Open Banking – Direct Mode

1. Introduction

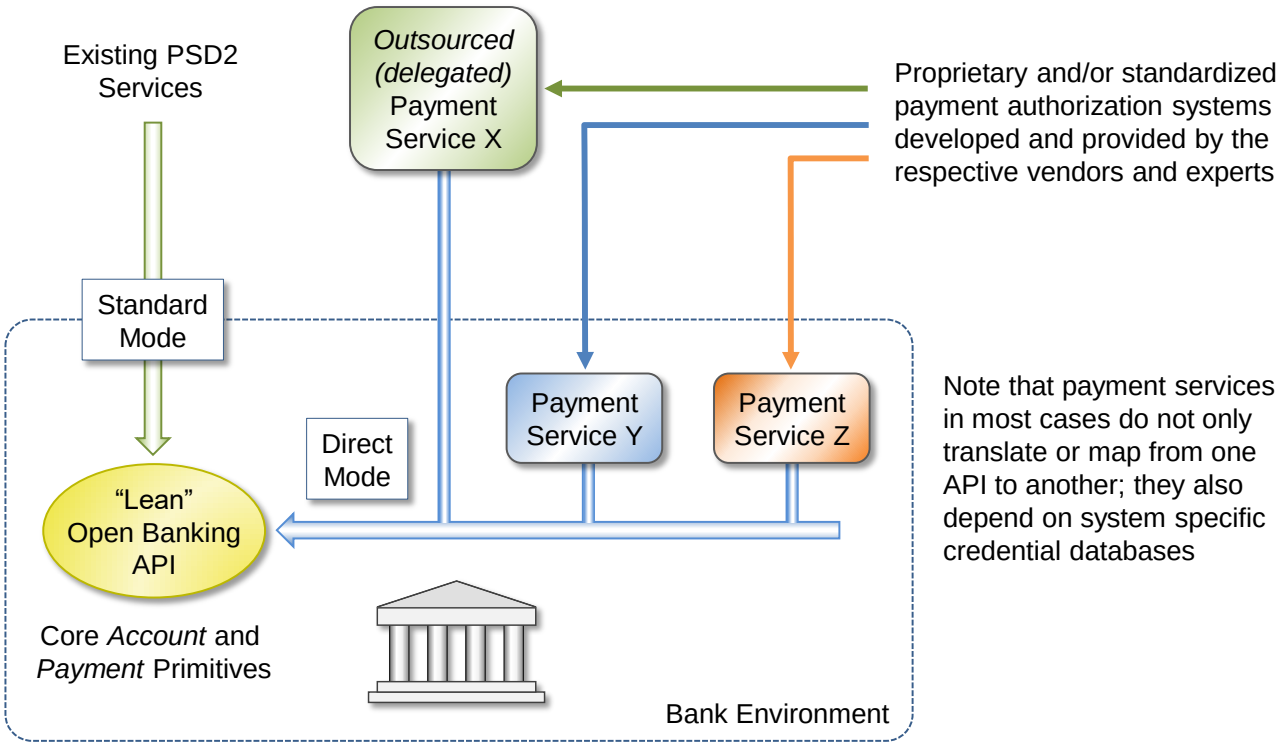
This standards proposal describes a way to extend the reach of Open Banking APIs, which preserves the platform concept ("*Bank Abstraction Layer*"), while enhancing the security/access model. The purpose of that includes:

- Enabling *payment innovation* without depending on Open Banking updates. It should even be possible to develop highly sophisticated applications only using an Open Banking "sandbox".
- The *decoupling of payment applications from the core* facilitates the design of stand-alone products potentially usable with any compatible Open Banking implementation.
- The *enhanced security model* permits creating payment systems that rival the best on the market, both with respect to convenience as well as to security.

The long-term goal is making Open Banking APIs *the natural foundation for all consumer payments*, which everyone (*including the banks that implement and maintain the APIs*) would benefit from.

Non-goal: PSD2 compliance. This may sound a bit strange but this proposal exposes Open Banking APIs in a *neutral manner*, making it possible for the market to build systems that are compliant with PSD2 as well as systems requiring mutual contracts between banks and external service providers.

The described system, from now on referred to as "Direct Mode" to distinguish it from the current way of accessing Open Banking APIs ("Standard Mode"). The illustration below shows how the different modes would be utilized.



2. Core Enhancements

Leveraging OAuth2

Since OAuth2 and associated OAuth2 tokens represent a core element in current Open Banking APIs, the Direct Mode keeps the concept intact, while changing some details around the implementation. In practical terms, this means that in order to use Open Banking APIs, the calling service must first obtain an OAuth2 token associated with the particular user. This can be accomplished in two ways:

1. Through a slightly upgraded OAuth2 "Authorize" and login process. Note that this is the only operation that requires the use of the default SCA solution for the particular Open Banking implementation.
2. Through a new method which "emulates" a user login. This method is only meant to be used by bank-internal services for setting up the Direct Mode API for a specific user.

