

# Swedbank PSD2 API developer documentation

- [Change log](#)
- [Definitions](#)
- [Introduction](#)
- [Connecting to API](#)
- [User authentication](#)
- [User authorisation using SCA](#)
- [Consent services](#)
- [Account information services API](#)
- [Payments initiation services API](#)
- [Requests signing](#)
- [Common errors](#)
- [Sandbox](#)
- [API versioning guidelines](#)
- [API v3 to v4 migration guidelines](#)

## Change log

We encourage providing feedback to improve our services by sending a mail to [openbanking@swedbank.com](mailto: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] <a href="#">Biometric data</a> support.
3.7	October 2020	Added <a href="#">common errors</a> chapter
3.7.1	November 2020	Notes about allAccounts availability only to AISP.
3.7.2	December 2020	Added state diagrams for <a href="#">Payments</a> and <a href="#">Authorisation</a> .
3.7.3	January 2021	Added additional info about errors in <a href="#">Common errors</a> chapter.
3.7.4	January 2021	Improvements and clarification to Future dated and recurrent/periodic payment sections.
3.8	March 2021	Introduced new feature: <a href="#">Account selection on bank side</a> .
3.9	March 2021	Introduced country based DNS for API usage for DNS problem mitigation. More info can be found on <a href="#">Sandbox and Production Connection overview</a> .
3.10	April 2021	[Latvia] Balances and transactions (up to 90 days) can be added as scopes to consent for less SCA.
4.0	July 2021	v3 to v4 migration changes and guidelines.

## 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 <a href="#">developer portal</a> .
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: <ul style="list-style-type: none"> <li>message parameters as part of the HTTP level (headers);</li> <li>message parameters by defining a resource path (URL path information);</li> <li>message parameters as part of the HTTPS body.</li> </ul>
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 <a href="#">API Explorer</a> .
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 requires 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 are 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 standards described in ISO20022 and the [NextGenPSD2 Access to Account Interoperability Framework - Implementation Guidelines V1.3.9 20210329.pdf](#) (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 2019 14:23:49 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. Please check [supported security devices](#).

# Connecting to API

## 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	SANDESS	SWEDSESS	-	<a href="https://se.psd2.api.swedbank.com">https://se.psd2.api.swedbank.com</a>
Lithuania	SANDLT22	HABALT22	<a href="https://psd2.api.swedbank.lt">https://psd2.api.swedbank.lt</a>	<a href="https://lt.psd2.api.swedbank.com">https://lt.psd2.api.swedbank.com</a>
Latvia	SANDLV22	HABALV22	<a href="https://psd2.api.swedbank.lv">https://psd2.api.swedbank.lv</a>	<a href="https://lv.psd2.api.swedbank.com">https://lv.psd2.api.swedbank.com</a>
Estonia	SANDEE2X	HABAE2X	<a href="https://psd2.api.swedbank.ee">https://psd2.api.swedbank.ee</a>	<a href="https://ee.psd2.api.swedbank.com">https://ee.psd2.api.swedbank.com</a>
Cooperating Savings Banks Sweden	SANDESS	SWEDSESS	-	<a href="https://se.psd2.api.swedbank.com">https://se.psd2.api.swedbank.com</a>

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

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.

## 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 PSD2 or PARTNER (e.g. {API-type} sandbox scope should be replaced with PSD2sandbox or PARTNERSandbox).

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](#).

## TPP onboarding on Developer portal

Onboard in the [Open Banking portal](#) by providing correct email. Please provide correct organisation name as it is 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 redirects the user back to the TPP application with authorisation code in the URI. Redirect URI must be on secure server (HTTPS) as OAuth2 flow sends sensitive data to it. When authorization is requested, the authorisation server validates 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 `client_secret` (`shared_secret`) is available. 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:

- BIC (check [connection overview](#) table);
- 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 does validation of signature and mutual TLS.

More details (including supported error scenarios) on Sandbox usage are provided in [Sandbox] [[#sandbox-details-chapter](#)].

## Production usage

For Production usage:

- BIC (check [connection overview](#) table);
- 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](mailto: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](mailto:openbanking@swedbank.com).

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

# User authentication

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

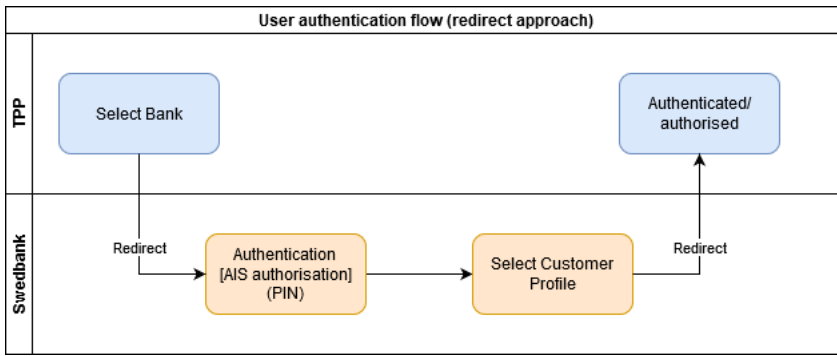
- As a private user person has access to customer owned accounts in the Swedbank;
- As a representative of a company user can only access accounts that user has been granted access to. Access rights to accounts are defined in the Internet bank agreement and transparently applied to API.

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:

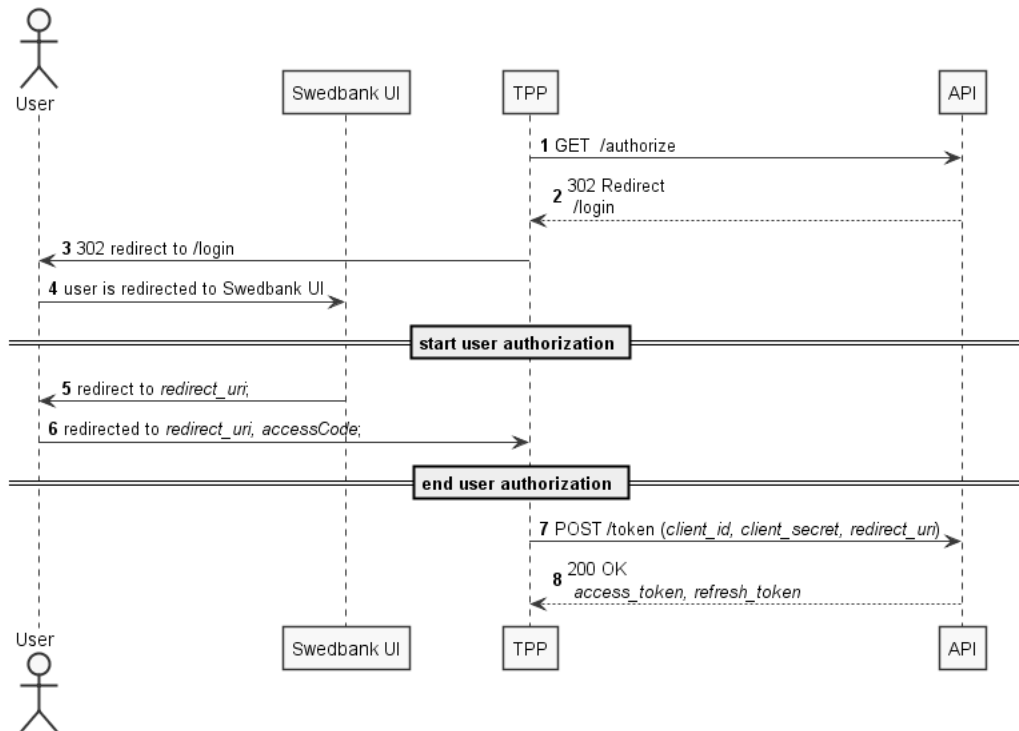
**[optional] AIS pre-authorization:** TPP can provide additional scopes (more details on [scopes chapter](#)) to get Account information service authorisation from user with single SCA;

**[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.

### Sequence diagram

#### User authentication diagram using OAuth 2.0 (redirect approach)



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 is 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?`

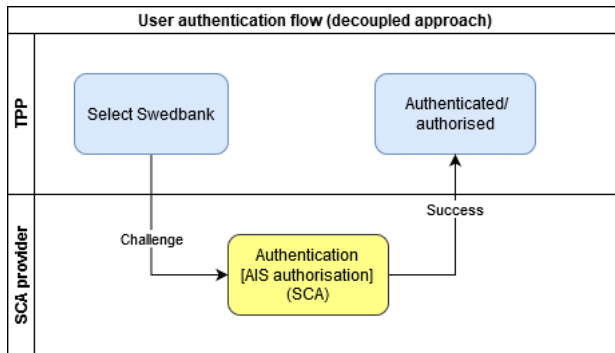
`grant_type=authorization_code&client_id=171c2ad437f0fd44d68b2ea3f89b76a28d&client_secret=079603ebf92446f8a70e0ddc11f8d18255-4c92-a975-4bd1916d4199&redirect_uri=https://www.swedbank.com/openbanking`

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

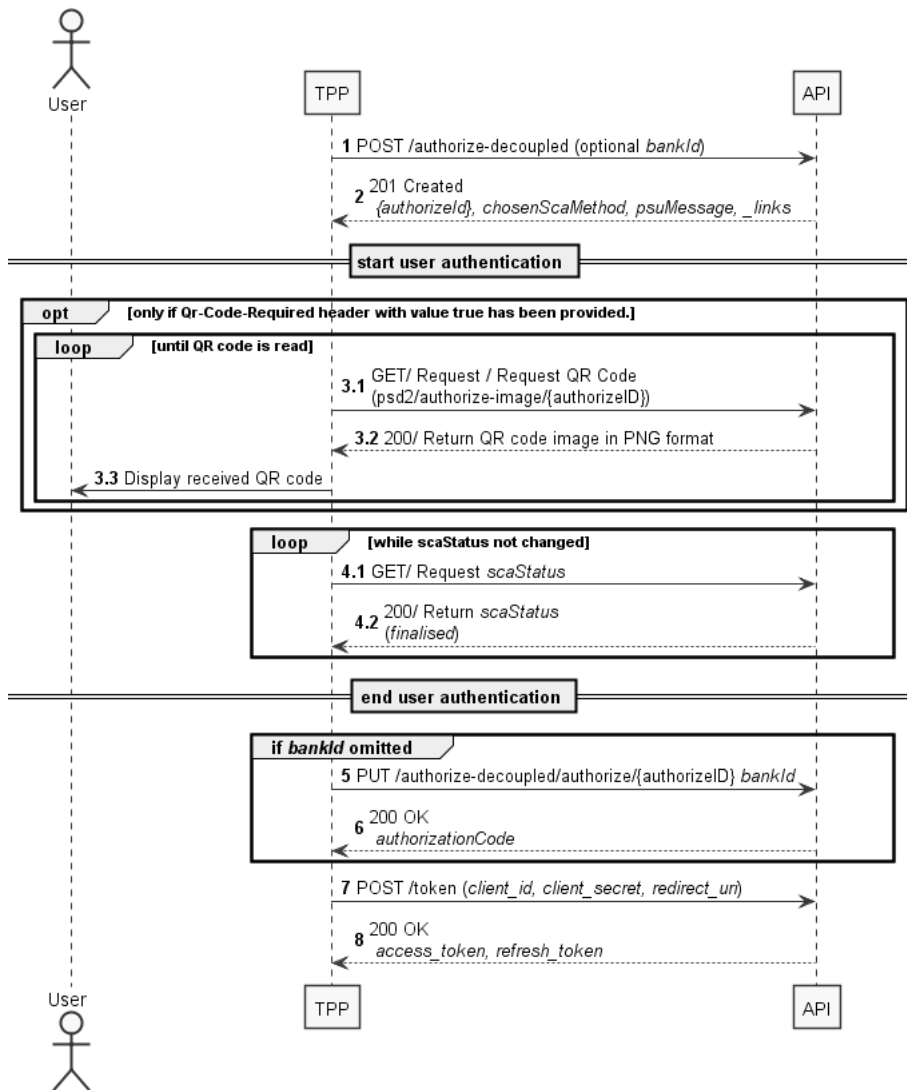


TPP may use API to get available login methods, participating Saving banks and other information needed for login process. This information should be cached and refreshed each 24 hours.

