Developer Documentation for Swedbank Open Banking Sandbox (BETA)

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.

As it is defined as a group service data may be queried from Sweden which includes Swedbank all the cooperating Savings Banks, Estonia, Latvia and Lithuania bank implementations. This also means that format and content may differ, one example is that Sweden does not use Euro as 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.

The 1.0 release of the Berlin Group X2SA specification was released 2018-02-08 and there are several items in specification that differ from current sandbox implementation and services that currently does not exist. Adjustments are to be made in upcoming releases.

Currently limited first versions of the targeted payment types are available. Everything is in early stages and will be subject to change over time until a stable release is defined and proper versioning applied.

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 an mail to openbanking@swedbank.com

Change log

- Security API added
- BICs changed from real BICs in production
- Definition updates
- Error codes update
- Formatting and typos ☺

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

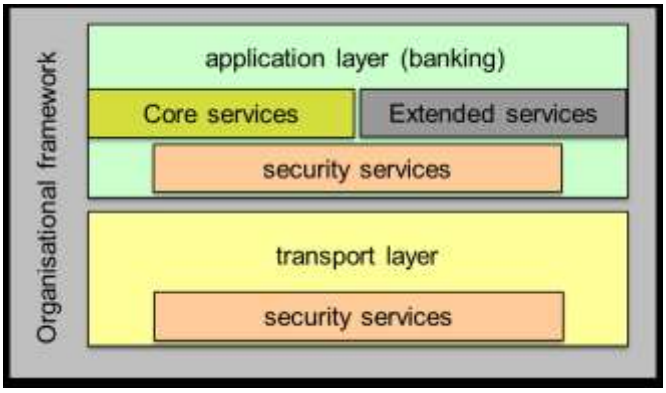
Sandbox is created in Swedbank Open bank initiative to support more detailed technical understanding of API's. Current version of API is based on static data and is subject to change. For using sandbox (compared to real API's) following configuration changes are needed:

- For Sandbox please use following BIC values:
  - For Swedbank Sweden – SANDSESS (production SWEDSESS)
  - For Swedbank Lithuania – SANDLT22 (production HABALT22)
  - For Swedbank Latvia – SANDLV22 (production HABALV22)
  - For Swedbank Estonia – SANDEE2X (production HABAEE2X)
- API is prefixed with /sandbox e.g. real API /v1/accounts is implemented as /sandbox/v1/accounts
- 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
- For sandbox OAuth2.0 requested scope must be scope=PSD2sandbox (production API will require scope=PSD2)

API Overview

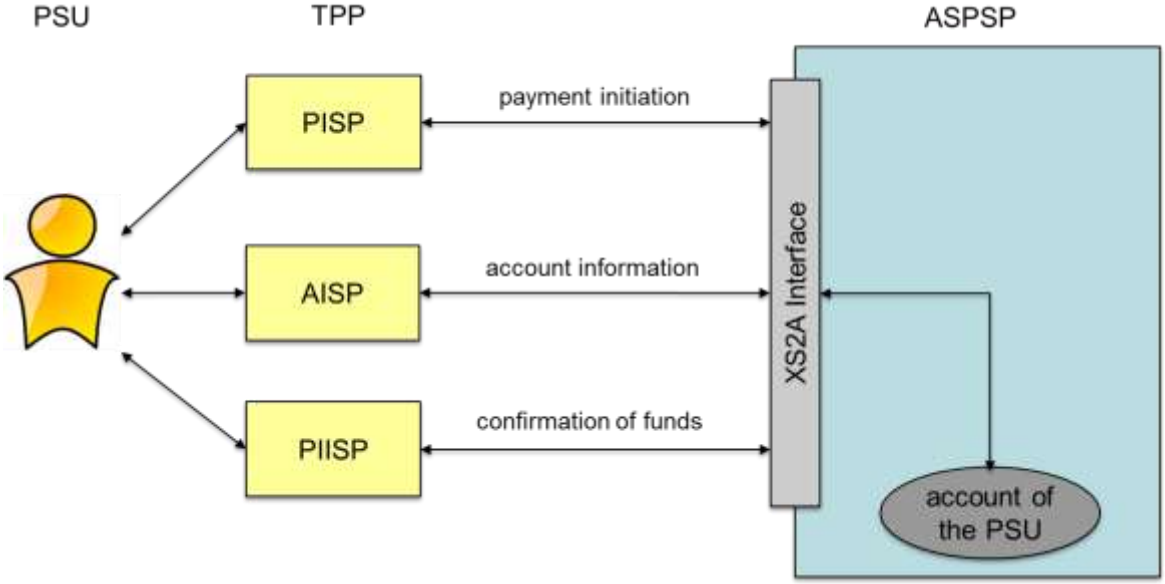
The API sandbox consists of **static mocked data** for

- Account Information Services (AIS) for Sweden only 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
- Security and consent services (SEC) to create consent and request an oauth token.



