Swedbank PSD2 API developer documentation

Change log

We encourage providing feedback to improve our services by sending a mail to openbanking@swedbank.com

Version	Date	Title
2.0	June 2019	Documentation for PSD2 API v2.
2.1	June 2019	API versioning guidelines.
2.2	August 2019	PSU Authentication using OAuth 2.0 decoupled; Decoupled Approach; POST signing basket, POST recurring/periodic payment.
2.3	September 2019	September 2019 PSU Authentication using OAuth 2.0 redirect; Initial consent for list of accounts.
2.4	January 2020	Standards; PSU Authentication using OAuth 2.0 decoupled; Decoupled Approach; Payments initiation services API Clarifications.
3.0	March 2020	[Sweden] New section "Add recipient"; Parallel consent support description.
3.1	May 2020	[Sweden] Updated auth diagrams to optionally include QR image fetching.
3.2	June 2020	[Sweden] An option to use additional scopes that allow simplification of AIS flow by decreasing number of SCAs. Optimization done in Authentication and Consent steps.
3.3	July 2020	[Sweden] Transactions over 90 days can be added as a scope to consent for less SCA.
3.4	August 2020	[Baltics] Future dated and recurring/periodic payment introduction.
3.5	September 2020	[Baltics] Added Payment cancellation for Recurring/periodic and future dated payment types
3.6	September 2020	[Baltics] Biometric data support
3.7	October 2020	Added common errors chapter
3.7.1	November 2020	Notes about allAccounts availability only to AISP
3.7.2	December 2020	Added state diagrams for Payments and Authorisation
3.7.3	January 2021	Added additional info about errors in Common errors chapter
3.7.4	January 2021	Improvements and clarification to Future dated and recurrint/periodic payment sections
3.8	March 2021	Introduced new feature: Account selection on bank side
3.9	March 2021	Introduced country based DNS for API usage for DNS problem mitigation. More info can be found on Sandbox and Production Connection overview
3.10	April 2021	[Latvia] Balances and transactions (up to 90 days) can be added as scopes to consent for less SCA.

Definitions

This section offers explanations to the terminology used throughout the document.

Definition	Description
TPP (Third Party Provider)	Third Party Provider (TPP) is a provider of an application that the user uses and that is not offered by the bank. TPP is the client/consumer/partner of the API and acts on behalf of the user.

Sandbox	Sandbox gives access to a small set of static data and it is used as an example to illustrate what would be returned when using the production API. The Sandbox can be reached within the developer portal.
API Call	API call is a request towards the API which receives a response. The API is by design stateless, it does not "remember" anything about previous requests, i.e. there is no session. Therefore every request made towards the API must contain certain headers so that the API can authenticate and authorize the use. Message parameters can be passed at different levels; • message parameters as part of the HTTP level (headers) • message parameters by defining a resource path (URL path information) • message parameters as part of the HTTPS body
Authentication	Authentication is the process of verifying that an individual, entity or website is who it claims to be. This authentication is later used to grant authorization to specific data and functions within a system.
SCA (Strong Customer Authentication)	The process of using a strong (2-factor) identification method to identify the customer.
Permission(s)	Permissions are stored in an access policy and are part of the consent given by the customer for the TPP. Permissions dictate what TPP is allowed to do with customer's data.
Consent	Consent is the agreement given by the customer to the TPP to retrieve user's data from the bank. Consent is stored and verified by the bank, but approved by the user. Consent may have different characteristics, like recurrence, expiration, etc.
Client	Client refers to the client of the API which is commonly the TPP application.
Swagger	An API design and documentation platform to aid in the development lifecycle. It is used to publish the documentation within the API Explorer.
User or PSU (Payments service user)	Refers to the bank customer, or representative of customer, who uses TPP application.
Customer	Private person or company (legal entity) which has agreement with the bank.
ASPSP (Account Servicing Payments Service Provider)	This is the account servicing provider, i.e, the Bank/Swedbank.
PISP (Payment Initiation Service Provider)	This is a service provider who can initiate a payment transaction on behalf of the customer.

Introduction

The purpose of this documentation is to help and guide developers around what is possible to access in terms of customer's data & services at Swedbank Group. This document describes best ways to use API, explains unclear parts of it and describes TPP API usage requirements and flows. All technical details on each endpoint can be found in swagger files in the Developer-portal -> Develop.

Please note that accessing data from Swedbank API includes Estonia, Latvia, Lithuania, Sweden and all the cooperating Savings Banks in Sweden. This also means that format, content and request parameters may differ, mainly driven by fact that Sweden does not use Euro as domestic payment currency.

Swedbank Open Banking is an invitation to all developers to build new products and services based on a set of APIs. As an Open Banking Developer you are someone trying to innovate by using financial data without being a regulated entity on the financial market. This will require a contract to access services provided and may also be a subject of specific terms and conditions.

APIs are offered as a part of the PSD2 regulation, as a subset of the Open Banking initiative. As a TPP (Third Party Payment service provider) within the PSD2 definition, you will be under supervision of the Financial Supervision Authority in your home member state in the European Union. With this comes a set of rights, as well as a set of obligations.

Standards

Swedbank Open Banking API is constructed following standarts described in ISO20022 and the Berlin Group standard NextGenPSD2 XS2A Framework Implementation Guidelines version 1.3.5 (BGS). API is built as group level API on semantical level, but at each session works only with one bank and one customer data and services. To be able to distinguish bank, country specific URL's are used or general URL with BIC as mandatory parameter in the request.

The exposure of data is done through RESTful services. API consumers should respect cache policy: VOLATILE. Most of API calls data is provided in JavaScript Object Notation (JSON) format. All API requests and responses must use a UTF-8 character encoding.

All dates in parameters & request body are represented in ISO 8601 date or date time format e.g.:

```
ISO 8601 Date - 2019-05-30
ISO 8601 DateTime - 2019-05-30T00:00:00+00:00
```

All dates in the HTTP headers are represented as RFC 7231 Full Dates

```
Tue, 21 May 201914:23:49 GMT
```

Identification and connection security

To ensure secure communication between TPP and Swedbank, and for TPP identification, Swedbank relies on the following qualified certificates:

- Qualified Website Authentication Certificate (QWAC);
- Qualified Certificate for Seals (QSEAL) certificates;

A qualified digital certificate is a public key certificate issued by a qualified trust service provider (QTSP) that ensures the authenticity and data integrity of an electronic signature, its accompanying message and/or attached data. This assures a link between the cryptographic keys used to secure the transaction and the entity the keys belong to. The respective TPP roles are also determined using these certificates of identification.

Transport layer Security (TLS)

The communication between the TPP and the Bank is always secured by using **TLS version 1.2** or higher.

To ensure secure communication between TPP and Swedbank, and for TPP identification, Swedbank relies on the Qualified Website Authentication Certificate (QWAC);

QWAC certificates are used to achieve Transport Layer Security (TLS) on the transport level.

Application layer security

Application level security is enabled by using Qualified Certificate for Seals (QSEAL) certificates to provide data integrity at application level.

Customer interaction patterns

From multiple options described in BGS we have selected to implement the following:

- preOAuth authorisation mode. It requires an authentication of a PSU in a pre-step, translating this authentication into an access token. Access token is mandatory for any other API call as described in BGS (4.3 Optional Usage of OAuth2 for PSU Authentication or Authorisation). For details, please, refer to documentation available in Open Banking portal.
- Based on multiple consent management models described in BGS in Swedbank API we support only detailed consent model. The TPP is submitting detailed consent information – user identification, services and account numbers affected – to the ASPSP for authorisation by the user. Valid consent is used to verify access to accounts information.
- Swedbank offers a redirect integration method as main way of integration for the TPP and the user. Decoupled integration method is supported with security means that are suitable for secure work with such way of integration. For decoupled integration method Mobile Bank-ID (SE), Smart-ID (LT, LV, EE), Mobile-ID (LT, EE) and Biometric data (LT,LV,EE) is supported.

Connecting to API

To be able to use and connect to the API TPP must fulfill these steps:

- Onboard to the Open Banking portal;
- Register application for Sandbox usage;
- Register application for Production usage;
- Setup OAuth2.0;
- Setup connection identification and security;

API types

There are two different API types under Swedbank API: PSD2 and Partner. Different API functionality depends on the selected scope (more info on scope usage can be found under Scopes chapter). Every API type name can be used as a prefix to construct scope names. Through out the documentation this prefix is described as `{API-type}` variable and should be replaced by corresponding API type PARTNER or PSD2 (e.g. `{API-type}sandbox` scope should be replaced with `PARTNERsandbox` or `PSD2sandbox`).

TPP onboarding on Developer portal

Onboard in the Open Banking portal by providing correct email. Please provide correct organisation name as it will be used in communication with customer. After few minutes email with link to further steps will arrive. Please keep username and follow best practices for password creation and management. Unauthorised use of this account may cause damage.

TPP application registration

- In Open Banking portal create an application. OAuth 2.0 standard parameters must be specified during application creation. TPP must use **different applications** for Sandbox and Production environment.
    - Add a `callback_url` - application URL where access code from OAuth 2.0 protocol is returned
    - Set type to `confidential`
    - Obtain a `client_id` - used in OAuth 2.0 protocol
    - Obtain a `client_secret` - used in OAuth 2.0 protocol
- Assign needed APIs to application.

Configuring OAuth2.0 flow for TPP

Some API endpoints does not require customer identification (login) or approval (signing) - in such cases OAuth2.0 token is not needed and configuration can be skipped.

TPP must register `redirect_uri` (link must support https protocol) for OAuth 2.0 flow in Open Bank portal. After a user successfully authorises in the bank, the authorisation server will redirect the user back to TPP application with authorisation code in the URI. Redirect URI must be on secure server (HTTPS) as OAuth2 flow will send sensitive data to it. When authorization is requested, the authorisation server will validate a redirection URI to ensure the URI in the request matches the registered one. So, the call-back URI that is registered in the application must match the `redirect_uri` that is passed as query parameter.

In the Auth tab you will see your `client_secret` (`shared_secret`). This is private information that should be kept safe; if it is lost it can potentially be used in malicious ways. It should NEVER EVER be placed in your frontend; it is used in backend communication ONLY. If needed `client_secret` can be regenerated.

Sandbox usage

Sandbox is created in Swedbank Open Banking initiative to support more detailed technical understanding and testing of API's. Current version of API is based on static data and is subject to change. For using Sandbox (compared to API's) following configuration changes are needed:

Swedbank Country	BIC Sandbox	BIC Production
Sweden	SANDSESS	SWEDSESS
Lithuania	SANDLT22	HABALT22
Latvia	SANDLV22	HABALV22
Estonia	SANDEE2X	HABAEE2X

Cooperating Savings Banks Sweden	SANDSESS	SWEDSESS

API Prefix: /Sandbox/{version}/accounts

OAuth 2.0 requested scope: scope=`{API-type}`sandbox

> Please note that in Sandbox OAuth 2.0 protocol is implemented, but in case you do not want to go through all the process in Authorization header - you may use hardcoded value – `Authorization: Bearer dummyToken`.

The API Sandbox consists of **static mocked data** needed to preliminary testing.

Sandbox can be used with dummy token or OAuth 2.0 authentication with fake data can be used. If QWAC or QSEAL certificates are provided Sandbox will do validation of signature and mutual TLS.

Production usage

API Prefix: /{version}/accounts

OAuth 2.0 requested scope: scope=`{API-type}`

Production usage is allowed only for TPPs which are licensed or have agreement with Swedbank.

For Production usage TPP must register public parts of TLS (QWAC) and signing (QSEAL) certificates in the Open Banking portal. These certificates are validated using OCSP and used for mutual TLS and verification of requests signatures. To streamline process please send us email to openbanking@swedbank.com.

It is strongly advised to perform API smoke test in production before opening it for wide usage. We encourage providing feedback in case of some unclarified places or some improvements are needed by sending a mail to openbanking@swedbank.com.

> Please note that application name and organization name will be visible to user and should be correct. TPP must use different applications (app-id's) for sandbox and production environment.