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

## User authentication diagram using OAuth 2.0 (decoupled approach)



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 should 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 once per new user authorisation. It grants access to transactions over 90 days data to selected accounts for one hour after consent's creation.

### PSD2 API available scopes

scope	SWEDSESS	HABALT22	HABALV22	HABAE2X
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 initiate 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` 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 is displayed to a user and the 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;

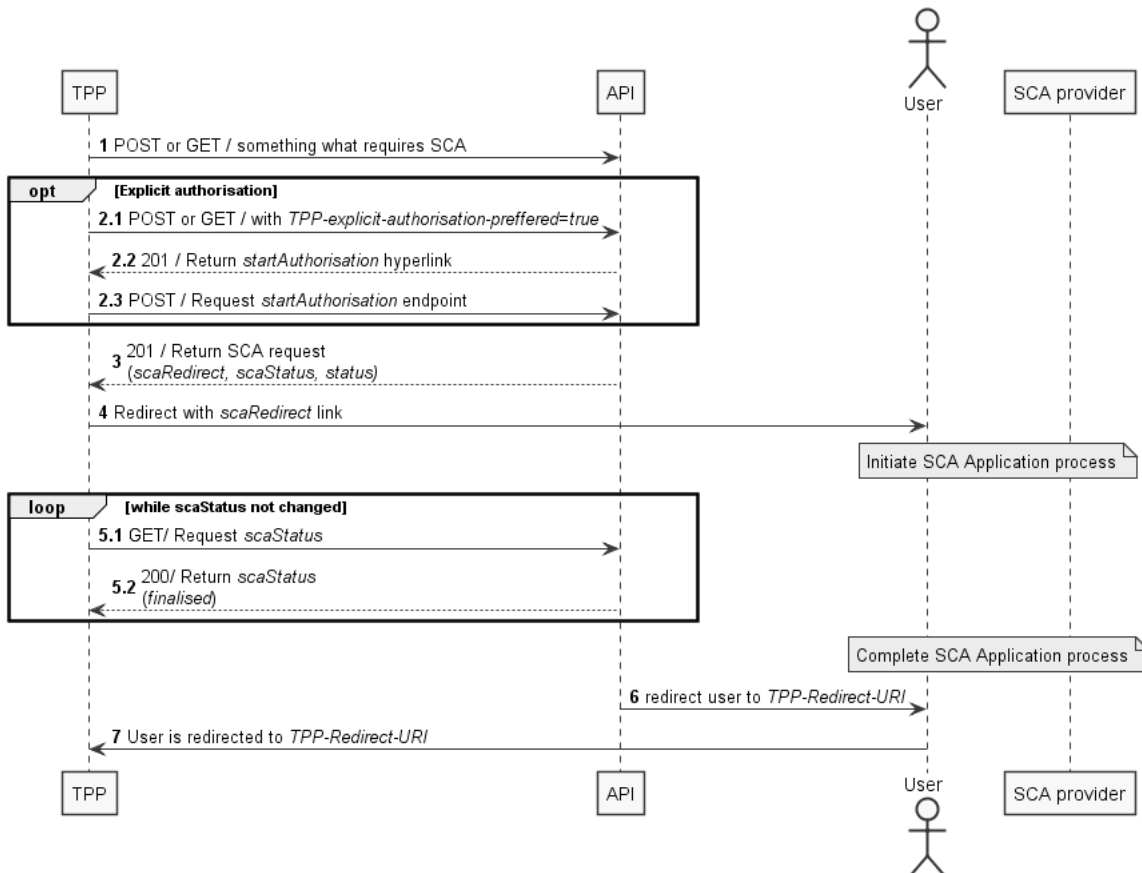
\* - limited to login and payment (up to 100 EUR 10 times in a row) signing;

## 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.

### User authorisation (redirect approach)



1. Any request which requires SCA is sent;

2. [Optional] Explicit authorisation:

1. If `TPP-explicit-authorization-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.

2. Response has 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 `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 is redirected to the URL provided earlier in the `TPP-Redirect-URI`;
7. TPP continues usual flow.

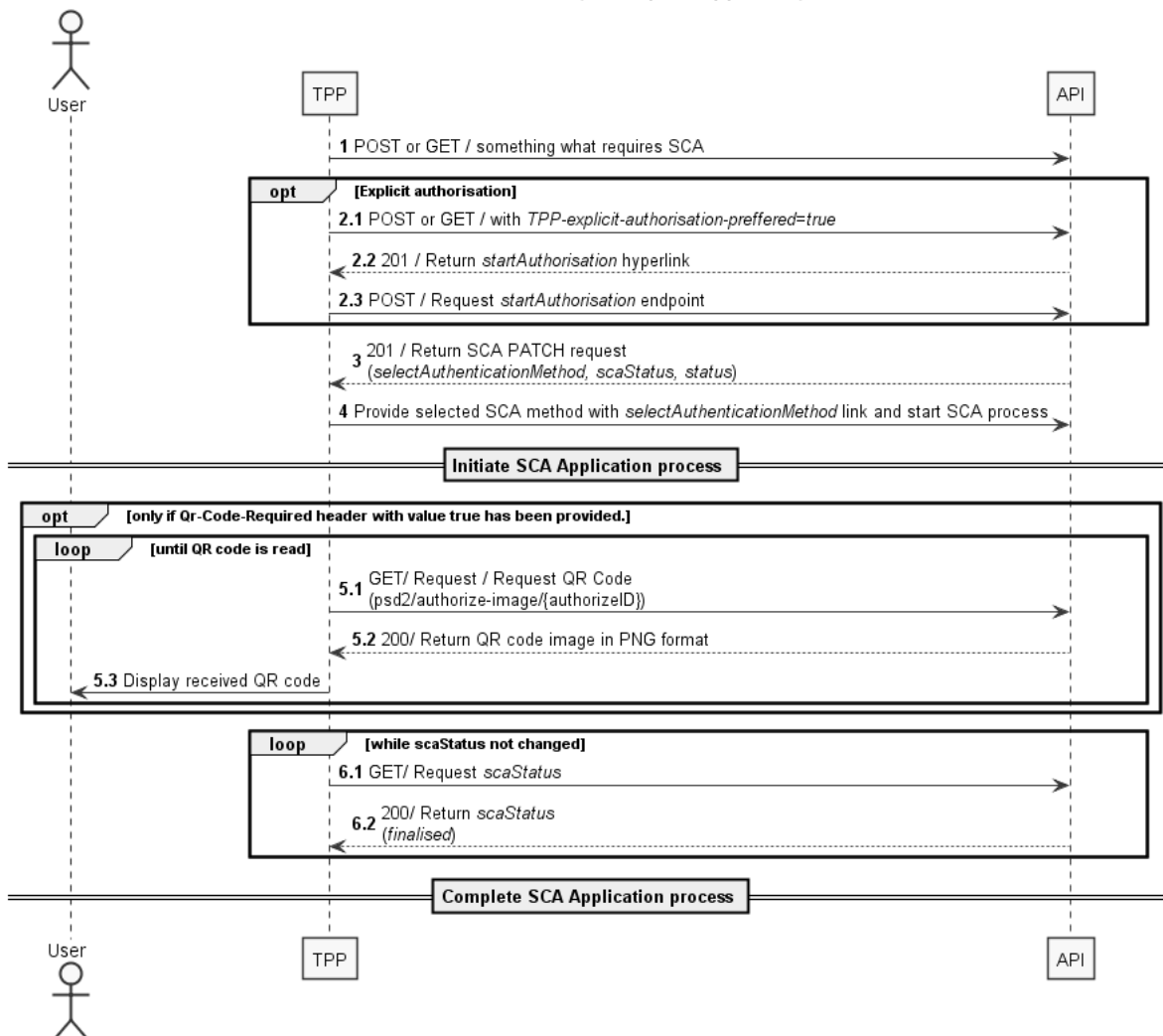
**Please note** that authorisation resource will be created automatically (unless specified otherwise over `TPP-Explicit-Authorisation-Preferred` header) by the Swedbank after the submission of the endpoint POST method. Explicit authorisation should only be used when decoupled approach is selected or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

### User authorisation (decoupled approach)



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 has 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 updates user's selected SCA method and starts 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 is created automatically (unless specified otherwise over `TPP-Explicit-Authoisation-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](http://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 gets 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 limitations described above.

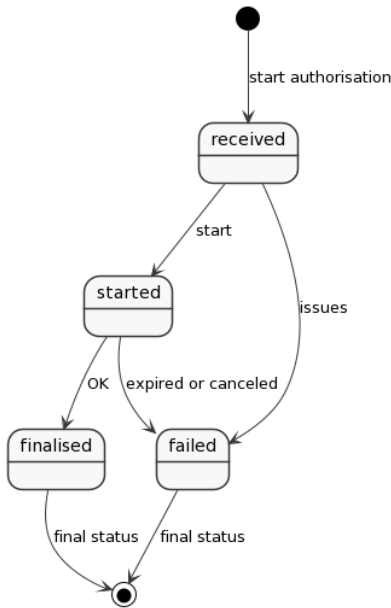
In decoupled flow TPP gets `SIMPLE_ID` as `authenticationMethodId` if biometric can be applied otherwise previously used method.

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.

## Authorisation state diagram



### 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.

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 hundreds 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 (one-time valid consent);
- Detailed consent (up to 90 days valid consent);
- Transactions Over 90 Days consent (one-time valid for 1 hour consent).

### Parallel consent support

It's important to note that from API v3 only one consent per Consent type (allAccounts, Detailed or Transactions Over 90 Days) 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 is autovalid if `{API-type}account_list` scope is present, otherwise it must be signed by PSU first. This scope is available only to AISP from 1st of March, 2021.