The implemented services fall in under core services in the above picture 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.

Please note: Limited security services and consent handling is available in this release for the purpose of showing how to get a token and create a consent, it is not always checked that the token you supply is actually valid in the data requests.

The communication between the Bank and the TPP is secured using TLS 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.

API HTTP Methods

- GET - This method reads a resource and returns it. It returns 200 on success.
- POST - This method creates a new resource. It returns 201 on success.

The following table shows methods supported by the APIs.

API	GET	POST
SEC	X	X
AIS	X	-
PIS	X	X

API Responses and Response Codes

This API consumes JSON objects and returns responses as JSON objects and in some cases in payment initiation for the Baltic banks an ISO XML in the form of a pain.001, pain.002 and a camt.05x.

Whenever XML is used this is specifically pointed out in the examples section to see how the response objects look like. The responses are returned are encoded in UTF-8 format and will be in *snake_case*.

The objects which the API returns as a response are specified in the API reference and they all contain some attributes that are common to all objects. For instance, every object has return code embedded in them which makes it easy for application developers to check whether the request was successful or not.

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.

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.

User
The user refers to the bank customer who uses the TPP application. (PSU)

ASPSP
This is the account servicing provider, i.e, the Bank.

Sandbox
Sandbox gives access to a small set of static data 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 or 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.

TPP-Transaction-ID
This is a unique identifier for the transaction sent from the TPP. It is a UUID generated at the TPP side. This can be used in certain circumstances to link several transactions into a concept of a "session" even though the API is stateless by design. For more information please see Berlin Group documentation. For now this can be set to anything, no validation is performed.

TPP-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 to share data from the bank. A 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). **Now this is something secret and should be kept very safe, if it is lost it can potentially be used in malicous ways and we don't want that, do we? 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 accoring to Oauth2 is defining what data you want to access. For this release of the API you should always set the scope to **PSD2sandbox**.

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.

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 followed by a string of characters (A-Z, a-z,0-9 and minus is allowed). This value will **MAY** be validated in some flows in the sandbox but not all. You should plan for inserting the oauth token you get from the oauth request flow. Also process-id will in future releases be renamed to TPP-Transaction-ID and request-id will be renamed to TPP-Request-ID according to Berlin Group field changes. These are mandatory headers throughout the API. Process-id must be a unique identifier for the transaction that is initiated and request-id must be unique per request.

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.
In this version of the API it is simply assumed that this has happened previously and you are passing in a Bearer token which currently is not validated.

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.

HTTP response	Text	Description
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	Validation error
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:


```
{"tpp_messages": [{
  "category": "ERROR",
  "code": "INVALID_REQUEST",
  "text": "Consent request is not valid json or does not pass json schema validation."
}]}
```


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.

Security and consent API

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. In terms of PSD2 the requirements on certificates and signing of payload are not available in this release.

Get consent from the PSU

There are two steps to getting the consent of the PSU, the first is POST'ing a 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.

