment request. Those should be approved by the PSU using the Flowe App.

A detailed description on the steps above follows below.

The OAuth2 pre-step is done with the **OAuth2 Authorization Code Grant**, where the **OAuth2 Client** is the TPP backend, which is confidential. That means that the <CLIENT_SECRET> should never be shared with the TPP Frontend (webapp/native app) but kept on a TPP server (or in a key vault accessed only by the backend). Same process applies for access tokens and refresh tokens retrieved during the authentication process.
OAuth2 Authorization Code Grant variants such as OAuth2 Authorization Code Grant with PKCE, that allows a webapp or native app to directly get an access token without passing thought a backend, are not allowed.

Another parameter that must be kept confidential is the <SUBSCRIPTION_KEY>.

TPP Frontend starts the OAuth2 flow redirecting the PSU User-Agent or opening a Web View to the Flowe IDP login form to allow the user to input their credentials, as explained later.

Any violation of the policy above will result in Flowe blacklisting the TPP.

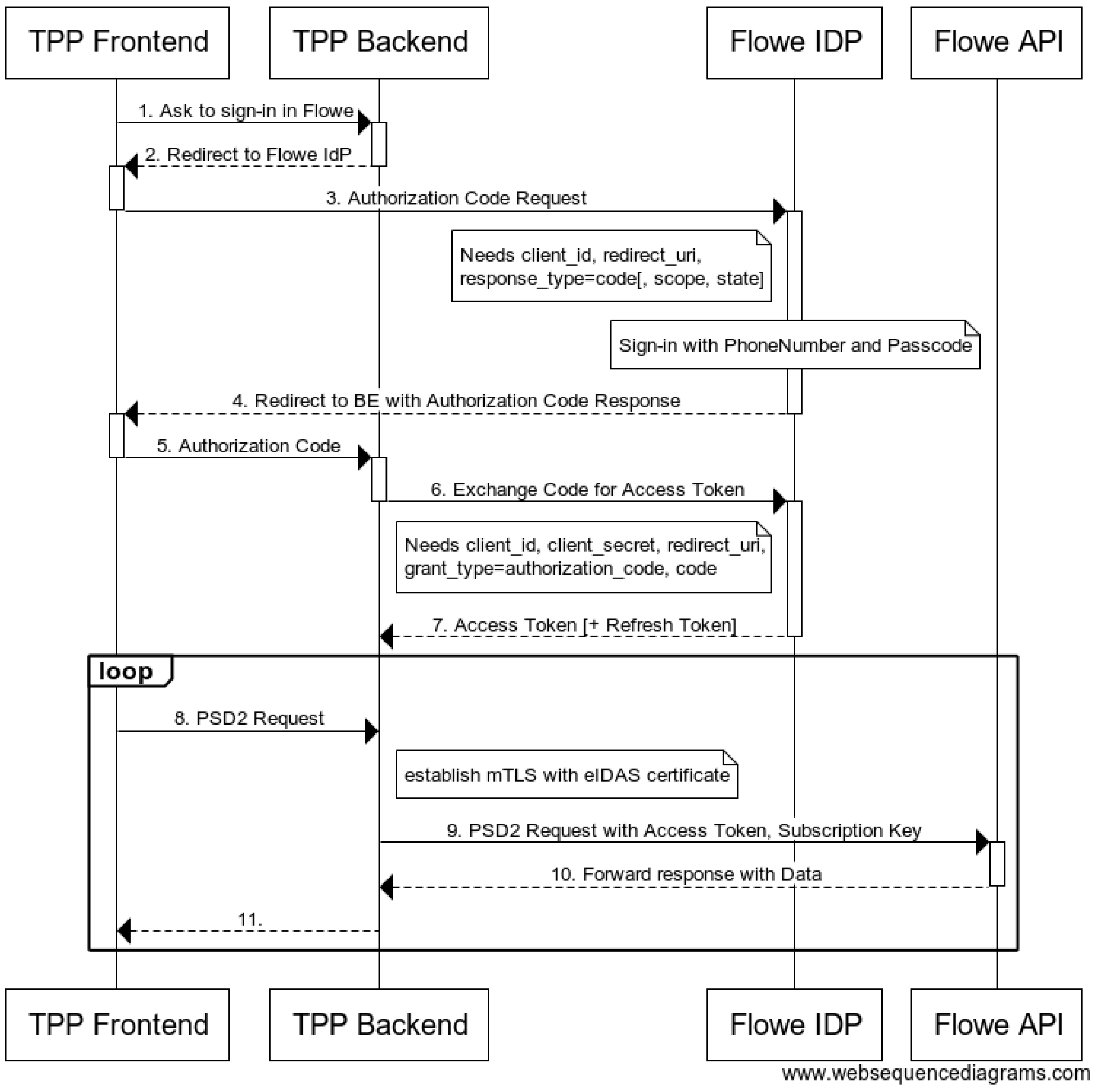
TPP Frontend
TPP Backend
Flowe IDP
Flowe API
1. Ask to sign-in in Flowe
2. Redirect to Flowe IdP
3. Authorization Code Request
Needs client_id, redirect_uri, response_type=code[, scope, state]
Sign-in with PhoneNumber and Passcode
4. Redirect to BE with Authorization Code Response
5. Authorization Code
6. Exchange Code for Access Token
Needs client_id, client_secret, redirect_uri, grant_type=authorization_code, code
7. Access Token [+ Refresh Token]
loop
8. PSD2 Request
establish mTLS with eIDAS certificate
9. PSD2 Request with Access Token, Subscription Key
10. Forward response with Data
11.
www.websequencediagrams.com

