Skandiabanken
Technical documentation Open Banking

1 Introduction

Welcome to Skandiabanken Open Banking solutions!

1.1 Background

This document and our information site are created to provide the information needed to access our API's. If you have questions or suggestions, you are welcome to contact us. Contact information can be found in section 1.3 Contact us and channels for communication.

Info site: www.skandia.se/openbanking
Portal: https://developer.skandia.se/open-banking/core-bank/
Test-Portal: https://developer.test.skandia.se/open-banking/core-bank/

1.2 Status

This section will clarify what we have in place and the status of the functionality. Please read the rest of this documentation for further information and details.

Functionality	Available	Comments
SCA	Yes	Redirect and decoupled solution offered using Swedish BankID.
TPP-identification (eIDAS)	Yes	We support eIDAS/QWAC. QSealC is currently not required but we are investigating it.
Relevant standards	Yes	Berlin Group NextGenPSD2 Framework version 1.3.6 OAuth2 including OpenID Connect TLS 1.2
AIS	Yes	Payment account information for private customers over 18 years. We have no business customers with payment accounts.
PIS	Yes	Domestic transfer, cross border payments, recurring payments, giro payments and signing basket.

During the timespan 04:00-04:30 each day our service provider restarts services and servers, which has impact on performance on all our channels (Online, App and Open Banking). For you to get the best experience, and in the end our common end customers, we recommend that during that time not to schedule batch calls. Response times will be much longer, and error rate will be higher. We apologize for this but we want to give you as much insight as possible to obtain the best possible service.

1.3 Contact us and channels for communication

If you have questions, feedback or general problems you are welcome to contact us via mail openbanking@skandia.se. This mailbox is mainly monitored during business hours. Please use this for technical questions or business inquires.

If you have very urgent issues outside business hours, please contact our customer and TPP support: https://www.skandia.se/kontakta-skandia/kontakta-oss/.
Mainly use phone: +46771555500
Open hours 07-23 Monday through Friday, 08-22 Saturday and Sunday, year around.

1.4 Abbreviations/Acronyms

Abbreviation/Acronym	Description
OTP	One-time password Sometimes Skandiabanken sends an OTP, one-time password, to the PSU for additional security. An OTP (6 digits, range 100000-999999) is sent to PSU's mobile phone number and should be entered by PSU and is then verified by Skandiabanken.

2 Document history

Version	Date	Description
1.39	2026-03-20	- Revised documentation to align with new API versions (PIS v4, AIS v3) - Added important note on error handling about timeout responses from API-Gateway Update PSU-IP-Address for Get Payment Status to not mandatory
1.38	2025-11-11	- Added information about possible returned http status code 429
1.37	2025-06-26	Fixed query parameters in 9.2.1 Authorize, initiate authentication and get available authentication methods
1.36	2025-05-27	Removed scope information and examples with url-encoded values
1.35	2025-05	Decoupled PSU authentication for AIS access and textual fixes
1.34	2025-04-11	- Added information about auth flow for new auth servers
1.33	2025-02-10	- Info RequestedExecutionDate: max 2 years in the future. - Info RequestedExecutionDate (cross border payment): max 6 months in the future. - New processing status: AMOUNT_LIMIT_EXCEEDED - Removed optional parameter CreditorAgentName
1.32	2024-12-19	- Signing basket operations in PIS v3: 8.10, 8.11, 8.12, and 0. - 10.1 Sequence diagram of decoupled SCA method.
1.31	2024-10-08	- 9 Decoupled SCA method
1.30	2024-05-06	New chapter added regarding PIS v3. Chapters regarding PIS v2 and PIIS was removed.
1.29	2024-03-20	
1.28	2024-02-13	Instruction on how to change certificate.
1.27	2023-10-10	Updates regarding HTTP 403.
1.26	2023-07-11	Updated section 8.3.4 and 8.3.6 regarding debtor account.
1.25	2023-06-27	- Updated information regarding change from 90 to 180 days consent for PSUs. - Corrected section 8.3.1 regarding Payment initiation – Request description. Removing unsupported headers TPP-Redirect-URI and TPP-Nok-Redirect-URI Corrected section 8.3.3 regarding Payment initiation - Domestic credit transfer request. Removing unsupported field DebtorAccount.Iban
1.24	2023-06-08	- Corrected section 8.11.2 regarding payment status. - Addition of examples for Giro payments, section 0.
1.23	2023-03-15	- Clarification of parameter “requestedExecutionDate” - Body description single payment initiation - GiroDomesticCreditTransfer as well as 0, 8.14.0 Execution rules for payments. - Section 6.3 General error messages is reworked and more detailed error messages are added - Section 7.3.5 HTTP 404 Resource unknown added
1.22	2022-10-24	- Move section version history to top of document. - Rework and added clarifications in section 5 Security and authentication of TPP including SCA of PSU - Correction of section 6.2.4 Refresh token. - More example responses under section 0 - Get Account List. - Clarification of responses on sections about AIS and PIS. - New section about expired token upon authorization of a payment.
		-

3 Test environment and sandbox

We currently support testing of the same services as we have in production.

To be able to use the test environment you need:

- Test or production eIDAS-certificate. If you use a test certificate:
  - The certificate must be issued by a publicly trusted CA. No self-signed certificates will be accepted.
  - The certificate must contain a correct Organization Identifier (2.5.4.9.7 = PSDXX-YYYY-ZZZZZZZZ)
  - The certificate must contain a QC Statement with the PSD2 roles that you intend to use.
- A test BankID. (Swedish BankID is currently the only supported SCA).
  - The BankID client must be configured as a test BankID client. More info about how this is done is available at demo.bankid.com

3.1 Onboarding of TPP in Test Environment

The process of onboarding in the Test Portal is the same as onboarding in the Production Environment but with a few exceptions.

1. You reach the test portal at https://developer.test.skandia.se/open-banking/core-bank/
   a. You will have to create separate users for the Test and Production Portals.
   b. Since there are separate users you will receive different Client-Ids in Test and Production.
2. You only need a test eIDAS-certificate and not a production certificate.
   a. You can use your production certificate, but it is not necessary.

Except for the above differences you can follow the instruction in 4 Onboarding of TPP in Production Environment when onboarding in the Test Environment.

3.2 Guidelines for the Test Environment

There is a limited number of test PSU's in the environment, please contact openbanking@skandia.se after onboarding to receive a specific test PSU to run tests on.

The account numbers used in testing must follow the logic of real accounts. You cannot test transfers to made up accounts. Notice that other TPPs also can see the test PSUs transactions history. Any account numbers used in testing can therefore be seen by other TPPs.

4 Onboarding of TPP in Production Environment

This section provides a process overview how a TPP can gain access to our production environment including the developer portal and API gateway.

1. Get an approval from a local NCA (Finansinspektionen in Sweden) as an AISP or PISP. If you are a bank you already have an approval.
2. Obtain a production eIDAS-certificate from a QTSP. QWAC is required.
    a. Note 1: you cannot use a CA root certificate
    b. Note 2: you cannot use a test certificate
3. Register in our portal and create apps and subscriptions. You will need the public key of your production eIDAS QWAC certificate in this process.
4. When you add a subscription to an app there will be a manual step for us to grant your app access. You will receive an e-mail when your app is granted access.
5. You can access the APIs (AIS and/or PIS) your certificate allows you to invoke

4.1 Create an account in the portal

Browse to the portal (https://developer.skandia.se/open-banking/core-bank).

To register an account:

1. Navigate to the portal. Click the “CREATE ACCOUNT” button (top right corner).
2. Enter the fields:
    a. user name (please make a note of the selected user name, it is required for login)
    b. email address (will be used for communication with your organization)
    c. phone (optional but recommended)
    d. first name
    e. last name
    f. consumer organization (defines the name of the collective under which client applications will be created)
    g. password and confirm password and captcha. Press the “Sign up” button
3. Open the email received from the portal and follow the activation link. This will activate the user account.
4. Login using the username and password selected by you

4.2 Adding a colleague to your consumer organization

It is possible to invite additional users to your consumer organization. This can be done while logged in as the consumer organization owner/creator or if you possess the administrator role of that specific consumer organization.

The new user is invited using an e-mail address. An e-mail containing a registration link is sent to the provided e-mail address. If the e-mail is registered to an existing user account, following the link will add that user to the specific consumer organization. If the e-mail is previously unused, following the link will require the user to step through the process for creating an account except that no new consumer organization needs to be created.

4.3 How to setup your apps and subscriptions

Product

The API Products view provides an overview of the products available in the portal. A product acts as a collection of APIs. To enable invocation of an API, you must subscribe with an app to a product’s plan containing the desired API.

API

An API is a set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service. The API can be protected by various security mechanism, including, but not limited to, Client-Id identification, MTLS and OAuth.

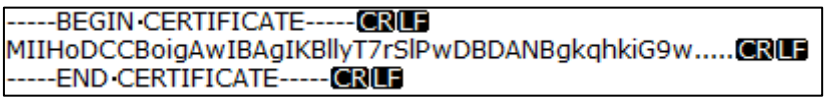
We expose a set of APIs which makes it possible for third parties to create applications using payment account information and payment initiation.

Plan
A plan represents a collection of APIs from a specific product. Subscribing to a plan allows the consuming app to invoke the APIs included in the plan given the correct security protocols are satisfied. A plan can contain terms of the usage of its APIs, including invocation rate limits and debit.

Apps
Apps are the components that are used to invoke the APIs. An app has its own set of security credentials, Client-Id and client secret.

To create an app, the following fields are required:

Field	Comment
Title	The name of the app (only visible to you and us)
QWAC Certificate	Paste the content of the public part of your eIDAS QWAC x509 Certificate in PEM format with base 64 encoding. Please provide PEM encapsulation (BEGIN CERTIFICATE, END CERTIFICATE) on separate lines with the base 64 encoded certificate on a single line between the encapsulations. This certificate will be used for both MTLS and to determine your TPP accessibility. The certificate should contain the whole certificate chain.
QSEAL Certificate (optional)	Paste the content of the public part of your eIDAS QSEAL x509 Certificate in PEM format with base 64 encoding. We currently do not validate QSEAL but will do later, by entering this field the process will be easier for you when we require QSEAL. The certificate should contain the whole certificate chain.
Application OAuth Redirect URL(s)	A list of redirect urls to be used in the OAuth Authorization code flow. These addresses will be used to receive the authorization code required when claiming the access token for the API invocation. The url: • Must follow the format (?=\w+:\/\/?*?\w+) • Cannot contain intermediate white spaces There is a timeout for how long an authorization code is valid, and that´s 60 seconds. If it takes longer than that to replace it with an access token, you will get the error message authorization code is invalid or expired.
Sign Redirect Url (optional)	Does not need to be included here since this information is not used. The redirect URL that the PISP wants to use must instead be included in the call.



***Above**: example of a correctly PEM-formatted public part of a QWAC Certificate*

After creating an app, Client-id and client secret will be provided. Take note, this is the only time the client secret will be shown for you. It is possible to reset the client secret by navigating to apps, select an app, click subscriptions, select the menu present in the top right corner of the credentials window and click reset client secret.

Step by step to create an App
1. While logged in, click the “APPS” label from the top menu
2. Click the “Create new app” button

3. Enter mandatory fields title, QWAC-certificate[1], application OAuth redirect URL(s) (minimum 1) and click the 'Submit' button
4. Take note of the API Key (Client-Id) and Secret and click the "Continue" button

Step by step to update an App
1. While logged in, click the "APPS" label from the top menu
2. Select and click the App to edit
3. Click the three vertical dots in the top right corner and select "Edit" from the drop-down menu
4. Change any of the available fields, including title, application OAuth Redirect URL(s) and Sign Redirect URL.
5. Click the 'Submit' button

Step by step to reset client secret
1. While logged in, click the "APPS" label from the top menu
2. Select and click the app to which you would like to reset client secret
3. Click the "Subscription" link located below the app name
4. Click the three vertical dots located in the top right corner of the credentials window and select "Reset Client Secret" from the drop-down menu
5. Click the "Reset" button
6. Check the 'Show' checkbox at to the top of the page
7. Take note of the new client secret

Step by step to delete an App
1. While logged in, click the "APPS" label from the top menu
2. Select and click the app to be deleted
3. Click the three vertical dots in the top right corner and select "Delete" from the drop-down menu
4. Click the "Delete" button

Subscription
A subscription defines a connection between a plan and an app. The plan defines the how and what of the subscription, including what APIs and how the APIs are accessible. The app defines the which of the subscriptions and hence details what credentials that are to be used against the gateway when attempting to invoke any of the APIs. Without a valid subscription, an app can never invoke a published API in the production environment.
To subscribe to a plan, go to the product overview page, select a product, click the plan you desire to subscribe to. Choose the App to which the subscription should be created for and confirm the subscription.

The act of requesting a new subscription will require a manual approval from us before the client app is enabled to invoke the API. It is possible to track the approval status by navigating to the app's subscriptions overview. If you know in advance that you need to get a fast approval, please contact us via mail and let us know a business day ahead.

Step by step to create a subscription
1. While logged in, click "API PRODUCTS" from the top menu
2. Select and click the product that the subscription should regard
3. Select and click the 'Subscribe' button of the desired Plan
4. Select and click the 'Select App' button of the desired app
5. Overview the subscription details and press the "Next" button
6. Click the "Done" button

Step by step to view Subscription status
1. While logged in, click the "APPS" label from the top menu
2. Select and click the app to which the subscription belongs to
3. Click the 'Subscription' link located below the app name
4. View the status of subscriptions in the subscriptions window

[1] The eIDAS certificate holds authorization/the role you have and regulates if you can access AIS and/or PIS. If you as an ASIP create an app/subscription for PIS, the subscription will be denied.