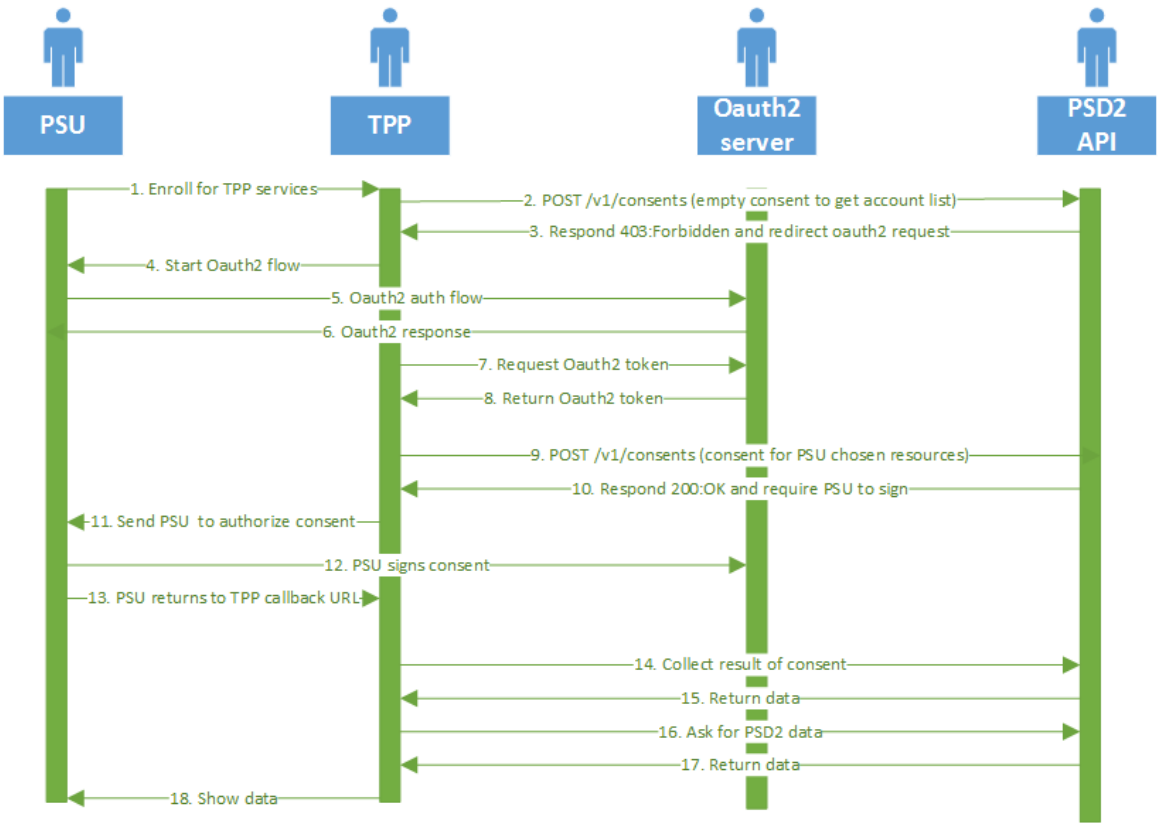
Prerequisites for oauth2.0

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

- registered in the Openbanking portal
- Created an application
    - Added a callback url
    - Set type to Confidential
    - Obtained your client_id
    - Obtained your Client_secret

After this you are ready to invoke the flow to request oauth authentication.

The Oauth2.0 flow should logically flow like this between the PSU, TPP and bank. The token is the identification method the TPP can use to act on behalf of the PSU to retrieve data without the users involvement. The callback url you registered in the application MUST match the redirect_uri you pass as query parameter.



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 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 Openbanking 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>&response_type=code&scope=PSD2sandbox&client_id=<your_registered_client_id>&redirect_uri=https://<wherever you want the client to come back to after Oauth2>&state=<your_state_string>* and approves Oauth request
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/sandbox/v1/authenticationpsd2/authorize/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”
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

11. TPP sends PSU to authorize
12. PSU authorizes
13. PSU Is sent back to TPP
14. TPP collects result of consent
15. Bank returns result of consent
16. TPP Request some data using token in Authorization header and referring to the consent-id of PSU
17. Is served some data from the bank
18. Show data to the user

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.

Signing the data requests

In this release no signing is mandated so this information is provided as information on upcoming releases and may change as this is still work in progress.

Once signing on the application layer is implemented the bank may mandate that requests are signed.

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	Mandatory. Must include • “Digest”, • “TPP-Transaction-ID”,

Elements of the “Signature” Header				
Element	Type	Condition	Requirement	Additional Requirement
			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 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.	• “TPP-Request-ID”, • “PSU-ID” (if and only if “PSU-ID” is included as a header of the HTTP-Request). • “PSU-Corporate-ID” (if and only if “PSU-Corporate-ID” is included as a header of the HTTP-Request). • “Date” **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]

More information and examples in upcoming releases.

API Examples Security and Consent

Below you can find examples how to use the API endpoints.

Example: Post consent
This example creates a consent resource based on the PSU:s orders.The endpoint uses POST only.