Figure 1

TPP Frontend can be a webapp hosted in the PSU browser or a native app installed on a PSU device.

For simplicity, we describe the case in which the Frontend is a webapp, but the instructions can be easily adapted to the case in which the TPP Frontend is a native app, which should use a web view in the interaction with the Flowe IDP.

Below are described the steps shown in Figure 1:

1. PSU asks the TPP to sign-in into Flowe

2. The TPP redirects the PSU to the following URL[1][2]:

   ```
   <IDP_TARGET_ENVIRONMENT_BASE_URI>/oauth2/v2.0/authorize?
   p=B2C_1A_Psd2_SignIn&
   client_id=<CLIENT_ID>&
   state=<STATE>&
   redirect_uri=<REDIRECT_URI>&
   scope=<SCOPES_STRING>&
   response_type=code&
   prompt=login
   ```

   Where <STATE> is a random string generated from and stored in the frontend[3].
3. The PSU insert their credentials
4. IDP redirects User-Agent to the following URL:

   ```
   <REDIRECT_URI>?state=<STATE>&code=<AUTHORIZATION_CODE_GRANT>
   ```

5. TPP frontend checks that <STATE> is equal to the one provided in step 2. If so, it extracts the <AUTHORIZATION_CODE_GRANT> from the URL and sends it to the TPP backend.
6. TPP backend performs the following HTTPS call[4] providing the TPP eIDAS QWAC certificate as the client certificate in the mutual TLS handshake.

   ```
   POST /oauth2/v2.0/token?p=B2C_1A_Psd2_SignIn HTTP/1.1
   Host: <IDP_TARGET_ENVIRONMENT_BASE_URI>
   Content-Type: application/x-www-form-urlencoded
   grant_type=authorization_code&
   scope=<SCOPE_STRING>&
   client_id=<CLIENT_ID>&
   client_secret=<CLIENT_SECRET>&
   code=<AUTHORIZATION_CODE_GRANT>
   ```

7. In case of success, Flowe IDP will return a response with status 200 and, in the application/json formatted payload, TPP backend extracts the following properties:
   - <ACCESS_TOKEN> in "`access_token`" property
   - <REFRESH_TOKEN> in "`refresh_token`" property

Steps after the 7 are not directly tied to TPP Authentication and Authorization but they are shown to explain how requests to Open Banking APIs should be performed and which path data exchange takes.

8. TPP frontend sends a request to TPP backend to perform an Open Banking API request
9. TPP Backend performs the corresponding request to Flowe Open Banking APIs with mutual. Additionally, the request must contain the following headers:
   - Authorization: Bearer <ACCESS_TOKEN>
   - P0-APIKey: <SUBSCRIPTION_KEY>
10. The Open Banking API responds. Note that the validity of the <ACCESS_TOKEN> is limited in time. After the <ACCESS_TOKEN> expiration the API will respond with HTTP status code 401. Without the PSU interaction, the TPP can request another access token using the

[1] To increase readability each query string parameter is shown in a separated line.
[2] Placeholders must be URL encoded.
[3] To understand what the purpose of the OAuth2 state parameter check https://auth0.com/docs/protocols/oauth2/oauth-state.
[4] To Increase readability each payload argument is shown in a separated line.