Step by step to remove a Subscription

1. While logged in, click the “APPS” label from the top menu
2. Select and click the app to which the subscription belongs to
3. Click the “Subscription” link located below the app name
4. Click the three vertical dots located to the right of the specific subscription located in the Subscriptions window
5. Select “Unsubscribe” from the drop-down menu
6. Click the “Unsubscribe” button

4.4 Update eIDAS-certificate

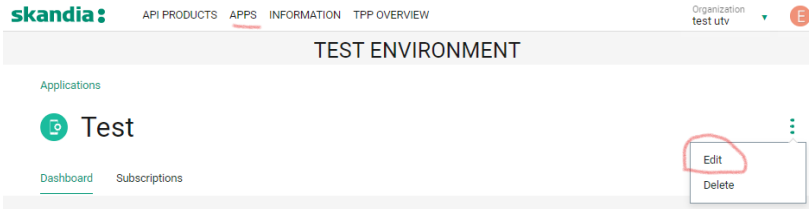
An eIDAS certificate has an expiration date and must be changed/updated before that date, otherwise the subscription will stop working. Below are instructions describing how to update an existing certificate.
An application can have multiple certificates active simultaneously. This allows you to add a new certificate before the old one expires, ensuring continuity of service.

If the new certificate is issued by a different Certificate Authority, you must contact our support before the certificate is used. In these cases, the new certificate must be manually added. Please ensure that support has completed this process before placing the certificate into production use.

Certificate updates can only be performed if the certificate roles are the same as in the existing certificate. If the roles differ, a new application must be created.

Step by step to update QWAC/Certificate

1. While logged in, click the “APPS” label from the top menu
2. Select and click the app to update
3. Click the three vertical dots
4. Select “Edit”



5. Paste the new QWAC/Certificate into the box “**New QWAC Certificate**” PEM, base-64
6. Click “Submit”

If checks are ok the new certificate will be updated and will immediately start being used going forward.
The old certificate will no longer be used and will automatically be removed when expiration date is passed.

If checks are not ok, read the message.

5 General information about API invocation

5.1 Gateway

Invoking the APIs requires three security measures:

1. Header “Client-Id” containing the client ID of your App created in the developer portal. The app holds a valid subscription containing the specific API.
2. MTLS with the eIDAS QWAC certificate provided when creating the App.
3. For AIS: An Authorization header containing a bearer access token of type reference, containing the scope demanded by the API definition.

5.2 General error messages

The APIs return errors that are common for all the available products as well as ones that are specific for the API product or method.

5.2.0 HTTP 401 Unauthorized

These are common 401 Unauthorized response messages and their implication:

- ***Invalid client id or secret:*** You are attempting to invoke the API using a Client-Id that is not valid for the specific API.
- ***Client certificates for mutual TLS in the API request doesn't match the registered certificate*:** The provided Client-Id is correct, but you are using a certificate that is not related to that Client-Id.
- ***Cannot find valid subscription for the incoming API request:*** The provided Client-Id is valid, but you are not authorized to invoke this API as that Client-Id (App) does not possess an approved subscription.
- ***Cannot pass the security checks that are required by the target API or operation, enable debug headers for more details:*** Could be one of the following three reasons.
    - Access token is not a valid reference token.
    - Access token has expired.
    - Access token does not contain the scope required by the API

5.2.1 HTTP 429 Too Many Requests

Too many requests. This error occurs when the client exceeds the allowed number of requests within a defined time window.

5.2.2 HTTP 500 Internal Server Error (Important Note on Error Handling)

When receiving an HTTP 500 Internal Server Error response from the API Gateway with the message:

```
{
   "httpCode": "500",
   "httpMessage": "URL Open error",
   "moreInformation": "Could not connect to endpoint"
}
```

This error does not indicate a server-side failure. Instead, it typically results from a timeout or network connectivity issue between the API Gateway and the downstream service.

Before retrying the operation, check the status of the payment or related process. The original request may have been completed successfully before the timeout occurred. Verifying the status helps prevent duplicate actions or inconsistencies.

5.2.3 HTTP 503 Service Unavailable

The service is unavailable. This can be caused by for example high load on the server or other unplanned issues. This error should be temporary and it is recommended to try again later.

5.3 Overview of available accounts and customers

The API:s give access to payment accounts owned by private consumers over the age of 18.

Summary of available accounts:

- “Allt i Ett-konto”. Transaction account, typically used for everyday transactions and bill payments
- “Sparkonto”. Typically, a savings account. Only domestic transfers possible, no bill payments etc.
- No fixed rate accounts are available
- The consumer must own the account
    - We have no co-owned accounts

6 AIS - Security and authentication of TPP including SCA of PSU

6.1 OAuth2

We use OAuth 2.0 for authentication and authorization. Our APIs are protected by the OAuth Authorization Code Grant type which is depicted in detail at https://oauth.net/2/.

6.2 Request Access code and ultimately Access token

To be granted access to the PSU’s accounts, the PSU needs to be authenticated by us. This is performed using a redirect or decoupled SCA approach. Swedish BankID is currently the only supported SCA.

The first step is to retrieve an authorization code. The code can be retrieved either with a redirect or decoupled approach.

- In the redirect approach the PSU is redirected to Skandia’s server and the code is returned in the redirect URI provided by you.
- In the decoupled approach the TPP calls Skandia Open Banking OAuth API and manages the user interface itself. The code is returned in the last call to the API.

When an authorization code has been retrieved, a token can be fetched from the token endpoint using the code.

The Swedish personal identity number for the person who authenticated can be found in the id token when scope openid is used.

6.2.0 Request for authorization, redirect

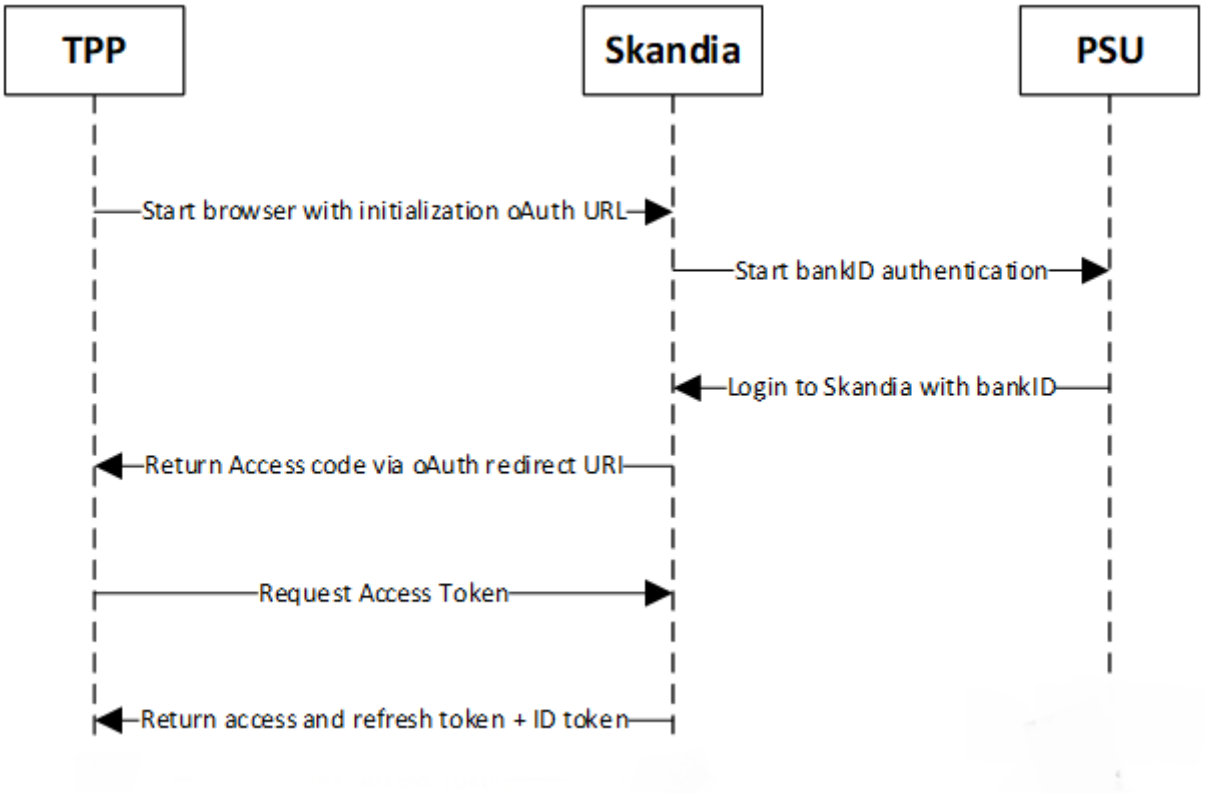


Figure 1 Overview of the authentication flow.

1. The TPP, makes a request to Skandia for the authentication of the PSU. In the request, your Client Id, a redirect URI and requested scopes must be provided.
2. The PSU identifies themselves using Swedish BankID
3. If BankID identification is successful, an access code is returned to the redirect URI provided by the TPP.
4. The TPP can now request access token using the code.

Use the client_id (denoted “Key” in Figure 2) and client_secret (denoted “Secret” in Figure 2) received when you created the app in the developer portal along with the redirect URI’s that you provided for that specific app.

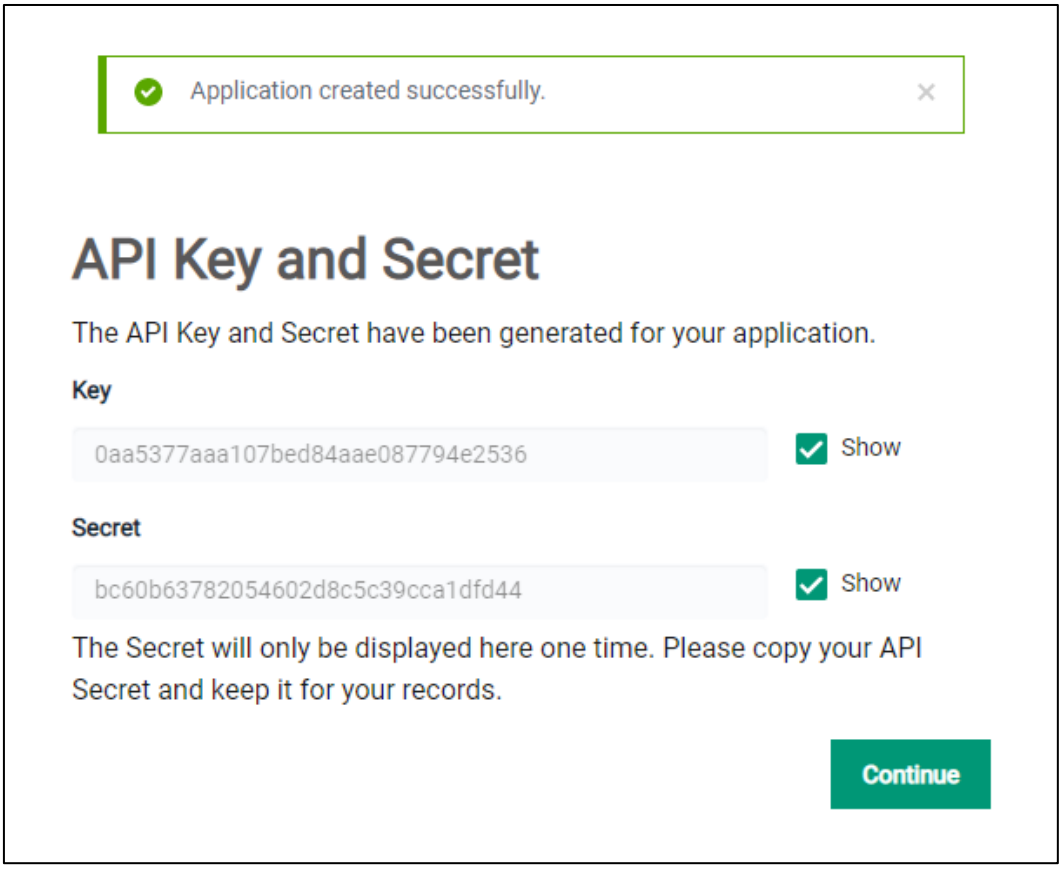


Figure 2: Example of view when an application has been created and the Key and Secret are shown.

The authorization request[2] should contain the following query parameters:

- **response_type:** code
- **client_id:** The client_id (denoted “Key” in Figure 2) of your App in the developer portal
- **redirect_uri:** The redirect URI that you have provided for the app in the portal. You cannot have a '/' as the last character in the URL, because it will be removed by the system. It must be URL encoded.
- **scope:** Should at least contain the API specific scope, i.e. psd2.aisp for AIS or psd2.pisp for PIS, and the openid scope. You must call on the format "scope=openid psd2.aisp psd2.pisp".
- **state:** A random string that should be validated to be identical in the redirect return after the user authorizes the app.
- **code_challenge:** When the App begins the authorization request, instead of immediately launching a browser, the client first creates what is known as a “*code verifier*“. This is a cryptographically random string using the characters A-Z, a-z, 0-9, and the punctuation characters -._~ (hyphen, period, underscore, and tilde), between 43 and 128 characters long.
  Once the app has generated the code verifier, it uses that to derive the *code challenge*. The code challenge is a Base64-URL-encoded string of the SHA256 hash of the code verifier.
- **code_challenge_method:** S256

[2] For production environment use: csts.skandia.se
For test environment use: csts-test.skandia.se/ct2

Example Authorization Request:

```
GET https://csts.skandia.se/prod/oauth/v2/oauth-authorize?
response_type=code
&client_id=0aa5377aaa107bed84aae087794e2536
&redirect_uri=https://localhost/
&scope=openid psd2.aisp
&state=ca17f9d039024a789493641d8cdbba14
&code_challenge=N1rZDhxSTs-WZ8-jpKOSlzxaLjFT8QWoczBSXVlItgw
&code_challenge_method=S256
```

The PSU is directed to Skandia's authentication server and prompted to identify themselves with BankID. After identification has been completed successfully, a code and state are returned to the redirect location. In this example, the PSU will be redirected to the below URL:

```
https://localhost/
?code=Im6_IkPhrd67m2gFw9upLilm9h32PmM51gUAAAAC
&state=ca17f9d039024a789493641d8cdbba14
```

6.2.1 Request for authorization, decoupled

See chapter 9, Decoupled SCA approach for AIS access.

6.2.2 Request for access token

The code received in the redirect or decoupled approach is used when requesting the access token.