This endpoint URL has the following form:

https://psd2.api.swedbank.com/sandbox/v1/consents

HTTP Headers

Attribute	Type	Condition	Description
TPP-Transaction-ID	UUID	Mandatory	
TPP-Request-ID	UUID	Mandatory	
Authorization	String		The oauth2 token you got from the collect oauth2 token flow if available or “dummyToken” if you want to bypass oAuth2 token checks
Content-Type	String	Mandatory	Must be set to application/json
Signature (future use)		Future use	A signature of the request by the TPP on application level. Not currently implemented.
TPP-Certificate (future use)	String	Future use	The certificate used to sign the request. Not currently implemented

HTTP Query parameters:

Attribute	Type	Condition	Description
bic	String	Mandatory	Swedish (bic=SANDSESS) or Baltic customer (bic=SANDEE2X, bic=SANDLT22, SANDLV22)
TPP-Redirect-Preferred	boolean	Optional	true means oauth flow preferred and is the only supported option. Always true in this release

Http Request body:

Attribute	Type	Condition	Description
Access	Account	Mandatory	The access definition to the resources agreed
Recurring_indicator	Boolean	Mandatory	indicating if this is recurring access or a one off access to account data
Valid_until	string	Mandatory	This parameter is requesting a valid until date for the requested consent. The content is the local ASPSP date in ISODate Format, e.g. 2017-10-30 If a maximal available date is requested, a date in far future is to be used: “9999-12-31”. The consent object to be retrieved by the GET Consent Request will contain the adjusted date.
Frequency_per_day	Integer	Mandatory	This field indicates the requested maximum frequency for an access per day. For a once-off access, this attribute is set to “1”.
Combined_service_indicator	boolean	Optional	If "true" indicates that a payment initiation service will be addressed in the same "session

The access is a JSON account structure in the below format:
Please note that sematical changes are needed to align to Berlin Group 1.0 spec. Will happen in upcoming releases.


```
{
"access_accounts" :
   [
            {"id": "LT377300010010269362","access": ["balance","transactions","payment"]},
        {"id": "LT377300010010270623","access": ["balance","transactions","payment"]},
            {"id": "LT747300010010270636","access": ["balance","transactions","payment"]},
```

```
            {"id": "LT377300010010270987","access": ["balance","transactions"]},
            {"id": "LT377300010010270568","access": ["confirmation-of-funds"]}
        ],
"recurring_indicator" : "true",
"valid_until" : "2019-01-01",
"frequency_per_day": 4
}
```


Example: Request Oauth2.0 authentication
This example request initiation of the Oauth2.0 authentication sequence of the PSU.

This endpoint URL has the following form:

https://psd2.api.swedbank.com/psd2/authorize

HTTP Headers

Attribute	Type	Condition	Description
Signature (future use)		Future use	A signature of the request by the TPP on application level. Not currently implemented.
TPP-Certificate (future use)	String	Future use	The certificate used to sign the request. Not currently implemented

HTTP Query parameters:

Attribute	Type	Condition	Description
bic	String	Mandatory	Swedish (bic=SANDSESS) or Baltic customer (bic=SANDEE2X, bic=SANDLT22, SANDLV22).
state	String	Optional	A random string used as an authentication parameter used to help mitigate CSRF attacks. Strongly advised to implement.
Client_id	String	Mandatory	The client_id of the TPP
Redirect_uri	String	Mandatory	The callback uri that should be used after oauth flow
response_type	String	Mandatory	Must be set to **code** which chooses the authorization_code flow of OAuth2.
scope	string	Mandatory	Use PSD2sandbox in the sandbox

The following cURL can be used to request Oauth2.0 authentication

```
curl "https://psd2.api.swedbank.com/psd/authorize/?bic=SANDSESS&scope=PSD2sandbox&client_id=<your client id>&response_type=code
```

redirect_uri must be set as a query parameter AND in the sandbox directly in the application definition.

Example: Collect Oauth2.0 token
This example uses POST to send data required to create the oauth2.0 tokens that are returned as part of the response. This endpoint supports only POST requests.

This endpoint URL has the following form:

https://psd2.api.swedbank.com/psd2/token

HTTP Headers

Attribute	Type	Condition	Description
Content-Type		Mandatory	Application/x-www-form-urlencoded
Signature (future use)		Future use	A signature of the request by the TPP on application level. Not currently implemented.
TPP-Certificate (future use)	String	Future use	The certificate used to sign the request. Not currently implemented

HTTP Query parameters:

Attribute	Type	Condition	Description
State	String	Optional	A random string used as an authentication parameter used to help mitigate CSRF attacks. Strongly advised to implement.
Client_id		Mandatory	The client_id of the TPP
Client_secret		Mandatory	The client_secret obtained when creating the application
Redirect_uri		Mandatory	The callback uri that should be used after oauth flow
Grant_type		Mandatory	must be set to **authorization_code**
Code		Mandatory	The code that was obtained from the oauth server

Account Information Services API

This is used to get information about accounts connected to a customer. A prerequisite for querying is that consent is collected already and consent information is passed in the request. In all our queries a BIC code is mandatory to specify in order to route the request to the correct backend. *Please note that Swedbank Sweden and cooperating Savings Banks share the same BIC.*

Browse API-Explorer for more information and refer to examples further down in this document.

