Developer Documentation for Swedbank Open Banking Sandbox (BETA)

Version Control

Version	Date	Author	Comment
1.0.1	February 2019	Open Banking API Development Team	This is the baseline version.

Release notes

- Added new payment endpoints:
  - */payments/sepa-credit-transfers* - Baltics only
  - */payments/instant-sepa-credit-transfers* – Baltics only
  - */payments/cross-border- credit-transfers*
  - */payments/se-international- credit-transfers* – Sweden only
  - */payments/{payment-id}/authorisations*
- Added consent authorisation endpoint
  - */consents/{consent-id}/authorisations*
- Added optional header "**Accept**" *to /accounts/{account-id}/transactions* endpoint - Baltics only
- Added Signing Messages at Application Layer - optional for all endpoints

The purpose of this documentation is to help and guide developers around what is possible to access in terms of data regarding customers of Swedbank Group.

The contents of this sandbox will grow over time and potentially also change significantly from the early iterations which may result in breaking backward compatibility. The sandbox facilitates the learning and should help developers to become familiar with methods of accessing data within the Swedbank Open Banking Initiative.

Please note, that accessing data from Swedbank includes all the cooperating Savings Banks in Sweden, Estonia, Latvia, Lithuania and Sweden bank implementations. This also means that format and content may differ, one example is that Sweden does not use Euro as domestic payment currency and other differences may also occur.

The API currently follows the Berlin Group specifications XS2A Interface Interoperability Framework and may be extended to cover further information, beyond what is covered by the PSD2 regulation.

Current sandbox implementation and services that exist are based on 1.2 release of the Berlin Group X2SA specification.

EXPLORE BGS1.2 IMPLEMENTATION GUIDELINES

The exposure of data is done through RESTful services and for the most part both requests and responses are in JavaScript Object Notation (JSON) format. In some cases for the Baltic services XML may be used, this is specifically indicated in the description of the applicable service.

We encourage you to provide feedback in order for us to improve our services by sending a mail to openbanking@swedbank.com

What is the difference between Open Banking and PSD2 services?

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 using financial data without being a regulated entity on the market. This will require a contract to access services provided and may also be subject to specific terms and conditions.

APIs offered as part of the PSD2 regulation is a subset of Open Banking. When PSD2 is applicable, as TPP (Third Party Payment Service Provider) within the PSD2 definition, you will be under supervision of the Financial Supervision Authority in your home member state in the European Union and with this comes a set of rights as well as a set of obligations.

Sandbox Overview

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

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

	Sandbox	**Production**
API Prefix	/sandbox /v1/accounts	/v1/accounts
OAuth2.0 requested scope	scope=PSD2sandbox	scope=PSD2

(OAuth2.0 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*)*

API Overview

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 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 token.

The implemented services fall under **core services** and is not covering all the requirements at this time.

High level overview of actors within the system as defined in the definitions chapter.

Swedbank is following the Berlin Group API specification and where it is needed extensions are made to cover the full set of requirements needed for the service.

The communication between the TPP and the Bank is always secured by using **TLS version 1.2** or higher. Later when the RTS is implemented additional checks against a TPP certificate will be implemented but at the time of this writing the requirements are not fixed.

The rest of the document will only describe the core services in the application layer.

Cache control - API consumers should respect cache policy: *VOLATILE.*

Signing the data requests

In this release **signing is optional** for all Open Banking API's endpoints with POST methods, so this information is provided as information on **upcoming releases**. According to BGS 1.2 requirements, signing on application layer will be **mandatory** on **upcoming releases**.