Body Parameters:

- **grant_type:** authorization_code
- **code:** Access code received in the redirect from the authorization request or received in the OauthCode response when using decoupled approach.
- **redirect_uri:** Same redirect URI that was used in the authorization request.
- **client_id:** The ID of your App created in the developer portal (denoted "Key" in Figure 2).
- **client_secret:** Is received when creating the App (denoted "Secret" in Figure 2).
- **code_verifier:** The same code_verifier used to derive **code_challenge** in the authorize call

Example Access Token Request:

```
POST https://csts.skandia.se/prod/oauth/v2/oauth-token
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code
&code=661e9996602e47f7b23039a441c91061
&redirect_uri=https://localhost/
&client_id=0aa5377aaa107bed84aae087794e2536
&client_secret=bc60b63782054602d8c5c39cca1dfd44
&code_verifier= MTIzNDU2NzkwMTIzNDU2NzkwMTIzNDU2NzkwMTIzNDU2Nzkw
```

Response

```
{
    "id_token": "XXXXXX",
    "token_type": "bearer",
    "access_token": "_0XBPWQQ_dfbff42a-612e-4aa2-8a65-2730533f3eb6",
    "refresh_token": "_1XBPWQQ_0bd893af-4282-40ee-babd-13a4b024afce",
    "scope": "openid psd2.aisp",
    "expires_in": 7200
}
```

6.2.3 Token types and its uses

To access the APIs, you need to provide a valid access token in the Authorization header. The access token is a short-lived token, valid only for 2 hours. After 2 hours you will get HTTP 401 Unauthorized. When the access token has expired, you can use the refresh token to get a new access token.
The access token can be refreshed for a maximum time of 180 day from the initial time of authentication. After 180 days the PSU needs to be authenticated again as previously described.

The ID token (of type OpenID Connect) is a representation of the identity (SSN, "*personnummer*") of the authenticated PSU. This enables you as a TPP to validate that the PSU using your services corresponds to the PSU authenticated by us.

6.2.4 Refresh token

Refreshing a token is possible by using the refresh_token provided in the response to the access token request. The refresh token is only valid for one time use. When a refresh token is used to issue a new access token, the response will contain the new access token and a new refresh token.
The request should contain the following body parameters:

- **grant_type**: refresh_token
- **refresh_token:** The refresh token received from the previous access token request.
- **client_id:** The client_id of your created app in the portal.
- **client_secret:** The client_secret received when creating the app.

Example Refresh Token Request:

```
POST https://csts.skandia.se/prod/oauth/v2/oauth-token
Content-Type: application/x-www-form-urlencoded

grant_type=refresh_token
&refresh_token=661e9996602e47f7b23039a441c91061
&client_id=0aa5377aaa107bed84aae087794e2536
&client_secret=bc60b63782054602d8c5c39cca1dfd44
```

Response

```
HTTP/1.1 200 OK
{
  "token_type": "bearer",
  "access_token": "_0XBPWQQ_a50f2da8-9671-4425-82ea-216803994e04",
  "refresh_token": "_1XBPWQQ_7e675c18-2e9c-417b-883f-6e470f6b6b75",
  "scope": "openid psd2.aisp",
  "expires_in": 7200
}
```

6.2.5 AIS consent

You as a TPP are responsible of handling the consent that the customer gives you and act according to it. We do not.

After authentication by the PSU you have access to account information for 180 days without a renewed authentication. You will have access to the accounts present at the time of the SCA and any new payment accounts the PSU creates.

If the PSU is authenticated again within the 180 days, you will receive a new token valid for another 180 days. If you for some reason, for example due to some agreement with the PSU, do not want access for another 180 days, you simply throw away the new token and continue using the old one.

7 AIS 3.0.0 – Operations

7.1 Introduction

The Account Information Services API is used to retrieve account information—such as transactions and balances—for customers' payment accounts.

Prerequisites:

1. You must have a registered user account in the portal (see section 4.1 Create an account in the portal)
2. You must have an app set up in the portal with a QWAC-type eIDAS certificate that identifies you as an AISP (see section 4.3 How to set up your apps and subscriptions)

Flow for first time access for a given customer:

1. The PSU agrees to let you use your services to retrieve account information from us
2. OAuth2/SCA – The customer is informed about sharing information with the TPP
    a. The OAuth token will contain the ID of the customer
3. AISP invokes the AIS operation of choice and receives a response with the requested information

Please view the swagger file for more information.

7.2 Common Headers

All AIS requests allow the following set of headers.

Name	Type	Information
Client-Id	String	In: header **Required: true** Description: The Client Id of the calling client
X-Request-ID	string	In: header **Required: true** Format: uuid Description: ID of the request, unique to the call, as determined by the initiating party.

7.3 Get Account List

The operation will return a list of all payment accounts for a given PSU. Only the accounts which is owned by the PSU will be listed.

GET /ais/v3/accounts {query-parameters}

7.3.0 Request description

The following parameters are available for the request.

Name	Type	Information
withBalance	Boolean	In: query Required: false Default: false Description: If true, balances are included in the response.

7.3.1 Response description

Name	Type	Description
Accounts	Array of AccountDetails	The list of accounts for the given customer
AccountDetails properties		
resourceId	string	Id of an account (used to call other AIS operations)
bban	string	Bban identifier (clearing + number)
bic	string	BIC code of Skandiabanken: SKIASESS
cashAccountType	string	Always “CACC”
currency	string	The currency code of the account ISO 4217
displayName	string	Does not work at the moment, shows same as “name”
iban	string	Iban of an account
name	string	Our product name of the account type Example: • “Allt-i-Ett konto” (transaction account) • “Sparkonto” (savings account)
ownerName	string	Name of account owner
usage	string	Always “PRIV” (private personal account, natural person)
_links		linked resources
_links		
self	string	link to get account details
balances	string	link to get balances
transactions	string	link to get transactions

7.3.2 HTTP 200 Response

When the PSU has accounts available in the channel, the response body is formatted as shown below, with one item in the accounts array per available account.


```
{
  "accounts": [
    {
      "resourceId": "957054871102373",
      "bban": "91598570120",
      "bic": "SKIASESS",
      "cashAccountType": "CACC",
      "currency": "SEK",
      "displayName": "Allt-i-Ett konto",
      "iban": "SE0791500000091598570120",
      "name": "Allt-i-Ett konto",
      "ownerName": "John Doe",
      "usage": "PRIV",
      "_links": {
        "self": {
          "href": "/ais/v3/accounts/957054871102373"
        },
        "balances": {
          "href": "/ais/v3/accounts/957054871102373/balances"
        },
        "transactions": {
          "href": "/ais/v3/accounts/957054871102373/transactions"
        }
      }
    }
  ]
}
```


7.3.3 HTTP 200 Response - no accounts available in the channel

When the PSU has no payment accounts available for viewing in the Open Banking channel, the API returns HTTP 200 with an empty array.


```
{  "accounts": [] }
```


7.3.4 HTTP 404 Resource unknown

When no accounts at all are found for the PSU, the response will be the following:


```
{
   "type": "https://datatracker.ietf.org/doc/html/rfc9110#section-15.5.5",
   "title": "Not found",
   "detail": "Account not found.",
   "code": "RESOURCE_UNKNOWN "
}
```

7.4 *Get Account Details*

The operation returns detailed information for the account identified by account-id.

GET /ais/v3/accounts/{account-Id} {query-parameters}

7.4.0 Request description

The following parameters are available for the request.

Name	Type	Information
account-id	String	In: path **Required: true** Description: This identification is denoting the addressed account. The account-id is retrieved by using a "Get Account List" call. The account-id is the "resourceId" attribute of the account structure.
withBalance	Boolean	In: query Required: false Default: false Description: If true, balances are included in the response.

7.4.1 Response description

Name	Type	Description
resourceId	string	Id of an account (used to call other AIS operations)
bban	string	Bban identifier (clearing + number)
bic	string	BIC code of Skandiabanken: SKIASESS
cashAccountType	string	Always “CACC”
currency	string	The currency code of the account ISO 4217
displayName	string	Does not work at the moment, shows same as “name”
iban	string	Iban of an account
name	string	Our product name of the account type. Example: • “Allt-i-Ett konto” (transaction account) • “Sparkonto” (savings account)
ownerName	string	Name of account owner
usage	string	Always “PRIV” (private personal account, natural person)
_links		
self	string	Link to get account details
balances	string	Link to get balances
transactions	string	Link to get transactions

7.4.2 Response example


```
{
  "accounts": [    {
      "resourceId": "957054871102373",
      "bban": "91598570120",
      "bic": "SKIASESS",
      "cashAccountType": "CACC",
      "currency": "SEK",
      "displayName": "Allt-i-Ett konto",
      "iban": "SE0791500000091598570120",
      "name": "Allt-i-Ett konto",
      "ownerName": "John Doe",
      "usage": "PRIV",
      "_links": {
        "self": {
          "href": "/v3/accounts/957054871102373"
        },
        "balances": {
          "href": "/v3/accounts/957054871102373/balances"
        },
        "transactions": {
          "href": "/v3/accounts/957054871102373/transactions"
        }
      }
    }
  ]
}
```

7.5 Get Balances

The operation will return the balances for the account addressed by account-id.

GET /ais/v3/accounts/{account-id}/balances

7.5.0 Request description

The following parameters are available for the request.

Name	Type	Information
account-id	String	In: path **Required: true** Description: This identification is denoting the addressed account. The account-id is retrieved by using a "Read Account List" call. The account-id is the "resourceId" attribute of the account structure.

7.5.1 Response description

Name	Type	Description
account		
bban	String	bban of an account
iban	String	Iban of an account
currency	String	the currency code of the account ISO 4217
balances		**Array of Balance**
balanceType ”closingBooked”		booked transactions
balanceAmount		
amount		Balance amount of booked credit and debit transactions when the request was initiated
currency	String	The currency of the amount
balanceType ”interimAvailable”		booked transactions
balanceAmount		
amount		Balance amount of booked credit and debit transactions when the request was initiated. Including reserved transactions.
currency	String	The currency of the amount
creditLimitIncluded	Boolean	Always “true”. This does not say if an account has a credit or not. It only states that; if the account has a credit, it is included in the amount.
referenceDate	Date	Now

7.5.2 Response example


```
{
    "account": {
      "bban": "91598570120",
      "iban": "SE0791500000091598570120",
      "currency": "SEK"
    },
    "balances": [
      {
        "balanceAmount": {
          "amount": "-1333.26",
          "currency": "SEK"
        },
        "balanceType": "closingBooked",
         "creditLimitIncluded": true,
         "referenceDate": "2026-03-20"
      },
      {
        "balanceAmount": {
          "amount": "8566.74",
          "currency": "SEK"
        },
        "balanceType": "InterimAvailable",
        "creditLimitIncluded": true,
        "referenceDate": "2026-03-20"
      }
    ]
  }
```

7.6 Get Transaction List of an Account

The operation returns the transactions for the account identified by account-id. The list of transactions is determined by the bookingStatus parameter and may include balances. Additional optional parameters—such as dateFrom and dateTo—can also be used to refine the query.

When requesting a list of transactions, a maximum number of transactions (200 booked. 50 pending) are returned per call. If the selected time period contains more than maximum number of transactions, a **next** link is provided. This link can be used to retrieve the next batch of up to maximum number of transactions, and so on. See the response example for details.

GET /ais/v3/accounts/{account-id}/transactions {query-parameters}

7.6.0 Request description

The following parameters are available for the request.

Name	Type	Information
account-id	String	In: path **Required: true** Description: This identification is denoting the addressed account. The account-id is retrieved by using a "Read Account List" call. The account-id is the "resourceId" attribute of the account structure.
bookingStatus	String	In: query Required: false. Mandatory when entryReferenceFrom isn't used. Ignored when entryReferenceFrom is used. Description: Permitted codes are "booked", "pending" and “information”. To get both booked and pending transactions you will need to call the operation twice.
dateFrom	String (date)	In: query Required: false Description: Starting date (including the date dateFrom) of the transaction list. For pending transactions, you cannot choose a past date since there are no pending transactions on earlier dates. For booked transaction you cannot choose a future date since there are no booked transactions after the current date. We recommend always using both dateTo and dateFrom to avoid confusion. Ignored when entryReferenceFrom is used.
dateTo	String (date)	In: query Required: false Description: End date (including the date dateTo) of the transaction list. For booked transaction you cannot choose a future date since there are no booked transactions after the current date. If dateTo is on a weekend the next weekday will also be shown in the response. This is because in our systems payments made on

		weekends are not visible as booked, until the next banking day, even if funds are drawn from the account. Ignored when entryReferenceFrom is used.
entryReferenceFrom	string	In: query Required: false Description: This data attribute is indicating that the AISP is in favour to get all transactions after the transaction with identification **entryReferenceFrom** alternatively to the above defined period. If the total number of transactions exceeds the maximum number of transactions to be included in a response the response will contain a next link that should be used to request the next batch of data. The pagination token is found in the query parameter **entryReferenceFrom** and will be unique for each pagination window. If this data element is contained the entries **dateFrom** and **dateTo** are ignored. Also bookingStatus and withBalance are ignored.
withBalance	Boolean	In: query Required: false Default: false Description: If true, balances are included in the response. Ignored when entryReferenceFrom is used.

7.6.1 Response description

Information for each transaction will differ depending on payment type.