Sandbox and Production Connection Overview

Sandbox is created in Swedbank Open Banking initiative to support more detailed technical understanding and testing of APIs. Current version of API is based on static data and is subject to change. In order to use API, either Sandbox or Production, there are different ways:

- BIC usage as a request parameter;
- Country based DNS. BIC is optional; **Recommended**
- Country specific URL. BIC is optional;

Below you can find BIC and URL information: | **Swedbank Country** | **BIC Sandbox** | **BIC Production** | **Country based DNS** | **Country based URL*** | | --- | --- | --- | --- | --- | | Sweden | SANDSESS | SWEDSESS | - | https://se.psd2.api.swedbank.com | | Lithuania | SANDLT22 | HABALT22 | https://psd2.api.swedbank.lt | https://lt.psd2.api.swedbank.com | | Latvia | SANDLV22 | HABALV22 | https://psd2.api.swedbank.lv | https://lv.psd2.api.swedbank.com | | Estonia | SANDEE2X | HABAEE2X | https://psd2.api.swedbank.ee | https://ee.psd2.api.swedbank.com | | Cooperating Savings Banks Sweden | SANDSESS | SWEDSESS | - | https://se.psd2.api.swedbank.com |

> * we recommend to use this country based URL as a fallback in case of network related errors (SSL, no response, timeouts).

	Sandbox	Production
API Prefix	/Sandbox/{version}/accounts	/{version}/accounts
OAuth 2.0 requested scope	scope=PSD2sandbox	scope=PSD2

> Please note that in Sandbox OAuth 2.0 protocol is implemented, but in case you do not want to go through all the process in Authorization header - you may use hardcoded value – `Authorization: Bearer dummyToken`.

The API Sandbox consists of **static mocked data** for:

- Account Information Services (**AIS**) for Sweden and Baltics as defined by article 67 in the PSD2 Directive.
- Payment Initiation services (**PIS**) for Sweden and Baltics as defined by Article 66 in the PSD2 Directive.
- Confirmation on the availability of funds (**CoF**) for Sweden and Baltics as defined by Article 65 in the PSD2 Directive.
- Security and consent services (**SEC**) to create consent and request an OAuth 2.0 token.

User authentication

In Swedbank user connecting to API can act as a private customer or as a representative of a company (legal entity):

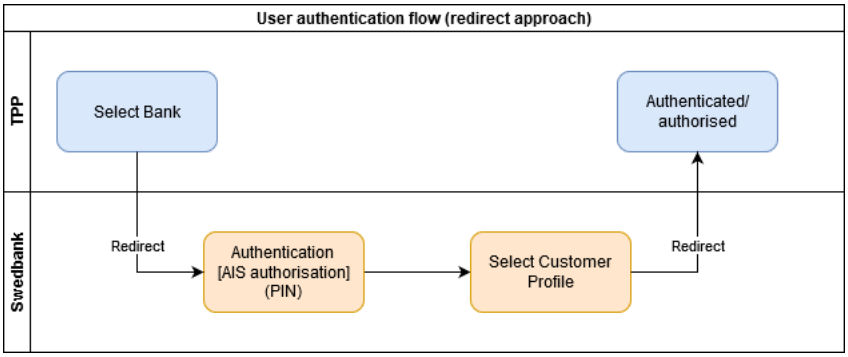
- As a private user person has access to customer owned accounts in the Swedbank;
- As a representative of a company user can only access accounts that user has been granted access to. Access rights to accounts are defined in the Internet bank agreement and transparently applied to API.

Swedbank in API implemented pre-OAuth authorisation mode. The token is the identification method that can be used by the TPP to act on behalf of the customer (e.g. to retrieve data without the user's involvement).

It requires an authentication of a user in a pre-step, translating this authentication into an access token. Swedbank API issues OAuth 2.0 access token for unique tuple of bank, user and target customer (account owner). It means that one OAuth 2.0 token represents one user in one bank accessing data for one target private or corporate customer owned accounts.

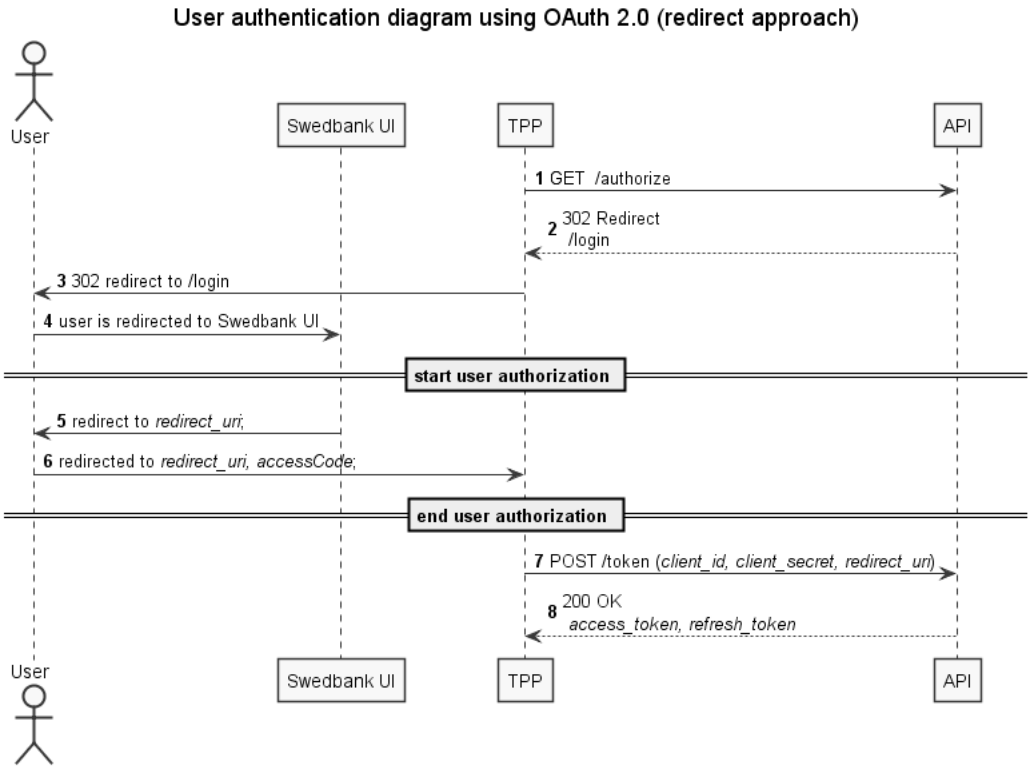
User authentication using OAuth 2.0 redirect approach diagrams

Simple flow diagram



1. User in TPP's environment selects Swedbank authentication method;
2. User is redirected to Swedbank UI where logs in and authenticates; **[optional] AIS pre-authorisation:** TPP can provide additional scopes (more details on scopes chapter) to get Account information service authorisation from user with single SCA. **[optional]** QR code may be displayed due to security reasons for BankID.
3. User selects customer profile which (s)he represents (private or corporate);
4. User is redirected back to TPP environment;

Sequnce diagram



1. TPP initiates OAuth 2.0 request from server side `https://psd2.api.swedbank.com/psd2/authorize?bic=SANDSESS&state=somestring&client_id=171c2ad437f0fd44d68b2ea3f89b76a28d&redirect_uri=https://www.swedbank.com/openba`
or
`https://{countryCode}.psd2.api.swedbank.com/psd2/authorize?state=somestring&client_id=171c2ad437f0fd44d68b2ea3f89b76a28d&redirect_uri=https://www.swedbank.com/openbanking&respons`
2. HTTP code 302 response with redirect link in `Location` header will be returned;
3. Forward the redirect link returned with HTTP code 302 to user browser;
4. User get redirected to Swedbank environment to `/login` page;

    in case of bic=`HABALT22` and additional scope `{API_type}account_list` is requested, user is presented with confirmation page on giving out list of accounts to TPP;
    in case of bic=`HABALV22` and additional scopes, `{API_type}balances` and/or `{API_type}transactions`, are requested, user is presented with confirmation page on giving out balance/statement (up to 90 days) data of his payment accounts;

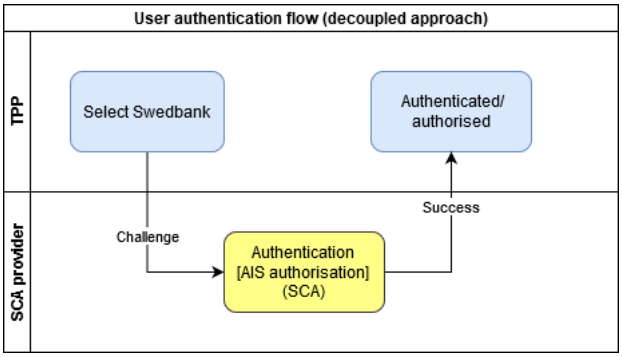
    User authentication procedure starts;
5. After login user is redirected to TPP `redirect_uri` with `access_code` and `state` parameters;
6. TPP from server side calls `/token` endpoint and exchange `access_code` to OAuth 2.0 token.
`https://psd2.api.swedbank.com/psd2/token?grant_type=authorization_code&client_id=171c2ad437f0fd44d68b2ea3f89b76a28d&client_secret=079603ebf92446f8a70e0ddc11f8d18255-4c92-a975-4bd1916d4199&redirect_uri=https://www.swedbank.com/openbanking`
or

```
https://{countryCode}.psd2.api.swedbank.com/psd2/token?
grant_type=authorization_code&client_id=171c2ad437f0fd44d68b2ea3f89b76a28d&client_secret=079603ebf92446f8a70e0ddc11f8d1
8255-4c92-a975-4bd1916d4199&redirect_uri=https://www.swedbank.com/openbanking
```

7. Get OAuth2.0 token for further API usage;
8. Response with created `access_token` and `refresh_token`;

User Authentication using OAuth 2.0 decoupled approach diagrams

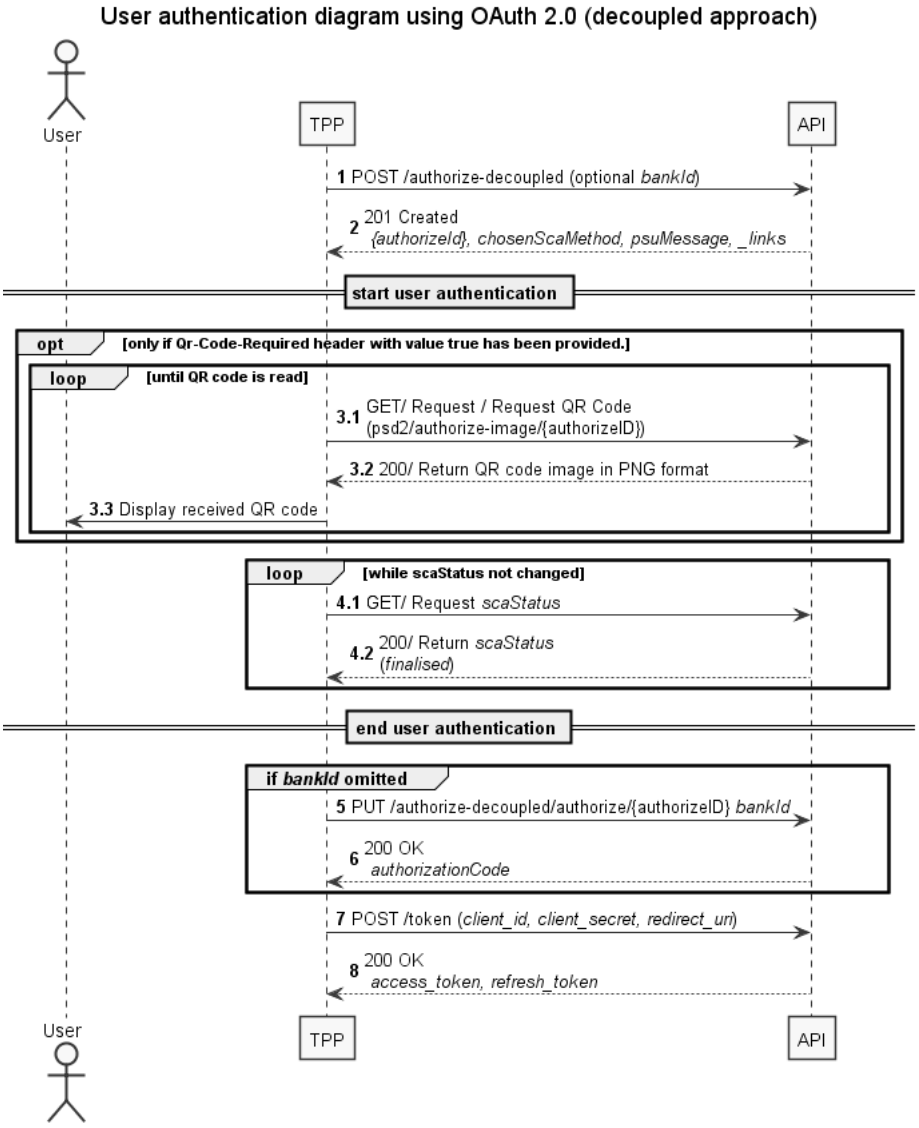
Simple flow diagram



TPP may use API to get available login methods, participating Saving banks and other information needed for login process. This information should be cached and refreshed each 24 hours;

1. User in TPP environment selects Swedbank authentication method, profile & information needed for selected authentication method;
2. User receives a challenge to security device and performs SCA; **[optional] AIS authorisation:** TPP can provide additional scopes (more details on scopes chapter) to get Account information service authorisation from user with single SCA.
3. User is back to TPP;

Sequence diagram



1. Initiate the OAuth 2.0 request with user's login details to Swedbank.

    In Sweden user can have engagement in one bank (Swedbank or Savingsbank) or in several. Already in this POST request TPP can provide `bankId` to which bank user's profile will be logged in. Alternatively `bankId` can be omitted. If user has engagement in one bank (Swedbank or Savings bank) user is logged in to that bank even if `bankId` is not provided. In case user has engagement in several banks TPP can proceed further omitting `bankId` and `bankId` can be provided later with PUT request;

2. TPP gets a response with created resource;
   User authentication starts;
3. SCA process starts on user's device.
    1. [Optional] Request QR code image (more about QR code information could be found in QR code chapter);
    2. TPP gets image in PNG format;
    3. TPP displays received image to user. Image is updated every 1s.
4. Check `scaStatus` every 3-5s until value 'finalised' or 'failed' is received

    1. TPP requests `scaStatus`;
    2. If SCA is OK and `bankId` was provided with POST or user profile has engagement in one bank (Swedbank or Savings bank), `scaStatus` is 'finalised' and response includes `authorizationCode`.