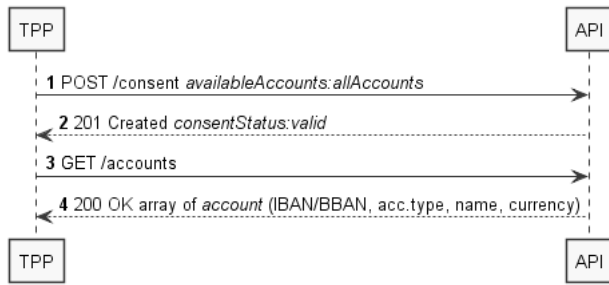
With allAccounts consent TPP gets list of accounts with this information:

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

### allAccounts consent diagrams

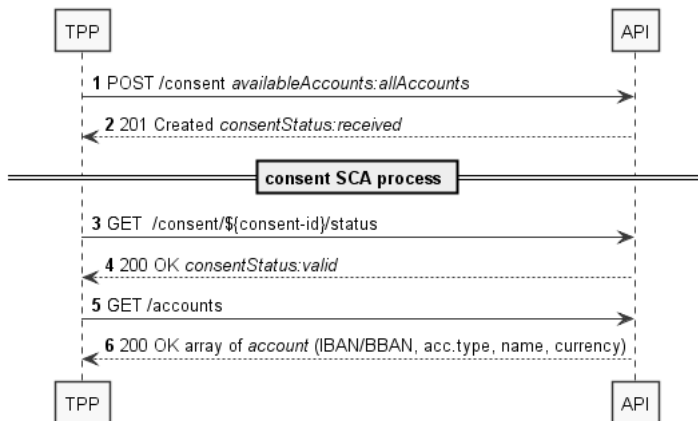
In case scope `{API-type}account_list` is granted

### Initial consent (redirect approach)



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

### allAccounts consent (redirect approach) 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 is valid automatically) and use it in `/accounts` endpoint. This consent is non recurring. 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` is requested user is 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

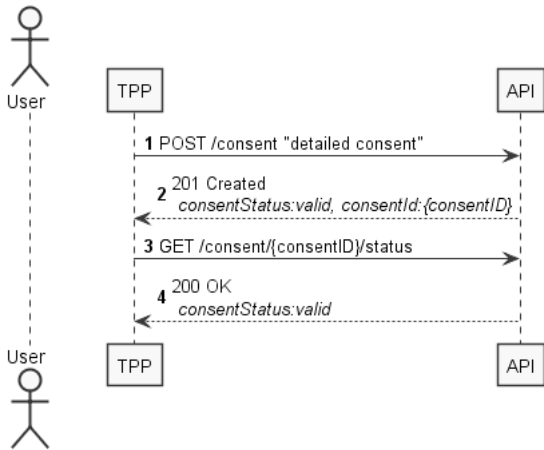
API uses **Detailed Consent** approach as described in BGS. Please refer to our [Developer Portal](#)->Develop for technical details on each endpoint, such as data models, response/request bodies and headers.

In order to get account details, TPP must create detailed consent. Detailed consent is auto valid if at least one of scopes `{API-type}account_balances` (Latvia and Sweden only), `{API-type}account_transactions` (Latvia and 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

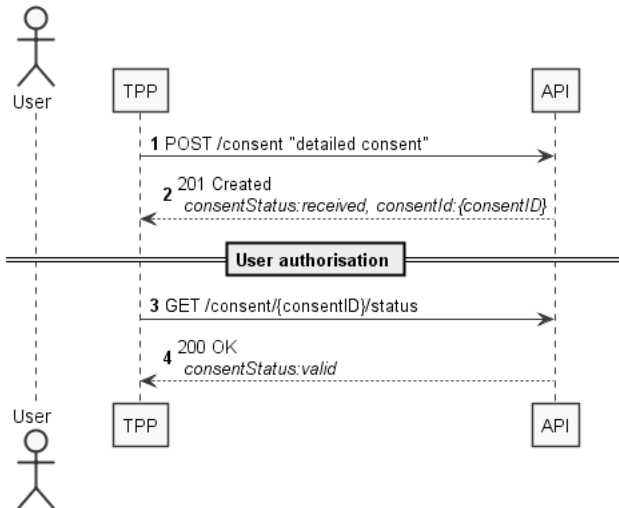
In case needed scopes are granted

### Detailed consent diagram (with granted scopes)



In case needed scopes are not granted

### Detailed consent diagram (without granted scopes)



## Transactions Over 90 Days consent

**Transactions Over 90 Consent** allows access of transactions over 90 days for one hour. Please refer to our [Developer Portal](#)->Develop for technical details on each endpoint, such as data models, response/request bodies and headers.

Transactions for 90 Days consent gives access to transactions history of multiple accounts and has to be confirmed by PSU with SCA. Consent's initiation is similar to Detailed Consent - IBANs of accounts should be listed under access's transactionsOver90Days field.

Sweden only: posted consent will be autovalid for the first time - once per customer's authorisation if scope `PSD2account_transactions_over90` is granted. Subsequent time customer will have to authorise each new consent request.

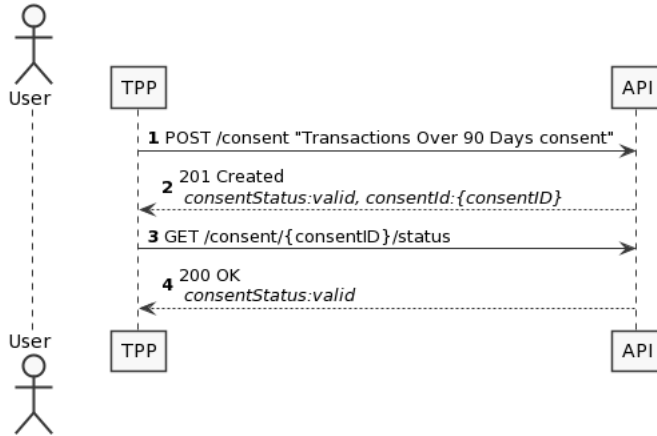
Baltics only: each consent for transactions over 90 days will have to be confirmed and authorised with SCA by PSU.

Consent can be signed by both redirect and decoupled approaches.

### Transactions Over 90 Days consent diagrams

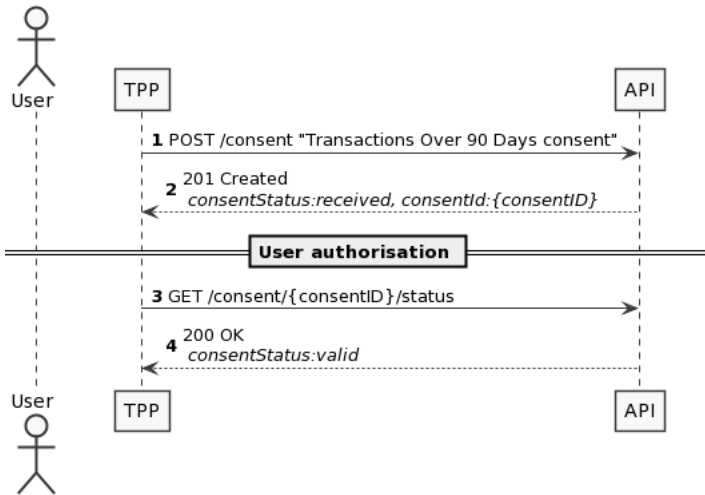
**Sweden only: in case posting consent first time**

### Transactions Over 90 Days consent diagram (Sweden, first time)



### Consent's signing process

#### Transactions Over 90 Days consent signing diagram



## Account information services API

Account information endpoint `/accounts` 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 is applied. Account information services can be called 4 times per 24h without user involvement. The headers `PSU-IP-Address` and `PSU-User-Agent` 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 requires person identification number or corporate identification number.

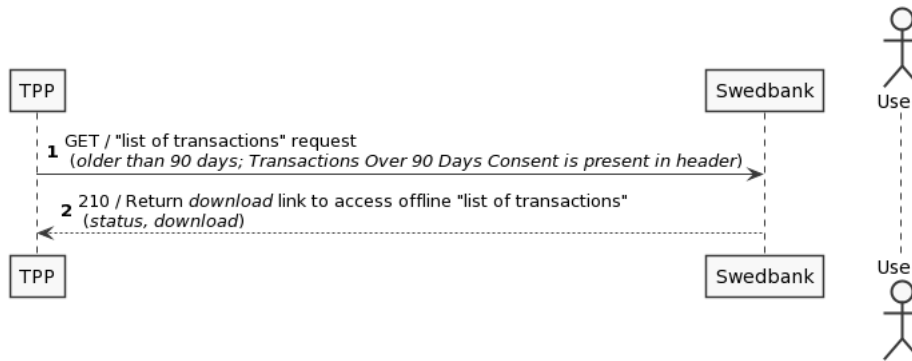
Please check `/accounts` endpoint in [Developer Portal](#)->Develop for technical details.

## Transaction list with data older than 90 days

In order to get transactions with data older than 90 days, there are two ways to do so:

- A recommended way is to use Transactions Over 90 Days consent to fetch transactions history. It allows access to selected accounts (GET/transactions) transactions as long as consent is valid (please check Consents section for more information). A flow of how to get `/download` link by using such consent:

### Transactions older than 90 days

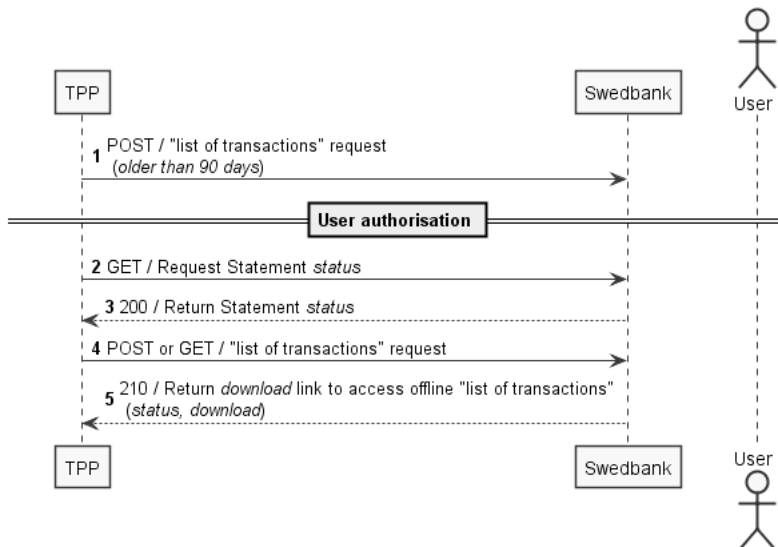


1. Call `GET /{version}/accounts/{accountID}/transactions` and use valid Transactions Over 90 Days consent ID;
2. This returns a HTTP 210 code with data containing `download_link`. Following this link it downloads the list of transactions as zipped file in JSON or XML format. This is called Offline mode.

When calling a download link the list of transactions is not available then HTTP 428 code is returned. Please retry in 5-10 seconds.

- Implementation remains where separate SCA for each account statement request required. Detailed consent can be used, but in that case user is asked to approve (SCA) the request beforehand. In such scenario, transactions over 90 days request must be POSTed first (`POST/transactions`). Without it, `GET/transactions` returns error indicating that user SCA is required. Following schema has to be applied (please note that download link might be returned while fetching transactions up to 90 days too, it happens in case statement size is very large):

### Transactions older than 90 days (redirect approach)



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 returns a HTTP 210 code with data containing `download_link`; Following this link it downloads the list of transactions as zipped file in JSON or XML format. This is called Offline mode.

When calling a download link the list of transactions is not available then HTTP 428 code is returned. Please retry in 5-10 seconds.

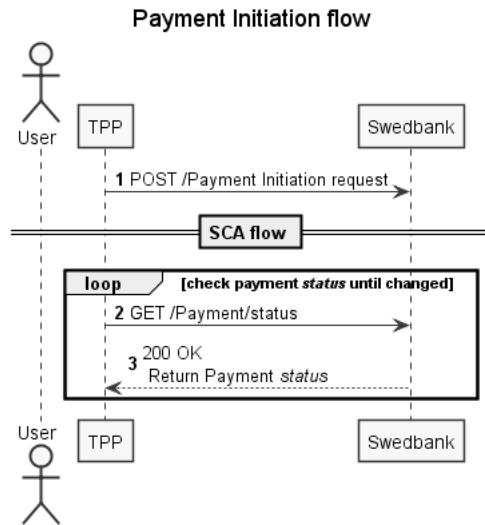
## Payments initiation services API



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 are 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.