Name	Type	Description
transactions	list	list of transactions for the given account
Account		
bban	String	Bban of the account
iban	string	Iban of the account
currency	string	Currency code of the account
Transactions		
booked	list	List of booked transactions (based on input parameter)
pending	list	List of pending transactions (based on input parameter)
information	list	Information regarding recurring payments
TransactionDetails		
transactionId	string	The Id of the transaction
entryReference	string	A reference based on execution date and time
endToEndId	string	End to end identity
creditorName	String	Creditor name
TransactionAmount		
amount	string	
currency	string	
CreditorAccount		
AdditionalInformationStructured		
startDate	String	The first applicable day of execution
frequency	String	monthly
endDate	String	The last applicable day of execution

executionRule	String	This data attribute defines the behaviour when recurring payment dates falls on a weekend or bank holiday.
Parameters of pending and booked list		
bookingDate	string	Date when entry was posted to the account
valueDate	string	Date when assets become available (credit)
Cross-border Payments		
creditorAgent	string	BIC code
debtorAgent	string	BIC code
debtorAgentProprietary	string	Bank code

7.6.2 Response example


```
{
  "account": {
    "bban": "91598570120",
    "iban": "SE0791500000091598570120",
    "currency": "SEK"
  },
  "transactions": {
    "booked": [
      {
        "transactionId": "915088937100081@YGCB0169@2021-02-04@2021-02-04-19.27.40.805936",
        "entryReference": "2021-02-04-19.27.40.805936",
        "bookingDate": "2021-02-04",
        "valueDate": "2021-02-04",
        "creditorName": "John Doe",
        "transactionAmount": {
          "amount": "-200",
          "currency": "SEK"
        },
        "creditorAccount": {
          "bban": "53460034661"
        },
        "_links": {
          "transactionDetails": {
            "href": "/ais/v3/accounts/915088937100081/transactions/915088937100081@YGCB0169@2021-
02-04@2021-02-04-19.27.40.805936"
          }
        }
      }
    ],
    "_links": {
      "account": {
        "href": "/ais/v3/accounts/915088937100081"
      },
      "next": {
        "href":
"/ais/v3/accounts/915088937100081/transactions?bookingStatus=booked&entryReferenceFrom=SlCoJaCDV-
qjwFhi98h_3Q.7VZEj6ysPjYTfV3p4Ido-V3hVyjC1nbTU9gvhwbgfIvsLENGjzvCdZRSBEedrwCwrKPegHalotbs-
jXEuf5DDOnT30UlxgGcwc8fRLNgzvIaeCGlzuFuFdg3xw5lmf1KWwxbmCw1ca4AnqTdM2vpsAke-q8-
wpAwlpdAPMRtSZRUJ512erm08tZrQgLdqQ-wBXIU4fmRrJOjrNC9DG-ocQTa9XQ17uWTECctyWUcBoU"
      }
    }
  }
}
```

7.7 Get Transaction Details

Reads transaction details from a given transaction addressed by transactionId on a given account addressed by account-id.

GET /ais/v3/accounts/{account-id}/transactions/{transactionId}

7.7.0 Request description

Name	Type	Information
account-id	String	In: path **Required: true** Description: This identification is denoting the addressed account, where the transaction has been performed. The account-id is retrieved by using a "Read Account List" call. The account-id is the "resourceId" attribute of the account structure.
transactionId	String	In: path **Required: true** Description: This identification is given by the attribute transactionId of the corresponding entry of a transaction list.

7.7.1 Response description

Information for each transaction will differ depending on payment type.

Name	Type	Description
TransactionDetails		
transactionId	string	This identification is given by the attribute transaction-id of the corresponding entry of a transaction list.
entryReference	string	A reference based on execution date and time
endToEndId	string	End to end identity
creditorName	String	Creditor name
Parameters of pending and booked transaction		
bookingDate	string	Date when entry was posted to the account
valueDate	string	Date when assets become available (credit)
TransactionAmount		
amount	string	
currency	string	
CreditorAccount		
Bban	String	
remittanceInformation Unstructured	string	Remittance information regarding the transaction
remittanceInformationStructuredArray		
reference	string	Remittance information regarding the transaction
AdditionainformationStuctured -> StandingOrderDetails		
startDate	String	The first applicable day of execution
frequency	String	monthly
endDate	String	The last applicable day of execution
executionRule	String	This data attribute defines the behaviour when recurring payment dates falls on a weekend or bank holiday.
multiplicator	integer	
dayOfExecution	String	
Cross-border Payments		

creditorAgent	string	BIC code
debtorAgent	string	BIC code
debtorAgentProprietary	string	Bank code
interbankSettlementDate	string	Execution date
regulatoryReportingCode	string	Payment tax code
AmountDetails		**instructedAmount** and **transactionAmount**
instructedAmount	string	
amount	string	
sourceCurrency	string	
targetCurrency	string	
exchangeRate	string	

7.7.2 Response example – pending transfer


```
{
  "transactionId": "957054871102373",
  "entryReference": "2021-02-04-19.27.40.805936",
  "bookingDate": "2030-02-02",
  "endToEndId": "0EAD3F14-35FB-4634-87F7-C48F26DCE42",
  "creditorName": "John Doe",
  "transactionAmount": {
    "amount": "7.07",
    "currency": "SEK"
  },
  "creditorAccount": {
    "bban": "53460034661"
  },
  "_links": {
    "transactionDetails": {
      "href": "/ais/v3/accounts/915088937100081/transactions/957054871102373"
    }
  },
  "remittanceInformationStructuredArray": [
    {
      "reference": "Own Account Text",
      "referenceType": "Dpdt",
    }
  ],
  "remittanceInformationUnstructured": ["Message to creditor"]
}
```

8 PIS 4.0.0 - Operations

8.1 Introduction

The PIS API is available for you to initiate payments.

Pre-requisites:

1. Have registered a user account in the portal
2. Have an app setup in the portal with an eIDAS-certificate of QWAC-type telling us you are an PISP

Please view the swagger file for more information.

8.2 Common headers

All PIS requests allow the following set of headers.

Name	Type	Description
Client-Id	String	In: header **Required: true** Description: The Client Id of the calling client
X-Request-ID	string	In: header **Required: true** Format: uuid Description: ID of the request, unique to the call, as determined by the initiating party.
PSU-IP-Address	String	In: header **Required: true (EXCEPT for status requests)** Description: The forwarded IP Address header field consists of the corresponding HTTP request IP Address field between PSU and TPP.
PSU-IP-Port	String	In: header Required: recommended by Skandia Description: The forwarded IP Port header field consists of the corresponding HTTP request IP Port field between PSU and TPP, if available
PSU-Accept	String	In: header Required: recommended by Skandia Description: The forwarded IP Accept header fields consist of the corresponding HTTP request Accept header fields between PSU and TPP, if available
PSU-Accept-Charset	String	In: header Required: recommended by Skandia Description: The forwarded IP Accept header fields consist of the corresponding HTTP request Accept header fields between PSU and TPP, if available
PSU-Accept-Encoding	String	In: header Required: recommended by Skandia Description: The forwarded IP Accept header fields consist of the corresponding HTTP request Accept header fields between PSU and TPP, if available
PSU-Accept-Language	String	In: header Required: recommended by Skandia Description: The forwarded IP Accept header fields consist of the corresponding HTTP request Accept header fields between PSU and TPP, if available

PSU-Device-ID	String	In: header Required: recommended by Skandia Description: UUID for a device, which is used by the PSU
PSU-Geo-Location	String	In: header Required: recommended by Skandia Description: The forwarded Geo Location of the corresponding http request between PSU and TPP
PSU-Http-Method	String	In: header Required: recommended by Skandia Description: HTTP method used at the PSU TPP interface, if available. Valid values are: * GET * POST * PUT * PATCH * DELETE
PSU-User-Agent	String	In: header Required: recommended by Skandia Description: The forwarded Agent header field of the HTTP request between PSU and TPP, if available

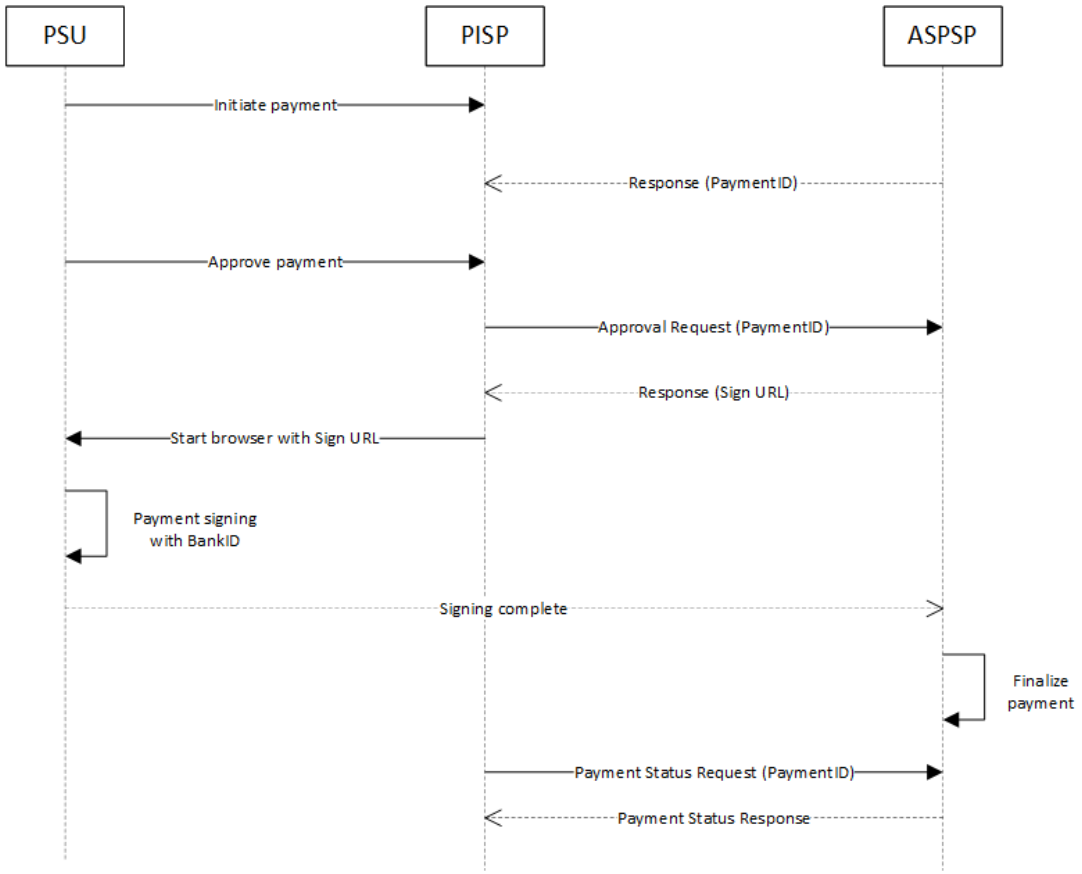
8.3

8.4 Payment Initiation flow

Skandiabanken is using a two-step Payment flow. The first step is for the TPP to Initiate the payment. The TPP can then approve and confirm the payment directly or at a later stage. (But no later than 24 hours after initiating the payment. In that case a new payment must be initiated.)

It is a single sign SCA approach, which means the customers do not need to be identified by the bank before initiating the payment. Instead, identification occurs at the same time as the signing of the payment. Skandiabanken offers both a redirect and a decoupled SCA approach. See section 9 for a description of the decoupled SCA approach.

Payment flow (with redirect SCA approach)



8.5 Payment initiation

The operation to initiate a payment is depending on the payment product in the path. We support request media type: application/json.

POST /{payment-service}/{payment-product}

8.5.0 Request description

Name	Type	Description
payment-service	String	In: path **Required: true** Description: Supported values for Payment Service is 'payments' or 'periodic-payments'
payment-product	String	In: path **Required: true** Description: Supported values for Payment Product is 'domestic-transfer', 'giro-payment' , 'cross-border-payment '

8.5.1 Request description, initiation of Domestic Transfer

Please view swagger file for details regarding formats.

Name	Type	Description
CreditorAccount		
Bban	String	Only supports Bban, not Iban. Clearing number 4-5 characters and account number 7-10 characters. When using Swedbank account, take away the fifth number in the clearingnr, the “9”.
DebtorAccount		
Bban	String	You cannot have “-“in the number.
Iban	String	Either 'Bban' or 'Iban' must be provide
EndToEndIdentification	String	The Id of the payment initiation request, maximum of 35 characters
InstructedAmount		
Amount	Number	Minimum 1 SEK. Max 6 integers and 2 decimals. Example: “123.50”
Currency	String	ISO 4217 code for the currency. Supported values: “SEK”
CreditorName	String	Required. Max 70 characters can be used.
RemittanceInformationStructuredArray		
reference	String	Max 12 characters can be used.
referenceType		Supports “DPDT”. DPDT is for reference to debtor account.
RemittanceInformationUnstructured	String	Text to creditor: Max 70 characters can be used.
requestedExecutionDate	String	Required. Max 2 years in the future.

Example Body – Domestic Transfer

```
{
  "creditorAccount": {
    "bban": "91500053920"
  },
  "debtorAccount": {
    "bban": "91500053904"
  },
  "endToEndIdentification": "<Your text(maxlength 35)>",
  "instructedAmount": {
    "amount": "10.5",
    "currency": "SEK"
  },
  "creditorName": "John Doe",
  "remittanceInformationStructuredArray": [
    {
      "reference": "<PSU TextToDebtor(maxlength 12)>",
      "referenceType": "Dpdt"
    }
  ],
  "remittanceInformationUnstructured": "<PSU TextToCreditor(maxlength 70)>",
  "requestedExecutionDate": "2024-06-15"
}
```

8.5.2 Request description, initiation of Giro Payment

Please view swagger file for details regarding formats.

Name	Type	Description
CreditorAccount		
giroNumber	String	Account number Identification represented by a GIRO number, only used in the creditor part.
giroType	String	Plusgiro or Bankgiro
DebtorAccount		Either 'Bban' or 'Iban' must be provided. Not both
Bban	String	You cannot have "-"in the number.
Iban	String	Either 'Bban' or 'Iban' must be provide
EndToEndIdentification	String	The Id of the payment initiation request, maximum of 35 characters
InstructedAmount		
Amount	Number	Minimum 1 SEK. Max 6 integers and 2 decimals. Example: "123.50"
Currency	String	ISO 4217 code for the currency. Supported values: "SEK"
RemittanceInformationStructuredArray		
reference	String	OCR number: 3-25 characters
referenceType		SCOR; Payment reference to Creditor in GIRO OCR.
RemittanceInformationUnstructured	String	Text to merchant: Max 140 characters can be used.
requestedExecutionDate	String	Required. If after 09:00, must be a future date. Max 2 years in the future.

