Open Banking API

Version 1.0.1
28 January 2022

Franx B.V.
Hessenbergweg 73
1101 CX Amsterdam
088-440 5500
service@franx.com

General Information

eIDAS certificate

Franx relies on your eIDAS certificate for every single API call. Accordingly, mTLS is enabled on our open banking domains and our OAuth2 server. TPPs should use the mTLS aliases for contacting the OAuth2 server but use the non-mTLS endpoints for PSU interactions (for example redirecting PSUs for an authorization code flow).

Key URLs

- **OAuth2 Discovery URL:**
  - UAT: https://login-uat001.franx.link/.well-known/openid-configuration
  - Prod: https://login.franx.com/.well-known/openid-configuration
- **Open banking URL:**
  - UAT: https://openbanking-uat001.franx.link
  - Prod: https://openbanking.franx.com

Response Errors

All response errors are RFC7807 compliant except status codes 401, and 403.

TPP registration

To access AIS APIs, registration is required in our OAuth2 server. We rely on your eIDAS certificate to complete the registration.

Name	Type	Value	Notes
redirect_uris	string[]		An array of redirection URI strings to use in the authorization_code flow.
platform	string	API	This should be the exact value.
client_url	string		URL string of a web page providing information about the TPP.
client_name	string		Human-readable string name of the TPP to be presented to the PSU during authorization.

Request example

```
curl --location --request POST 'https://login-tpp-uat001.franx.link/register' \
--header 'Content-Type: application/json' \
--cert eIDASCertificate.crt \
--key eIDASCertificate.key \
--data-raw '{
    "client_name": "My TPP",
    "client_uri": "http://my-tpp.com",
    "redirect_uris": [
        "https://my-tpp.com/callback"
    ],
    "platform": "api"
}'
```

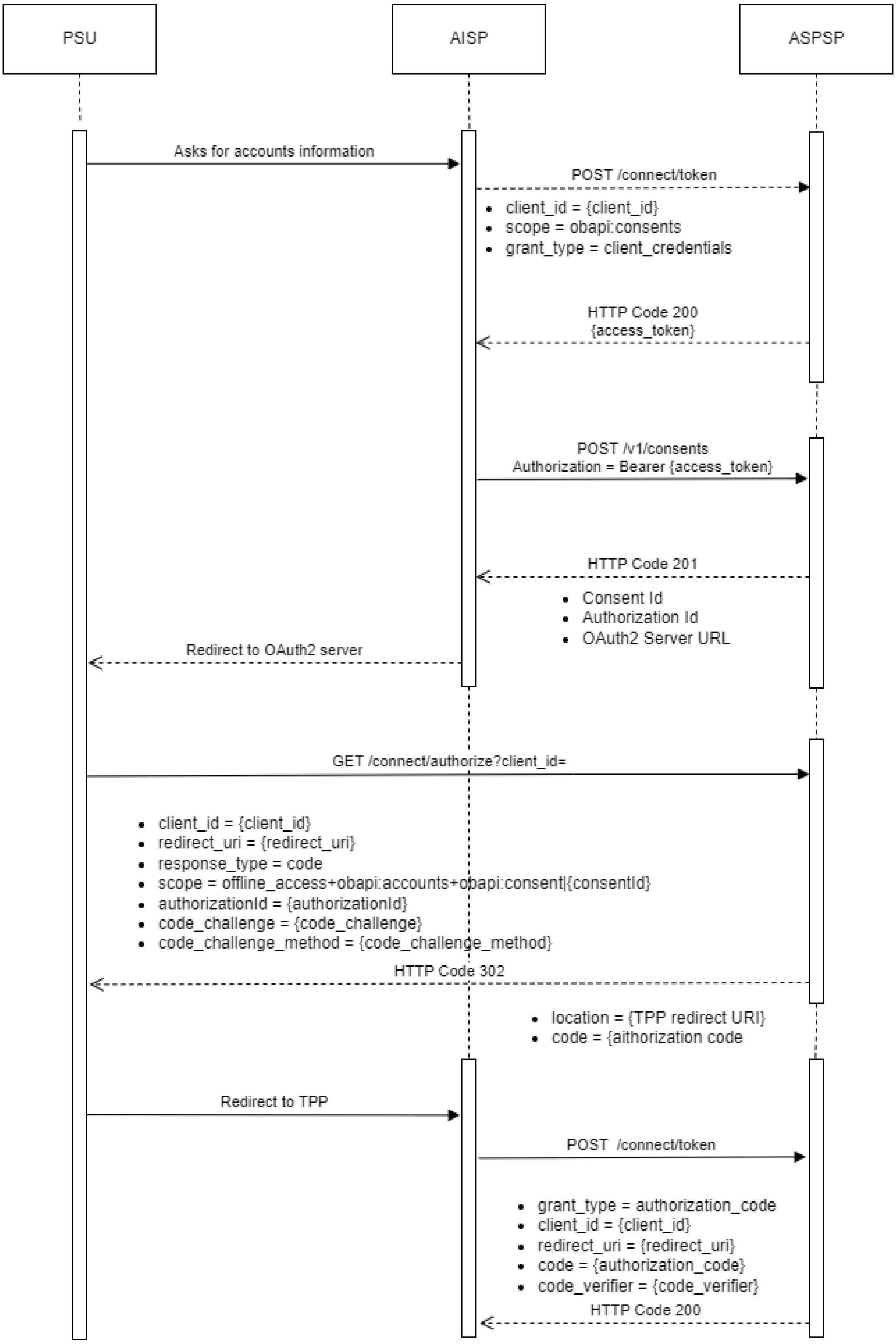
Response example

```
{
    "client_id": "PSDNL-DNB-R163102",
    "client_name": "My TPP",
    "client_uri": "http://my-tpp.com",
    "redirect_uris": [
        "https://my-tpp.com/callback"
    ],
    "scope": "obapi:consents obapi:consent obapi:accounts obapi:balances obapi:transactions"
}
```