## 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. Check possible [payment statuses and descriptions](#);
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 on Bank's UI. However, such payments have some limitations:

- It's applicable in Redirect approach only;
- International payments are supported only in Baltic countries;
- 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 is set on payments object. If needed, `GET /payment/{paymentId}` endpoint can be called to get it. Please refer to Swagger for more information.

## SE specifics

### Duplicate payment

In Sweden two payments are considered to be duplicates when all of the following fields contain identical information: type of payment(BG/PG), date, amount, debit account, credit account(BG/PG number), remittance information (OCR number/unstructured text). 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 has to add it so there are additional steps the user and TPP must complete.

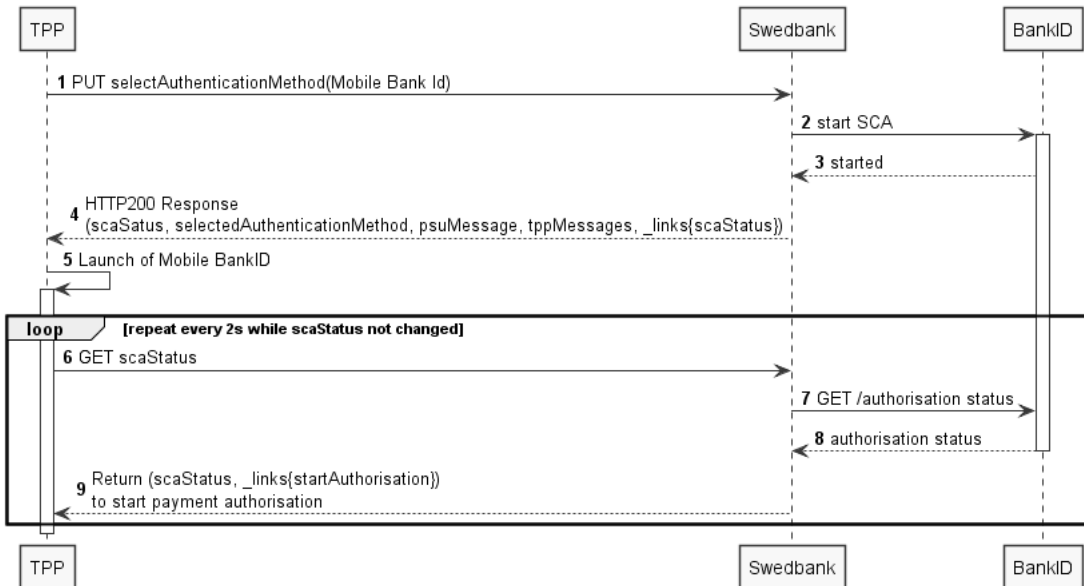
### Add recipient in redirect approach

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.

## Decoupled Add recipient SCA (Sweden)



### 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, it means that Recipient authorisation is initiated, so TPP additionally needs to authorise the payment.

5. Mobile BankID application is launched on the 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.

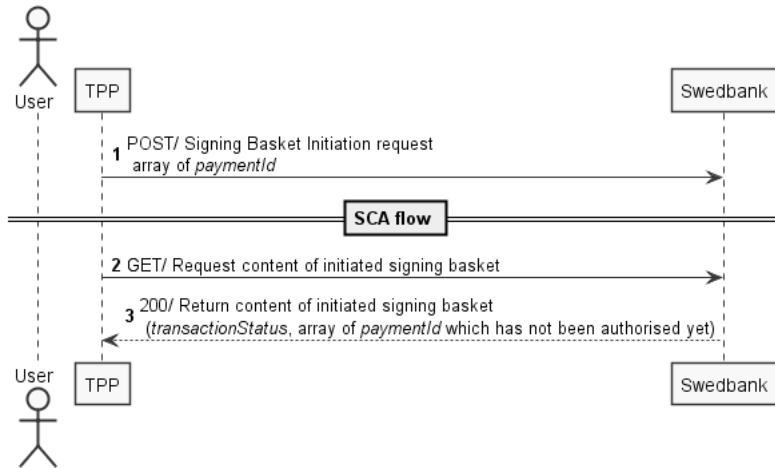
7. BankID service is checked for status;
8. SCA status is returned to Swedbank;
9. If status changes to 'finalised' TPP can start decoupled payment authorisation by following `startAuthorisation` steering link (see section [User authorisation using SCA](#)). For any other status please check error messages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

## POST 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.

## Signing Basket Initiation diagram



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 is returned;
  - Array of `paymentId` that are not signed is 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.

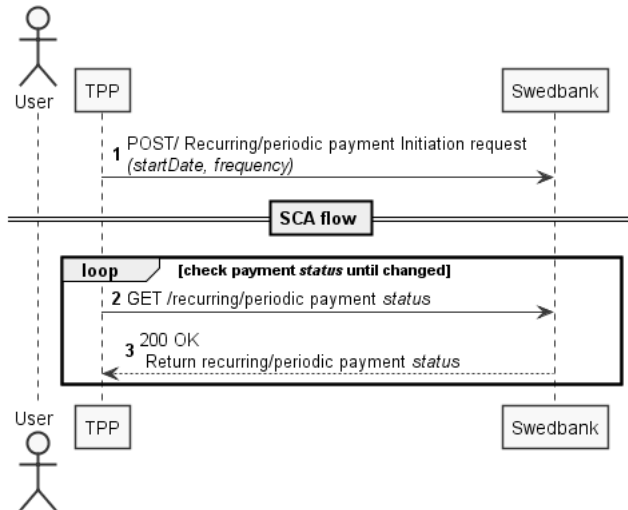
### Recurring/periodic payments in Sweden

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

## Recurring/Periodic Payment Initiation flow



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. Check possible [payment statuses and descriptions](#).

## 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 cancellation process:

1. Initiate payment cancellation by calling `DELETE /{version}/{payment-service}/{paymentId}`;
2. Check payment status until you get CANC for successfully cancellation 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.)

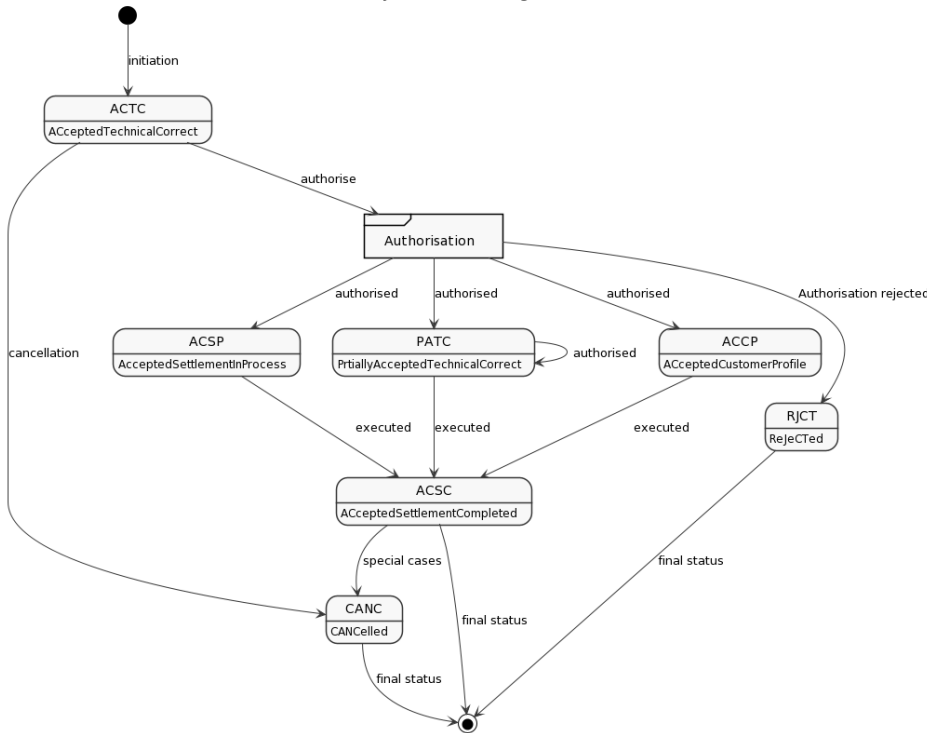
### Sweden specific

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 transitions through its 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

Payment state diagram



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 has 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.

<b>algorithm-ID</b>	string	Mandatory	The <b>algorithm</b> 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 <b>keyId</b> ) of this Request. It must be <b>rsa-sha256</b> or <b>rsa-sha512</b> .
<b>Headers</b>	string	Mandatory	The <b>headers</b> 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 <b>Date</b> 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: <b>Digest</b> , <b>X-Request-ID</b> , <b>Date_Optional</b> , <b>Date</b> Optional: <b>PSU-IP-Address</b> , <b>TPP-Redirect-URI</b> No other entries may be included.
<b>Signature</b>	string	Mandatory	The <b>signature</b> parameter is a base 64 encoded digital signature, as described in RFC 4648 [RFC4648], Section 4. The client uses the <b>algorithm</b> and <b>headers</b> signature parameters to form a canonical <b>signing string</b> . This <b>signing string</b> is then signed with the key associated with <b>keyId</b> and the algorithm corresponding to <b>algorithm</b> . The <b>signature</b> 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 is 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 2017 15:02:37 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 generates different signature than signing string.

When signature algorithm is used the signed request should 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 2017 15:02:37 GMT
Digest: SHA-256=F6gJ/9CFG6cIe1kuR4xmuH8u7TVaC2K7hDixR2rRa8w=
Signature: keyId="SN=9FA1,CA=Q049RC1U01VTVCBDQSAyLTEgMjAxNSxPPURUcnVzdCBhbWJILEM9REU= ", 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  $\geq$  400) response body contains element `tppMessages` with details of error. TPP should use combination of HTTP status code and `tppMessages[0].code` to identify error and make decision on how to proceed with error.

```

{
  "transactionStatus": "RJCT",
  "tppMessages": [
    { "category": "ERROR",
      "code": "FUNDS_NOT_AVAILABLE",
      "text": "The payment could not be completed because of insufficient funds." }
  ]
}

```

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.