Example Body – Giro Payment with message

```
{
  "creditorAccount": {
    "giroNumber": "235-9750",
    "giroType": "Bankgiro"
  },
  "debtorAccount": {
    "bban": "91598570120"
  },
  "endToEndIdentification": "<Your text (maxlength 35)>",
  "instructedAmount": {
    "amount": "10.5",
    "currency": "SEK"
  },
  " RemittanceInformationUnstructured ": "<Text to merchant, 140 characters>",
  "requestedExecutionDate": "2024-05-03"
}
```

Example Body – Giro Payment with OCR

```
{
  "creditorAccount": {
    "giroNumber": "901950-6",
    "giroType": "Plusgiro"
  },
  "debtorAccount": {
    "bban": "91598570120"
  },
  "endToEndIdentification": "<Your text (maxlength 35)>",
  "instructedAmount": {
    "amount": "10.5",
    "currency": "SEK"
  },
  "remittanceInformationStructuredArray": [
    {
      "reference": "<OCR number>",
      "referenceType": "Scor"
    }
  ],
  "requestedExecutionDate": "2024-05-03"
}
```

8.5.3 Request description, initiation of Cross Border Payments

Note that a cross-border payment can only be authorized individually. It cannot be included in a signing basket.

Name	Type	Description
chargeBearer	string	DEBT (debtor) or SHAR (shared)
CreditorAccount		
Iban	String	
Bban		

Country	String	Country Code
CreditorAddress		
streetName	string	Max length 70 characters
buildingNumber	string	
townName	string	
postCode	string	
country	string	
debtorAccount		
Bban	String	You cannot have “-“in the number.
Iban		Either 'Bban' or 'Iban' must be provide
EndToEndIdentification	String	The Id of the payment initiation request, maximum of 35 characters
InstructedAmount		
Amount	Number	Minimum 1 SEK. Max 6 integers and 2 decimals. Example: “123.50”
Currency	String	ISO 4217 code for the currency.
creditorAgent	string	BICFI. If no BICFI then use the field creditorAgentProprietary with Bank code
creditorAgentProprierity	string	Bank code. Max 35 characters
creditorName	string	Name of the account holder for recipient account. Max 70 characters
regulatoryReportingCode	String	Tax code
RemittanceInformationUnstructured	String	Text to merchant, max-length 140 characters.
requestedExecutionDate	String	Required. Max 6 months in the future.

Example Body


```
{
  "chargeBearer": "SHAR",
  "creditorAccount": {
    "iban": "ES9500810617620001996110",
    "country": "ES"
  },
  "creditorAddress": {
    "streetName": "Carrer de Garcilaso",
    "buildingNumber": "114",
    "townName": "Barcelona",
    "postCode": "08027",
    "country": "ES"
  },
  "debtorAccount": {
    "bban": "91598570120"
  },
  "endToEndIdentification": "<Your text (maxlength 35)>",
  "instructedAmount": {
    "amount": "10.5",
    "currency": "EUR"
  },
  "creditorAgent": "BSABESBBXXX",
  "creditorName": "Carlos Jose",
  "regulatoryReportingCode": "462",
  "remittanceInformationUnstructured": "<Text to creditor, 140 characters>",
  "requestedExecutionDate": "2024-11-20"
}
```

8.5.4 Request description, initiation of Recurring Domestic Transfer

Note that a recurring transfer can only be authorized individually. It cannot be included in a signing basket.

Name	Type	Description
CreditorAccount		
Bban	String	Only supports Bban, not Iban. Clearing number 4-5 characters and account number 7-10 characters. When using Swedbank account, take away the fifth number in the clearingnr, the “9”.
DebtorAccount		
Bban	String	You cannot have “-“in the number.
Iban	String	Either 'Bban' or 'Iban' must be provide
EndToEndIdentification	String	The Id of the payment initiation request, maximum of 35 characters
Frequency	String	Only supported frequency is “Monthly”
InstructedAmount		
Amount	Number	Minimum 1 SEK. Max 6 integers and 2 decimals. Example: “123.50”
Currency	String	ISO 4217 code for the currency. Supported values: “SEK”
CreditorName	String	Required. Max 70 characters can be used.
RemittanceInformationStructuredArray		
reference	String	Max 12 characters can be used.
referenceType		Supports “DPDT”. DPDT is for reference to debtor account.
RemittanceInformationUnstructured	String	Text to creditor: Max 70 characters can be used.
StartDate	String	Required.
EndDate	String	

Example Body


```
{
  "creditorAccount": {
    "bban": "91598570130"
  },
  "debtorAccount": {
    "bban": "91598570120"
  },
  "endToEndIdentification": "EAD3F14-35FB-4634-87F7-C48F26DCE42C",
  "instructedAmount": {
    "amount": "10.5",
    "currency": "SEK"
  },
  "creditorName": "John Doe",
  "remittanceInformationStructuredArray": [
    {
      "reference": "Reference",
      "referenceType": "Dpdt"
    }
  ],
  "remittanceInformationUnstructured": "<PSU TextToCreditor(maxlength 70)>",
  "frequency": "Monthly",
  "startDate": "2024-11-30",
  "endDate": "2025-02-28"
}
```


8.5.5 Response example

The response will have the same content regardless of the payment type.


```
{
  "paymentId": "915087014007096",
  "transactionStatus": "RCVD",
  "_links": {
    "startAuthorisation": {
      "href": "/pis/v4/payments/domestic-transfer/915087014007096/authorisations"
    },
    "self": {
      "href": "/pis/v4/payments/domestic-transfer/915087014007096"
    },
    "status": {
      "href": "/pis/v4/payments/domestic-transfer/915087014007096/status"
    }
  }
}
```

8.6 Start the authorization process for a payment

POST /{payment-service}/{payment-product}/{payment-id}/authorisations

8.6.0 Request description

Name	Type	Description
PaymentService	String	In: path **Required: true** Description: Supported values for Payment Service is 'payments' or 'periodic-payments'
PaymentProduct	String	In: path **Required: true** Description: Supported values for Payment Product is Swedish 'domestic-transfer', 'giro-payment' or 'cross-border-payment'.
paymentId	String	In: path **Required: true** Description: The id of the created transaction
TPP-Nok-Redirect-URI	String	In: header **Required if using redirect SCA approach** Format: uri. Description: URI of the TPP, where the transaction flow shall be redirected to in case of a negative result of the redirect SCA approach.
TPP-Redirect-URI	String	In: header **Required if using redirect SCA approach** Format: uri. Description: URI of the TPP, where the transaction flow shall be redirected to in case of a successful result of the redirect SCA approach.
TPP-Redirect-Preferred	boolean	In: header Description: You must choose Redirect or Decoupled.
TPP-Decoupled-Preferred	boolean	In: header Description: You must choose Redirect or Decoupled.

8.6.1 Response description

Name	Type	Description
TransactionsStatus	String	Status of the transaction.
SigningId	String	Signing ID (decoupled)
_links		
scaRedirect.href	String	The SCA link (redirect)
scaDecoupled.href	String	The SCA link (decoupled)
self.href	String	Link to get the transaction details
status.href	String	Link to get the status of the payment

8.6.2

8.6.3 Response example - redirect


```
{
  "scaStatus": "Started",
  "transactionStatus": "ACSP",
  "_links": {
    "scaRedirect": {
      "href": "https://signservice.skandia.se/openbanking?signing_id=50459996-e859-4981-aef5-52419ef0a6a1&client_id=i_openbanking_short"
    },
    "self": {
      "href": "/pis/v4/payments/domestic-transfer/915094062009002"
    },
    "status": {
      "href": "/pis/v4/payments/domestic-transfer/915094062009002/status"
    }
  }
}
```


8.6.4 Response example - decoupled


```
{
  "scaStatus": "Started",
  "transactionStatus": "ACSP",
  "signingId": "cb6804e1-98b9-4c81-aea2-2dd447e2f3a0",
  "_links": {
    "scaDecoupled": {
      "href": "/pis/v3/payments/signing/cb6804e1-98b9-4c81-aea2-2dd447e2f3a0/authorize"
    },
    "self": {
      "href": "/pis/v4/payments/domestic-transfer/915010012009096"
    },
    "status": {
      "href": "/pis/v4/payments/domestic-transfer/915010012009096/status"
    }
  }
}
```

8.7 Get Payment information
Returns the content of a payment initiation.

GET /{payment-service}/{payment-product}/{payment-id}

8.7.0 Request description

Name	Type	Description
PaymentService	String	In: path **Required: true** Description: Supported values for Payment Service is ‘payments’ or ’periodic-payments’.
PaymentProduct	String	In: path **Required: true** Description: Supported values for Payment Product is Swedish ‘domestic -transfer’, ‘giro-payment’ or ’cross-border-payment’
paymentId	String	In: path **Required: true** Description: The id of the created transaction

8.7.1 Response description
Information about a payment.

Name	Type	Description
EndToEndIdentification	String	The Id of the payment initiation request
CreditorName	String	Creditor name
DebtorAccount		
Bban	String	
CreditorAccount		
Bban	String	
InstructedAmount		
Amount	Number	
Currency	String	ISO 4217 code for the currency. Supported values: “SEK”
RemittanceInformation Unstructured	String	Remittance information regarding the transaction
remittanceInformation Structured	String	Remittance information regarding the transaction
requestedExecutionDate	String	The date when the amount is requested to be drawn from the debtor account
transactionStatus	String	The status of the payment. Can be RCVD, ACSP, ACSC, PDNG, CANC or RJCT.

8.7.2 Response example


```
{
  "creditorAccount": {
    "bban": "91598616325"
  },
  "debtorAccount": {
    "bban": "91598570120"
  },
  "endToEndIdentification": "<Your text(maxlength 35)>",
  "creditorName": "John Doe",
  "instructedAmount": {
    "amount": "10.5",
    "currency": "SEK"
  },
  "remittanceInformationUnstructured": "<PSU text(maxlength 70)>",
  "requestedExecutionDate": "2024-05-02",
  "transactionStatus": "RCVD"
}
```

8.8 Payment Cancellation

This method initiates the cancellation of a payment addressed by paymentId.

DELETE /{payment-service}/{payment-product}/{payment-id}

8.8.0 Request description

Name	Type	Description
PaymentService	String	In: path **Required: true** Description: Supported values for Payment Service is 'payments' or 'periodic-payments'
PaymentProduct	String	In: path **Required: true** Description: Supported values for Payment Product is Swedish 'domestic-transfer', 'giro-payments' or 'cross-border-payment'
paymentId	String	In: path **Required: true** Description: The id of the created transaction
TPP-Nok-Redirect-URI	String	In: header **Required if using redirect SCA approach** Format: uri. Description: URI of the TPP, where the transaction flow shall be redirected to in case of a negative result of the redirect SCA approach.
TPP-Redirect-URI	String	In: header **Required if using redirect SCA approach** Format: uri. Description: URI of the TPP, where the transaction flow shall be redirected to in case of a successful result of the redirect SCA approach.
TPP-Redirect-Preferred	boolean	In: header Description: You must choose Redirect or Decoupled.
TPP-Decoupled-Preferred	boolean	In: header Description: You must choose Redirect or Decoupled.

8.8.1 Response description

Status information about a cancellation payment initiation.

Name	Type	Description
TransactionsStatus	String	Status of the transaction. Which ever status the transaction has at the moment. Status does not change by initiating cancellation, only by signing cancellation.
SigningId	String	Signing ID (decoupled)
_links		
scaRedirect.href	String	The SCA link (redirect)
scaDecoupled.href	String	The SCA link (decoupled)
self.href	String	Link to get the transaction details
status.href	String	Link to get the status of the payment

8.8.2 Response example - redirect


```
{
  "scaStatus": "Started",
  "transactionStatus": "RCVD",
  "_links": {
    "scaRedirect": {
      "href": "https://signservice.skandia.se/openbanking?signing_id=861aa538-5871-4140-9d3b-650c1765d898&client_id=i_openbanking_short"
    },
    "self": {
      "href": "/pis/v4/payments/domestic-transfer/915044062009002"
    },
    "status": {
      "href": "/pis/v4/payments/domestic-transfer/915044062009002/status"
    }
  }
}
```


8.8.3 Response example - decoupled


```
{
  "scaStatus": "Started",
  "transactionStatus": "RCVD",
  "signingId": "e3147ff1-a779-4605-bfed-a8416fd14e06",
  "_links": {
    "scaDecoupled": {
      "href": "/pis/v3/signing/e3147ff1-a779-4605-bfed-a8416fd14e06/authorize"
    },
    "self": {
      "href": "/pis/v4/payments/domestic-transfer/915094062009001"
    },
    "status": {
      "href": "/pis/v4/payments/domestic-transfer/915094062009001/status"
    }
  }
```

8.9 Get Payment status request

Check the transaction status of a payment.

GET /{payment-service}/{payment-product}/{payment-id}/status

8.9.0 Request description

Name	Type	Description
PaymentService	String	In: path **Required: true** Description: Supported values for Payment Service is 'payments' or 'periodic-payments'
PaymentProduct	String	In: path **Required: true** Description: Supported values for Payment Product is Swedish 'domestic-transfer', 'giro-payment' or 'cross-border-payment'
paymentId	String	In: path **Required: true** Description: The id of the created transaction
PSU-IP-Address	String	In: header Required: recommended by Skandia Description: The forwarded IP Address header field consists of the corresponding HTTP request IP Address field between PSU and TPP. It shall be contained if and only if this request was actively initiated by the PSU

8.9.1 Response description

Status information about a payment initiation.