Consent Establishment

In order to access all AIS APIs, it is necessary to establish consent between the TPP, the APSP, and the PSU. Currently, Franx offers the Redirect OAuth2 approach to do that.

In this approach, the AISP has to proceed with an OAuth2 authorization in order to retrieve a time-limited access token.
This access token is mandatory to access all the AIS Open Banking APIs. It is associated with the consent established. A refresh token is also returned. TPP needs to keep this refresh token to obtain an access token for the period of the consent validity.


PSU
AISP
ASPSP
Asks for accounts information
POST /connect/token
client_id = {client_id}
scope = obapi:consents
grant_type = client_credentials
HTTP Code 200
{access_token}
POST /v1/consents
Authorization = Bearer {access_token}
HTTP Code 201
Consent Id
Authorization Id
OAuth2 Server URL
Redirect to OAuth2 server
GET /connect/authorize?client_id=
client_id = {client_id}
redirect_uri = {redirect_uri}
response_type = code
scope = offline_access+obapi:accounts+obapi:consent|{consentId}
authorizationId = {authorizationId}
code_challenge = {code_challenge}
code_challenge_method = {code_challenge_method}
HTTP Code 302
location = {TPP redirect URI}
code = {aithorization code
Redirect to TPP
POST /connect/token
grant_type = authorization_code
client_id = {client_id}
redirect_uri = {redirect_uri}
code = {authorization_code}
code_verifier = {code_verifier}
HTTP Code 200

01 - Obtaining an access token for the consent APIs

Request example

```
curl --location --request POST 'https://login-tpp-uat001.franx.link/connect/token' \
--cert eIDASCertificate.crt \
--key eIDASCertificate.key \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=client_credentials' \
--data-urlencode 'scope=obapi:consents' \
--data-urlencode 'client_id=PSDNL-DNB-R163102' \
```

Response example

```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6.....",
    "expires_in": 300,
    "token_type": "Bearer",
    "scope": "obapi:consents",
    "bearer_type": "pop"
}
```

02 - Creating a consent resource

Request example

```
curl --location --request POST 'https://openbanking-uat001.franx.link/v1/consents' \
--cert eIDASCertificate.crt \
--key eIDASCertificate.key \
--header 'X-Request-ID: 3f100918-124f-40f1-b964-9c382abc63be' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer eyJhbGciOiJSUzI1.....' \
--data-raw '{
    "access": {
        "accounts": [],
        "balances": [],
        "transactions": []
    },
    "combinedServiceIndicator": false,
    "frequencyPerDay": 4,
    "recurringIndicator": true,
    "validUntil": "2022-01-01"
}'
```