## Service unspecific errors

These errors can be found in any endpoint.

market	status	code	text
Sweden	400	BAD_REQUEST	Bad request data: {0}
Sweden	400	FORMAT_ERROR	Format of certain request fields are not matching the XS2A requirements due to {0}
Sweden	500	INTERNAL_ERROR	Internal Error. Reference: {0}
Sweden	500	INTERNAL_ERROR	Service unavailable
Sweden	403	INTERNET_BANK_AGREEMENT	Internet bank agreement missing. Please direct customer to the bank to sign internet bank agreement
Sweden	403	KYC_INVALID	Please direct customer to the online banking to fill KYC.
Sweden	400	NEW_BANKID_AUTH_OCCURRED	New BankID authentication occurred
Sweden	403	OPEN_BANK_AGREEMENT	Internet bank/Open bank agreement missing. Please direct customer to the bank to sign agreement.
Sweden	401	PSU_CREDENTIALS_INVALID	Login/authorisation failed due to incorrect user ID or challenge response
Sweden	403	RESOURCE_EXPIRED	Requested resource has expired
Sweden	404	RESOURCE_NOT_FOUND	The addressed resource is unknown
Sweden	403	SCA_METHOD_INVALID	Chosen SCA method invalid or unsupported
Sweden	403	SIGNING_FAILED	Unexpected error
Sweden	400	STATUS_INVALID	Authorisation resource already signed.
Sweden	401	USER_INTERRUPTION	Interrupted by user
Sweden	403	USER_RIGHTS_OPENB	Not authorised to perform the request in Open Banking.
Sweden	403	USER_RIGHTS_ACCOUNT	Not authorized to perform the request
Baltics	400	FORMAT_ERROR	Mandatory parameter bic is missing or has unsupported value
Baltics	400	FORMAT_ERROR	Parameter psuData.personalID for scaMethod SMART_ID missing
Baltics	400	FORMAT_ERROR	JSON schema validation failure: \$.psuData.phoneNumber: does not match the regex pattern {0}
Baltics	400	HEADER_INVALID	Mandatory header \Authorization\ is wrong format
Baltics	403	VERIFICATION_FAILED	Wrong PSU-ID and/or PSU-CORPORATE-ID

Baltics	403	INTERNET_BANK_AGREEMENT	Internet bank agreement is not active or none eligible accounts available
Baltics	500	INTERNAL_ERROR	Internal Error
Baltics	500	INTERNAL_ERROR	Could not decode sessionData
Baltics	500	INTERNAL_ERROR	Technical error
Baltics	403	RESOURCE_EXPIRED	Requested resource has expired
Baltics	404	RESOURCE_NOT_FOUND	Unknown authorizationId
Baltics	404	RESOURCE_NOT_FOUND	{0}
Baltics	403	SCA_METHOD_INVALID	Chosen SCA method invalid or unsupported
Baltics	401	SCA_REQUIRED	Statement requires SCA
Baltics	405	SERVICE_INVALID	POST method allowed only
Baltics	503	SERVICE_UNAVAILABLE	Service is temporarily not available. Try again in few minutes or use different login method
Baltics	403	SIGNING_FAILED	Unexpected error
Baltics	403	SIGNING_FAILED	Phone number is not provided
Baltics	403	SIGNING_FAILED	PSU credentials invalid
Baltics	403	SIGNING_FAILED	SCA was already initiated for payment ID {0} and authorisation ID {1}
Baltics	403	SIGNING_FAILED	Please direct customer to unblock account
Baltics	401	TIMEOUT	The session has timed out.
Baltics	400	TOKEN_EXPIRED	Provided refresh_token expired
Baltics	401	TOKEN_INVALID	Could not match OAuth token to TPP - authorization_link:
Baltics	403	USER_RIGHTS_ACCOUNT	Not authorized to perform the request
Baltics	403	UNAUTHORISED	Not authorised to perform the request
Baltics	401	USER_INTERRUPTION	Interrupted by user
Baltics	400	USER_NOT_FOUND	Login unsuccessful! Please try again.
Baltics	400	USER_NOT_FOUND	Incorrect user ID
Baltics	403	VERIFICATION_FAILED	Verification of user or owner of account has failed
All	503	TEMPORARY_UNAVAILABLE	Planned system maintenance ongoing

## AIS specific errors

These are common errors found in AIS endpoints.

market	status	code	text
All	429	ACCESS_EXCEEDED	Consent daily limit {0} is exceeded for {1} account service {2}
All	403	CONSENT_EXPIRED	Consent expired
All	400	CONSENT_INVALID	No valid consent found
Sweden	401	SCA_REQUIRED	Authorise {0} account statement from {1} to {2}
Baltics	400	PERIOD_INVALID	Statement maximum date range (from to) exceeded
Baltics	400	PRODUCT_INVALID	Statement in xml format is not available for private users
Baltics	428	RESOURCE_PENDING	Statement is not yet available. Try again later

## PIS and Baskets specific errors

These are common errors found in `/payments` and `/signing-baskets` endpoint.

market	status	code	text
Sweden	403	ACTION_REQUIRED	Direct customer to the bank to complete this task
Sweden	403	DAILY_LIMIT	The payment limit per day is exceeded
Sweden	400	FUNDS_NOT_AVAILABLE	The payment could not be completed because of insufficient funds
Sweden	403	OPEN_BANK_AGREEMENT	Internet bank/Open bank agreement missing. Please direct customer to the bank to sign agreement
Sweden	403	PAYMENT_PRODUCT_AGREEMENT	Payment service agreement for a payment/transfer missing
Sweden	400	PAYMENT_FAILED	The payment request failed due to {0}
Sweden	400	PAYMENT_FAILED	The payment request failed due to recurrence details provided
Sweden	400	PAYMENT_FAILED	The payment request failed due to frequency details missing
Sweden	403	PAYMENT_PRODUCT_LIMIT	Payment product limit is exceeded
Sweden	400	PRODUCT_INVALID	Payment product is not supported in basket
Sweden	400	PRODUCT_INVALID	The addressed payment product is not available



Sweden	404	RESOURCE_NOT_FOUND	One or more payments IDs are incorrect
Sweden	404	RESOURCE_NOT_FOUND	Incorrect basket ID
Sweden	404	RESOURCE_NOT_FOUND	Basket or authorization ID are incorrect
Sweden	404	RESOURCE_NOT_FOUND	The addressed resource is unknown
Sweden	403	SCA_METHOD_INVALID	To add a new recipient, activate Mobile BankID for extended use.
Sweden	403	SCA_METHOD_INVALID	Chosen SCA method invalid or unsupported
Sweden	400	STATUS_INVALID	Authorisation resource already signed.
Sweden	400	STATUS_INVALID	Cannot delete basket
Sweden	400	STATUS_INVALID	Authorisation resource is in an invalid state.
Sweden	401	TIMEOUT	Mobile id challenge expired
Sweden	403	USER_ACCOUNT_LIMIT	The amount limit for signing is exceeded
Sweden	403	USER_RIGHTS_ADD_RECIPIENT	Not authorised to sign recipient
Sweden	403	USER_RIGHTS_ACCOUNT	Not authorised to perform the request
Baltics	403	DAILY_LIMIT	The payment limit per day is exceeded
Baltics	400	FUNDS_NOT_AVAILABLE	The payment could not be completed because of insufficient funds
Baltics	400	FUNDS_NOT_AVAILABLE	The payment could not be completed because of not enough funds for fee
Baltics	403	LIMIT	Amount exceeds limit
Baltics	403	MONTHLY_LIMIT	The payment limit per month is exceeded
Baltics	400	PAYMENT_FAILED	The payment initiation POST request failed due to {invalid cut-off time}
Baltics	400	PAYMENT_FAILED	The payment request failed due to {0}
Baltics	400	PAYMENT_FAILED	The payment request failed due to missing debtor
Baltics	400	PAYMENT_FAILED	The payment request failed due to invalid debtor account
Baltics	400	PAYMENT_FAILED	The payment request failed due to invalid account number
Baltics	400	PAYMENT_FAILED	The payment request failed due to duplicate payment order
Baltics	400	PAYMENT_FAILED	The payment request failed due to payment amount too low
Baltics	400	PAYMENT_FAILED	The payment request failed due to currency invalid
Baltics	400	PAYMENT_FAILED	The payment request failed due to identification code missing or invalid
Baltics	400	PAYMENT_FAILED	The payment request failed due to debtor identification code missing or invalid
Baltics	400	PAYMENT_FAILED	The payment request failed due to creditor identification code missing or invalid
Baltics	400	PAYMENT_FAILED	The payment request failed due to payment amount invalid
Baltics	400	PAYMENT_FAILED	The payment request failed due to execution date can't be in the future
Baltics	400	PAYMENT_FAILED	The payment request failed due to incorrect bank identifier
Baltics	400	PAYMENT_FAILED	The payment request failed due to invalid creditor bank
Baltics	400	PAYMENT_FAILED	The payment request failed due to invalid bic
Baltics	400	PAYMENT_FAILED	The payment request failed due to missing remittanceInformationStructured reference
Baltics	400	PRODUCT_INVALID	Payment service is available in redirect integration only
Baltics	400	PRODUCT_INVALID	Payment product invalid
Baltics	400	PRODUCT_INVALID	Cross-border instant payment not supported
Baltics	400	PRODUCT_INVALID	Intrabank payment cannot be instant
Baltics	400	PRODUCT_INVALID	Recurring payment has more than 1 payment initiation
Baltics	400	RECIPIENT_INVALID	Invalid creditor account number
Baltics	400	RECIPIENT_INVALID	Inconsistent with end customer
Baltics	400	RECIPIENT_INVALID	Missing creditor address
Baltics	400	RECIPIENT_INVALID	Invalid creditor name
Baltics	401	USER_INTERRUPTION	Interrupted by user
Baltics	401	USER_RIGHTS_ACCOUNT	Not authorized to perform the request

## Most common errors in integration phase

These errors indicate errors in integration with API.

market	status	code	text	action
All	400	CERTIFICATE_INVALID	No certificate provided in developer portal, in Application setup	Endpoint requires proper certificates configured and enabled
All	400	FORMAT_ERROR	Mandatory parameter bic is missing or has unsupported value	Mandatory parameter bic is missing or has unsupported value

All	400	FORMAT_ERROR	Parameter dateTo is in future	Request to /transactions endpoint
All	400	FORMAT_ERROR	Client_id mismatch on OAuth2.0 invocation and token exchange	Errors in OAuth configuration or implementation
All	400	FORMAT_ERROR	Missing parameter refresh_token	Missing parameter
All	400	FORMAT_ERROR	Consent request schema validation failed: {json_schema_error}	Invalid json provided
All	400	FORMAT_ERROR	Payment product {payment_product} schema validation failed: {json_schema_error}	Invalid json provided
All	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
All	400	FORMAT_ERROR	Missing required parameter or duplicate	Missing required parameter or duplicate
All	400	HEADER_INVALID	Mandatory header {header name} is wrong format	Header has to have correct value
All	400	HEADER_INVALID	Mandatory header is missing: {header name}	Header has to be specified
All	404	INVALID_REQUEST	Statement for transactions over 90 days must be posted first	Check documentation for statements over 90 days
All	400	SCOPE_INVALID	Scope(s) ({scopes}) needs main scope specified	Invalid scopes specified in /psd2/authorize request
All	405	SERVICE_INVALID	Got: {http_verb} allowed: {http_verb}	Endpoint does not support specified HTTP verb
All	405	SERVICE_INVALID	HTTP method {http_verb} not supported for {endpoint}	Endpoint does not support specified HTTP verb

## Sandbox

### Sandbox setup

Sandbox environment can be used for understanding API details, creating integration to API. Most popular scenarios can be tested in sandbox. For better quality of integration with API negative flow handling should be implemented. For test purposes you can find common error cases in our sandbox and adopt your solution to handle such errors.

You can reproduce error cases according to special values used for UserId (login flow) and payment amount (payment initiation flow) parameters.

All special values are provided in the tables below. Furthermore, you can find list of special users in login widget (redirect login flow).