> *By building on OAuth2, secure enrollment of virtual payment credentials by external or outsourced service providers can be performed without any prior knowledge of users.*

Authenticating services using the Direct Mode

In order to *securely separate* services using the Standard Mode and the Direct Mode, the **recommended** way would be through the TTP client certificates where services using the Direct Mode would typically use client certificates issued by the bank itself using a dedicated PKI.

> *In addition to internal bank services, the Direct Mode is only intended to be used by selected and contracted parties.*

Static user identity (SUID)

The primary enhancement needed is that a *static user identity* (SUID) is established during the Direct Mode API initiation phase. This would in most cases be the same identity as used in other parts of a bank's IT-system. The reason for this requirement is that *OAuth2 tokens may be renewed and there is also a need for an identity for logging and administration purposes*.

After this step has been performed the SUID and the two associated OAuth2 tokens (primary and refresh), would typically be stored in a local database in the caller service in order to enable future Direct Mode sessions, which is required for payment operations.

Design option: since Direct Mode is operating at a higher privilege than the Standard Mode, OAuth2 tokens used by Direct Mode services may live longer, possibly even indefinitely.

Human identity option

For certain use cases like the creation of virtual payment credentials, it would be beneficial if the initialization step also provided a human name.

The following printout shows an example of an enhanced OAuth2 authorization response:

```
{
  "access_token": "619763e4-cf77-4d2f-838e-1f6c6b634040",
  "token_type": "Bearer",
  "expires_in": 3600,
  "refresh_token": "da1bdd53-bed9-4cb7-9c62-0bbe0356d90b",
  "scope": "xyz",
  "static_user_identity": "479262777",
  "human_name": "Jane Doe"
}
```

The elements within the red rectangle are the ones described in this section.

Below is a *non-normative* database, here expressed in SQL code:

```
CREATE TABLE OAUTH2TOKENS (
    SUID            VARCHAR(50) NOT NULL UNIQUE,      -- Static User ID
    HumanName       VARCHAR(50) NULL,                 -- Optional human name
    AccessToken     CHAR(36)    NOT NULL UNIQUE,      -- For API access
    RefreshToken    CHAR(36)    NOT NULL UNIQUE,      -- For refreshing AccessToken
    Expires         TIMESTAMP   NOT NULL,             --        -"-
    PRIMARY KEY (SUID)
);
```

A background process is assumed to maintain “freshness” of the OAuth2 tokens in the local database.

3. Calling Direct Mode APIs

To invoke an Open Banking API method in Direct Mode the caller must in some way have obtained a SUID that it uses to extract the currently associated OAuth2 token from the local database.

SCA and consent exemption

In the Direct Mode the Open Banking API (after the initial setup), is assumed to be called by a service which in some way have authenticated the user as well as dealt with possible consents. Due to that, the Open Banking API when called in Direct Mode **must not** request SCA or consents. However, to maintain proper API order, the caller must still call the same APIs and use the same parameters (with the exceptions described in the next section), as it would using the Standard Mode.

4. Sample Operation

Below is an example of how a payment operation can be performed using the NextGenPSD2 API.

Payment Request

```
POST /v2/payments/credit-transfers?app-id=l77262db3b8475473b8b702c8ea4eaa136 HTTP/1.1
User-Agent: Java HttpClient 1.3
Host: psd2.example-bank.com
Date: Mon, 20 May 2020 10:33:40 GMT
PSU-IP-Address: 203.48.103.79
PSU-IP-Port: 8442
PSU-Http-Method: GET
PSU-User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
TPP-Redirect-URI: https://example-ttp.com/paymentsuccess
TPP-Nok-Redirect-URI: https://example-ttp.com/operationfailed
X-Request-ID: e0229fa5-98ce-48aa-90b9-1e088083b7de
Authorization: Bearer e378ce15-a4cf-441c-82b1-9b61861ec248
Content-Type: application/json
Content-Length: 4563

{
  "creditorAccount": {
    "iban": "FR7630002111110020050012733"
  },
  "debtorAccount": {
    "iban": "FR7630004003200001019471656"
  },
  "debtorAccountStatementText": "payback time!",
  "instructedAmount": {
    "amount": "152.00",
    "currency": "EUR"
  }
}
```

The shadowed elements would not be used or emitted since they have no use in the Direct Mode.

Continued on the next page….

Payment Response

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: 967

{
  "transactionStatus": "ACTC",
  "paymentId": "0762e6c2-8a67-4a71-9217-b033afc0a77b",
  "_links": {
    "scaStatus": {
      "href": "/v2/payments/0762e6c2-8a67-4a71-9217-b033afc0a77b/...."
    },
    "scaRedirect": {
      "href": "https://psd2.example-bank.com/performsca...."
    },
    "self": {
      "href": "/v2/payments/0762e6c2-8a67-4a71-9217-b033afc0a77b"
    },
    "status": {
      "href": "/v2/payments/0762e6c2-8a67-4a71-9217-b033afc0a77b/status"
    }
  }
}
```

The payment response would preferably not include the shadowed SCA-related elements since they do not apply to the Direct Mode.

5. Login Emulation Method

T.B.D.

Version: 0.13, anders.rundgren.net@gmail.com, 2020-06-02