Response example

```
{
    "consentStatus": "Received",
    "consentId": "kGlBxVQ_uUtisxwqPhe45WIkwbAnU-WK",
    "_links": {
        "scaOAuth": {
            "href": "https://login-uat001.franx.link/.well-known/openid-configuration"
        },
        "self": {
            "href": "/v1/consents/kGlBxVQ_uUtisxwqPhe45WIkwbAnU-WK"
        },
        "status": {
            "href": "/v1/consents/kGlBxVQ_uUtisxwqPhe45WIkwbAnU-WK/status"
        },
        "scaStatus": { "href": "/v1/consents/kGlBxVQ_uUtisxwqPhe45WIkwbAnU-
WK/authorisations/dxXU27zn-VmA8hfZ5JvU5-Q7QS02znDj" }
    }
}
```

03 - Redirecting PSU to ASPSP’s OAuth server

TPP needs to redirect PSU to a location at the OAuth2 server address. Here you can find an example:
GET https://login-uat001.franx.link/connect/authorize?client_id=PSDNL-DNB-R163102&redirect_uri=https://my-tpp.com/callback&code_challenge=LsyAaWafqw2JVw8tjIjDtocjz72mwdXOJB2nhYkZrNo&code_challenge_method=S256&response_type=code&scope=offline_access+obapi:accounts+obapi:balances+obapi:transactions+obapi:consent|kGlBxVQ_uUtisxwqPhe45WIkwbAnU-WK&authorizationId=dxXU27zn-VmA8hfZ5JvU5-Q7QS02znDj

The result of the above GET request would be a redirection from the OAuth2 server with the authorization code parameter. Consider “EE6AB039A2DA8A871EBA8D31A35ABFD896D63D3AE95CB411235157DE98F7035A” as an example of the produced authorization code.

04 - Obtaining an access token for AIS APIs

Request example

```
curl --location --request POST 'https://login-tpp-uat001.franx.link/connect/token' \
--cert eIDASCertificate.crt \
--key eIDASCertificate.key \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=authorization_code' \
--data-urlencode 'code=EE6AB039A2DA8A871EBA8D31A35ABFD896D63D3AE95CB411235157DE98F7035A' 
\
--data-urlencode 'client_id=PSDNL-DNB-R163102' \
--data-urlencode 'redirect_Uri=https://my-tpp.com/callback' \
--data-urlencode 'code_verifier=SomeRandomValueThatIsAtLeast43CharactersLong'
```

Response example

```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6I.....",
    "expires_in": 300,
    "token_type": "Bearer",
    "refresh_token": "F61366C07F4FF645FF17623F685C0.....",
    "scope": "offline_access obapi:accounts obapi:balances obapi:transactions 
obapi:consent|kGlBxVQ_uUtisxwqPhe45WIkwbAnU-WK",
    "bearer_type": "pop"
}
```

Obtaining an access token using a refresh token for the future calls

Request example

```
curl --location --request POST 'https://login-tpp-uat001.franx.link/connect/token' \
--cert eIDASCertificate.crt \
--key eIDASCertificate.key \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=refresh_token' \
--data-urlencode 'refresh_token=F61366C07F4FF645FF17623F685C0.....' \
--data-urlencode 'client_id=PSDNL-DNB-R163102' \
```

Response example

```
{
    "access_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6I.....",
    "expires_in": 300,
    "token_type": "Bearer",
    "refresh_token": "F61366C07F4FF645FF17623F685C0.....",
    "scope": "offline_access obapi:accounts obapi:balances obapi:transactions 
obapi:consent|kGlBxVQ_uUtisxwqPhe45WIkwbAnU-WK",
    "bearer_type": "pop"
}
```

Managing consents

To access Consent APIs, it is essential to obtain an access token via the client_credentials flow.

POST /v1/consents

This method creates a consent resource, defining access rights to dedicated accounts of a given PSU-ID. These accounts are addressed explicitly in the method as parameters as a core function.

Franx only supports the **Bank Offered Consent** model. That being said, the access property should remain as an empty object.

Header Parameters

Name	Type
X-Request-ID (required)	The ID of the request, unique to the call.
Authorization (required)	The obtained access token via the client_credentials flow.

Response Errors

Name	Status	Title	Description
FORMAT_ERROR	400	Bad Request	One or more validation errors occurred.

GET /v1/consents/{consentId}

Returns the content of an account information consent object. This is returning the data for the TPP especially in cases, where the consent was directly managed between ASPSP and PSU.