#### Service unspecific errors

SANDLT22, SANDLV22, SANDEE2X:

Special value	Error code	Description	How to reproduce	Result
UserId = 444444 ("User is not valid")	USER_NOT_FOUND	When in decoupled login flow some provided information doesn't match. For example, phone number doesn't belong to user.	<ul style="list-style-type: none"> <li>Start decoupled login with userID 444444.</li> </ul>	Response: {"tppMessages": [{"category": "ERROR", "code": "USER_NOT_FOUND", "text": "Incorrect user ID code personal code or phone number"}]}
UserId = 333333 ("User with failing SCA on consent signing")	SIGNING_FAILED	Common error, when something went wrong on resource signing.	<ul style="list-style-type: none"> <li>Get valid token for userID 333333.</li> <li>Initiate consent with redirect signing and try to sign it.</li> </ul>	Unexpected error in redirect. Authorization status response: {"scaStatus": "failed", "tppMessages": [{"category": "ERROR", "code": "SIGNING_FAILED", "text": "Unexpected error"}]}
authenticationMethodId = SIMPLE_ID	SCA_METHOD_INVALID	In decoupled signing flow (consent or payment), when provided unsupported authentication method.	<ul style="list-style-type: none"> <li>Get valid token for any user.</li> <li>Initiate payment or consent with decoupled signing.</li> <li>Provide unsupported authentication method id in PUT request. For example: {"authenticationMethodId": "SIMPLE_ID"}</li> </ul>	Response: {"scaStatus": "failed", "tppMessages": [{"category": "ERROR", "code": "SCA_METHOD_INVALID", "text": "Chosen SCA method invalid or unsupported"}]}
-	USER_INTERRUPTION	Common error, when user cancels signing in redirect flow.	<ul style="list-style-type: none"> <li>Get valid token for any user.</li> <li>Initiate any resource (consent, statement, payment) with redirect signing.</li> <li>Open redirect signing page and click "Cancel" button.</li> </ul>	User is forwarded to Nok-Redirect-Uri. Authorization status response: {"scaStatus": "failed", "tppMessages": [{"category": "ERROR", "code": "USER_INTERRUPTION", "text": "Interrupted by user"}]}

SANDESS:

Special value	Error code	Description	How to reproduce	Result
---------------	------------	-------------	------------------	--------

Userld = 444444444444 ("User is not valid")	USER_NOT_FOUND	When in decoupled login flow some provided information doesn't match. For example, personal code doesn't belong to user.	<ul style="list-style-type: none"> <li>Start decoupled login with userID 444444444444.</li> </ul>	Response: {"tppMessages": [{"category": "ERROR", "code": "USER_NOT_FOUND", "text": "Incorrect user ID code personal code or phone number"}]}
-	USER_INTERRUPTION	Common error, when user cancels signing in redirect flow.	<ul style="list-style-type: none"> <li>Get valid token for any user.</li> <li>Initiate any resource (consent, statement, payment) with redirect signing.</li> <li>Open redirect signing page and click "Cancel" button.</li> </ul>	User is forwarded to Nok-Redirect-Uri. Authorization status response: {"scaStatus": "failed", "tppMessages": [{"category": "ERROR", "code": "USER_INTERRUPTION", "text": "Interrupted by user"}]}

#### AIS specific errors

SANDLT22, SANDLV22, SANDEE2X:

Special value	Error code	Description	How to reproduce	Result
Userld = 555555 ("User without bank agreement")	INTERNET_BANK_AGREEMENT	When user is missing agreement to use internet bank	<ul style="list-style-type: none"> <li>Get valid token for userID 555555.</li> <li>Try to initiate consent.</li> </ul>	Response on consent initiation: {"tppMessages": [{"category": "ERROR", "code": "INTERNET_BANK_AGREEMENT", "text": "Internet bank agreement is not active or none eligible accounts available"}]}

SANDEESS:

Special value	Error code	Description	How to reproduce	Result
Userld = 333333333333 ("User without completed KYC")	KYC_INVALID	When user hasn't filled KYC information in internet bank.	<ul style="list-style-type: none"> <li>Get valid token for userID 333333333333.</li> <li>Try to initiate consent.</li> </ul>	Response on consent initiation: {"tppMessages": [{"category": "ERROR", "code": "KYC_INVALID", "text": "Please direct customer to the online banking to fill KYC."}]}

#### PIS specific errors

SANDLT22, SANDLV22, SANDEE2X:

Special value	Error code	Description	How to reproduce	Result
Userld = 666666 ("User is blocked")	SIGNING_FAILED	When user account is blocked for some reasons.	<ul style="list-style-type: none"> <li>Get valid token for userID 666666.</li> <li>Try to initiate payment.</li> </ul>	Response on payment initiation: {"tppMessages": [{"category": "ERROR", "code": "SIGNING_FAILED", "text": "Please direct customer to unblock account"}]}
amount = 30	FUNDS_NOT_AVAILABLE	When overall payment amount (with fees and etc) is larger, than user's account balance.	<ul style="list-style-type: none"> <li>Try to initiate payment with "amount": "50".</li> </ul>	Response on payment initiation: {"transactionStatus": "RJCT", "tppMessages": [{"category": "ERROR", "code": "FUNDS_NOT_AVAILABLE", "text": "The payment could not be completed because of not enough funds for fee"}]}
amount = 51	DAILY_LIMIT	When user's daily payment limit is exceeded.	<ul style="list-style-type: none"> <li>Try to initiate payment with "amount": "51".</li> </ul>	Response on payment initiation: {"transactionStatus": "RJCT", "tppMessages": [{"category": "ERROR", "code": "DAILY_LIMIT", "text": "The payment limit per day is exceeded"}]}
amount = 52	MONTHLY_LIMIT	When user's monthly payment limit is exceeded.	<ul style="list-style-type: none"> <li>Try to initiate payment with "amount": "52".</li> </ul>	Response on payment initiation: {"transactionStatus": "RJCT", "tppMessages": [{"category": "ERROR", "code": "MONTHLY_LIMIT", "text": "The payment limit per month is exceeded"}]}
amount = 53	USER_RIGHTS_ACCOUNT	When user is missing rights to initiate payment with provided debtor account.	<ul style="list-style-type: none"> <li>Try to initiate payment with "amount": "53".</li> </ul>	Response on payment initiation: {"transactionStatus": "RJCT", "tppMessages": [{"category": "ERROR", "code": "USER_RIGHTS_ACCOUNT", "text": "Not authorized to perform the request"}]}

-	FUNDS_NOT_AVAILABLE	When payment amount is larger, than user's account balance.	<ul style="list-style-type: none"> <li>You should get desired account balance.</li> <li>Try to initiate payment with this account and amount larger, than account balance.</li> </ul>	Response on payment initiation: {"transactionStatus": "RJCT", "tppMessages": [{"category": "ERROR", "code": "FUNDS_NOT_AVAILABLE", "text": "The payment could not be completed because of insufficient funds"}]}
---	---------------------	---	---	--

SANDESS:

Special value	Error code	Description	How to reproduce	Result
Userld = 555555555555 ("User without extended Mobile-ID")	SCA_METHOD_INVALID	When user hasn't extended Mobile BankID	<ul style="list-style-type: none"> <li>Get valid token for userID 555555555555.</li> <li>Initiate payment with redirect signing.</li> <li>Open redirect signing page and click "Add recipient".</li> </ul>	User see error message. On "Back" click user is forwarded to Nok-Redirect-Uri. Authorization status response: {"scaStatus": "failed", "tppMessages": [{"category": "ERROR", "code": "SCA_METHOD_INVALID", "text": "To add a new recipient activate Mobile BankID for extended use."}]}
amount = 30	ACTION_REQUIRED	When payment failed AML validation.	<ul style="list-style-type: none"> <li>Try to initiate payment with "amount": "30".</li> </ul>	Response on payment initiation: {"transactionStatus": "RJCT", "tppMessages": [{"category": "ERROR", "code": "ACTION_REQUIRED", "text": "Direct customer to the bank to complete this task"}]}
amount = 31	PAYMENT_FAILED	When payment has duplicate (BG/PG).	<ul style="list-style-type: none"> <li>Try to initiate payment with "amount": "31".</li> </ul>	Response on payment initiation: {"transactionStatus": "RJCT", "tppMessages": [{"category": "ERROR", "code": "PAYMENT_FAILED", "text": "The payment/transfer contains errors"}]}

## 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 available V[n+1] version available	-
Release date	Newsletter, Developer portal	V[n] version available V[n+1] version available	V[n] version available V[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.

## API v3 to v4 migration guidelines

### Introduction

In October of 2021 4th major version of API will be released. This chapter describes changes in v4 to smoothen migration from API v3.

Market	Change	Impact	Description
All	<code>combinedServiceIndicator</code>	breaking change	Its value should always be set to false
All	<code>app-id</code> required	breaking change	If TPP uses several application with the same QWAC then <code>app-id</code> is mandatory.
All	<code>allAccounts</code> consent is not autovalid	breaking change	<code>PSD2account_list</code> scope has to be added in order to get account list.
All	<code>DELETE /payments/{payment-product}/{paymentID}</code>	breaking change	Response code on successful payment cancellation was changed from 200 to 204 and no response body is returned.
All	<code>PSU-Http-Method</code> and <code>PSU-IP-Port</code> headers	improvement	API call without <code>PSU-Http-Method</code> and <code>PSU-IP-Port</code> headers will NOT count towards the number of TPP calls without PSU interaction.
All	Balance type label has been changed	breaking change	To follow BGS, balance type label has been changed from <code>authorised</code> to <code>interimBooked</code>
BB	Redirection to TPP changed from POST to GET	improvement	If user cancels signing in redirect flow (consent, statement, payment), then redirection to TPP goes through GET method.
BB	GET Detailed consent response body updated	breaking change	Response body won't contain empty list of access service if it wasn't requested while POSTING the consent.
Sweden	New way to fetch transactions over 90 days	breaking change	New Consent is introduced to fetch transactions over 90 days for multiple accounts with one SCA, it allows a limited time access to fetch transactions history. Please check Consents section for more information.
Sweden	Updated valid consent's response	breaking change	To follow BGS, in case of valid GET/consent scenario, response will contain next link to fetch all accounts in both 'allAccounts' and detailed consents scenarios.
Sweden	<code>allAccounts</code> <code>consent recurringIndicator</code> always set to false	breaking change	<code>allAccounts</code> is a one time consent that doesn't support recurrence, so its value should always be set to <code>false</code>
Sweden	POST Basket authorisation's response body updated	breaking change	Response has been updated by removing <code>authorisationId</code> field and renaming <code>transactionStatus</code> to <code>scaStatus</code>
Sweden	GET consent response returns next link for <code>allAccounts</code>	breaking change	Response of both <code>allAccounts</code> and <code>Detailed</code> consents has been updated with a next link that should be used to fetch account list if needed.

### Errors migration

Error codes and messages have been improved in V4. The list of changed errors is provided in the tables below.