        In case user profile has engagement in several banks and `bankId` was omitted with POST, error message is returned asking to provide `bankId`.

5. If `bankId` is not provided in the 1st request then TPP should provide `bankId` in case user profile has engagement in several banks.
6. Response includes `authorizationCode`.
7. Call `/token` endpoint and exchange `access_code` to OAuth 2.0 token.
   `https://psd2.api.swedbank.com/psd2/token?grant_type=authorization_code&client_id=171c2ad437f0fd44d68b2ea3f89b76a28d&client_secret=079603ebf92446f8a70e0ddc11f8d18255-4c92-a975-4bd1916d4199&redirect_uri=https://www.swedbank.com/openbanking`
   or
   `https://{countryCode}.psd2.api.swedbank.com/psd2/token?grant_type=authorization_code&client_id=171c2ad437f0fd44d68b2ea3f89b76a28d&client_secret=079603ebf92446f8a70e0ddc11f8d18255-4c92-a975-4bd1916d4199&redirect_uri=https://www.swedbank.com/openbanking`
8. Get OAuth2.0 tokens for further API usage;

- Please note that some login methods (Bank-Id, Smart-Id) may raise error if there is ongoing login session.

Scopes used in user authentication

In Swedbank API OAuth2.0 scopes are used to express in high level requested access to data and functionality. Scopes are requested in OAuth2.0 flow and may be granted by user. Final list of granted scopes is provided in OAuth2.0 response. Table describes dependency of scopes and functionality impacted for each BIC.

Scopes description

- `{API-type}` - specifies partner role used to connect to API. Supported values `PSD2` or `PARTNER`;
- `{API-type}account_list` - this scope is needed in case TPP needs customer list of accounts. If such scope is requested, user is informed about TPP request during login process and if scope is granted, allAccounts consent is returned automatically valid;
- `{API-type}account_balances` - this scope is needed in case TPP needs customer account balances. If such scope is requested, user is informed about TPP request during login process and if scope is granted, detailed consent with balances request is returned automatically valid;
- `{API-type}account_transactions` - this scope is needed in case TPP needs customer account transaction list/statment. If such scope is requested, user is informed about TPP request during login process and if scope is granted, detailed consent with transactions request is returned automatically valid;
- `{API-type}account_transactions_over90` - this scope is needed in case TPP wants to get transactions over 90 days without SCA. It grants access to transaction data up to four times per 24 hours for each user's account.

PSD2 API available scopes

scope	SWEDSESS	HABALT22	HABALV22	HABAEE2X
PSD2	mandatory	mandatory	mandatory	mandatory
PSD2account_list	optional	optional	optional	optional
PSD2account_balances	optional	not supported	optional	not supported
PSD2account_transactions	optional	not supported	optional	not supported
PSD2account_transactions_over90	optional	not supported	not supported	not supported

Samples

To inititate authentication for user TPP should use following request with different `scope` parameter values:

```
GET    https://psd2.api.swedbank.com/psd2/authorize?bic=SWEDSESS&state=1&client_id=123123132132132&redirect_uri=https%3A%2F
```

TPP uses API in pure PISP way - account information is not needed

- scope=`PSD2`

TPP agreed with customer on sharing account list

- scope=`PSD2 PSD2account_list`

TPP agreed with customer on sharing account list and balances

- scope=`PSD2 PSD2account_list PSD2_account_balances`

Access and refresh token validity

`access_token` is required to access protected resources. Access token is going to be valid for a limited period (provided in json element `expires_in`, usually 1 hour). When access token expires, `refresh_token` will have to be provided in order to obtain a renewed access token using `.../psd2/token?grant_type=refresh_token...` . Although access token can be renewed at any time using refresh token, due to performance reasons access token should be renewed only when it expires.

```
{
   "access_token":"sf681f6s8-fs81f68ase-86as16e8f-3351eafs851",
   "token_type":"bearer",
   "expires_in":3600,
   "refresh_token":"asefsef566-86asf-yiky6-a6e86j",
   "scope":"PSD2"
}
```

A refresh token carries the information necessary to obtain a renewed access token. You can request a new access token until the refresh token is valid. Validity of OAuth 2.0 refresh token is subject to agreement (usually defined as 90 days). After expiration of refresh token refresh of token fails with error:

```
{
   "tppMessages":
   [
      {"category" : "ERROR",
       "code" : "FORMAT_ERROR",
       "text" : "Provided refresh_token expired"}
   ]
}
```

TPP can request customer to repeat authentication process to get new refresh token to continue Swedbank API services usage.

User authorisation using SCA

Some API calls requires SCA (user must approve/sign request by using PIN). SCA can be implemented either by using redirect or decoupled methods. Default integration method is redirect if nothing is selected.

In redirect mode API generates links and TPP must redirect user to these pages. In these pages corresponding `payment/consent/...` information will be displayed to user and user authorises using security device.

In decoupled integration mode SCA is performed without displaying bank pages. Process relies on third party identity provider (e.g. Bank-ID, Smart-ID, Mobile-ID). In decoupled mode API generates SCA requests to third party identity provider, user gets details of requested authorisation action in his security device and approves it using PIN code.

> Authorisation resource is valid 15 minutes if not stated differently. Later authorisation status is set to `'failed'`.

Authentication procedures issued by Swedbank to users

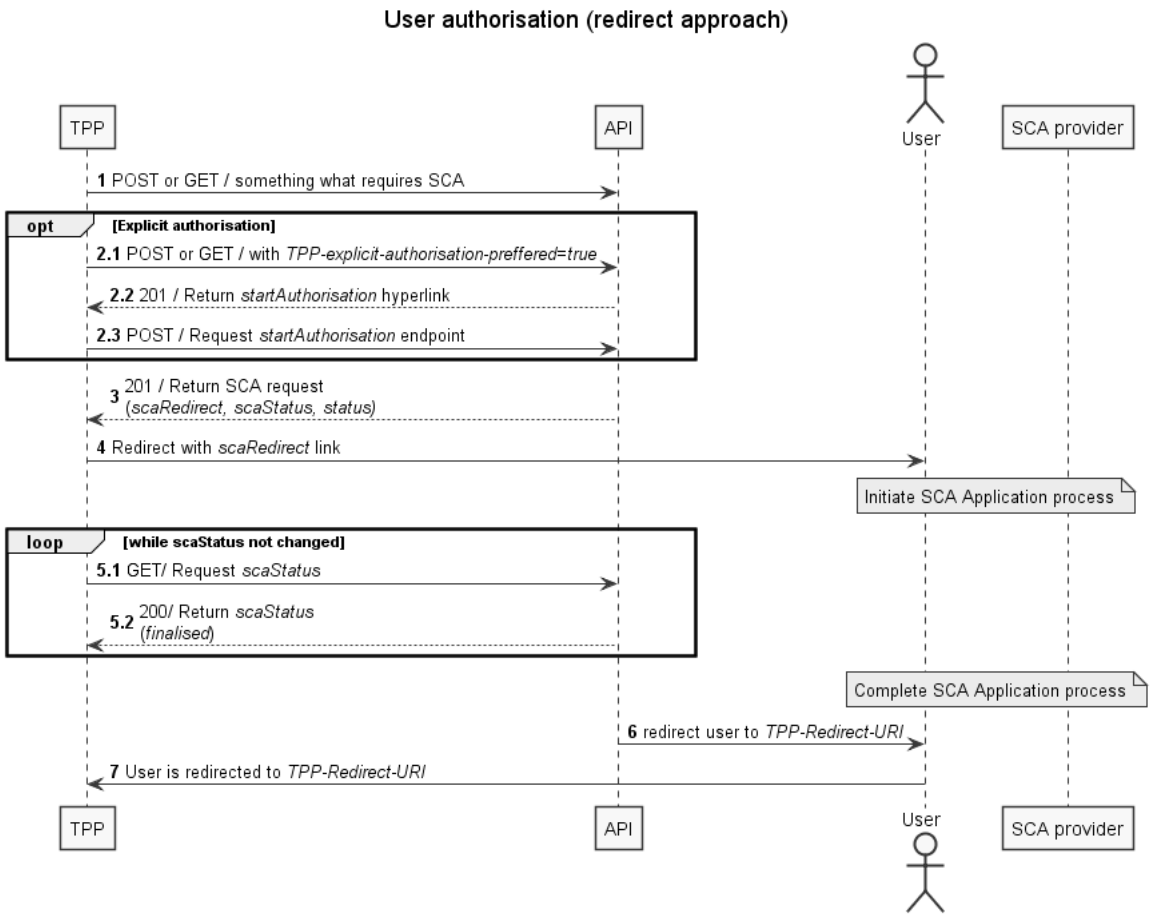
Country	Sweden (SE)/ Saving banks	Latvia (LV)	Estonia (EE)	Lithuania (LT)
Smart-ID	-	R/D	R/D	R/D
Mobile-ID	-	-	R/D	R/D
Card-ID	-	R	R	-
Mobile BankID	R/D	-	-	-
BankID on card	R	-	-	-
Security token	R	R	R	R
Biometric data*	-	R/D	R/D	R/D

> R – Redirect approach; D – Decoupled approach; * - limited to login and payment (up to 100 EUR 10 times in a row) signing;

Authorisation redirect approach

In redirect integration method TPP needs to provide redirects URIs in the `TPP-Redirect-URI` header in case when SCA is required. Redirects happen after the user has completed the SCA process in Swedbank API user interface (UI) and have to be redirected back to TPP.

> `TPP-Redirect-URI` header is required for all SCA requests in redirect integration method.



1. Any request which requires SCA is sent.
2. [Optional] Explicit authorisation
    1. If `TPP-explicit-authorisation-preferred` is set to 'true', explicit authorisation is started.

    Note: Explicit authorisation allows more detailed control of authorisation process needed for decoupled integration method or countersigning (check section Authorisation decoupled approach). It is not advised for redirection integration.

    1. Response will have steering link in `startAuthorisation` parameter and TPP must request it.
    2. Start authorisation and return `selectAuthenticationMethod` link. More technical details are available on Developer portal.