When you include a signature then 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 these 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 they need to validate the signature. It could be an SSH key fingerprint, a URL to machine-readable key data, an LDAP DN, etc. Management of keys and assignment of 'keyId' is out of scope for this document.	Serial Number of the TPP's certificate included in the "Certificate" header of this request.
algorithm-ID	string	Mandatory	The `algorithm` parameter is used to specify the digital signature algorithm to use when generating the signature. Valid values for this parameter can be found in the Signature Algorithms registry located at http://www.iana.org/assignments/signature-algorithms and MUST NOT be marked "deprecated"	The algorithm must identify the same algorithm for the signature as presented in the certificate (Element "keyId") of this Request. It must identify SHA-256 or SHA-512 as Hash algorithm.
Headers	string	Optional	The `headers` parameter is used to specify the list of HTTP headers included when generating the signature for the message. If specified, it should be a lowercased, quoted list	Mandatory. Must include • "Digest", • "X-Request-ID", • "Date" Optional.

			of HTTP header fields, separated by a single space character. If not specified, implementations MUST operate as if the field were specified with a single value, the \`Date\` header, in the list of HTTP headers. Note that the list order is important, and MUST be specified in the order the HTTP header field-value pairs are concatenated together during signing.	• "PSU-IP-Address" • "TPP-Redirect-URI" **No other entries may be included.**
Signature	string	Mandatory	The \`signature\` parameter is a base 64 encoded digital signature, as described in RFC 4648 [RFC4648], Section 4. The client uses the \`algorithm\` and \`headers\` signature parameters to form a canonicalized \`signing string\`. This \`signing string\` is then signed with the key associated with \`keyId\` and the algorithm corresponding to \`algorithm\`. The \`signature\` parameter is then set to the base 64 encoding of the signature.	[No additional Requirements]

For more information, explore API examples documentation.

Definitions

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

This documentation is published and managed within the API-Explorer; make sure you use the latest version.

Actor	Description
Third Party Provider	Third Party Provider (TPP) is the provider of an application which the user uses and is not offered by the bank. TPP is the client/consumer of the API and acts on behalf of the user through ***consent***. Before the PSD2 deadline, for a bank customer to access their data via a TPP, following conditions must be met: • TPP must have a valid agreement with Swedbank. • The TPP must be enrolled on Swedbank's Open Banking platform and register the third party application before they can use the API.
PSU (User)	The user refers to the bank customer who uses the TPP application.
ASPSP	This is the account servicing provider, i.e, the Bank/Swedbank.
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 live API. The sandbox can be reached within the developer portal at ***developer.swedbank.com.***
API Call	API call is a request towards the API which receives a response. The API is by design stateless, it does not "remember" anything about previous requests, i.e. there is no session. Therefore every request made towards the API must contain certain headers so that the API can authenticate and authorize the use. Message parameters can be passed at different levels; • message parameters as part of the https level (https header) • message parameters by defining a resource path (URL path information) • message parameters as part of the https body
Authentication	Authentication is the process of verifying that an individual, entity or website is who it claims to be. This authentication is later used to grant authorization to specific data and functions within a system. **SCA**(Strong Customer Authentication) is the process of using a strong (2-factor) identification method to identify the customer.
Authorization	Authorization is to validate an authenticated user against a defined access policy and will lead to granting or denying access to the data and functions defined in the policy.
Permission	Permissions are stored in an access policy and are part of the consent given by the user for the TPP. Permissions dictate what the TPP is allowed to do with the user data.
X-Request-ID	This is a unique identifier for the individual request by the TPP. It is a UUID and is generated by TPP. For now this can be set to anything, no validation is performed.
Consent	Consent is the agreement given by the user (PSU) to the third party provider (TPP) to share data from the bank. Consent is stored by the bank and validated by the user according to PSD2 or in the way agreed with an unregulated entity. The consent may have a duration or just be used for a single API call. The given consent will be available for the user to list and revoke within the bank services.
OAuth2	OAuth 2 is an authorization framework that enables applications to obtain limited access to user accounts on an HTTP service.