Name	Type	Description
TransactionsStatus	String	Status of the transaction. Possible values [RCVD, ACSP, ACSC, PDNG, CANC, RJCT]
ProcessingStatus	String	Status of the payment initiation state. Possible values [PENDING, PROCESSED, CANCELLED, UNPROCESSABLE, INSUFFICIENT_FUNDS, AMOUNT_LIMIT_EXCEEDED]

8.9.2 Response body example

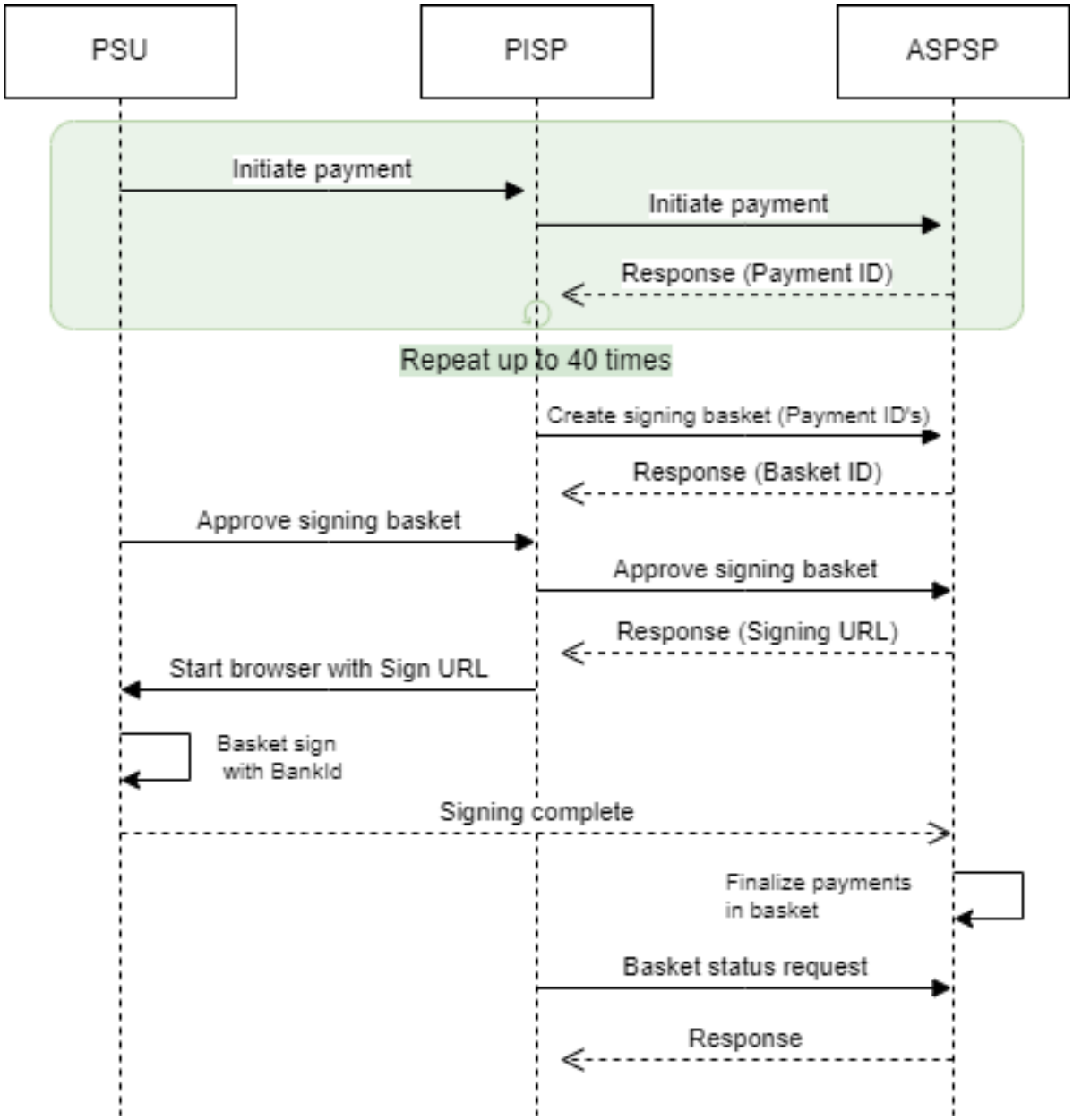

```
{
  "transactionStatus": "RCVD",
  "processingStatus": "PENDING"
}
```

8.10 Signing Basket flow

Signing baskets are a way to sign multiple payments with only one SCA. The process of a signing basket consists of first initiating payments using the methods described in 8.5 Payment initiation. Then the signing basket can be created with up to 40 payments. The created basket can then be authorized with one SCA. Upon authorization, the payments must all have the status RCVD.

As with the single payment flow, it is recommended that the PISP checks the status of the basket to verify the status of the included payments after the signing is completed.

If you wish to cancel the payments within a signing basket, each individual payment must be cancelled using the method described in section 8.8. The basket itself cannot be deleted.



8.11 Create signing basket

Operation to start up the basket for signing multiple payments. The payments within one signing basket must be created by the same PISP that is creating the basket. Also note that the payments within the signing basket must be created the same day that the signing basket is created.

Note! The creation of a signing basket does not allow the inclusion of recurring payments or foreign payments. These payment types can only be authorized individually.

POST /signing-baskets

8.11.0 Request description

```
{
  "paymentIds": [
    "915017003008080",
    "915067003008080",
    "915017003008081"
  ]
}
```

8.11.1 Response description

Name	Type	Description
BasketId	String	ID of the basket. Used for authorization and status requests.
Payments		
PaymentId	String	ID of the contained payment.
TransactionStatus	String	Current status of the payment.
_links		
StartAuthorisation.href	String	Link to start the authorization.
status.href	String	Link to get the status of the payment

8.11.2 Response example


```
{
    "basketId": "915067003008081",
    "payments": [
      {
        "paymentId": "915017003008080",
        "transactionStatus": "RCVD"
      },
      {
        "paymentId": "915067003008080",
        "transactionStatus": "RCVD"
      },
      {
        "paymentId": "915017003008081",
        "transactionStatus": "RCVD"
      }
    ],
    "_links": {
      "startAuthorisation": {
        "href": "/pis/v4/signing-baskets/915067003008081/authorisations"
      },
      "status": {
        "href": "/pis/v4/signing-baskets/915067003008081/status"
     }
   }
 }
```


8.12 Authorize signing basket

POST /signing-baskets/{basketId}/authorisations

8.12.0 Request description

Name	Type	Description
basketId	String	In: path **Required: true** Description: The id of the created basket
TPP-Nok-Redirect-URI	String	In: header **Required if using redirect SCA approach** Format: uri. Description: URI of the TPP, where the transaction flow shall be redirected to in case of a negative result of the redirect SCA approach.
TPP-Redirect-URI	String	In: header **Required if using redirect SCA approach** Format: uri. Description: URI of the TPP, where the transaction flow shall be redirected to in case of a successful result of the redirect SCA approach.

TPP-Redirect-Preferred	boolean	In: header Description: You must choose Redirect or Decoupled.
TPP-Decoupled-Preferred	boolean	In: header Description: You must choose Redirect or Decoupled.

8.12.1 Response description

Name	Type	Description
scaStatus	String	Status of the SCA process. Possible values are
SigningId	String	Signing ID (decoupled)
Payments		
paymentId	String	ID of the contained payment.
transactionStatus	String	Current status of the payment.
_links		
scaRedirect.href	String	The SCA link (redirect)
scaDecoupled.href	String	The SCA link (decoupled)
self.href	String	Link to get the transaction details
status.href	String	Link to get the status of the payment

8.12.2 Response example – redirect


```
{
    "scaStatus": "Started",
    "payments": [
      {
        "paymentId": "915017003008080",
        "transactionStatus": "ACSP"
      },
      {
        "paymentId": "915067003008080",
        "transactionStatus": "ACSP"
      },
      {
        "paymentId": "915017003008081",
        "transactionStatus": "ACSP"
      }
    ],
    "_links": {
      "scaRedirect": {
        "href": "https://signservice.skandia.se/openbanking?signing_id=86574156-0c9c-426d-b1ad-
34b0a1afe697&client_id=i_openbanking_short"
      },
      "status": {
        "href": "/pis/v4/signing-baskets/915067003008081/status"
      }
    }
  }
```

8.12.3 Response example – decoupled


```
{
    "scaStatus": "Started",
    "signingId": "a7815968-bb0c-4e26-af2b-4ea8dd0cd2c3",
    "payments": [
      {
        "paymentId": "915028013008006",
        "transactionStatus": "ACSP"
      },
      {
        "paymentId": "915078013008006",
        "transactionStatus": "ACSP"
      },
      {
        "paymentId": "915028013008007",
        "transactionStatus": "ACSP"
      }
    ],
    "_links": {
      "scaDecoupled": {
        "href": "/pis/v3/signing/a7815968-bb0c-4e26-af2b-4ea8dd0cd2c3/authorize"
      },
      "status": {
        "href": "/pis/v4/signing-baskets/915078013008007/status"
      }
    }
  }
```

8.13 Signing basket status
Method for checking the status of the payments within the specified basket.

GET /signing-baskets/{basketId}/status

8.13.0 Request description

Name	Type	Description
basketId	String	In: path **Required: true** Description: The id of the created basket

8.13.1 Response description

Name	Type	Description
BasketId	String	ID of the basket
Payments		
PaymentId	String	ID of the transaction
TransactionsStatus	String	Status of the transaction. Possible values [RCVD, ACSP, ACSC, PDNG, CANC, RJCT]
ProcessingStatus	String	Status of the payment initiation state. Possible values [PENDING, PROCESSED, CANCELLED, UNPROCESSABLE, INSUFFICIENT_FUNDS, AMOUNT_LIMIT_EXCEEDED]

8.13.2 Response example


```
{
  "basketId": "915067003008081",
  "payments": [
    {
      "paymentId": "915017003008080",
      "transactionStatus": "ACSC",
      "processingStatus": "PROCESSED"
    },
    {
      "paymentId": "915017003008081",
      "transactionStatus": "ACSC",
      "processingStatus": "PROCESSED"
    },
    {
      "paymentId": "915067003008080",
      "transactionStatus": "ACSC",
      "processingStatus": "PROCESSED"
    }
  ]
}
```


8.14 Rules and different statuses for Payments
This section will contain information that is necessary for TPPs to create a good user experience around payments. Here are answers about when payments can be made and when those payments are booked and what status a payment gets in different situations.

8.14.0 Execution rules for payments

Product	Initation Time	Execution Time
RequestedExecutionDate as current day on a business day		
Domestic-Transfer	Before 13:45	Same day (These transfers can not be cancelled.)
	After 13:45	Next business day. (These transfers can not be cancelled.)
Domestic-Transfer within Skandiabanken	Same day (regardless if business day or not). (These transfers can not be cancelled.)	
Giro-Payment	You can only choose same day as RequestedExecutionDay before 09:00. After that you must choose a later date. These transactions can be cancelled by PSU in other channels.	
Cross-border-transfers	1-2 days to EU/EES in Euros, 2-3 days to EU/EES in other currencies, 3-5 for other countries.	
RequestedExecutionDate on a non-business day		
Domestic -Transfer	As current date on a non-business day	Next business day. (These transfers cannot be cancelled.)
	On a business day	First business day after RequestedExecutionDate. (Transfers can be cancelled.)
Giro-Payment	Giro cannot be initiated with a RequestedExecutionDay that is on a non-business day	
Cross-border -transfers	Same as for business days.	

* If the recipient bank is located outside the EU / EEA, the bank cannot provide any guarantees regarding execution times for payment transactions.

8.14.1 Examples of transaction status codes during different stages of PIS operation.

Operation	Day of initiation	Requested Execution Day	Status after Initiate	Status after approve & sign	Status booked transaction
Domestic -Transfer (same day)	Monday	Monday	RCVD	ACSC	ACSC
Domestic -Transfer (weekend)	Saturday	Saturday	RCVD	ACSC	ACSC. Funds transferred on Monday.
Domestic -Transfer (future)	Monday	Wednesday	RCVD	ACSP	ACSC
Domestic -Transfer (future weekend)	Monday	Saturday	RCVD	ACSP	ACSC. Funds transferred on Monday.
Giro-Payment (before 09:00 same day)	Monday	Monday	RCVD	ACSC	ACSC
Giro-Payment (weekend)	-	-	-	-	-

Giro-Payment (future)	Monday	Wednesday	RCVD	ACSP	ACSC
Payment cancellation	----------	----------	Current status	CANC	---------

Meaning of different transaction status codes

- RCVD - Received - Payment initiation has been received by the receiving agent.
- ACSP -AcceptedSettlementInProcess - All preceding checks such as technical validation and customer profile were successful and therefore the payment initiation has been accepted for execution.
- ACSC - AcceptedSettlementCompleted - Settlement on the debtor's account has been completed.
- CANC – Cancelled – Payment has been cancelled and will not be executed.
- PNDG – Pending – A future payment did not have enough funds on the account on Requested Execution Day and a new try will be made.
- RJCT – Rejected. Payment has been rejectad.

Meaning of different processing status codes

- PENDING – Payment has been initiated but authorisation or cancellation has not been completed.
- PROCESSED – Payment has been successfully authorised.
- CANCELLED – Payment has been successfully cancelled.
- UNPROCESSABLE – Something went wrong during the payment authorisation or cancellation. For example, if the SCA validation failed.
- INSUFFICIENT_FUNDS – Payment was rejected due to insufficient funds on the debtor account.
- AMOUNT_LIMIT_EXCEEDED – Payment was rejected because the daily bank limit has been exceeded.

Please be advised that if a payment is scheduled to be executed on a future date, then the transaction may still become rejected but still have the processing status “PROCESSED”. For example, if there are insufficient funds on the debtor account at the time of execution.

9 Decoupled SCA approach for AIS access

To be granted access to the PSU's accounts, the PSU needs to be authenticated by Skandia. This is performed using either a redirect or decoupled approach. This chapter describes how to fetch an authorization code with the decoupled approach. The code can then be used to get a token. See chapter 6 for more information about AIS security, redirect SCA approach and how to fetch a token from code.

Three different authentication methods are offered. All are variants of the Swedish BankID application:

- BankID on file (could be used when TPP application is running on PC)
- Mobile BankID on same device as the TPP application (could be used when TPP app/web application is running on mobile device)
- Mobile BankID on other device than the TPP application (could be used when TPP app/web application is running on PC or mobile device)

In case Mobile BankID on other device is used, the Swedish personal identification number of the person who will authenticate must be provided. When BankID on file/same device is used the identity is taken from the BankID certificate that is used in the BankID application.