3. If request requires user SCA and `TPP-explicit-authorisation-preferred` is omitted or false in response `_links` object `scaRedirect` link is returned.
4. Redirect the user using `scaRedirect` steering link to Swedbank UI environment, where user completes SCA flow.
5. Check `scaStatus` every 5s until value 'finalised' or 'failed' is received

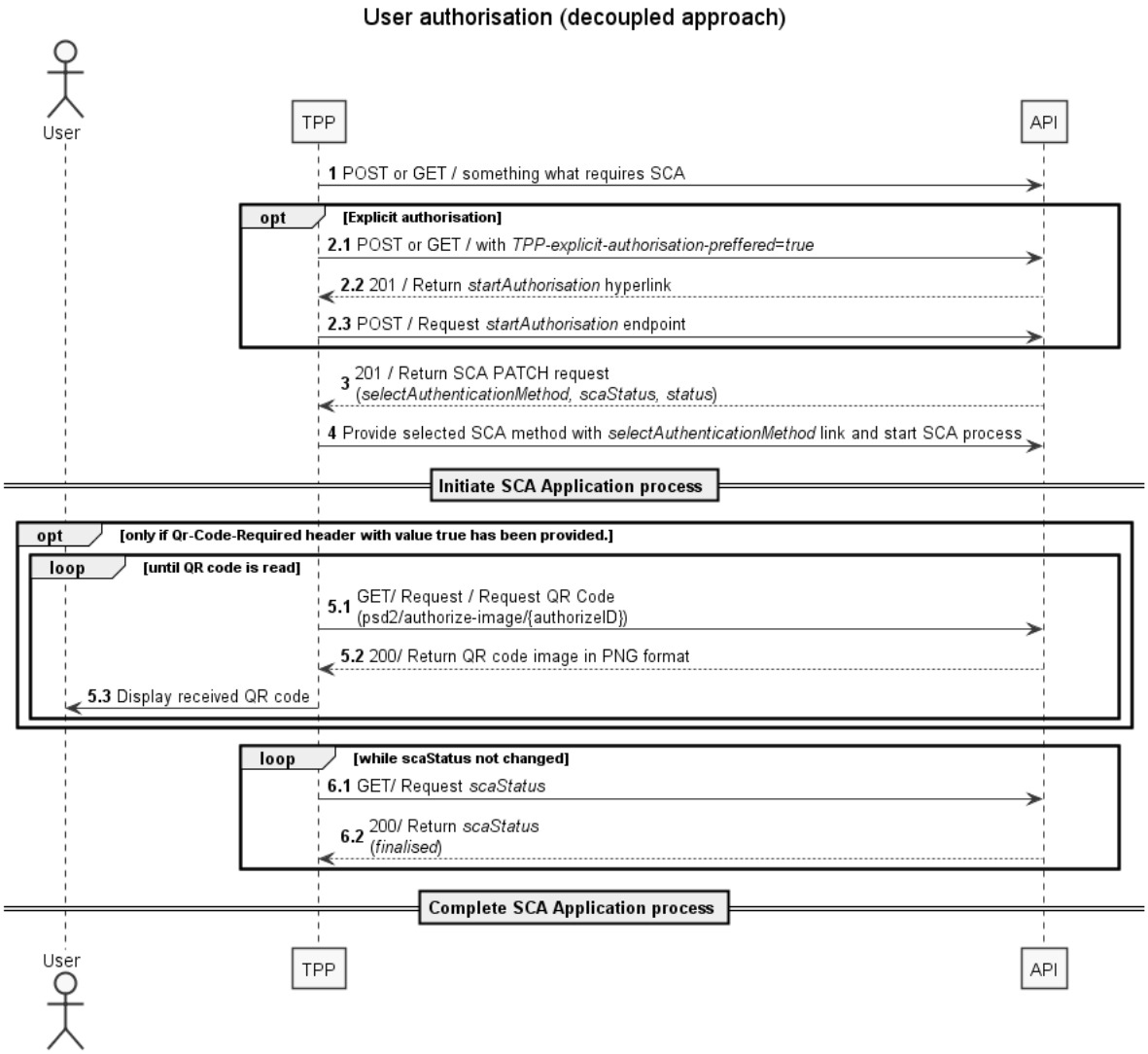
    1. TPP requests `scaStatus`;
    2. Response with `scaStatus:finalised`;
       If authorisation has failed, new authorisation may be created and processed.
6. After successfully completing the SCA flow the user will be redirected to the URL provided earlier in the `TPP-Redirect-URI`.
7. TPP continues usual flow;

> **Please note** that authorisation resource will be created automatically (unless specified otherwise over `TPP-Explicit-Authorisation-Preferred` header) by the Swedbank after the submission of the endpoint POST method. Explicit authorisation should only be used when decoupled approach is selected or when countersigning of the payment is required.

Authorisation decoupled approach

In order to start decoupled SCA approach TPP needs to provide header parameter `TPP-Redirect-Preferred` with value 'false'. With decoupled SCA approach Swedbank will ask the user to do SCA via dedicated SCA application which is independent from the online banking front end. TPP must inform the user about SCA by sending a corresponding user messages and challenge code.

After the SCA having been processed between Swedbank and user, the TPP then needs to ask for the result of transaction to be completed.



1. Any request which requires SCA is sent.
2. Explicit authorisation
    1. [Optional] If `TPP-explicit-authorisation-preferred` is set to 'true', explicit authorisation is started. Explicit authorisation allows more detailed control of authorisation process needed for decoupled integration method or countersigning.
    2. Response will have steering link in `startAuthorisation` parameter and TPP must request it.
    3. Start authorisation and return `selectAuthenticationMethod` link. More technical details are available on Developer Portal.
3. If request requires user SCA and `TPP-explicit-authorisation-preferred` is omitted or 'false' in response `_links` object `selectAuthenticationMethod` link is returned.
4. Provide selected SCA method by using the `selectAuthenticationMethod` PUT to Swedbank environment, this will update user's selected SCA method and will start SCA process by sending a challenge to user's SCA application.
   Note: TPP must check whether customer's IP address match Bank ID service – in case it doesn't, `Qr-Code-Required:true` header must be present in this request.
5. SCA process starts on user's device.

    1. [Optional] Request QR code image (more about QR code information could be found in QR code chapter);
    2. TPP gets image in PNG format;
    3. TPP displays received image to user. Image is updated every 1s.
6. Check `scaStatus` every 5s until value 'finalised' or 'failed' is received
    1. TPP requests `scaStatus`;
    2. Response with `scaStatus:finalised`;
       If authorisation has failed, new authorisation may be created and processed.
7. User completes authentication process.

> **Please note** that authorisation resource will be created automatically (unless specified otherwise over `TPP-Explicit-Authorisation-Preferred` header) by the Swedbank after the submission of the endpoint POST method. Explicit authorisation should only be used when decoupled approach is selected or when countersigning of the payment is required.

BankID

Swedbank in Sweden supports BankID (Smart card which is called *BankID on card*, mobile phone application - *Mobile BankID*) citizen identification solution.

QR code

Mobile BankID service has QR code authentication method which is described on www.bankid.com page. Mobile BankID identification is used for user authentication and authorisation flows.

Current Mobile BankID 5.1 versions has dynamic QR code support which means that QR code is refreshed every second in order to improve security.

Biometric data

Biometric in LT, LV and EE is based on usage of Swedbank mobile application. It is limited to mobile device biometric capabilities.

When biometric login process is started user gets notification on mobile device. Following this notification activates Swedbank mobile application for Biometric authorisation.

Application displays login message and hash code (must be displayed to user) and requires biometric confirmation (e.g. fingerprint or face recognition sensor).

After biometric confirmation is successfully done - user is redirected to TPP (or TPP will get result while calling `/scaStatus` endpoint).

Biometric signing

Biometric signing is applicable to payment signing limited up to 100 EUR, not more than 10 payments in a row. Counter is reset with other SCA method applied in Swedbank Mobile application. Different SCA method does not reset the counter in Internet bank or other channels.

> eg. 5 payments authorised with `SIMPLE_ID` (Biometric data) and then >100 EUR payment is authorised with `MOBILE_ID` in Internet bank. This doesn't reset the counter. eg. 8 payments authorised with `SIMPLE_ID` and then 5 EUR payment is signed in Swedbank Mobile application with `SMART_ID`. This resets counter and you can sign 10 payments with `SIMPLE_ID` again.

If user has created token using biometric data - payment is signed with biometric data too.

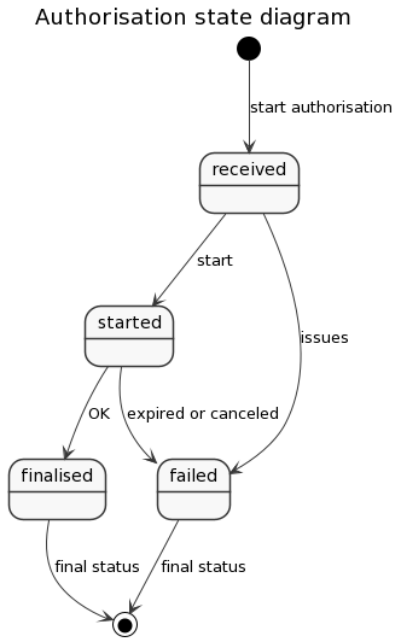
In redirect flow user is automatically offered to use previously applied signing method if payment can't be signed with `SIMPLE_ID` due to limititations described above.

In decoupled flow TPP gets `SIMPLE_ID` as `authenticationMethodId` if biometric can be applied otherwise previously used method.

Swedbank Mobile application displays signing message, challenge code (must be displayed to user) and requires biometric data confirmation (e.g. fingerprint or face recognition sensor). After biometric data confirmation is successfully done - user is redirected to TPP (or TPP gets result while calling `/scaStatus` endpoint).

Authorisation status

There are several states in authorisation flow. In order to make it more clear which states can transition and which are final we provide this state diagram. State is returned in the `scaStatus` parameter.



State descriptions

- `received` - An authorisation or cancellation-authorisation resource has been created successfully.
- `started` - The addressed SCA routine has been started. In a redirect approach PSU has followed scaRedirect URL, in a decoupled approach PSU started interaction in SCA app but not yet authorised.
- `finalised` - The SCA routine has been finalised successfully (including a potential confirmation command). This is a final status of the authorisation resource.
- `failed` - The SCA routine failed. This is a final status of the authorisation resource. This is a final status of the authorisation resource.

> TPP should not call for `scaStatus` after final state is received. There are TPPs which after receiving final `scaStatus` request status again and again for tens and hunderds of times.

Consent services

Customer consent is mandatory to access Account information services. Consent defines what kind of account information rules are agreed with User.

API supports these types of consent:

- allAccounts consent;
- Detailed consent.

> **Note:** Maximum consent validity period is 90 days.

Parallel consent support

It's important to note that from API v3 only one consent per Consent type (allAccounts or Detailed) can be valid – creation of a new, the same type Consent, would invalidate the previous one. However, it wouldn't affect consent of another type.

> For example, there are two consents associated with a customer, allAccounts and Detailed consent. In case a new allAccounts consent would be created, previous one would expire automatically, but Detailed Consent – remain active.

allAccounts consent

According to legal requirements TPP is required to obtain customer's consent in advance of accessing account list. This consent is called **allAccounts consent**.

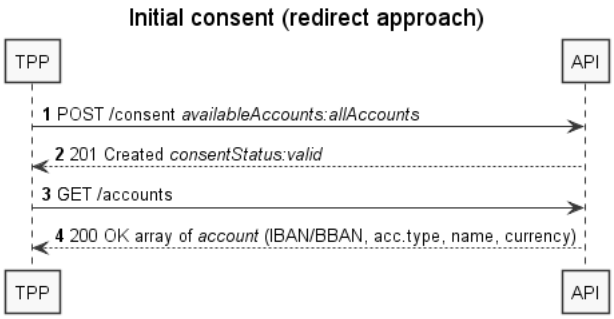
allAccounts consent will always be valid, `{API-type}account_list` scope is not mandatory. This scope is available only to AISP from 1st of March, 2021. That might be changed in the future, so to add this scope is recommended anyway.

With allAccounts consent TPP gets list of accounts with this information:

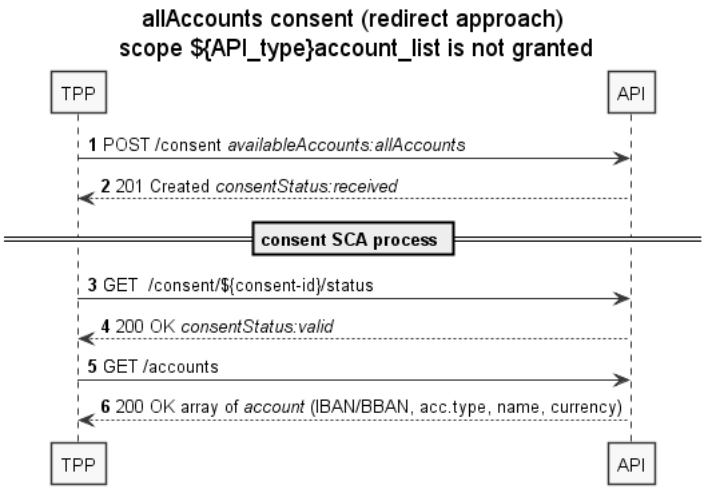
- Account number (IBAN);
- Account type;
- Account currency;
- Account alias.

allAccounts consent diagrams

In case scope `{API-type}account_list` is granted



In case scope `{API-type}account_list` is not granted



Only in Lithuania

Due to Lithuanian law about banking secrecy additional check must be done before giving access to customer account list. If TPP needs account list, TPP must express it by adding additional scope. TPP can get account list if scope `{API-type}account_list` is granted and TPP has valid OAuth 2.0 token. To get it TPP must create consent with `availableAccounts` defined (consent will be automatically valid) and use it in `/accounts` endpoint. This consent is non reccuring. This yields additional screen in customer's login flow to inform end user about agreeing to give out list of accounts.