<REFRESH_TOKEN> with the following HTTPS request[4].

```
POST /oauth2/v2.0/token?p=B2C_1A_Psd2_SignIn HTTP/1.1
Host: <IDP_TARGET_ENVIRONMENT_BASE_URI>
Content-Type: application/x-www-form-urlencoded
grant_type=refresh_token&
scope=<SCOPES_STRING>&
client_id=<CLIENT_ID>
client_secret=<CLIENT_SECRET>
refresh_token=<REFRESH_TOKEN>
```

The response contains a new <ACCESS_TOKEN> and another <REFRESH_TOKEN>.
The TPP could use the new access token, but it is not useful to use the new refresh token since, although all refresh tokens have a 90 days lifetime, 90 days after step 3 the Identity Provider requires to restart the authentication process from step 2, refusing old refresh tokens.

The PSU data always passes through the TPP backend and never goes directly from the TPP Frontend to the Flowe Open Banking APIs.

4. Consume AIS/PIS APIs

TPP backend can consume AIS/PIS Open Banking APIs, which are described in the developer portal and downloadable in the form of OpenAPI (previously known as Swagger) specification.

Flowe Open Banking APIs SCA Approach is the *Decoupled SCA Approach with Implicit Start of Authorization (with oAuth2 pre-step already shown)*.
The details on this flow are described in the Berlin Group PSD2 Standard v1.3.

4.1 AIS

In the following scenario, the TPP acts like an AISP.

1. The TPP backend perform a consent request

```
POST /openbankingais/v1/consents HTTP/1.1
Host: <PSD2_TARGET_ENVIRONMENT_BASE_URI>
Content-Type: application/json
P0-APIKey: <SUBSCRIPTION_KEY>
Authorization: Bearer <ACCESS_TOKEN>
{
   "access": {
        "availableAccounts": "allAccounts",
        "allPsd2": "allAccounts"
    },
    "recurringIndicator": true,
    "validUntil": "2020-08-31",
    "frequencyPerDay": 4,
    "combinedServiceIndicator": false
}
```

2. Note that Flowe adopts the Global Consent approach, that is the TPP requests access to all the PSU entities (accounts, balances, transactions) with one consent request. The PSU can accept all or nothing.
   Additionally, Flowe adopts the following policy over the parameters above:

   a. “recurringIndicator” must always be true. In case you need “One-Shot” consent just set “frequencyPerDay”: 1 and “validUntil: “<TODAY_UTC>”, where <TODAY_UTC> is a placeholder representing the current day in UTC
   b. “validUntil”: cannot be greater than <TODAY_UTC> + 90 days
   c. “frequencyPerDay”: cannot be greater than 4.
3. If TPP is authenticated and authorized, the response will be the following

```
HTTP Status Code 200
Consent-Type application/json

{
    "consentId": "AAACT20143X9S9GJ6M",
    "_links": {
        "status": {
            "href": "/v1/consents/AAACT20143X9S9GJ6M/status"
        }
    },
    "consentStatus": "received"
}
```

4. TPP should show a message to the PSU asking them to approve the consent request that the PSU device will receive as a push notification (and, as a fallback, as a to do list item in the Flowe app main page). Flowe chose to not populate the “psuMessage” header to let the TPPs choose the language given the PSU preferences in their application.
5. TPP can now start polling with a reasonable frequency (no more than 1/sec) the consent status API.

```
GET /openbankingais/v1/consents/AAACT20143X9S9GJ6M/status HTTP/1.1
Host: <PSD2_TARGET_ENVIRONMENT_BASE_URI>
P0-APIKey: <SUBSCRIPTION_KEY>
Authorization: Bearer <ACCESS_TOKEN>
```

   Before any customer action, the response will be the following

```
HTTP Status Code 200
Content-Type: application/json

{
    "consentStatus": "received"
}
```