The information can be categorized in following categories:

- Account list
- Balances
- Transaction history

Response Format

The AIS API response format is in JSON and looks like the following.

Response: HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Encoding: gzip
Content-Type: application/json;charset=UTF-8

Content-Length: 203
Date: Fri, 13 Oct 2017 12:56:52 GMT

{"account_list" : [
  {"id": "AsdF01234EfgH4567",
"name":"privatkonto",
"currency":"SEK",
“product”:”privatkonto”,
"account_type":"CACC ",
"iban":"xxxxxxxx",
“bic”:”SANDSESS”,
“clearingnumber”:”xxxx”,
“accountnumber”:”xxxx”,
},
{"id": "AbcD1234eFgH568",
"name":"Privatkonto",
"currency":"SEK",
“product”:”Privatkonto”
"account_type":"CACC ",
"iban":"xxxxxxx",
“bic”:”SANDSESS”,
“clearingnumber”:”xxxx”,
“accountnumber”:”xxxx”,
}
]}

The Response header will show the http response code, encoding, content type as well as the creation time. The response contains the actual payload of the response, and the endpoint in question defines its structure.

API Examples AIS

Below you can find examples how to use the API endpoints.

Example: List Accounts
This example lists all payment accounts on the user.

This endpoint URL has the following form:

https://psd2.api.swedbank.com/sandbox/v1/accounts

HTTP Headers

Attribute	Type	Condition	Description
TPP-Transaction-ID	UUID	Mandatory	
TPP-Request-ID	UUID	Mandatory	
Authorization	String		The oauth2 token you got from the collect oauth2 token flow if available or “dummyToken” if you want to bypass oAuth2 token checks

Signature (future use)		Conditional	A signature of the request by the TPP on application level. Not currently implemented.
TPP-Certificate (future use)	String	Conditional	The certificate used to sign the request. Not currently implemented

HTTP Query parameters:

Attribute	Type	Condition	Description
bic		Mandatory	Swedish (bic=SANDSESS) or Baltic customer (bic=SANDEE2X, bic=SANDLT22, SANDLV22).
with-balance	boolean	Optional	true means oauth flow preferred and is the only supported option

This endpoint supports only GET requests.

The following cURL can be used to fetch account list from this endpoint.

```
curl "https://psd2.api.swedbank.com/sandbox/v1/accounts/?bic=SANDSESS"
-H "Authorization: Bearer dummyToken"
-H "Process-ID: AZXS3456"
-H "Request-ID: 12345SGHDF"
-H "Date: Thu, 01 Dec 1994 16:00:00 GMT"
```

Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Encoding: gzip
Content-Type: application/json;charset=UTF-8
Content-Length: 203
Date: Fri, 13 Oct 2017 12:56:52 GMT
```


```
{"account_list": [
    {
    "id": "AsdF01234EfgH4567",
    "currency": "SEK",
    "product": "privatkonto",
    "account_type": "CACC",
    "iban": "SE4880000123459876543219",
    "bic": "SANDSESS",
    "bban": "1234-5,987 654 321-9",
    "clearingnumber": "1234-5",
    "account_number": "987 654 321-9",
    "balances": [{"booked":      {
      "amount":         {
        "currency": "SEK",
        "content": 100
      },
      "date": "2017-11-02"
```