#### Service unspecific runtime errors updated

market	v4	v3
Sweden	400 BAD_REQUEST Bad request data: {0}	400 BAD_REQUEST_DATA Bad request data
Sweden	500 INTERNAL_ERROR Internal Error. Reference: {0}	500 INTERNAL_ERROR Internal server error. Reference:

Sweden	500 INTERNAL_ERROR Service unavailable.	503 SERVICE_UNAVAILABLE Service unavailable.
Sweden	403 INTERNET_BANK_AGREEMENT Internet bank agreement missing. Please direct customer to the bank to sign internet bank agreement	401 IB_AGREEMENT_EXCEPTION Internet bank agreement missing. Please direct customer to the bank to sign internet bank agreement
Sweden	403 INTERNET_BANK_AGREEMENT Internet bank agreement missing. Please direct customer to the bank to sign internet bank agreement	401 MISSING_BANK_AGREEMENT Internet bank agreement not OK. Please direct customer to the bank to sign agreement
Sweden	401 PSU_CREDENTIALS_INVALID Login/authorisation failed due to incorrect user ID or challenge response	401 PSU_CREDENTIALS_INVALID The PSU-ID cannot be matched by the addressed ASPSP or is blocked, or a password resp. OTP was not correct.
Sweden	403 SCA_METHOD_INVALID Chosen SCA method invalid or unsupported	400 UNSUPPORTED_SCA_METHOD Executing payments with id-card is not available in LT
Sweden	400 STATUS_INVALID Authorisation resource already signed.	400 AUTHORISATION_SIGNED Authorisation resource already signed.
Sweden	403 USER_RIGHTS_OPENB Not authorised to perform the request in Open Banking.	403 OPENB_RIGHTS_ON_ACCOUNT Not authorised to perform the request in Open Banking
Baltics	400 FORMAT_ERROR Parameter psuData.personalID for scaMethod SMART_ID missing	400 FORMAT_ERROR For scaMethod SMART_ID missing or empty parameter psuData.personalID
Baltics	400 HEADER_INVALID Mandatory header \Authorization\ is wrong format	400 FORMAT_ERROR Mandatory header \Authorization\ is wrong format
Baltics	403 VERIFICATION_FAILED Wrong PSU-ID and/or PSU-CORPORATE-ID	401 FORMAT_ERROR Wrong PSU-ID and/or PSU-CORPORATE-ID
Baltics	500 INTERNAL_ERROR technical error	500 TECHNICAL_ERROR technical error
Baltics	403 INTERNET_BANK_AGREEMENT Internet bank agreement is not active or none eligible accounts available agreement	401 MISSING_BANK_AGREEMENT Internet bank agreement is not active or none eligible accounts available
Baltics	404 RESOURCE_NOT_FOUND Unknown authorizationId	400 FORMAT_ERROR Unknown authorizationId
Baltics	404 RESOURCE_NOT_FOUND {0}	404 RESOURCE_UNKNOWN {0}
Baltics	404 RESOURCE_NOT_FOUND The addressed resource is unknown	403 SERVICE_BLOCKED The addressed authorisation resource is unknown
Baltics	403 SCA_METHOD_INVALID Chosen SCA method invalid or unsupported	400 UNSUPPORTED_SCA_METHOD Requested SCA method is not available. Please acquire new access token or choose different method
Baltics	403 SCA_METHOD_INVALID Chosen SCA method invalid or unsupported	401 AGREEMENT_NOT_FOUND Smart-ID agreement is missing
Baltics	403 SCA_METHOD_INVALID Chosen SCA method invalid or unsupported	401 NOT_SUITABLE_CERTIFICATE_FOR_SIGNING Smart-ID confirmation failed. This operation can only be confirmed if your Smart-ID account has been registered with Mobile-ID or ID card.
Baltics	403 SCA_METHOD_INVALID Chosen SCA method invalid or unsupported	400 error_scaMethodNotAvailable Selected confirmation method not available. Please change signing method
Baltics	405 SERVICE_INVALID POST method allowed only	405 SERVICE_INVALID Got: GET allowed: POST

Baltics	503 SERVICE_UNAVAILABLE Service is temporarily not available. Try again in few minutes or use different login method.	401 SERVICE_NOT_AVAILABLE Service is temporarily not available. Try again in few minutes or use different login method.
Baltics	403 SIGNING_FAILED Unexpected error	401 DOCUMENT_UNUSABLE Your Smart-ID account is locked. Please, unlock Smart-ID and try again!
Baltics	403 SIGNING_FAILED PSU credentials invalid	400 PASSWORD_INVALID
Baltics	403 SIGNING_FAILED Phone number is not provided	401 PHONE_ABSENT
Baltics	403 SIGNING_FAILED PSU credentials invalid	401 PSU_CREDENTIALS_INVALID
Baltics	403 SIGNING_FAILED SCA was already initiated for payment ID{0} and authorisation ID{1}	401 SCA_FAILED
Baltics	403 SIGNING_FAILED Unexpected error	401 SIM_ERROR
Baltics	403 SIGNING_FAILED Please direct customer to unblock account	401 USER_COULD_NOT_BE_UNBLOCKED User account is blocked. You can unblock it by calling to consultation centre or by visiting any Swedbank branch
Baltics	401 TIMEOUT The session of signing has timed out.	401 EXPIRED_TRANSACTION
Baltics	400 TOKEN_EXPIRED Provided refresh_token expired	400 FORMAT_ERROR Provided refresh_token expired
Baltics	401 TOKEN_INVALID Could not match OAuth token to TPP - authorization_link:	401 TOKEN_UNKNOWN Could not match OAuth token to TPP - authorization_link:
Baltics	403 UNAUTHORISED Not authorised to perform the request	401 ACCOUNT_UNUSABLE
Baltics	401 USER_INTERRUPTION Interrupted by user	401 MID_USER_CANCELLED You pressed the 'cancel' button on your phone
Baltics	401 USER_INTERRUPTION Interrupted by user	401 USER_CANCEL You have cancelled the signing by Mobile-ID in your phone.
Baltics	401 USER_INTERRUPTION Interrupted by user	401 USER_REFUSED User cancelled request in Smart-ID app.
Baltics	400 USER_NOT_FOUND Login unsuccessful! Please try again.	401 USER_NOT_FOUND Login unsuccessful! Please try again.
Baltics	400 USER_NOT_FOUND Incorrect user ID	401 USER_NOT_VALID Incorrect user ID
Baltics	403 USER_RIGHTS_ACCOUNT Not authorized to perform the request	400 AG01 Debtor account service not allowed
Baltics	403 USER_RIGHTS_ACCOUNT Not authorized to perform the request	400 AG01 NO_RIGHTS
Baltics	403 USER_RIGHTS_ACCOUNT Not authorized to perform the request	401 USER_RIGHTS_ON_ACCOUNT Not authorized to perform the request
Baltics	403 VERIFICATION_FAILED Verification of user or owner of account has failed	403 RESOURCE_UNKNOWN Verification of user or owner of account has failed
All	400 FORMAT_ERROR Malformed URL	500 FORMAT_ERROR Malformed URL

All	403 SERVICE_BLOCKED TPP was blocked	500 SERVICE_BLOCKED TPP was blocked
All	400 CERTIFICATE_EXPIRED Expired QWAC certificate	400 FORMAT_ERROR Expired QWAC certificate
All	400 CERTIFICATE_INVALID Invalid certificate:{0}	400 FORMAT_ERROR Invalid certificate: {0}
All	400 CERTIFICATE_REVOKED QWAC certificate revoked	400 FORMAT_ERROR QWAC certificate revoked
All	400 HEADER_INVALID Body digest mismatch: {0}	400 FORMAT_ERROR Body digest mismatch: {0}
All	400 PERIOD_INVALID Mandatory header \Date\" is in future"	400 FORMAT_ERROR Mandatory header \Date\" is in future"
All	400 PERIOD_INVALID Mandatory header \Date\" is in past"	400 FORMAT_ERROR Mandatory header \Date\" is in past"
All	400 PERIOD_INVALID Parameter dateTo is in future	400 FORMAT_ERROR Parameter dateTo is in future
All	403 PERIOD_INVALID Parameter dateTo is older then parameter dateFrom	403 FORMAT_ERROR Parameter dateTo is older then parameter dateFrom
All	403 PSU_CREDENTIALS_INVALID {0}	403 RESOURCE_UNKNOWN {0}
All	403 SCA_METHOD_INVALID Chosen SCA method invalid or unsupported	400 UNSUPPORTED_METHOD Unsupported login method: {0}
All	400 SCHEMA_INVALID {0}	400 FORMAT_ERROR {0}
All	400 SCOPE_INVALID {0}	400 FORMAT_ERROR {0}
All	401 TIMEOUT sessionData expired, please start authorisation again	400 FORMAT_ERROR sessionData expired, please start authorisation again

#### AIS specific errors updated

market	v4	v3
Sweden	403 CONSENT_EXPIRED Consent expired	401 CONSENT_EXPIRED The consent was created by this TPP but has expired and needs to be renewed
Sweden	400 CONSENT_INVALID No valid consent found	404 CONSENT_UNKNOWN The consent-ID cannot be matched by the ASPSP relative to the TPP
Baltics	400 CONSENT_INVALID No valid consent found	404 CONSENT_UNKNOWN The consent-ID cannot be matched by the ASPSP relative to the TPP
Baltics	400 CONSENT_INVALID No valid consent found	401 CONSENT_INVALID No valid consent found
Baltics	400 PRODUCT_INVALID Statement in xml format is not available for private users	400 INVALID_FORMAT

#### PIS specific errors updated

market	v4	v3



Sweden	403 ACTION_REQUIRED Direct customer to the bank to complete this task	403 ACTION_REQUIRED Direct customer to the bank to complete this task or call support: 0771-53 00 55
Sweden	403 ACTION_REQUIRED Direct customer to the bank to complete this task	403 ANTI_FRAUD_CHECK You need to contact the bank to complete this task. Please call our internet and mobile services support as soon as possible so that we can help you. Internet and Mobile Services Support: 0771-53 00 55.
Sweden	403 DAILY_LIMIT The payment limit per day is exceeded	403 EXCEEDED_PAYMENT_LIMIT The payment limit per day is exceeded
Sweden	400 FUNDS_NOT_AVAILABLE The payment could not be completed because of insufficient funds	401 INSUFFICIENT_FUNDS The payment/transfer could not be completed because of insufficient funds
Sweden	403 PAYMENT_PRODUCT_AGREEMENT Credit transfer agreement for a payment/transfer missing	401 MISSING_CT_AGREEMENT Credit transfer agreement for a payment/transfer missing
Sweden	400 PAYMENT_FAILED The payment request failed due to cut-off time for payment in selected currency has expired	400 PASSED_CUTOFF_TIME Cut-off time for payment in selected currency has expired
Sweden	400 PAYMENT_FAILED The payment request failed due to recurrence details provided	400 FREQUENCY_DETAILS_PRESENT Recurrence details provided
Sweden	400 PAYMENT_FAILED The payment request failed due to frequency details missing	400 FREQUENCY_DETAILS_MISSING Recurrence details missing
Sweden	400 PAYMENT_FAILED The payment request failed due to {0}	400 INVALID_CT The payment/transfer contains errors
Sweden	400 PAYMENT_FAILED The payment request failed due to invalid BIC/ natID	400 INVALID_BIC_OR_NATID The payment/transfer contains errors: invalid BIC/natID
Sweden	400 PAYMENT_FAILED The payment request failed due to maximum character length exceeded	400 EXCEEDED_FIELD_LENGTH The payment/transfer contains errors : maximum character length exceeded {0}
Sweden	400 PAYMENT_FAILED The payment request failed due to signing amount too small	400 AMOUNT_TOO_SMALL Signing amount is too small
Sweden	400 PAYMENT_FAILED The payment request failed due to invalid bic provided	400 INVALID_BIC Invalid provided bic
Sweden	403 PAYMENT_PRODUCT_LIMIT Payment product limit is exceeded	403 AMOUNT_EXCEEDS_LIMIT The amount exceeds the Monetary limit.
Sweden	403 PAYMENT_STATE_INVALID Invalid state for signing	403 NOTHING_TO_SIGN Nothing to sign
Sweden	400 PRODUCT_INVALID The addressed payment product is not available.	400 RECURRING_PAYMENT_ID Recurring paymentID provided
Sweden	400 PRODUCT_INVALID The addressed payment product is not available.	400 NON_RECURRING_PAYMENT_ID Non recurring paymentID provided
Sweden	400 RECIPIENT_INVALID The recipient is invalid	400 INVALID_RECIPIENT The recipient is invalid
Sweden	400 RECIPIENT_INVALID The recipient is invalid	400 INVALID_RECIPIENT_BIC The recipient is invalid.