Header Parameters

Name	Type
X-Request-ID (required)	The ID of the request, unique to the call.
Authorization (required)	The obtained access token via the client_credentials flow.

Response Errors

Name	Status	Title	Description
CONSENT_UNKNOWN	403	Forbidden	The consent was not found.

GET /v1/consents/{consentId}/status

Reads the status of an account information consent resource.

Header Parameters

Name	Type
X-Request-ID (required)	The ID of the request, unique to the call.
Authorization (required)	The obtained access token via the client_credentials flow.

Response Errors

Name	Status	Title	Description
CONSENT_UNKNOWN	403	Forbidden	The consent was not found.

GET /v1/consents/{consentId}/authorisations

Returns a list of all authorization IDs which have been created. This function returns an array of hyperlinks to all generated authorization.

Header Parameters

Name	Type
X-Request-ID (required)	The ID of the request, unique to the call.
Authorization (required)	The obtained access token via the client_credentials flow.

Response Errors

Name	Status	Title	Description
CONSENT_UNKNOWN	403	Forbidden	The consent was not found.

DELETE /v1/consents/{consentId}

The TPP can delete an account information consent object if needed.

Header Parameters

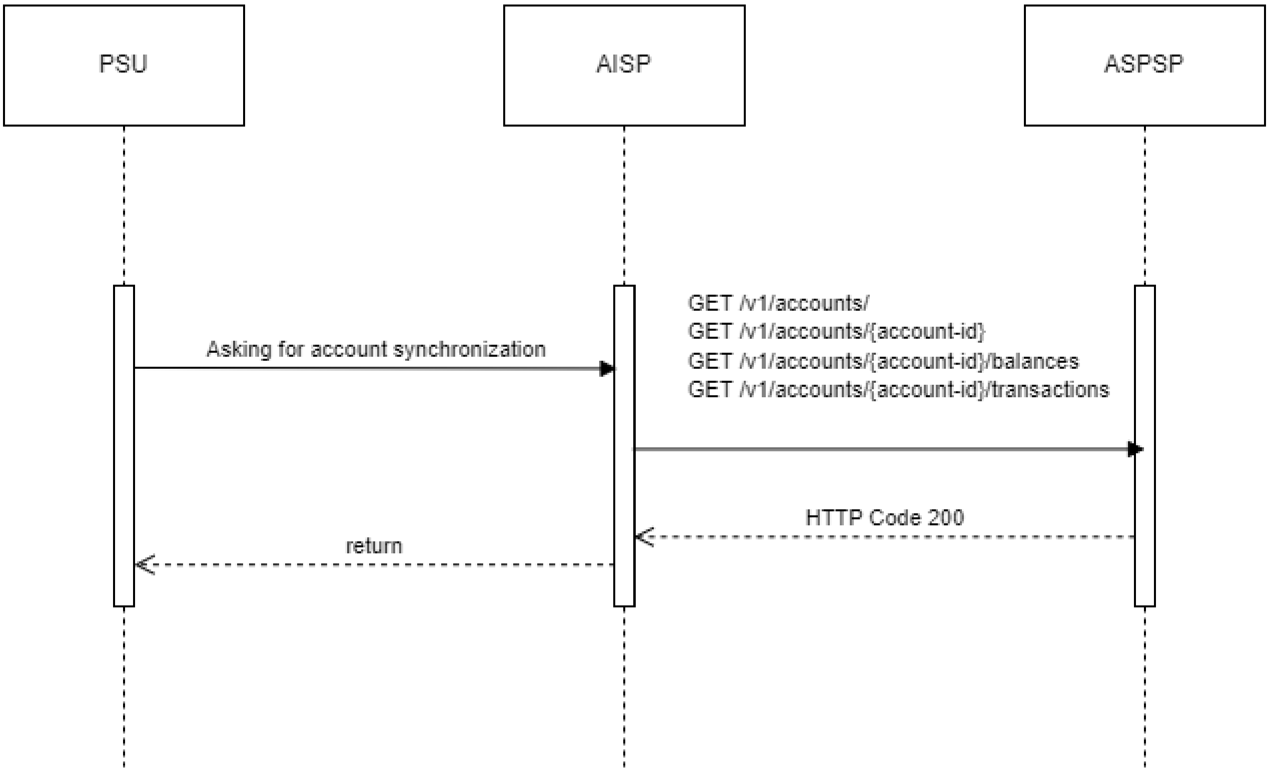
Name	Type
X-Request-ID (required)	The ID of the request, unique to the call.
Authorization (required)	The obtained access token via the client_credentials flow.