Oauth2.0 flow	Since the API is built for backend communication and not designed to handle direct client communication it is mandated that the ***grant_type*** will be ***authorization_code*** and will require the use of a ***client_secret*** and a ***client_id***.
Client_secret	A client secret is obtained by registering in the Openbanking portal and logging in to create your application. In the ***Auth*** tab you will see your client_secret (shared_secret). **This is private information and should be kept very 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.**
Client_id	A *client_ id* is obtained by registering in the Openbanking portal and logging in to create your application. In the ***Auth*** tab you will see your *client_id* (API Key).
Redirect_uri	This is the URI where the user should be sent after authorization according to Oauth2.0 is completed, i.e somewhere on the TPP side. You define this when creating your application in the portal.
Scope	A scope according to Oauth2 is defining what data you want to access. For this release of the API you should set the scope to **PSD2sandbox** in a sandbox environment or **PSD2** in a production environment.
State	In Oauth2 a random string is defined as state and is later verified by the TPP. The main purpose is to avoid some cross site request forgery (CSRF) attacks.
Client	The client refers to the client of the API which is commonly the TPP application.
Swagger	An API design and documentation platform to aid in the development lifecycle. It is used to publish the documentation within the **API-Explorer**.
Date	Standard **http header** element date and time. This is a mandatory header parameter for each request.

Connecting to API

To be able to use and connect to the API there are few requirements:

- In the request header **Authorization** with a value of **Bearer** (also called **token authentication**) followed by a string of characters (A-Z, a-z, 0-9 and minus is allowed). This value **WILL** be validated in **ALL** flows in the sandbox.
- **X-Request-ID** must be unique per request. This is a mandatory header throughout the API.
- **Date** must be added to request header. This is a mandatory header throughout the API.

Applications

The TPP provides the application(s), and they are the clients of the API. The application can refer to a website, mobile application, etc. which uses the API. The resource owner (application user) grants the permission for the TPP application (consumer of the API) to use API resources. This permission is given by giving the **CONSENT** after authentication.

Error Codes and Responses

Every response returned by this API has a response code. Response codes can be used to check the result of the requests e.g. was the request successful or did it fail.

The following table shows the return codes used by AIS API.

HTTP response	Text	Description
200	OK	Request was fulfilled.
201	Created	The request has been fulfilled, resulting in the creation of a new resource.
204	No Content	Indicates that request was performed but no content is returned.
302	Found	Redirect.
400	Bad request	Similar to 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided.
401	Unauthorized	Similar to 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided.
403	Forbidden	The request was valid, but the server is refusing action. The user might not have the necessary permissions for a resource.
404	Not Found	The requested resource could not be found but may be available in the future.
405	Invalid input	The method is not allowed on the specific endpoint.
408	Request timeout	The individual request timed out.
415	Unsupported media type	The media type supplied is unsupported by the bank.
429	Too many requests	The TPP has exceeded the number of requests allowed.
503	Service unavailable	The server is currently unavailable

Additional information about the error may be passed using the data element “tpp_messages” in any JSON response message.

EXAMPLE FROM API-EXPLORER

https://psd2.api.swedbank.com/sandbox/v1/consents/?BIC=SANDSESS

```
{ "tppMessages": [{j

   "category": "ERROR",

   "code": "TOKEN_INVALID",

   "text": "additional text information of the ASPSP up to 512 characters"

 }]

}
```

Please note that the underlying data for the account is same and static in all endpoints of the AIS API. However, some endpoints return more data from this model, e.g. account listing endpoint returns more account data compared to account details endpoint even though the underlying data model is the same.

Authentication and security

Overview of an authentication

This is the API used for authentication and authorization purposes. You can use it to invoke the Oauth2 flows as well as posting the consent for the PSU.

Get consent from PSU

There are two steps to getting the consent of the PSU, the first is POST'ing consent to the consent API and then invoking the Oauth2.0 flow to have the customer confirming the consent with strong customer authorization. After this the TPP can collect the Oauth token to use as the authentication mechanism to use on behalf of the PSU when using the PSD2 API.