```
    }}]
  },
    {
    "id": "AbcD1234eFgH568",
    "currency": "SEK",
    "product": "ungdomskonto",
    "account_type": "CACC",
    "iban": "SE4880000123451234567890",
    "bic": "SANDSESS",
    "bban": "1234-5,123 456 789-0",
    "clearingnumber": "1234-5",
    "account_number": "123 456 789-0",
    "balances": [{"booked":      {
      "amount":         {
        "currency": "SEK",
        "content": 35000
      },
      "date": "2017-11-02"
    }}]
  }
]}
```


Example: Fetch Account Information on specific account

In this example, we fetch account information by ACCOUNT-ID which can be found by listing the accounts.

The endpoint URL has the following form:

https://psd2.api.swedbank.com//sandbox/v1/accounts/{ACCOUNT-ID}

HTTP Headers

Attribute	Type	Condition	Description
TPP-Transaction-ID	UUID	Mandatory	
TPP-Request-ID	UUID	Mandatory	
Authorization	String		The oauth2 token you got from the collect oauth2 token flow if available or “dummyToken” if you want to bypass oAuth2 token checks
Signature (future use)		Conditional	A signature of the request by the TPP on application level. Not currently implemented.
TPP-Certificate (future use)	String	Conditional	The certificate used to sign the request. Not currently implemented

HTTP Query parameters:

Attribute	Type	Condition	Description
bic		Mandatory	Swedish (bic=SANDSESS) or Baltic customer (bic=SANDEE2X, bic=SANDLT22, SANDLV22).
with-balance	boolean	Optional	true means oauth flow preferred and is the only supported option

This endpoint supports only GET HTTP method.

Response:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Encoding: gzip
Content-Type: application/json;charset=UTF-8
Content-Length: 203
Date: Fri, 13 Oct 2017 13:06:16 GMT
```


```
{
  "id": "AbcD1234eFgH568",
  "currency": "SEK",
  "product": "ungdomskonto",
  "account_type": "CACC",
  "iban": "SE4880000123451234567890",
  "bic": "SANDSESS",
  "bban": "1234-5,123 456 789-0",
  "clearingnumber": "1234-5",
  "account_number": "123 456 789-0",
  "balances": [{"booked":    {
    "amount":       {
      "currency": "SEK",
      "content": 35000
    },
    "date": "2017-11-02"
  }}]
}
```


Example: Fetch Account Information on specific account

This is how it looks if you query an unsupported BIC.

```
GET https://psd2.api.swedbank.com//sandbox/v1/accounts/AsdF01234EfgH4567/?bic=SANDSESS
Authorization: Bearer dummyToken
Response:       HTTP/1.1 404 Not Found
                Server: Apache-Coyote/1.1
                Content-Type: text/plain;charset=UTF-8
                Content-Length: 70
                Date: Fri, 13 Oct 2017 13:15:13 GMT
                Connection: close

                {"tpp_messages" : [{"category" : "ERROR",
                "code" : "NOT_IMPLEMENT"}]}
```

Example: Read Transaction list

The following request will fetch transactions from the account by ACCOUNT-ID. Note that this query also requires parameters DateFrom and DateTo.

These parameters mentioned above control the time range from between which to fetch the transactions, and they are mandatory. They are sent as query parameters.

The endpoint URL has the following form:
https://psd2.api.swedbank.com//sandbox/v1/accounts/{ACCOUNT-ID}/transactions

This endpoint supports only GET method.
The **{ACCOUNT-ID}** URL parameter(s) can be found from account listing request above.

HTTP Headers

Attribute	Type	Condition	Description
Process-ID	UUID	Mandatory	
Request-ID	UUID	Mandatory	
Authorization	String		The oauth2 token you got from the collect oauth2 token flow if available or “dummyToken” if you want to bypass oAuth2 token checks
Signature (future use)		Future use	A signature of the request by the TPP on application level. Not currently implemented.
TPP-Certificate (future use)	String	Future use	The certificate used to sign the request. Not currently implemented

HTTP Query parameters:

Attribute	Type	Condition	Description
bic		Mandatory	Swedish (bic=SANDSESS) or Baltic customer (bic=SANDEE2X, bic=SANDLT22, SANDLV22).
with-balance	boolean	Optional	true means oauth flow preferred and is the only supported option
date_from	ISODate	Mandatory	Starting date of transaction list
date_to	ISODate	Optional	Ending date of transaction list, if empty then now is assumed