6. A push notification is sent to the PSU device.
7. The PSU accepts the TPP requests with SCA.
8. The consent status request will return the consent status “valid”. If otherwise, the PSU refused the consent request, the consent status will be “rejected”.
9. If the PSU accepted the consent request, the TPP can now perform AISP API calls, such as querying the list of accounts of the PSU.

```
GET /openbankingais/v1/accounts HTTP/1.1
Host: <PSD2_TARGET_ENVIRONMENT_BASE_URI>
Consent-Id: AAACT20143X9S9GJ6M
P0-APIKey: <SUBSCRIPTION_KEY>
Authorization: Bearer <ACCESS_TOKEN>
```

Note that the Consent-ID cannot be used more than “frequencyPerDay” parameter in 24 hours, specified during consent request, for **unattended requests**. If the consent usage exceeds that limit, server responds with Http Status Code 429.
To indicate that an API request is **attended** the **PSU-IpAddress** header must be populated with the PSU user agent IP address.

4.2 PIS

Currently, Flowe support “payments” as Payment-Service and “sepa-credit-transfer” as Payment-Product.

In the following scenario, the TPP acts like a PISP.

1. The TPP backend submits a payment request

```
POST /openbankingpis/v1/payments/sepa-credit-transfers HTTP/1.1
Host: <PSD2_TARGET_ENVIRONMENT_BASE_URI>
Content-Type: application/json
P0-APIKey: <SUBSCRIPTION_KEY>
Authorization: Bearer <ACCESS_TOKEN>
{
    "debtorAccount": {
        "iban": "<DEBTOR_IBAN>"
    },
    "creditorAccount": {
        "iban": "<CREDITOR_IBAN>"
    },
    "creditorName": "<CREDITOR_NAME>",
    "instructedAmount": {
        "amount": "<AMOUNT>",
        "currency": "EUR"
    },
    "chargeBearer": "DEBT",
    "remittanceInformationUnstructured": "<REMITTANCE_INFORMATION>"
}
```

Note that only IBAN is allowed as an account reference.

2. If the input fields are correct, the response will be the following

```
HTTP Status Code 201
Content-Type: application/json

{
  "transactionStatus": "RCVD",
  "_links": {
    "scaStatus": {
      "href": "/v1/payments/sepa-credit-
transfers/PI20213149S1068Z/authorisations/PI20213149S1068Z-AUTH-1"
    },
    "self": {
      "href": "/v1/payments/sepa-credit-transfers/"
    },
    "status": {
      "href": "/v1/payments/sepa-credit-transfers/PI20213149S1068Z/status"
    }
  },
  "paymentId": "<PAYMENT_ID>",
  "transactionFeeIndicator": "false"
}
```

3. TPP can start polling, with a reasonable frequency (no more than 1/sec), the payment status API.

```
GET /openbankingpis/v1/payments/sepa-credit-transfers/<PAYMENT_ID>/status
HTTP/1.1
Host: <PSD2_TARGET_ENVIRONMENT_BASE_URI>
P0-APIKey: <SUBSCRIPTION_KEY>
Authorization: Bearer <ACCESS_TOKEN>
```

   Before any customer action, the response will be the following

```
HTTP Status Code 200
Content-Type: application/json

{
    "transactionStatus": "RCVD"
}
```

4. A push notification is sent to the PSU device (and, as a fallback, as a to do list item in the Flowe app main page) and the PSU is involved.
5. Depending on the approval or the refusal of the PSU we have respectively case a or case b.
   a. The “transactionStatus” changes to “ACSP”, which means that the payment request has been approved and it is in processing.
      Depending on the cut-off times it will become “ACSC”, and payment is set to the clearing network.
      In case the clearing network rejects the payment with an incoming pac.002 “RJCT” then the status of the payment changes to "RJCT".
   b. The “transactionStatus” changes to “CANC”. The same happens if the TPP cancels the payment request before the PSU sends its input.

The SCA status of a given payment request can be retrieve using the following API, where <PAYMENT_AUTHORIZATION_ID> can be obtained in the _links.scaStatus.href field from the payment request submission response shown above.

```
GET /openbankingpis/v1/payments/sepa-credit-transfers/<PAYMENT_ID>/
authorisations/<PAYMENT_AUTHORIZATION_ID>
Host: <PSD2_TARGET_ENVIRONMENT_BASE_URI>
P0-APIKey: <SUBSCRIPTION_KEY>
Authorization: Bearer <ACCESS_TOKEN>
```