If scope is not granted - accounts consent has to be approved by user with SCA.

In other countries, `{API-type}account_list` scope is not mandatory.

This works only for redirect flow. In decoupled flow additional scope is not supported in current API version.

In order to be able to get allAccounts consent TPP is required to add additional scope `{API-type}account_list` in Developer portal (**Publish/Apps/{YOUR_APP}/Edit/Auth**). It should be set like that: `{API-type} {API-type}account_list` (`{API-type}` should be replaced with PSD2 in these cases).

If in OAuth flow additional scope `{API-type}account_list` will be requested user will be presented with additional confirmation on giving out list of accounts to TPP. After user confirmation TPP gets consent with `allAccounts` as 'valid'

> Example: `https://psd2.lt.api.swedbank.com/psd2/authorize?response_type=code&client_id=CLIENT_ID&redirect_uri=SOME_URL&state=SOME_STATE&scope=PSD2 PSD2account_list`

Next step is to make consent request POST `/consents` with attribute `availableAccounts:allAccounts` in a request body.

Consent status is set to 'valid' if scopes were set properly and if user has approved it, then TPPs can fetch available accounts.

If in OAuth flow scope `{API-type}account_list` is not requested or granted - user is not presented with confirmation on giving out list of accounts to TPP. TPP receives consent with `allAccounts` only if user approves it with SCA. To redirect user back to TPP `redirect_uri` header parameter is required with such requests.

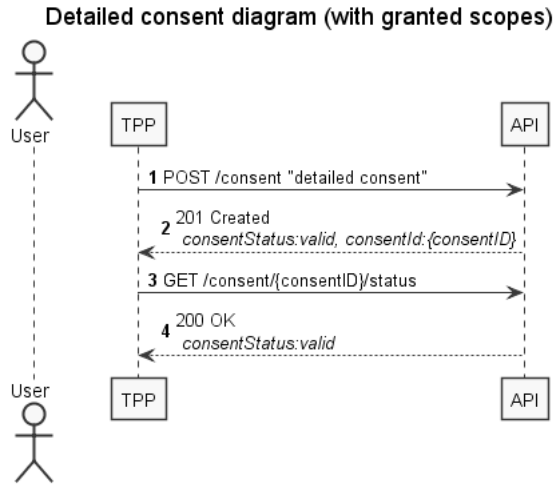
Detailed consent

API uses **Detailed Consent** approach as described in BGS, 6 Account Information Service, Consent Models. Please refer to our Developer Portal->Develop for technical details on each endpoint, such as data models, response/request bodies and headers.

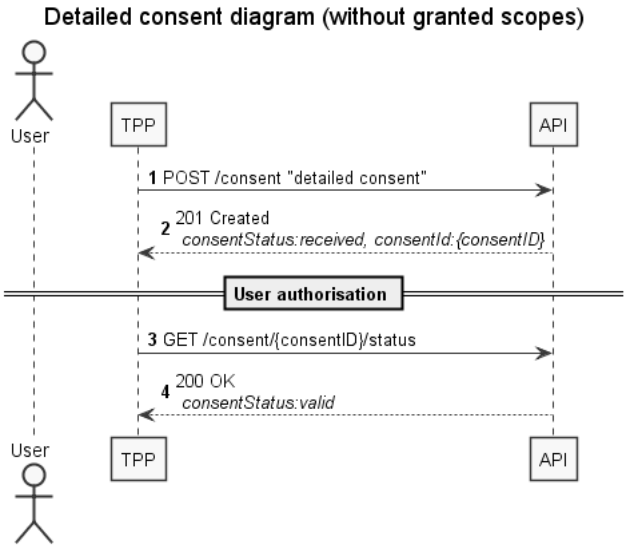
In order to get account details, TPP must create detailed consent. Detailed consent will be auto valid if at least one of scopes `{API-type}account_balances` (Latvia and Sweden only), `{API-type}account_transactions` (Latvia and Sweden only) or `{API-type}account_transactions_over90` (Sweden only) are granted. Otherwise, if TPP requests to access more information than granted by scopes, detailed consent must be approved by user with SCA. During SCA TPP submits the detailed consent information – user identification, services and account numbers affected – to Swedbank for authorisation by the user. Consent details are displayed to the user, when user performs SCA.

Detailed consent diagrams

In case needed scopes are granted



In case needed scopes are not granted



Account information services API

Account information endpoints should be used by TPP to get list of accounts and related information. Account API is available to receive information which can be categorised in the following categories:

- Account list
- Balances
- Transaction history/statement

All calls to `/accounts` endpoint requires valid consent (see Consent services chapter). TPP can access the same payment account information as the user can access through another digital channels. This includes the following data from accounts:

- Account information (account number, IBAN, account type by ISO20022, allowed currencies, account name);
- Available and booked account balances (currency, amount, date);
- Account statement (booking date, transaction date, value date, payment details, amount, and transaction currency). Account statement is provided in JSON format (for Baltic banks CAMT.052, CAMT.053 formats are supported) as described in BGS.

Please refer to our Developer Portal->Develop for technical details on each endpoint, such as data models, response/request bodies and headers.

Account information request limits

Requests to the account information API does not require direct user involvement once the consent is given, but on such requests limit will be applied. Account information services can be called 4 times per 24h without user involvement. The headers `PSU-IP-Address`, `PSU-IP-Port`, `PSU-User-Agent`, `PSU-HTTP-Method` must be provided if the user is present at TPP environment has asked for account access in real-time. When user is involved, request limitation is not applied.

When consent expires (validity of the consent is described in Consent services), TPP needs to resubmit consent and request SCA with user involvement.

Verification of user and account owner

In API there is provided a possibility to help TPP having accounts endpoint access to verify user. TPP might have a need to verify user ID and customer ID/account owner. The functionality is especially valuable in redirect approach where user acts in the bank environment.

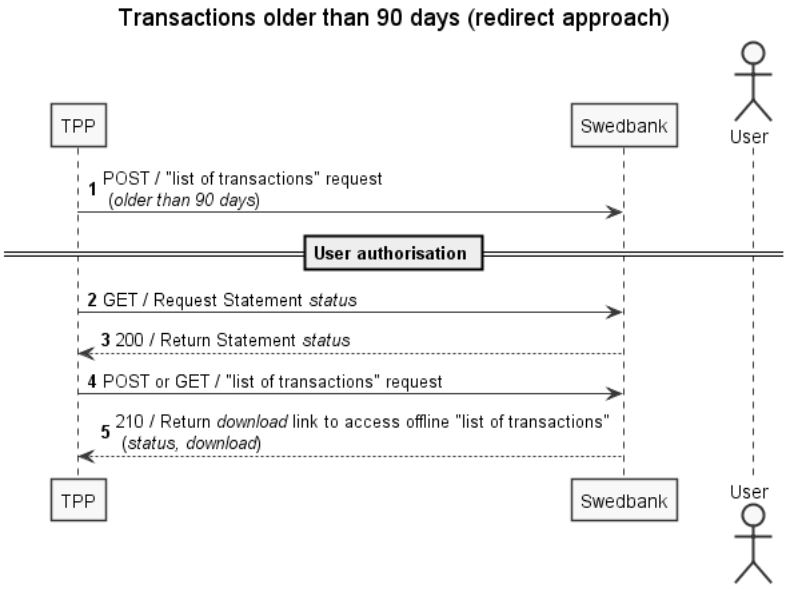
Verification can be done in `/accounts` endpoint by specifying headers:

- `PSU-ID` – specifying user id in the header TPP can verify if person connected to API is the same as provided by TPP. In Sweden `PSU-ID` is personal identification number while in Baltics it is generated number of user id.
- `PSU-Corporate-ID` – specifying the header TPP can verify if user acts on behalf of provided customer ID or if user is account owner. In other words, verification is performed to check if `PSU-Corporate-ID=Target-Reg-Nr`. `PSU-Corporate-ID` can be person or corporate. In both Sweden and Baltics this header will require person identification number or corporate identification number.

Please check `/accounts` endpoint in Developer Portal->Develop for technical details.

Transaction list with data older than 90 days

In order to get transactions with data older than 90 days, 'PSD2account_transactions_over90' scope can be used. It would allow TPP to request transaction data up to four times per 24 hours for each account (it is recommended to request transactions for a larger period of time rather than request smaller periods multiple times). Otherwise, in case scope above wasn't used, or in case TPP requests data for more than four times, user will be asked to approve (SCA) the request beforehand. In such case error is returned indicating that user SCA is required. Same schema is valid for large statements responding with big amount of data. Following schema has to be applied:



1. Initiate `POST /{version}/accounts/{accountID}/transactions` using same parameters as in GET request and with empty body. Start SCA with redirect approach as preferred integration method (see User authorisation using SCA).
2. Check `scaStatus` until value 'finalised' or 'failed' is received.
3. Check statement status and use it if it is 'signed'
4. Call either `POST` or `GET /{version}/accounts/{accountID}/transactions`.
5. This will now return a `HTTP 210` code with data containing `download_link`. Following this link will download list of transactions as zipped file in JSON or XML format. This is called Offline mode. If list of transactions when calling a download link is not available, `HTTP 428` code with the request to try later will be returned.

Payments initiation services API

Payment initiation endpoints should be used by TPP to initiate, authorize and start payment processing. In case of corporate user trying to initiate payment, his rights will be validated according user's access rights and limits to corporate accounts stored in corresponding agreement in bank.

Please refer to our Developer Portal->Develop for technical details on each endpoint, such as data models, response/request bodies and headers.

SE specifics

Duplicate payment

In Sweden two payments cannot have the same date, amount, debit account and credit account. This will lead to a Logical Duplicate error and payment will be rejected.

Add recipient

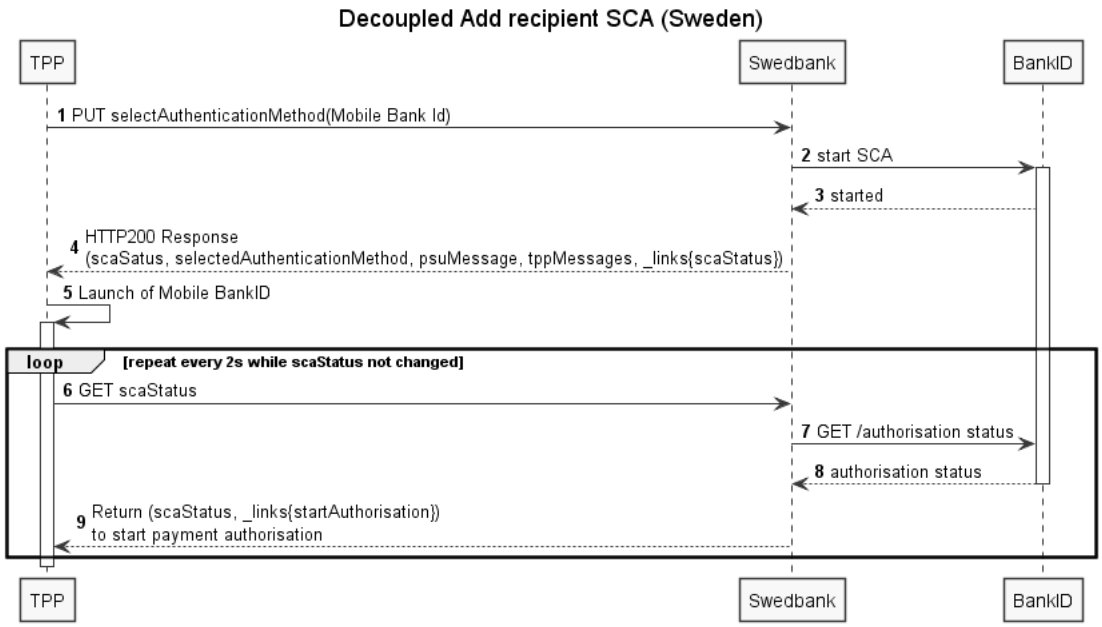
In Sweden Swedbank payment requires registration of recipient. If recipient is not registered user will have to add it so there are additional steps user and TPP must complete.