The TPP is responsible for providing a graphical user interface (GUI) to the PSU, starting the BankID application and checking authentication status until success or authentication flow is aborted. For a successful authentication flow, the PSU must not only authenticate in the BankID application, but also pass some checks such as valid KYC status.

The authentication endpoints can be found in Skandia Open Banking OAuth API. The TPP needs to apply for a subscription to get access. The API contains 5 endpoints. Request and response details are available in the developer portal, https://developer.skandia.se/open-banking/core-bank, where the TPP can also request subscription.

- GET /auth/authorize: Initiates the BankID authentication and returns available authentication methods.
- POST /auth/{identifySessionId}/idmethod: Selection of authentication method.
- GET /auth/{identifySessionId}/bankid: Get authentication status.
- POST /auth/{identifySessionId}/otp: Verify OTP.
- DELETE /auth/{identifySessionId}: Cancel authentication flow.

The response models (BankIdResponse, OneTimePasswordResponse, etc.) are specific for each endpoint but have a generic content with a lot of properties to suit the different "response types" that occur during authentication depending on the authentication state. The "id" property denotes the actual "response type". The TPP must check the "id" property to know what kind of response that has been returned and to know what to do next. For each "response type" only some of the properties in the model contains a value.

The different "response types" (value for "id" property) are:

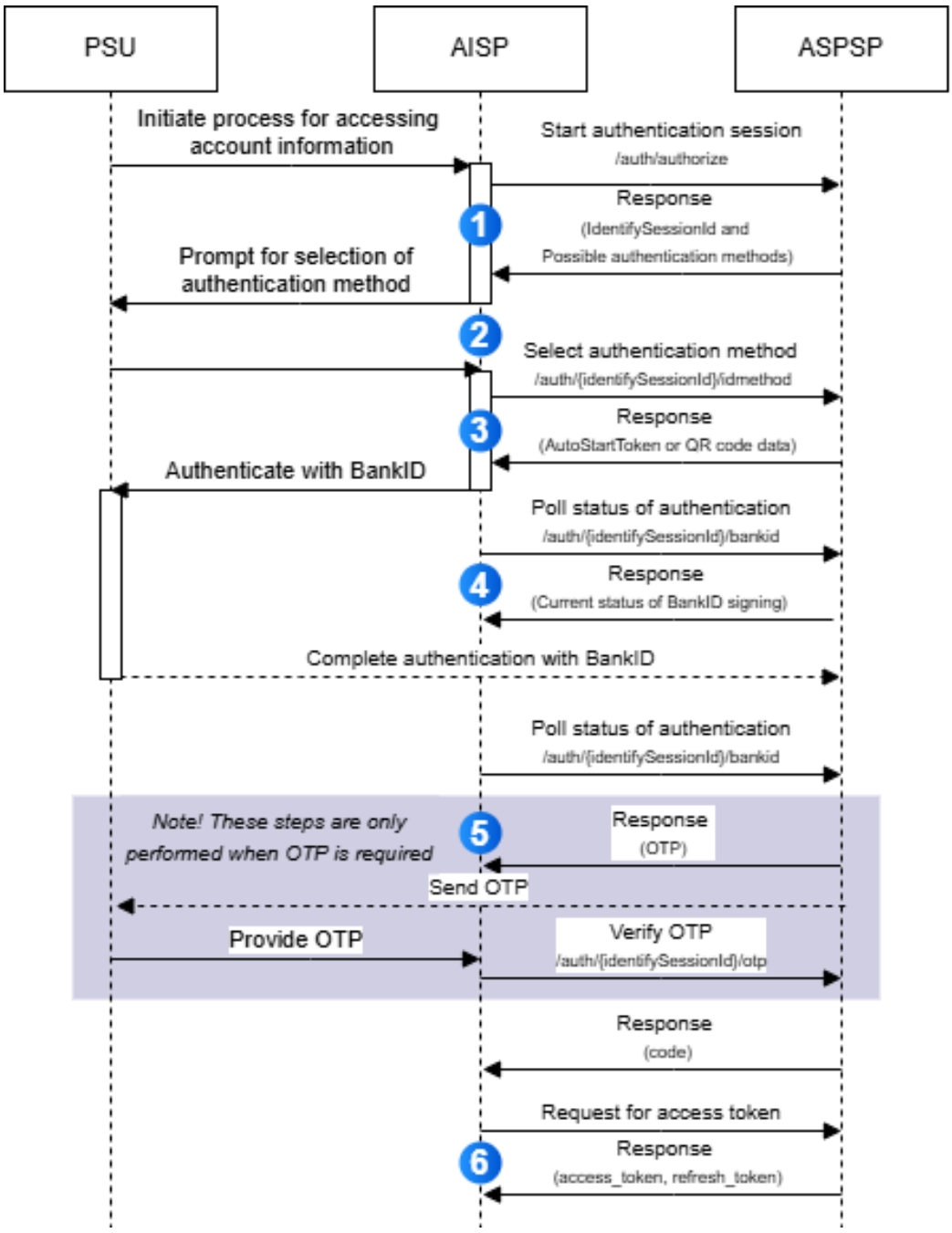
- IdMethods: A list of available authentication methods is returned.
- BankId_AutoStart: The BankID application should be started.
- BankId_QRCode: A QR code should be generated and displayed in GUI.
- BankId_Status: The BankID order is pending.
- Otp: SMS with OTP is sent to PSU and it should be able enter OTP in TPP GUI.
- OauthCode: Authentication including checks as KYC status is successful. Contains code to be used in token request.
- IdentifyAborted: Authentication flow has been aborted.

Since the same "response type" can be returned from several endpoints they are described in more detail in a separate chapter including response body JSON and with action to take.

The "response type" given in the description of endpoints below is the value for the "id" property in the response.

9.1 Decoupled authentication flow

The decoupled authentication flow is described in general terms in the diagram below. Depending on what authentication method is selected and on whether OTP is required or not, the process may differ a bit. Refer to the description of the steps for more details.



1. The TPP initiates the authentication flow with a call to the authorize endpoint. All Skandia's available authentication methods are returned.
2. TPP must analyze if the application is running on a PC or mobile device and present applicable authentication methods in GUI for PSU to select.
3. The PSU selects authentication method.

    - Mobile BankID on other device was selected. Swedish personal identification number of the PSU must be provided in the call to the idmethod when BankID on other device is selected.

1. Response contains QR code data.
2. TPP generates QR code from QR code data and displays the QR code to the PSU.
3. PSU starts the BankID app on other device and scans the QR code.
4. PSU enters the security code in the BankID app.

- BankID on file/Mobile BankID on same device selected

1. Response contains BankID application start information (autoStartToken).
2. TPP starts the BankID application according to BankID instruction.
3. PSU enters the security code in the BankID application.

4. Immediately after the QR code is shown/BankID application is started the TPP should check for authentication status every second as long as there is a pending BankID order.
   The response to the authentication status check could be of different types where different actions should be taken.

   - In the case Mobile BankID on other device and PSU hasn't yet scanned the QR code new QR code data is contained in the response and the QR code should be updated in GUI. Continue to check for authentication status.
   - In case there is a pending BankID order, the current status is returned. Show status for PSU and continue checking status.
   - In case PSU has successfully authenticated in the BankID application the full Skandia authentication flow can in some circumstances require OTP. Then continue with step 6 below, the GUI must allow PSU to enter OTP.
   - Authentication flow complete (in case all checks, as KYC status, are fulfilled as well). An OAuth authorization code is returned. Continue with a call to the token endpoint to fetch access and refresh token using the code.
   - Authentication flow aborted due to some issue (PSU cancelled, KYC not fulfilled, etc.). Final state. Show message to PSU.

5. In case of OTP
   1. TPP shows GUI where PSU can enter OTP.
   2. PSU enters OTP received in SMS.
   3. TPP sends OTP for validation.
   4. OTP validation response should be handled.
      - In case valid OTP (and other checks as KYC status fulfilled) the authentication flow is complete and an OAuth authorization code is returned. Continue with a call to the token endpoint to fetch access and refresh token using the code.
      - Invalid OTP and not too many attempts. It should be possible for PSU to retry.
      - Invalid OTP and too many attempts. Authentication flow aborted, final state. Show message to PSU.
6. The code is exchanged for an access token and a refresh token.

9.2 Decoupled authentication endpoints

Mandatory request headers for all decoupled authentication requests.

Name	Type	Description
Client-Id	String	The Client Id of the calling client Example : b77191d3ea614176ba9f6afc9805a320
X-Request-Id	String	ID of the request, unique to the call, as determined by the initiating party. Example : 99391c7e-ad88-49ec-a2ad-99ddcb1f7721
PSU-IP-Address	String	The forwarded IP Address header field of the HTTP request between PSU and TPP.

The authorize and idmethod endpoints have 4 additional required request headers (PSU-Channel and PSU-Device-ID are mandatory, all 4 are mandatory for a web application):

Name	Type	Description
PSU-Channel	String	The channel used by the PSU. If the PSU is using a web browser on desktop or mobile phone the value should be 'Web'. If the PSU is using a mobile application the value should be 'App'. Example: WEB
PSU-Device-ID	String	When running from an app: The identifier of the device your client is running on. This is used to uniquely identify the device and should be a value that is not tied to a single user of the device. Preferably, it should remain the same even if your app is reinstalled. When running from a web page: The identifier of the device running your client. Use a web cookie or the hash of one. This value should be unique to the user's browser and persist across sessions. Example: f1e3813ab36f114d4b0c2b3636617511467adb353ce8e5ae6c83500d932f2269
PSU-User-Agent	String	When running from an app: optional (not used) When running from a web page: The forwarded User-Agent header field of the HTTP request between PSU and TPP. Example: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0
PSU-Referring-Domain	String	When running from an app: optional (not used) When running from a web page: The fully qualified domain name from where the request originates from. Example: skandia.se

9.2.0 Authorize, initiate authentication and get available authentication methods
GET /auth/authorize

This is the first operation to be called in the decoupled authentication flow and it returns the authentication methods Skandiabanken offers. The response also contains an identifySessionId to be used when calling other endpoints in the API.

The authorization request requires the following query parameters:

- **responseType (required):** code
- **redirectUri (conditional):** A redirect URI that you have provided for the app in the portal. It must be URL encoded. This parameter may be omitted in case only one redirect_uri has been registered and scope openid isn't used. Otherwise it's required.
- **scope (required):** Should at least contain the API specific scope, psd2.aisp for AIS. The openid scope can be included. You must call on the format "scope=openid psd2.aisp".
- **state (optional):** A random string that should be validated to be identical in the OAuthCode response, the last response in the decoupled flow where also the OAuth authorization code is included.
- **codeChallenge (required):** The client first creates what is known as a "*code verifier*". This is a cryptographically random string using the characters A-Z, a-z, 0-9, and the punctuation characters -._~ (hyphen, period, underscore, and tilde), between 43 and 128 characters long.
  Once the app has generated the code verifier, it uses that to derive the *code challenge*. The code challenge is a Base64-URL-encoded string of the SHA256 hash of the code verifier. The code verifier will be sent in the token request later. See RFC 7636 - Proof Key for Code Exchange by OAuth Public Clients.
- **codeChallengeMethod (required):** S256

Authorization request example:

```
GET https:// apis.skandia.se/open-banking/core-bank/api.openbanking.identify/v1/auth/authorize?
responseType=code
&redirectUri=https://localhost/
&scope=openid psd2.aisp
&state=ca17f9d039024a789493641d8cdbba14
&codeChallenge=N1rZDhxSTs-WZ8-jpKOSlzxaLjFT8QWoczBSXVlItgw
&codeChallengeMethod=S256
```

Response example

```
{
  "id": "IdMethods",
  "identifySessionId": "7478a48c-35f9-4f7f-b9c9-1b2b3b9ad581",
  "availableMethods": [
    "BankIdSameDevice",
    "MobiltBankIdSameDevice",
    "MobiltBankIdOtherDevicePnr"
  ]
}
```

9.2.1 Select authentication method
POST /auth/{identifySessionId}/idmethod

The TPP must analyse if application is running on PC or mobile device and show only applicable methods for the device from the authentication methods returned from authorize request for PSU to choose. Applicable authentication methods should be shown in GUI by TPP and one should be selected by the PSU.

With this operation the authentication method is selected and a corresponding BankID order will be created. Depending on selected method, either BankID application start information or QR code data is returned.

Those are used by TPP to either start the BankID application or show a QR code.

Request examples

Example 1:

```
{
  "selectedMethod": "BankIdSameDevice"
}
```

Example 2:

```
{
  "selectedMethod": "MobiltBankIdOtherDevicePnr",
  "officialId": "199001012385"
}
```

Response types

- BankId_AutoStart
- BankId_QRCode
- IdentifyAborted

9.2.2 Get authentication status

GET /auth/{identifySessionId}/bankid

With this operation the current status of the authentication can be fetched. This operation should be called every second as long as the BankID order is pending (response type BankId_QRCode or BankId_Status).

Response types

- BankId_QRCode
- BankId_Status
- Otp
- OauthCode
- IdentifyAborted

9.2.3 Verify OTP

POST /auth/{identifySessionId}/otp

This operation verifies an OTP entered by the PSU. OTP format is 6 digits. Range 100000-999999.

Request example

```
{
  "otpCode": 123456
}
```

Response types

- Otp (otp_invalid, correct format)
- OauthCode
- IdentifyAborted

In case an OTP with invalid format or range is sent in the request, a bad request response will be returned.

Response example (400)

```
{
  "type": "https://datatracker.ietf.org/doc/html/rfc9110#section-15.5.1",
  "title": "One or more validation errors occurred",
```

```
  "detail": "OtpCode: 'Otp Code' must be greater than or equal to '100000'.",
  "code": "FORMAT_ERROR"
}
```


9.2.4 Cancel authentication flow

DELETE /auth/{identifySessionId}

This operation should be called if the PSU has cancelled the authentication in the TPP GUI.
In case a pending BankID order exists, the order will be cancelled. A new BankID order cannot be created in case there is a pending order so this operation is important to call in case PSU cancels so a new order can be created immediately.