Response Errors

Name	Status	Title	Description
CONSENT_UNKNOWN	403	Forbidden	The consent was not found.
RESOURCE_BLOCKED	400	Bad Request	Consent is not valid anymore.

Account Information Services

To access all AIS APIs, it is necessary to establish consent between the TPP, the APSP, and the PSU.
***Note**: TPP must provide the access token related to the given consent too.*


PSU
AISP
ASPSP
Asking for account synchronization
GET /v1/accounts/
GET /v1/accounts/{account-id}
GET /v1/accounts/{account-id}/balances
GET /v1/accounts/{account-id}/transactions
HTTP Code 200
return

GET /v1/accounts

Reads the identifiers of the available payment account together with booking balance information, depending on the consent granted.

Query Parameters

Name	Type
withBalance	If contained, this function reads the list of accessible accounts including the booking balance.

Header Parameters

Name	Type
X-Request-ID (required)	The ID of the request, unique to the call.
Authorization (required)	The access token for the related consent.

Response Errors

Name	Status	Title	Description
CONSENT_UNKNOWN	400	Bad Request	The consent was not found.
RESOURCE_BLOCKED	400	Bad Request	Consent is not valid anymore.

GET /v1/accounts/{account-id}

Reads details about an account, with balances where required. It is assumed that consent of the PSU to this access is already given and stored on the ASPSP system. The addressed details of this account depend then on the stored consent addressed by consentId, respectively the OAuth2 access token.

***NOTE**: The account-id can represent a multicurrency account. In this case, the currency code is set to "XXX".*

Query Parameters

Name	Type
withBalance	If contained, this function reads the list of accessible accounts including the booking balance.

Header Parameters

Name	Type
X-Request-ID (required)	The ID of the request, unique to the call.
Authorization (required)	The access token for the related consent.

Response Errors

Name	Status	Title	Description
CONSENT_UNKNOWN	400	Bad Request	The consent was not found.
RESOURCE_BLOCKED	400	Bad Request	Consent is not valid anymore.
RESOURCE_UNKNOWN	404	Not Found	The account-id not found.

GET /v1/accounts/{account-id}/balances

Reads account data from a given account addressed by "account-id".

Header Parameters

Name	Type
X-Request-ID (required)	The ID of the request, unique to the call.
Authorization (required)	The access token for the related consent.

Response Errors

Name	Status	Title	Description
CONSENT_UNKNOWN	400	Bad Request	The consent was not found.
RESOURCE_BLOCKED	400	Bad Request	Consent is not valid anymore.
RESOURCE_UNKNOWN	404	Not Found	The account-id not found.

GET /v1/accounts/{account-id}/transactions

Reads transaction reports or transaction lists of a given account addressed by "account-id".

Query Parameters

Name	Type
withBalance	If contained, this function reads the list of accessible accounts including the booking balance.
dateFrom (required)	Starting date (inclusive of the date dateFrom) of the transaction list.
dateTo	End date (inclusive the data dateTo) of the transaction list, default is "now" if not given.
bookingStatus (required)	Permitted codes are: • "booked"

Header Parameters

Name	Type
X-Request-ID (required)	The ID of the request, unique to the call.
Authorization (required)	The access token for the related consent.
Accept	Permitted values are: • application/json • text/mt940

Response Errors

Name	Status	Title	Description
CONSENT_UNKNOWN	400	Bad Request	The consent was not found.
RESOURCE_BLOCKED	400	Bad Request	Consent is not valid anymore.
RESOURCE_UNKNOWN	404	Not Found	The account-id not found.
PARAMETER_NOT_CONSISTENT	400	Bad Request	MT940 format is not supported for the specified account-id.
PARAMETER_NOT_SUPPORTED	400	Bad Request	The specified bookingStatus is not supported.
PERIOD_INVALID	400	Bad Request	The specified dates are not valid.
FORMAT_ERROR	400	Bad Request	The requested format is not supported.

Postman

You can try our APIs with the two postman files attached to this pdf document.

References

- NextGenPSD2 XS2A Framework
- The OAuth 2.0 Authorization Framework
- Problem Details for HTTP APIs