Add recipient in redirect approach

In redirect approach user must register new recipient in Swedbank UI in payment authorisation screen so SCA is required.

Add recipient in decoupled approach

In decoupled approach Add recipient flow control is done from TPP side.



Prerequisites for this flow

- user doesn't have recipient in the list;
- user authentication (section User Authentication using OAuth 2.0 decoupled) is already done;
- Payment initiation (section POST Single payment) is already done;

Decoupled Add recipient SCA diagram description

1. Initiate payment authorisation using `PUT /{version}/payments/se-domestic-credit-transfers/authorisation` with `selectAuthenticationMethod` selected.

   Note: TPP doesn't know whether user has recipient registered in the system so payment authorisation is expected by TPP in this step.

2. SCA is initiated with BankID.

   Note: In this step Swedbank identifies that recipient is not registered yet so starts recipient authorisation before payment authorisation.

3. Status is received that SCA successfully started.
4. Response code `200 OK` is returned. `psuMessage` should be delivered to user.

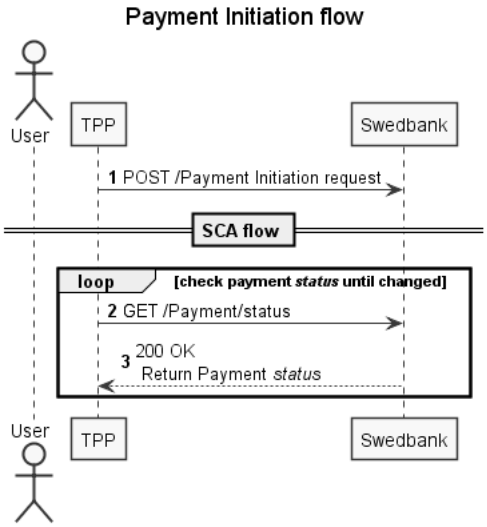
   Note: if `tppMessage` is returned, this means that Recipient authorisation is initiated, so TPP additionally will need authorise payment.

5. Mobile BankID application is launched on user device.
6. [loop] Check `scaStatus` until value 'finalised' or 'failed' is received. If authorisation has failed, but payment is correct, new authorisation may be created and processed.

   Note: initiated authorisation has expiration, so there is no sense to query for `scaStatus` longer than expiration time. For more information check User authorisation using SCA section.

7. BankID service is checked for status.
8. SCA status is returned to Swedbank.
9. If status changes to 'finalised' TPP can start decoupled payment authorisation by following `startAuthorisation` steering link (see section User authorisation using SCA). For any other status please check error messages for more information.

POST single payment



1. Initiate payment using `POST /{version}/payments/{payment-product}` with correct payment details provided in request body;
    - Start SCA according to preferred integration method.
    - Check SCA status until value 'finalised' or 'failed' is received. If authorisation has failed, but payment is correct, new authorisation may be created and processed.
    - Initiated authorisation expires after 5 mins, so there is no sense to query for `scaStatus` longer.
2. Check payment status by calling `GET /{version}/payments/{payment-product}/{paymentID}/status` endpoint;
    - RJCT – payment has an error. New payment with correct details has to be initiated.
    - ACSC – payment is authorised and is ready for settlement process.
    - ACCP – future dated payment is authorised. Settlement process will start on execution date, and then status will change to ACSC.
    - PATC – payment is authorised and requires counter signing. User may authorise it in Internet bank or additional authorisation must be created with OAuth token of countersigner and processed accordingly.
3. Payment status is returned;

Account selection on bank side

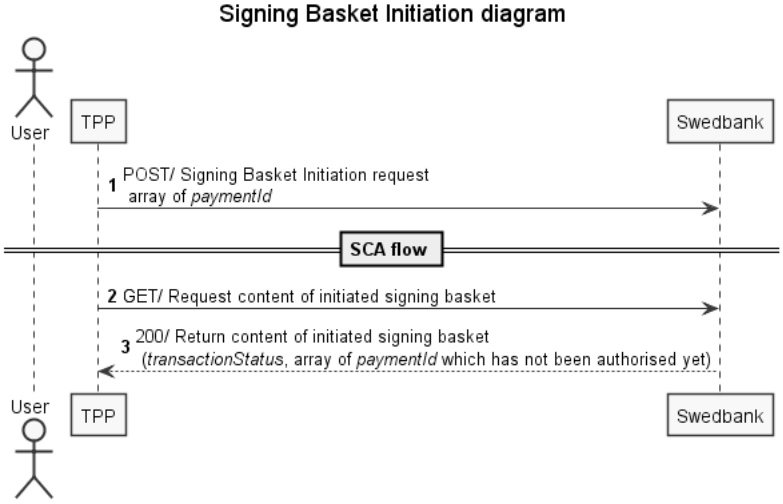
Single domestic payments can be initiated without debtor account details provided in a request. In such case, customer would be able to select desired account while signing payment on bank's side. However, such payments have some limitations:

1. Such payments cannot be executed in decoupled approach;
2. International payments are supported only in Baltic states;
3. SE specific: such payments cannot be put into a Basket - it can only be executed as a single payment.

After successful payment's execution selected debtor account will be set on payments object. If needed, `GET /payment/{paymentId}` endpoint can be called to fetch it. Please refer to Swagger for more information.

POST signing basket (SE only)

Payment Initiation Service offers creation of a signing basket resource for authorising several payments with one SCA. Signing basket is supported for Swedish banks only. Signing basket can be initiated on behalf of Swedbank Private or Corporate user for domestic transfers/payments only. Currently redirect mode only is available for basket payments. Maximum number of payments allowed in the basket is 100.



1. Initiate signing basket using `POST /{version}/signing-baskets` with correct list of `paymentId` provided in request body.

    Complete SCA according to preferred integration method. If authorisation has failed, but the signing basket is correct, new authorisation may be created and processed.

2. Check signing basket content by calling `GET /{version}/signing-baskets/{basketID}` endpoint.
3. Signing basket transaction status will be returned.
    1. Array of `paymentId` that are not signed will be returned.

Signing basket status can be checked by calling `GET /{version}/signing-baskets/{basketID}/status` endpoint.

Signing basket can be cancelled by calling `DELETE /{version}/signing-baskets/{basketID}` endpoint.

Recipient signing

All new recipients in payment basket can be authorised with one SCA. Maximum number of new recipients in the basket is 100. Please find more information for add recipient flow in chapter Add recipient.

Recurring/periodic payment

Payment Initiation Service offers initiation of a recurring/periodic payment request. Recurring/Periodic payments are always registered for a future payment date. Payment is executed a regular intervals after the first payment date.

Recurring/periodic payments in Baltics (redirect only)

Recurring payments in Baltics enabled for private and corporate customers, SEPA domestic payments in redirect approach only.

Recurring payment is set through `/periodic-payments/sepa-credit-transfers` endpoint by adding additional mandatory input parameters: `startDate` (first payment date), `frequency` and optional `endDate` parameter.

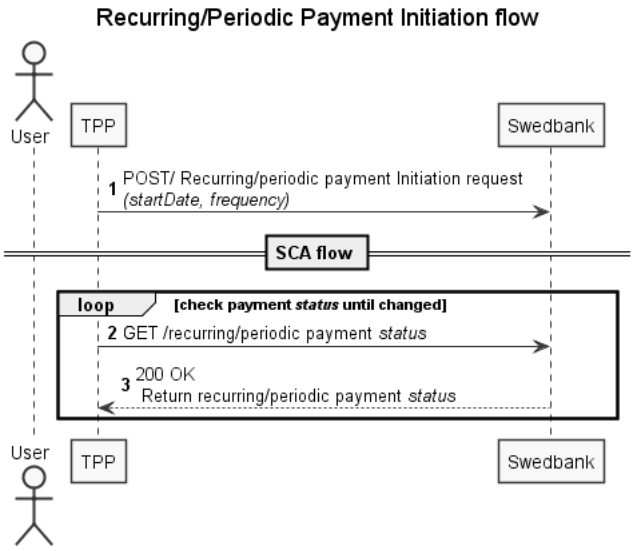
When concluding recurring payment agreement, customer should accept Swedbank service conditions.

Recurring/periodic payments in Sweden

Recurring/periodic payments in Sweden enabled for private and corporate customer, domestic and international in both redirect and decoupled approach.

Recurring payment is set through `/periodic-payments/{payment-product}` endpoint by adding additional mandatory input parameters: `startDate` (first payment date), `frequency`.

Recurring/periodic payment sequence diagram



1. Initiate payment using `POST /{version}/periodic-payments/{payment-product}` with `startDate`, `frequency` and other payment details provided in request body. Then start SCA according to preferred integration method.
2. Check SCA status until value 'finalised' or 'failed' is received.

    If authorisation has failed, but payment is correct, new authorisation may be created and processed.

3. Check payment status by calling `GET /{version}/periodic-payments/{payment-product}/{paymentID}/status` endpoint
    - RJCT – payment has an error. New payment with correct details has to be initiated.
    - ACSC – payment is authorised and is ready for settlement process.
    - ACCP – future dated payment is authorised. Settlement process will start on execution date, and then status will change to ACSC.
    - PATC – payment is authorised and requires counter signing. User may authorise it in Internet bank or additional authorisation has to be created with OAuth token of countersigner and processed accordingly.

Future dated payment

Payment Initiation Service offers initiation of a future dated payment request. Future payments are always registered for a future payment date.

Future dated payment is set through credit transfers endpoints (see more details in payment flow) by adding optional execution date parameter `requestedExecutionDate` YYYY-MM-DD which specifies when payment should be executed.

Execution date cannot be past and is max 1 year in the future.

Payment cancellation

Call `DELETE /{version}/{payment-service}/{payment-product}/{paymentId}` to delete the corresponding payment resource. PSU authorisation of deletion is not needed.

Payment cancelation process:

1. Initiate payment cancelation by calling DELETE: /{version}/{payment-service}/{paymentId};
2. Check payment status until you get CANC for successfully cancelation performed. In case payment can't be cancelled - error is returned. More technical details on Developer portal, API explorer.

Baltics specific

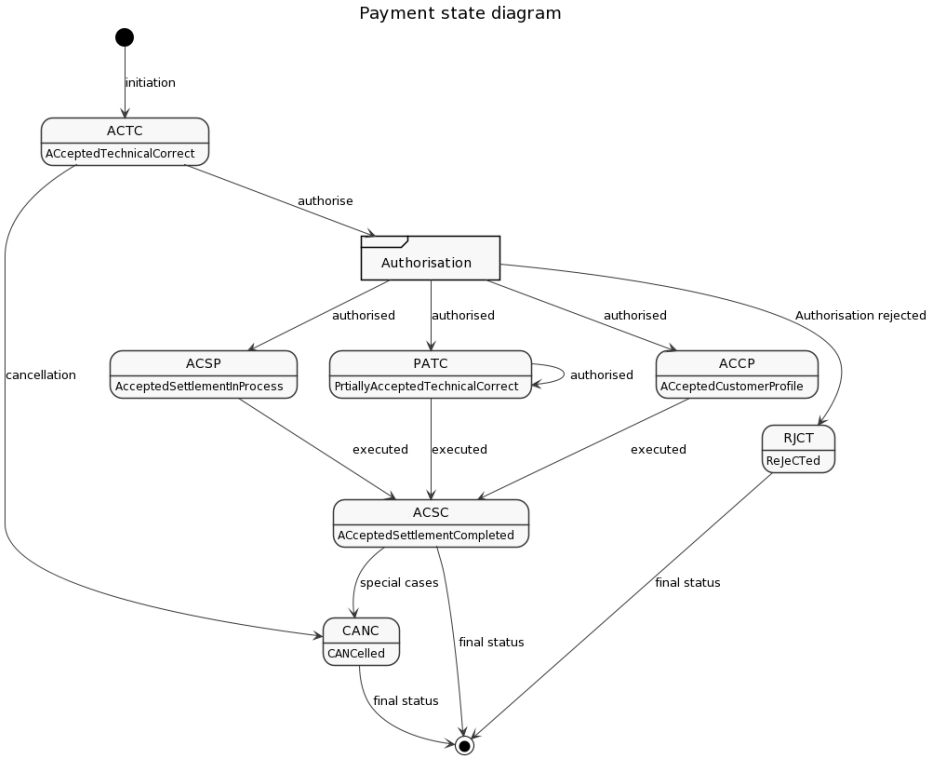
While payment is in status ACTC, ACSP or ACCP - TPP can cancel initiated or authorised immediate or future dated payment before payment execution. (In Latvia only - recurring payment before payment execution can be cancelled.)