Sweden	400 RECIPIENT_INVALID The recipient is invalid	400 INVALID_RECIPIENT_ACC_NUMBER The recipient account number is invalid.
Sweden	404 RESOURCE_NOT_FOUND One or more payments IDs are incorrect	404 BAD_PAYMENT_ID One or more payments IDs are incorrect
Sweden	404 RESOURCE_NOT_FOUND Incorrect basket ID	404 BAD_BASKET_ID Incorrect basket ID
Sweden	404 RESOURCE_NOT_FOUND Basket or authorization ID are incorrect	404 BAD_BASKET_OR_AUTHORIZATION_ID Basket or authorization ID are incorrect
Sweden	403 SCA_METHOD_INVALID To add a new recipient, activate Mobile BankID for extended use.	403 MOBILE_ID_EXCEPTION To add a new recipient, activate Mobile BankID for extended use.
Sweden	403 SCA_METHOD_INVALID Chosen SCA method invalid or unsupported	403 INVALID_SCA_METHOD Chosen SCA device invalid
Sweden	403 SIGNING_FAILED Unexpected error	500 UNSUCCESSFUL_SIGNING Signing was unsuccessful
Sweden	400 STATUS_INVALID Authorisation resource already signed.	400 BASKET_ALREADY_AUTHORISED Basket has related authorisation resource
Sweden	400 STATUS_INVALID Cannot delete basket	400 BAD_BASKET_STATUS Cannot delete basket
Sweden	400 STATUS_INVALID Authorisation resource is in an invalid state.	400 AUTHORISATION_RESOURCE_INVALID Authorisation resource is in an invalid state.
Sweden	403 TIMEOUT Mobile id challenge expired	403 AUTHORIZATION_FAILED Mobile id challenge expired or cancelled
Sweden	403 USER_ACCOUNT_LIMIT The amount limit for signing is exceeded	403 EXCEEDED_SIGNED_AMOUNT The amount limit for signing is exceeded
Sweden	403 USER_RIGHTS_ADD_RECIPIENT Not authorised to sign recipient	403 USER_RIGHTS_ADD_RECIPIENT Not authorised to sign recipient
Sweden	403 USER_RIGHTS_ACCOUNT Not authorised to perform the request	403 USER_RIGHTS_ON_ACCOUNT Not authorised to perform the request
Sweden	403 BASKET_LIMIT Number of payments in the basket is exceeded.	403 EXCEEDED_BASKET_LIMIT Basket limit of 100 payments is exceeded
Sweden	400 PRODUCT_INVALID Payment product is not supported in basket	403 BAD_PAYMENT_STATUS One or more payments are incorrect
Baltics	403 DAILY_LIMIT The payment limit per day is exceeded	400 NARR dailyLimitExceeded
Baltics	400 FUNDS_NOT_AVAILABLE The payment could not be completed because of insufficient funds	400 AM04 Insufficient funds
Baltics	400 FUNDS_NOT_AVAILABLE The payment could not be completed because of insufficient funds	400 AM04 NOT_ENOUGH_FUNDS

Baltics	400 FUNDS_NOT_AVAILABLE The payment could not be completed because of not enough funds for fee	400 NARR notEnoughFundsForFee
Baltics	403 MONTHLY_LIMIT The payment limit per month is exceeded	400 NARR monthlyLimitExceeded
Baltics	400 PAYMENT_FAILED The payment request failed due to unknown transaction parameters	400 AG03 TRANSACTION_NOT_SUPPORTED
Baltics	403 USER_RIGHTS_ACCOUNT Not authorized to perform the request	400 AG01 Account not allowed
Baltics	400 PAYMENT_FAILED The payment initiation POST request failed due to {invalid cut-off time}	400 TM01 INVALID_CUT_OFF_TIME
Baltics	400 PAYMENT_FAILED The payment request failed due to missing debtor	400 MISSING_DEBTOR
Baltics	400 PAYMENT_FAILED The payment request failed due to invalid debtor account	400 AC01 INCORRECT_ACCOUNT_NUMBER
Baltics	400 PAYMENT_FAILED The payment request failed due to invalid account number	400 AC01 INVALID_IBAN
Baltics	400 PAYMENT_FAILED The payment request failed due to currency invalid	400 AM03 INVALID_CURRENCY
Baltics	400 PAYMENT_FAILED The payment request failed due to duplicate payment order	400 AM05 Payment order already exists!
Baltics	400 PAYMENT_FAILED The payment request failed due to payment amount too low	400 AM06 TOO_LOW_AMOUNT
Baltics	400 PAYMENT_FAILED The payment request failed due to currency invalid	400 AM11 INVALID_TRANSACTION_CURRENCY
Baltics	403 LIMIT Amount exceeds limit	400 AM12 INVALID_AMOUNT
Baltics	400 PAYMENT_FAILED The payment request failed due to identification code missing or invalid	400 BE15 INVALID_IDENTIFICATION_CODE
Baltics	400 PAYMENT_FAILED The payment request failed due to debtor identification code missing or invalid	400 BE16 INVALID_DEBTOR_IDENTIFICATION_CODE
Baltics	400 PAYMENT_FAILED The payment request failed due to creditor identification code missing or invalid	400 BE17 INVALID_CREDITOR_IDENTIFICATION_CODE
Baltics	400 RECIPIENT_INVALID Invalid creditor name	400 BE22 MISSING_CREDITOR_NAME

Baltics	400 PAYMENT_FAILED The payment request failed due to payment amount invalid	400 CH20 DECIMAL_POINT_NOT_ COMPATIBLE_WITH_CURRENCY
Baltics	400 PAYMENT_FAILED The payment request failed due to execution date can't be in the future	400 DT01 Requested execution date can't be in the future
Baltics	400 PAYMENT_FAILED The payment request failed due to dynamic text	400 MD03 Payment details or reference number must be set!
Baltics	400 PAYMENT_FAILED The payment request failed due to dynamic text	401 NARR GENERAL_ERROR_NARR
Baltics	400 PAYMENT_FAILED The payment request failed due to incorrect bank identifier	400 RC01 BANK_IDENTIFIER_INCORRECT
Baltics	400 PAYMENT_FAILED The payment request failed due to invalid creditor bank	400 RC04 INVALID_CREDITOR_BANK_IDENTIFIER
Baltics	400 PAYMENT_FAILED The payment request failed due to invalid bic	400 RC05 INVALID_BIC_IDENTIFIER
Baltics	400 PAYMENT_FAILED The payment request failed due to invalid bic	400 RC07 INVALID_CREDITOR_BIC_IDENTIFIER
Baltics	400 PAYMENT_FAILED The payment request failed due to missing remittanceInformationStructured reference	400 RR07 REMITTANCE_INFORMATION_INVALID
Baltics	400 PRODUCT_INVALID Payment service is available in redirect integration only	400 INVALID_FORMAT
Baltics	400 PRODUCT_INVALID Payment product invalid	400 FF03 INVALID_PAYMENT _TYPE_INFORMATION
Baltics	400 PRODUCT_INVALID Cross-border instant payment not supported	400 NARR Cross-border instant payment not supported
Baltics	400 PRODUCT_INVALID Intrabank payment cannot be instant	400 NARR Intrabank payment cannot be instant!
Baltics	400 PRODUCT_INVALID Instant payment amount limit exceeded	400 NARR instantPaymentAmountTooBig
Baltics	400 PRODUCT_INVALID Recurring payment has more than 1 payment initiation	400 SERVICE_INVALID Recurring payment has more than 1 payment initiation
Baltics	400 RECIPIENT_INVALID Invalid creditor account number	400 AC03 INVALID_CREDITOR_ACCOUNT_NUMBER
Baltics	400 RECIPIENT_INVALID Inconsistent with end customer	400 BE01 INCONSISTENT_WITH_END_CUSTOMER
Baltics	400 RECIPIENT_INVALID Missing creditor address	400 BE04 MISSING_CREDITOR_ADDRESS

Baltics	400 PRODUCT_INVALID Only recurring DOMESTIC payments are allowed	400 SERVICE_INVALID Only recurring DOMESTIC payments are allowed
Baltics	403 SCA_METHOD_INVALID Chosen SCA method invalid or unsupported	400 UNSUPPORTED_SCA_METHOD Requested SCA method is not available. Please acquire new access token or choose different method
Baltics	403 SCA_METHOD_INVALID Chosen SCA method invalid or unsupported	400 UNSUPPORTED_SCA_METHOD Executing payments with id-card is not available in LT

**Integration errors updated**

market	v4	v3
All	401 APPLICATION_UNKNOWN Incorrect application status. Please try again later	401 MEDIATOR_APP_BLOCKED Service is blocked
All	401 APPLICATION_UNKNOWN Incorrect application status. Please try again later	404 MEDIATOR_APP_UNKNOWN Mediator APP not found
All	401 APPLICATION_UNKNOWN Incorrect application status. Please try again later	404 MEDIATOR_UNKNOWN
All	403 CERTIFICATE_REVOKED QWAC certificate revoked	400 FORMAT_ERROR QWAC certificate revoked
All	500 INTERNAL_ERROR technical error	500 TECHNICAL_ERROR technical error
All	400 CERTIFICATE_INVALID {0}	400 FORMAT_ERROR {0}
All	400 CERTIFICATE_INVALID No certificate provided in developer portal, in Application setup	400 FORMAT_ERROR No certificate provided in developer portal, in Application setup
All	400 CERTIFICATE_INVALID Provided Certificate doesn't match header: {0}	400 FORMAT_ERROR Provided Certificate doesn't match header: {0}
All	400 CERTIFICATE_INVALID QWAC certificate is waiting for verification, please try in an hour	400 FORMAT_ERROR QWAC certificate is waiting for verification, please try in an hour
All	403 CERTIFICATE_INVALID Unknown certificate used for SSL	403 SERVICE_BLOCKED Unknown certificate used for SSL
All	400 CERTIFICATE_INVALID Unsupported certificate algorithm: {0}	400 FORMAT_INVALID Unsupported certificate algorithm: {0}
All	400 HEADER_INVALID {0}	400 FORMAT_ERROR {0}
All	400 HEADER_INVALID {0}	400 FORMAT_ERROR {0}
All	400 HEADER_INVALID Mandatory header \"Authorization\" is wrong format"	400 FORMAT_ERROR Mandatory header \"Authorization\" is wrong format"
All	400 HEADER_INVALID Mandatory header \"Date\" is wrong format"	400 FORMAT_ERROR Mandatory header \"Date\" is wrong format"
All	400 HEADER_INVALID Mandatory header is missing: {0}	400 FORMAT_ERROR Mandatory header is missing: {0}
All	400 HEADER_INVALID Mandatory header value is empty: {0}	400 FORMAT_ERROR Mandatory header value is empty: {0}

All	403 HEADER_INVALID Missing mandatory header: {0}	403 FORMAT_ERROR Missing mandatory header: {0}
All	404 PRODUCT_INVALID Payment product invalid	404 PRODUCT_UNKNOWN Bad payment product
All	403 ROLE_INVALID No AISP role	403 SERVICE_BLOCKED No AISP role
All	403 ROLE_INVALID No CBPISP role	403 SERVICE_BLOCKED No CBPISP role
All	403 ROLE_INVALID No PISP role	403 SERVICE_BLOCKED No PISP role