Prerequisites for oauth2.0

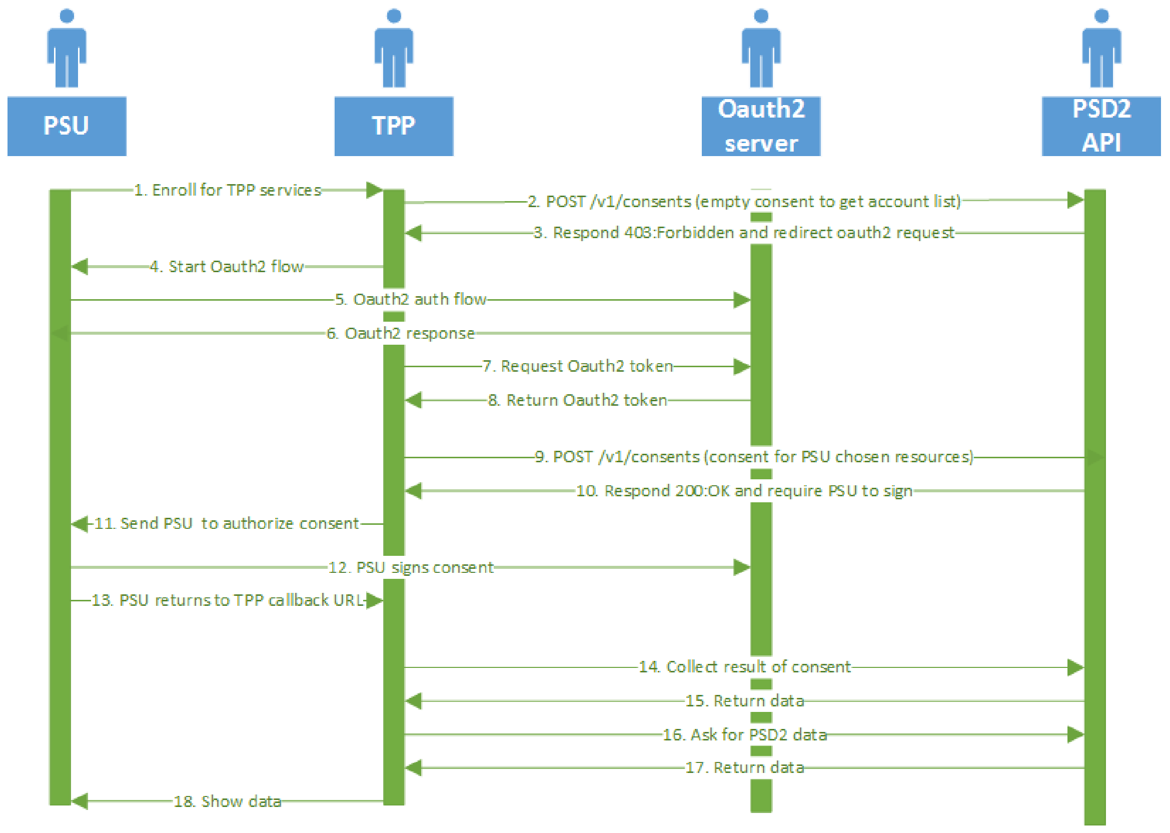
In order to invoke the Oauth2.0 flow you need to have:

- Registered in the Open Banking portal
- Created an application
    - Added a callback ***url***
    - Set type to ***confidential***
    - Obtained your ***client_id***
    - Obtained your ***client_secret***

After these steps you are ready to invoke the flow to request Oauth authentication**.**

The token is the identification method the TPP can use to act on behalf of the PSU to retrieve data without the user's involvement. The callback ***URL*** you registered in the application **MUST** match the ***redirect_uri*** you pass as query parameter.