Sweden specific

TPP can cancel initiated payment and if payment/transfer is not yet authorised by PSU (while payment resource is in status ACTC).

Payment status

In this section payment (transaction) life cycle is described and payment available states and transitions are provided. Every payment trasitions through it's life cycle from initial status to the final status. The overall life cycle transaction status of the payment. The transaction status is filled with codes of the ISO 20022



Status descriptions

- `ACTC` - AcceptedTechnicalValidation. Payment has been initiated successfully meaning that authentication and syntactical and semantic validation are successful. Now authorisation needs to be performed in order to finalize payment. More info in user authorisation chapter.
- `CANC` - Cancelled. Status of cancelled Payments. Payment initiation has been cancelled before execution. This is final status.
- `ACSP` - All preceding checks such as technical validation and customer profile were successful, but payment did not make it before cut-off time and trading room, therefore the payment initiation hasP been accepted for execution. This means that payment is being put on hold and is going to be processed once trading room is available again.
- `PATC` - Status of partially authorised payments within a multilevel SCA process (countersigning, multisigning). Payment initiations which are at least authorised by one PSU, but which are not yet finally authorised by all applicable PSU.
- `ACCP` - AcceptedCustomerProfile. Preceding check of technical validation was successful. Customer profile check was also successful. Payment has been successfully processed and prepared to be debited when future date becomes current date. Once payment is executed status changes to ACSC.

  Future dated payment gets this status after authorisation (signing). Recurring payment has this status until end date.

- `ACSC` - AcceptedSettlementCompleted. Authentication and syntactical and semantical validation is successful. Payment has been successfully processed and debited meaning that settlement on the debtor's account has been completed. This is final status except very special cases.

  Future dated payment gets this status on execution date when account is debited.

- `RJCT` - Rejected. Payment initiation or individual transaction included in the payment initiation has been rejected. This is final status.

  Please note that if payment is in final status it can't be transitioned to another status. This means that TPP should stop calling for payment status.

Requests signing

Certain API calls requires additional security by requesting request signing. QSEAL certificate is used by TPP to sign needed requests (usually HTTP POST requests related to initiation of payments), so it guarantees data integrity and ensures correctness of data origin. Signing of all requests may become mandatory in upcoming

releases.

When signature is included a digest header is required as described in RFC3230. The only allowed hash algorithms are SHA-256 and SHA-512.

The signature header must contain following fields:

Elements of the `Signature` Header

Element	Type	Condition	Requirement	Additional Requirement
keyId	string	Mandatory	The `keyId` field is an opaque string that the server can use to look up the component that is needed for the signature validation.	`Serial Number` of the TPPs certificate declared in Developer Portal and `CA name`. If `CA name` contains national characters due to the fact that HTTP headers are transferred using limited charset (ISO/IEC 8859-1) it should be in base64 encoding.
algorithm-ID	string	Mandatory	The `algorithm` parameter is used to specify the digital signature algorithm that is used to generate the signature.	The algorithm must identify the same algorithm for the signature as presented in the certificate (Element `keyId`) of this Request. It must be `rsa-sha256` or `rsa-sha512`.
Headers	string	Mandatory	The `headers` parameter is used to specify the list of HTTP headers included when generating the signature for the message. If specified, it should be a lowercased, quoted list of HTTP header fields, separated by a single space character. If not specified, implementations MUST operate as if the field was specified with a single value, the `Date` header, in the list of HTTP headers. Note that the list order is important and MUST be specified in the order the HTTP header field-value pairs are concatenated together during signing.	Mandatory. Must include: `Digest`, `X-Request-ID`, `Date_Optional`, `Date` Optional: `PSU-IP-Address`, `TPP-Redirect-URI` No other entries may be included.
Signature	string	Mandatory	The `signature` parameter is a base 64 encoded digital signature, as described in RFC 4648 [RFC4648], Section 4. The client uses the `algorithm` and `headers` signature parameters to form a canonical `signing string`. This `signing string` is then signed with the key associated with `keyId` and the algorithm corresponding to `algorithm`. The `signature` parameter is then set to the base 64 encoding of the signature.	Signature

Inclusion of a signature in payments request:

```
POST https://psd2.api.swedbank.com/Sandbox/v3/payments/sepa-credit-transfers
Content-Type:   application/json
X-Request-ID:   99391c7e-ad88-49ec-a2ad-99ddcb1f7721
PSU-IP-Address: 123.456.7.89
PSU-User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0
TPP-Redirect-URI:   HTTPS://client.example.com/cb&code_challenge_method="S256"
Date:   Sun, 06 Aug 2017 15:02:37 GMT
{
    "instructedAmount": {
      "currency": "EUR",
      "amount": "1"
    },
    "debtorAccount": {
      "iban": "LT657300010066666666"
    },
    "creditorAccount": {
      "iban": "LT187300010177777777"
    },
    "remittanceInformationUnstructured": "Test payment 2019-02-18T11:03:38.086",
    "endToEndIdentification": "string",
    "creditorAgent": "HABALT22",
    "requestedExecutionDate": "2019-02-18",
    "creditorName": "PERSON NAME"
}
```

Digest header is set to *[digest calculation algorithm] = [calculated body digest encoded in base64]*.

Only **SHA-256** digest algorithm is supported, therefore body in SHA-256 will equal to: `F6gJ/9CFG6cIe1kuR4xmuH8u7TVaC2K7hDixR2rRa8w=`

So header would look: `Digest: SHA-256=F6gJ/9CFG6cIe1kuR4xmuH8u7TVaC2K7hDixR2rRa8w=`

Signature header is created by combining:

- `keyId`="SN=[Signing certificate serial number],CA=***base64***([Signing certificate issuer DN name])",
- `algorithm`="[certificate algorithm type – rsa-sha256 or rsa- sha512]",
- `headers`="[part is space separated list of lowercase headers that are included into header signature]",
- `signature`="[base64 encoded `signingString` signature]".

`signingString` is combined of lowercase header names, in same order as provided in headers part, with values divided by Unix new line sign – "\n", e.g:

```
digest: SHA-256=F6gJ/9CFG6cIe1kuR4xmuH8u7TVaC2K7hDixR2rRa8w=
x-request-id: 99391c7e-ad88-49ec-a2ad-99ddcb1f7721
```

```
tpp-redirect-uri: HTTPS://client.example.com/cb&code_challenge_method="S256"
date: Sun, 06 Aug 201715:02:37 GMT
```

UNIX command for generating signature:

```
echo -en "${signingString}" | openssl dgst -sha256 -binary -sign client_signing.key -passin "pass:changeit" | base64 -w 0
```

> Please note that signing file will generate different signature than signing string.

When signature algorithm is used the signed request will be:

```
POST https://psd2.api.swedbank.com/Sandbox/v3/payments/sepa-credit-transfers
Content-Type: application/json
X-Request-ID: 99391c7e-ad88-49ec-a2ad-99ddcb1f7721
PSU-IP-Address: 192.168.8.78
PSU-User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0
TPP-Redirect-URI: HTTPS://client.example.com/cb&code_challenge_method="S256"
Date: Sun, 06 Aug 201715:02:37 GMT
Digest: SHA-256=F6gJ/9CFG6cIe1kuR4xmuH8u7TVaC2K7hDixR2rRa8w=
Signature: keyId="SN=9FA1,CA=Q049RC1UUlVTVCBDQSAyLTEgMjAxNSxPPURUcnVzdCBHbWJILEM9REU= ", algorithm="rsa-sha256", headers="d
{
   "instructedAmount": {
      "currency": "EUR",
      "amount": "1"
   },
   "debtorAccount": {
      "iban": "LT657300010066666666"
   },
   "creditorAccount": {
      "iban": "LT187300010177777777"
   },
   "remittanceInformationUnstructured": "Test payment 2019-02-18T11:03:38.086",
   "endToEndIdentification": "string",
   "creditorAgent": "HABALT22",
   "requestedExecutionDate": "2019-02-18",
   "creditorName": "PERSON NAME"
}
```

Common errors

In Swedbank Openbanking API errors are reported using HTTP status codes and parsing response body; In case of error (HTTP status code >= `400`) response body will contain element `tppMessages` with details of error. TPP should use combination of HTTP status code and `tppMessages[0].code` to identify error and make decission on how to proceed with error.

```
{
    "transactionStatus": "RJCT",
    "tppMessages": [
        { "category": "ERROR",
          "code": "AMOUNT_EXCEEDS_LIMIT",
          "text": "The amount exceeds the Monetary limit." }
        ]
    }
```

In case HTTP status code is in `4xx` range - means that request was incorrect and depending on `tppMessages[0].code` may be repeated or must be changed before repeating request. HTTP status code is in `5xx` range states errors on bank side and repeating request after some time may remove error.

Most frequent error codes and associated messages are sorted in ascending order by error code, text. In API v4 error codes and texts refinement is planned.

Common errors in any request

These errors can be found in any endpoint.

status	code	Text	action
400	FORMAT_ERROR	QWAC certificate revoked	QWAC certificate was revoked or expired. Please provide valid certificate.
401	TOKEN_UNKNOWN	Could not match OAuth token to TPP - authorization_link: ${url}	Technical refresh of OAuth2.0 token is needed. May happen on any endpoint upon expiration or invalidation of access_token.
401	TOKEN_INVALID	OAuth token not valid for given service or resource - authorization_link: ${url}	Login must be done to get new OAuth2.0 token
401	MISSING_BANK_AGREEMENT	Internet bank agreement missing. Please direct customer to the bank to sign internet bank agreement	SE: Customer should review status of his internet bank agreement.
401	MISSING_BANK_AGREEMENT	Internet bank agreement is not active or none eligible accounts available	BB: Internet bank agreement not active or access rights to accounts are missing. Please direct customer to the bank to review internet bank agreement

401	MISSING_CT_AGREEMENT	Credit transfer agreement for a payment/transfer missing	No valid agreement for this type of payment/transfer, please review internet bank agreement
403	KYC_INVALID	Please direct customer to the online banking to fill KYC.	SE: Customer should visit internet bank or branch and fill in KYC questionnaire.
403	SERVICE_BLOCKED	Unknown certificate used for SSL	Declare SSL certificate in Developer portal and manke sure it is enabled
404	RESOURCE_NOT_FOUND	The addressed resource not found.	The addressed resource not found
429	TOO_MANY_PARALLEL_REQUESTS	Too many parallel invocations of endpoint.	Too frequent invocation of endpoint.
429	TOO_MANY_REQUESTS	Too frequent invocation of endpoint.	Too frequent invocation of endpoint. Please wait before repeating request.
200	RESOURCE_EXPIRED	Requested resource has expired	Resource can't be used any more as it had expired
500	INTERNAL_ERROR	Internal ERROR	Internal technical error or general service unavailability. Retry after some time.
500	INTERNAL_ERROR	Internal server error. Reference: ${error_ref}	Internal technical error or general service unavailability. Retry after some time.