Response: HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Encoding: gzip
Content-Type: application/json;charset=UTF-8
Content-Length: 1105
Date: Fri, 13 Oct 2017 13:06:16 GMT


```
{
  "transactions": {
    "booked": [
      {
        "credit_debit": "Debited",
        "amount" : {"currency" : “SEK”, content: 343 } ,
        "booking_date": "2017-10-30",
        "transaction_date": "2017-10-28",
        "value_date": "2017-10-30",
```

```
        "remittance_information": "Fackavgift",
        "balances": {
          "booked": {
            "amount": {
              "currency": "SEK",
              "content": 10000
            },
            "date": "2017-11-02"
          }
        }
      },
      {
        "credit_debit": "Credited",
        "amount" : {"currency" : “SEK”, content: 2365 } ,
        "booking_date": "2017-10-25",
        "transaction_date": "2017-10-24",
        "value_date": "2017-10-25",
        "remittance_information": "Löneinkomst",
        "balances": {
          "booked": {
            "amount": {
              "currency": "SEK",
              "content": 35000
            },
            "date": "2017-11-02"
          }
        }
      }
    ]
  }
}
```


Payment Initiation Services API

This is used to submit a payment initiation request on behalf of a customer. It is assumed that the account information has been obtained from the customer in some way prior to the request most commonly through a query of the list accounts request. In all our queries a BIC code is mandatory to specify in order to route the request to the correct backend. See examples for more information.

Please note that there is a 15 minute timeout from the registration of a payment by a TPP until the customer signs, if this time is exceeded then the payment is deleted and the TPP will need to request a new payment initiation.

API Examples PIS

Below you can find examples how to use the API endpoints.

Example: Initiate SEPA payment with pain.001 XML
This example intiates a Domestic, Intrabank or SEPA payment using a pain.001 XML. This is only valid for the Baltic BIC's. This endpoint supports only POST requests.

This endpoint URL has the following form:

https://psd2.api.swedbank.com//sandbox/v1/payments/pain.001-sepa-credit-transfers

HTTP Headers

Attribute	Type	Condition	Description
TPP-Transaction-ID	UUID	Mandatory	
TPP-Request-ID	UUID	Mandatory	
Authorization	String		The oauth2 token you got from the collect oauth2 token flow if available or “dummyToken” if you want to bypass oAuth2 token checks
Signature (future use)		Future use	A signature of the request by the TPP on application level. Not currently implemented.
TPP-Certificate (future use)	String	Future use	The certificate used to sign the request. Not currently implemented
Content-type	string	Mandatory	application/xml

HTTP Query parameters:

Attribute	Type	Condition	Description
bic		Mandatory	Baltic customer (bic=SANDEE2X, bic=SANDLT22, SANDLV22).

Request Body

A pain.001 structure corresponding to the payment product.

The structure of the pain.001 is validated that it conforms to the schema.

Example: Initiate Swedish domestic payment

This example intiates a Swedish payment in SEK and allows the use of BBAN, IBAN or Swedish Bank or PostGiro payments. Only SANDSESS is allowed as BIC. This endpoint supports only POST requests.

This endpoint URL has the following form:

https://psd2.api.swedbank.com//sandbox/v1/payments/se-domestic-ct

HTTP Headers

Attribute	Type	Condition	Description
TPP-Transaction-ID	UUID	Mandatory	
TPP-Request-ID	UUID	Mandatory	
Authorization	String		The oauth2 token you got from the collect oauth2 token flow if available or “dummyToken” if you want to bypass oAuth2 token checks
Signature (future use)		Future use	A signature of the request by the TPP on application level. Not currently implemented.
TPP-Certificate (future use)	String	Future use	The certificate used to sign the request. Not currently implemented

Content-Type	String	Mandatory	application/json

HTTP Query parameters:

Attribute	Type	Condition	Description
bic		Mandatory	Swedish (bic=SANDSESS)

Example with IBAN

{
"instructed_amount" : {"currency" : "SEK" , "content" : 123},
"debtor_account" : { "iban":"SE4880000123459876543219"},
"creditor_account": {"iban":"SE4880000123459876543219"},
"remittance_information_unstructured" : "Ref Number Mottagare-1234567890"
}

Example with BBAN

{
"instructed_amount" : {"currency" : "SEK" , "content" : 123},
"debtor_account" : { "bban":"1234-5,987 654 321-9"},
"creditor_account": {"bban":"1234-5,987 654 321-9"},
"remittance_information_unstructured" : "Ref Number Mottagare-1234567890"
}

Example BG account BBAN

{
"instructed_amount" : {"currency" : "SEK" , "content" : 123},
"debtor_account" : { "bban":"1234-5,987 654 321-9"},
"creditor_account": {"bban":"BG 2345-6789"},
"remittance_information_unstructured" : "OCR Number 1234567890"
}

Common Issues

In this section, we will collect the common issues that the users of the API face. This section will be updated over time.