PSU Authentication with OAUTH2 and initial consent


PSU
TPP
Oauth2 server
PSD2 API
1. Enroll for TPP services
2. POST /v1/consents (empty consent to get account list)
3. Respond 403:Forbidden and redirect oauth2 request
4. Start Oauth2 flow
5. Oauth2 auth flow
6. Oauth2 response
7. Request Oauth2 token
8. Return Oauth2 token
9. POST /v1/consents (consent for PSU chosen resources)
10. Respond 200:OK and require PSU to sign
11. Send PSU to authorize consent
12. PSU signs consent
13. PSU returns to TPP callback URL
14. Collect result of consent
15. Return data
16. Ask for PSD2 data
17. Return data
18. Show data

Walking through of example flow in runtime

1. The PSU interacts with the TPP to enroll for PSD2 services through TPP.
2. The TPP posts the consent for applicable PSD2 resources at the bank. If resources are not known then it is assumed that list of accounts method is used first to be able to specify details of the consent.
3. The bank responds with a 403: forbidden, since no Oauth token is passed in the request and a next link to send PSU into the Oauth flow with strong customer authentication to authorize request according to article 10 if the TPP is a regulated entity or according to contractual agreement if Open Banking partner.
4. The TPP responds with the Oauth2 link to the PSU.

5. PSU accesses link ***https://psd2.api.swedbank.com/psd2/authorize?bic=<BIC_spec>&state=<your_state_string>&client_id=<your_registered_client_id>&redirect_uri=https://<whatever you want the client to come back after Oauth2>&response_type=code&scope=PSD2sandbox***
   and approves Oauth request
   e.g.:
   https://psd2.api.swedbank.com/psd2/authorize?bic=SANDSESS&state=somstring&client_id=l71c2ad437f0fd44d68b2ea3f89b76a28d&redirect_uri=https://www.swedbank.com/openbanking&response_type=code&scope=PSD2sandbox

```
curl --request GET \ --url
'https://psd2.api.swedbank.com/psd2/authorize?bic=SANDSESS&state=somstring&client_id=l71c2
```

```
ad437f0fd44d68b2ea3f89b76a28d&redirect_uri=https://www.swedbank.com/openbanking&response_t
ype=code&scope=PSD2sandbox' \ --header 'date: 2018-10-19'
```

6. And is sent back to <redirect_uri>?code=<access_code>&state=<your_state_string>
7. TPP requests Oauth2 token by calling **https://psd2.api.swedbank.com/psd2/token?grant_type=authorization_code&client_id=<your_client_id>&client_secret=<your_client_secret>&code=<access_code>&redirect_uri=https://<your_redirect_uri>**with a Content-Type header set to ***"application/x-www-form-urlencoded"***
   e.g.:
   https://psd2.api.swedbank.com/psd2/token?grant_type=authorization_code&client_id=l71c2ad437f0fd44d68b2ea3f89b76a28d&client_secret=079603ebf92446f8a70e0ddc11f8d11d&code=fa020756-8255-4c92-a975-4bd1916d4199&redirect_uri=https://www.swedbank.com/openbanking

```
curl --request POST \ --url
'https://psd2.api.swedbank.com/psd2/token?grant_type=authorization_code&client_id=l71c2ad4
37f0fd44d68b2ea3f89b76a28d&client_secret=079603ebf92446f8a70e0ddc11f8d11d&code=fa020756-
8255-4c92-a975-4bd1916d4199&redirect_uri=https://www.swedbank.com/openbanking' \ --header
'Content-Type: application/x-www-form-urlencoded' --header 'date: 2018-10-19'
```

8. The Oauth2 access token and refresh token is returned in a JSON response
9. TPP Posts the details on what resources the customer wants to share with TPP passing Oauth token
10. Bank responds with 200:OK and asks for PSU authorization

	Validity period
Oauth token	90 days

The refresh token can be used to get a new access token once the access token has expired for the lifetime of the refresh token. Beyond that a new Oauth consent flow with the PSU is required again.

For more information explore API examples documentation.