Common errors in authentication and authorisation

These are common errors found in OAuth2.0 related endpoints or resource signing process.

status	code	Text	action
200	USER_INTERUPTION	Interrupted by user	Login or sign process is cancelled by user.
200	UNSUCCESSFUL_SIGNING	Signing was unsuccessful	Signing of payment or other resource has failed
200	AUTHORIZATION_FAILED	Mobile id challenge expired or cancelled	SE: Mobile id challenge expired or cancelled.
200	NEW_BANKID_AUTH_OCCURRED	New BankID authentication occurred	SE: Another Mobile id authentication had started, this one is had failed.
200	EXPIRED_TRANSACTION	The session of signing has timed out.	The session of signing has timed out
200	USER_CANCEL	You have cancelled the signing by Mobile-ID in your phone.	User had cancelled sign process
200	USER_CANCEL	You pressed the "cancel" button on your phone, which terminated signing.	User had cancelled sign process
200	USER_CANCEL	Login process canceled by PSU	Login process canceled by PSU
200	USER_REFUSED	User cancelled request in Smart-ID app.	User cancelled request in Smart-ID app
200	MOBILE_ID_EXCEPTION	To add a new recipient, activate Mobile BankID for extended use.	Visit internet bank or branch and extend Mobile BankID.
400	USER_ID_NOT_VALID	Incorrect code, user ID or personal ID	BB: incorrect user data
401	USER_NOT_FOUND	Login unsuccessful! Please try again.	Login unsuccessful due to invalid user-id format.
400	FORMAT_ERROR	Wrong UserId parameter	You have entered an incorrect social security number, please try again.
400	FORMAT_ERROR	Provided refresh_token expired	In endpoint /psd2/token?grant_type=refresh_token technical refresh of token had failed due to refresh token expiration or due to refresh_token invalidation. Repeat login process to get new token.
400	FORMAT_ERROR	Provided authorization code expired	In endpoint /psd2/token?grant_type=authorization_code - timeout occured for login session. Repeat login process to get new token.
400	FORMAT_ERROR	The given client credentials were not valid	In endpoint /psd2/token?grant_type=authorization_code - timeout occured for login session. Repeat login process to get new token.
400	FORMAT_ERROR	No profile available	SE: during login process specified target reg. nr. is incorrect or there is no active profile
400	FORMAT_ERROR	Unknown authorizationId	Authorization had expired or id is invalid
400	FORMAT_ERROR	sessionData expired, please start authorisation again	Ussually identifies timeout in login process. Please repeat login.
400	FORMAT_ERROR	Other login session is ongoing	Please wait for until ongoing session will expire
400	FORMAT_ERROR	Given redirect_uri () does not match the initial code request one.	redirect_uri must match the one declared in developer portal
401	USER_NOT_VALID	Incorrect user ID, code, personal code or phone number	Incorrect user ID, code, personal code or phone number
401	USER_RIGHTS_ON_ACCOUNT	Not authorized to perform the request	Insufficient user rights to sign credit transfer
403	RESOURCE_ALREADY_SIGNED		Resource is allready signed
403	FORMAT_ERROR	Wrong PSU-ID and/or PSU-CORPORATE-ID	Login failed due to incorrect PSU* parameters
404	SERVICE_BLOCKED	The addressed authorisation resource is unknown	The addressed authorisation resource is unknown

409	MOBILE_ID_CHALLENGE_CONFLICT	MOBILE_ID_CHALLENGE_CONFLICT	SE: Another Mobile id authentication had started, this one is had failed.
500	INTERNAL_ERROR	Could not decode sessionData	Ussually identifies timeout in login process. Please repeat login.

Errors in consent/accounts flow

These are common errors found in `/consent` and `/accounts` endpoints.

status	code	Text	action
400	FORMAT_ERROR	Format of certain request fields are not matching the XS2A requirements due to {0} Invalid	Incorrect format of request field
400	INVALID_REQUEST	Consent expired	New consent has to be created.
400	INVALID_REQUEST	Consent not valid	Invalid consent id
400	INVALID_REQUEST	validUntill date is wrong.	validUntil date is wrong
400	INVALID_REQUEST	validUntill is in past.	validUntil date is wrong
400	INVALID_REQUEST	Consent can not be deleted, as it is in ${status} status.	Consent is incorrect state
400	BAD_REQUEST_DATA	Bad request data No available accounts	No suitable accounts for consent
400	INVALID_REQUEST	validUntill exceeds ${days} days period.	Time limit for consent is exceeded
401	SCA_REQUIRED	Statement requires SCA	Statment with transactions older than 90 days requires user approval
401	CONSENT_EXPIRED	The consent was created by this TPP but has expired and needs to be renewed.	New consent has to be created.
401	CONSENT_INVALID	No valid consent found	Consent is invalidated or expired.
401	CONSENT_INVALID	No consent was found	Consent is invalidated or expired.
401	CONSENT_INVALID	No consent for account ID ${accid} ${detail}	Current consent does not allows to access account details
401	CONSENT_INVALID	Consent with expiration date ${date_until} is not valid.	Invalid consent body. Date until is in past.
403	RESOURCE_UNKNOWN	Verification of user or owner of account has failed	Account owner or user specified in PSU* headers are not matching real ones. Check APi documentation.
403	RESOURCE_UNKNOWN	The addressed resource is unknown.	Specified resource is incorrect or not valid anymore
404	RESOURCE_UNKNOWN	Unknown offline statement resource	Unknown offline statement resource
428	RESOURCE_PENDING	Statement is not yet available. Try again later	Statment generation is in progress, please repeat request later
428	PRECONDITION_REQUIRED	Resource is not ready. Please try again later	Statment generation is in progress, please repeat request later
429	ACCESS_EXCEEDED	The access on the account has been exceeding the consented multiplicity per day.	Limit of requests performed without active customer request is exceeded, if consent is reccurent it will reset limit in 24h.
429	ACCESS_EXCEEDED	Consent daily limit ${counter} is exceeded for ${account} account service [${TRANSACTIONS, ACCOUNT_STATEMENT, AVAILABLE_BALANCE, ACCOUNTS_ACCESS}]	Repeat request after reccurent consent reset.

Errors in payments flow

These are common errors found in `/payments` endpoint.

status	code	Text	action
200	INVALID_CT	The payment/transfer contains errors	Payment details are incorrect, new payment with correct details has to be created.
201	RECIPIENT_MISSING	PSU must add recipient to recipient list first.	Please confirm the recipient to make the payment. When you have added the recipient, the payment can be made. The recipient will thereafter be stored in your recipient list. This can be done using API or internet bank.
200	NARR	debitHasFailed	BB: payment was signed, but debiting failed. May happen due to multiple reasons.
400	AG01	Account not allowed	BB: account is not available to payments due to agreement limitations
400	AG01	Double accept is required for account	BB: payment has to be countersigned (*functionality not available now).
400	AG01	Debtor account service not allowed	BB: account is not available to payments due to agreement limitations
400	AC03	Invalid creditor account number	BB: creditor account is incorrect or not suitable for payments
400	BE01	The account number and the name do not coincide!	Creditor account number and creditor name do not coincide!
400	AM04	Insufficient funds	BB: payment amount with fee is bigger than funds available
400	NARR	dailyLimitExceeded	BB: daily payments limit defined in bank agreement is exceeded

400	NARR	instantPaymentAmountTooBig	BB: limit of instant payment schema is exceeded
400	NARR	monthlyLimitExceeded	BB: monthly payments limit defined in bank agreement is exceeded
400	NARR	notEnoughFundsForFee	BB: payment amount with fee is bigger than funds available
400	NARR	notEnoughFundsForFeeinServiceAccount	BB: payment amount with fee is bigger than funds available
400	CT_INVALID	The payment/transfer contains errors : creditor_account	Payment details are incorrect, new payment with correct details has to be created.
400	CT_INVALID	The payment/transfer contains errors : debtor_account	Payment details are incorrect, new payment with correct details has to be created.
400	CT_INVALID	The payment/transfer contains errors: {field_name}	Payment details are incorrect, new payment with correct details has to be created.
400	BAD_REQUEST_DATA	Bad request data : date is not in the future	Future payment date is not in future
400	INVALID_REQUEST	Wrong payment id	Incorrect payment id in request
400	INVALID_RECIPIENT	The recipient account number is invalid	SE: The recipient account number is invalid
401	EXCEEDED_SIGNED_AMOUNT	The amount limit for signing is exceeded	Payment amount is too big to be performed due to limits defined in Internet bank agreement.
401	INSUFFICIENT_FUNDS	The payment/transfer could not be completed because of insufficient funds.	Balance is to low to make a payment. Topu p balance in order to proceed with payment signing.
401	EXCEEDED_PAYMENT_LIMIT	The payment limit per day is exceeded.	The allowed payment limit per day has been reached. Payment cannot be signed.
401	INSUFFICIENT_FUNDS	The payment/transfer could not be completed because of insufficient funds	Payment amount is bigger than account balance or account limit.
403	SERVICE_BLOCKED	Canceling of recurring payment is not allowed	BB: Canceling of recurring payment is not allowed

Most common errors in integration phase

These errors indicate errors in integration with API.

status	code	Text	action
400	FORMAT_ERROR	Mandatory parameter bic is missing or has unsupported value	Mandatory parameter bic is missing or has unsupported value
400	FORMAT_ERROR	Parameter dateTo is in future	Request to /transactions endpoint
400	FORMAT_ERROR	Client_id missmatch on OAuth2.0 invokation and token exchange	Errors in OAuth configuration or implementation
400	FORMAT_ERROR	Missing parameter refresh_token	Missing parameter
400	FORMAT_ERROR	Mandatory header ${header name} is wrong format	Header has to have correct value
400	FORMAT_ERROR	Mandatory header is missing: ${header name}	Header has to be specified
400	FORMAT_ERROR	Consent request schema validation failed: ${json_schema_error}	Invalid json provided
400	FORMAT_ERROR	Payment product ${payment_product} schema validation failed: ${json_schema_error}	Invalid json provided
400	FORMAT_ERROR	Format of certain request fields are not matching the XS2A requirements due to null or missing data	Invalid request due to missing or invalid parameters
400	FORMAT_ERROR	No certificate provided in developer portal, in Application setup	Endpoint requires proper certificates configured and enabled
400	FORMAT_ERROR	Scope(s) (${scopes}) needs main scope specified	Invalid scopes specified in /psd2/authorize request
400	FORMAT_ERROR	Missing required parameter or dublicate	Missing required parameter or dublicate
404	INVALID_REQUEST	Statement for transactions over 90 days must be posted first.	Check documentation for statements over 90 days
405	SERVICE_INVALID	Got: ${http_verb} allowed: ${http_verb}	Endpoint does not support specified HTTP verb
405	SERVICE_INVALID	HTTP method ${http_verb} not supported for ${endpoint}	Endpoint does not support specified HTTP verb

API versioning guidelines

Introduction

This chapter describes Swedbank API versioning guidelines in order to have a consistent approach towards versioning and to allow quick changes in APIs, the following guidelines are to be followed.

Versioning policy

- In API we follow principle of releasing minimum number of versions. API version stays the same if changed or new functionality does not break backwards compatibility.
- Try to keep the same version for all versioned API endpoints.
- Each change in API is tried to implement by not breaking backward compatibility. Such changes should not require TPP integration redevelopment and should not increase version number:

    - Adding of new functionality or new endpoint;
    - Adding optional parameters (fields, headers, attributes) in request and response;
    - Reducing or extending of enumeration range when used as output parameters;
    - Extending of enumeration range when used as input parameters;
    - Removal or loosening of constraint for input fields validated via server-side business logic;
    - Adding additional error message;
- New version is created in case of:
    - Removal of used endpoint or functionality;
    - Making mandatory field (fields, headers, attributes) in request;
    - Datatype or semantic of field changes;
    - Changing request validation logic makes it more restrictive;
    - Enumeration range is reduced when used as input parameters (if values of enumeration are not provided as list in output);
    - Removal of endpoint (or URI change);
- Only major version is used in API URI endpoint. E.g. https://psd2.api.swedbank.com/**v3**/payments/sepa-credit-transfers

Documentation and information about changes in API versions

Non-breaking changes

Timeline	Communication	Sandbox	Production
Release date	Newsletter, Developer portal, PDF documentation, Swagger	Non-breaking change V[n].x version available	Non-breaking change V[n].x version available

- When non-breaking changes are released, general documentation, technical documentation (swagger) and Sandbox environment for changes not requiring new major API version is released no later than Production date; Information is provided in Developer Portal and newsletter.

Breaking changes

Timeline	Communication	Sandbox	Production
Release date - 3 months	Newsletter, Developer portal, PDF documentation, Swagger	V[n] version availableV[n+1] version available	-
Release date	Newsletter, Developer portal	V[n] version availableV[n+1] version available	V[n] version availableV[n+1] version available
Release date + 1 month	Newsletter, Developer portal	V[n+1] version available	V[n+1] version available

- PDF documentation, technical documentation (swagger) and Sandbox environment for new major API version is released at least 3 months before Production release. Information is provided in Developer Portal and newsletter.
- API is planning to support not more than 2 major API versions in production in parallel;
    - Parallel running is planned for 1 month.
    - In special cases (if TPP makes request and it is technically possible) parallel run can be extended up to 3 months.
- Inform consumers when deprecating a version;
    - Consumers must be notified at least 3 months before deprecating a version.
    - Consumers must not start using deprecated parts of APIs.
    - Deprecation must be part of the API specification.
    - Set Sunset header for deprecated API's by specifying API's end date.