Response type

- IdentifyAborted

9.3 Decoupled authentication response types

The response models IdMethodResponse, BankIdResponse, OneTimePasswordResponse contains an “id” property denoting the actual “response type”. This chapter describes the different response types. Since the same “response type” can be returned from several endpoints they are described in detail here instead of duplicating the information as part of each endpoint. Each type is described with example JSON, action to take by the TPP and what endpoint to call as the next step.

9.3.0 IdMethods

The response from authorize endpoint.

Response example


```
{
  "id": "IdMethods",
  "identifySessionId": "7478a48c-35f9-4f7f-b9c9-1b2b3b9ad581",
  "availableMethods": [
    "BankIdSameDevice",
    "MobiltBankIdSameDevice",
    "MobiltBankIdOtherDevicePnr"
  ]
}
```


Next step: Show appropriate authentication methods in GUI and PSU should select one. Call endpoint idmethod with the selected authentication method. If MobiltBankIdOtherDevicePnr is selected a Swedish personal identification number must be included in the call to idmethod.

9.3.1 BankId_AutoStart

The BankID application should be started with the autoStartToken given in the response. The BankID application should be started in different ways depending on if the TPP application is a native app or a web application and if executing on iOS/Android or Desktop. The TPP is responsible to find out the proper start command and possible check and handle start errors or if the BankID app not installed. Information about how to start the BankID application can be found in BankID documentation: https://developers.bankid.com/getting-started/frontend/autostart.

Response example


```
{
  "id": "BankId_AutoStart",
  "autoStartToken": "00000000-aaaa-0000-aaaa-000000000001"
}
```


Next step: Get authentication status.

9.3.2 BankId_QRCode

A QR code should be generated from qrCodeText and displayed in GUI. Some tips and recommendations can be found in BankID documentation, https://developers.bankid.com/getting-started/frontend/qr-code.

Response example


```
{
  "id": "BankId_QRCode",
  "qrCodeText": "bankid.00000000-aaaa-0000-aaaa-
000000000002.0.09e65831d6d269e50eb5e3db2ae09a35fb7d1eb98eeada3e944e5dd407770fcc"
}
```


Next step: Get authentication status (after 1 second).

9.3.3 BankId_Status

The BankID authentication order is pending with statusCode given in the response.
A status text should be displayed in GUI.
All status codes (hint codes), action and recommended status message to be displayed in GUI can be found in BankID documentation. See https://developers.bankid.com/api-references/auth--sign/collect and https://developers.bankid.com/resources/user-messages.

Responses examples


```
{
  "id": "BankId_Status",
  "statusCode": "OutstandingTransaction"
}
```

```
{
  "id": "BankId_Status",
  "statusCode": "UserSign"
}
```


Next step: Get authentication status (after 1 second).

9.3.4 Otp

An SMS with a one-time password has been sent to PSU's mobile phone number.
TPP should display a field in GUI where PSU can enter the OTP (6 digits, 100000-999999).

Response example


```
{
  "id": "Otp"
}
```


Next step: Send entered OTP for verification.

Response from OTP validation endpoint in case of incorrect OTP (valid OTP format) contains a statusCode with otp_invalid:


```
{
  "id": "Otp",
  "statusCode": "otp_invalid"
}
```


PSU should be able to enter OTP again and it should be sent to the OTP validation endpoint.

9.3.5 OauthCode

When the PSU has successfully authenticated in the BankID application and other checks, such as OTP verification and KYC status check, are fulfilled, the authentication flow is complete.

Response example


```
{
  "id": "OauthCode",
  "code": "kYmW0O0ST6w4BWQ9H2TjJqIo3JDuntfIEUkAAAAC",
  "state": "mystate"
}
```


Next step: In case state has been sent in authorize request verify it has same value.

9.3.6 Request access token

The code obtained in the preceding steps is required when requesting the access token.
For detailed instructions on retrieving the final token, refer to chapter 6.2.2 Request for access token

9.3.7 IdentifyAborted

The authentication flow has been aborted. The response is a 200/OK with a response body that contains the reason and a message (reasonDescription) in Swedish that can be shown for the PSU.
This is a final state. Calling an endpoint with an authentication in this state results in an error.

Category	Reason	Description
BankID	BankID_AlreadyInProgress	BankID authentication order could not be created due to a BankID order for this user is already in progress. See BankID documentation, https://developers.bankid.com/api-references/errors.
	BankID_ExpiredTransaction, BankID_CertificateErr, BankID_UserCancel, etc	BankID authentication failed. See BankID documentation for an exhaustive list, descriptions and action to take, https://developers.bankid.com/api-references/auth--sign/collect.
	BankID_QRTimeout	QR code wasn't scanned within time limit.
OTP	Otp_SecureMobileNumberMissing	No mobile phone number for OTP registered for PSU. PSU needs to contact Skandia's customer service.
	Otp_MaxAttemptsExceeded	Invalid OTP given too many times.
PIN	Policy_Pin_Change	PSU needs to log in to www.skandia.se and change PIN.
KYC	Kyc_NotAnswered	PSU needs to log in to www.skandia.se and answer KYC questions.
ECondition	EConditions_NotApproved	PSU needs to log in to www.skandia.se and accept conditions for E services.
Miscellaneous	Cancel	Confirmation response to successful cancel request.
	Unknown_Reason	Technical error. Contact Skandia's customer service if the error persists.

Response examples

BankID


```
{
  "id": "IdentifyAborted",
  "reason": "BankID_AlreadyInProgress",
  "reasonDescription": "En identifiering eller underskrift för det här personnumret är redan påbörjad. Försök igen.",
}
```

```
{
  "id": "IdentifyAborted",
  "reason": "BankID_CertificateErr",
  "reasonDescription": "Det BankID du försöker använda är för gammalt eller spärrat. Använd ett annat BankID eller hämta ett nytt."
}
```

```
{
  "id": "IdentifyAborted",
  "reason": "BankID_UserCancel",
  "reasonDescription": "Åtgärden avbruten."
}
```

```
{
  "id": "IdentifyAborted",
  "reason": "BankID_QRTimeout",
  "reasonDescription": "Giltighetstiden för QR-koden för att starta BankID har gått ut."
}
```


OTP


```
{
  "id": "IdentifyAborted",
  "reason": "Otp_SecureMobileNumberMissing",
  "reasonDescription": "Vi har inget mobilnummer för engångskoder till dig. Kontakta Skandias kundservice för att registrera ditt mobilnummer så att vi kan skicka SMS med engångskoder till dig."
}
```


PIN


```
{
  "id": "IdentifyAborted",
  "reason": "Policy_Pin_Change",
  "reasonDescription": "Du behöver byta din PIN-kod. Det kan du göra på https://www.skandia.se/"
}
```


Miscellaneous


```
{
  "id": "IdentifyAborted",
  "reason": "Cancel",
  "reasonDescription": "Identiferingen/signeringen avbröts."
}
```

```
{
  "id": "IdentifyAborted",
  "reason": "Unknown_Reason",
  "reasonDescription": "Ett tekniskt fel har uppstått. Kontakta Skandias kundservice om felet kvarstår."
}
```

10 Decoupled SCA approach for signing PIS operations

For PIS v.3 Skandiabanken offers both a redirect and a decoupled SCA approach. In both cases the actual signing by the PSU is done using the Swedish BankID application. Three different signing methods exist:

- BankID on file (could be used when TPP application is running on PC)
- Mobile BankID on same device as the TPP application (could be used when TPP app/web application is running on mobile device)
- Mobile BankID on other device than the TPP application (could be used when TPP app/web application is running on PC or mobile device)

The TPP selects redirect or decoupled SCA approach with the request header TPP-Redirect-Preferred or TPP-Decoupled-Preferred when calling the authorization/cancellation endpoints.

In the redirect SCA approach, a signing URL to Skandiabanken signing website is returned to the TPP and the PSU should be directed to this page by the TPP. The Skandiabanken website handles all the interactions with the PSU, starting the BankID application and completing the payment or payment cancellation after a successful signing.

In the decoupled SCA approach the TPP is responsible for providing a graphical user interface (GUI) to the PSU, starting the BankID application, checking for signing status and completing the payment or payment cancellation after a successful signing.

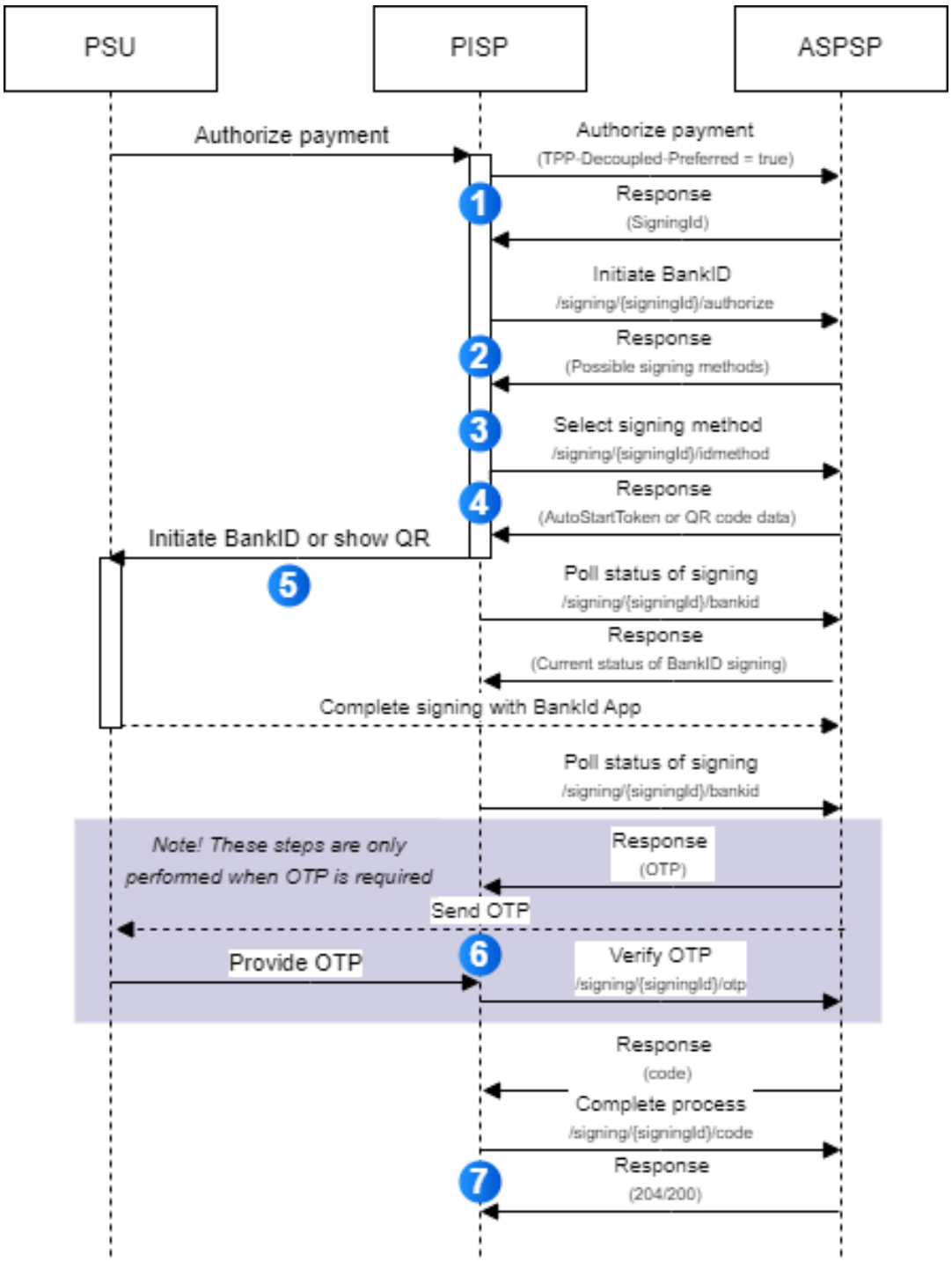
The PIS API contains 6 endpoints related to decoupled SCA approach:

- GET /signing/{signingId}/authorize: Start endpoint for decoupled flow, returns available signing methods
- POST /signing/{signingId}/idmethod: Selection of signing method
- GET /signing/{signingId}/bankid: Get signing status
- POST /signing/{signingId}/otp: Verify OTP
- DELETE /signing/{signingId}: Cancel signing flow
- PATCH /signing/{signingId}/code: Complete payment or payment cancellation after successful signing

Request and response details are available in the developer portal, https://developer.skandia.se/open-banking/core-bank. Note! The signing service is exposed as a separate product, Skandia Open Banking API - PIS Signing Services, also requiring a subscription.

10.1 The decoupled signing flow

The decoupled signing flow is described in general terms in the below diagram. Depending on what signing method is selected and on whether OTP is required or not, the process may differ a bit. Refer to the description of the steps for more details.



1. The TPP calls the authorisation/payment cancellation endpoint with request header TPP-Decoupled-Preferred set to true. The response contains the path to the first decoupled endpoint to call and a signingId to use in other decoupled requests.
2. The TPP calls the endpoint returned in authorization/payment cancellation (authorize) to get available signing methods.
3. The TPP presents applicable signing methods in GUI for PSU.

4. The PSU selects signing method.

    - Mobile BankID on other device was selected

        5. Response contains QR code data.
        6. TPP generates QR code from QR code data and displays the QR code to the PSU.
        7. PSU starts the BankID app on other device and scans the QR code.
        8. PSU enters the security code in the BankID app.

    - BankID on file/Mobile BankID on same device selected

        4. Response contains BankID application start information (autoStartToken).
        5. TPP starts the BankID application according to BankID instruction.
        6. PSU enters the security code in the BankID application.

5. Immediately after the QR code is shown/BankID application is started the TPP should check for signing status every second as long as there is a pending BankID order.
   The response to the signing status check could be of different types where different actions should be taken.

    - In the case Mobile BankID on other device and PSU hasn't yet scanned the QR code new QR code data is contained in the response and the QR code should be updated in GUI. Continue to check for signing status.
    - In case there is a pending BankID order, the current status is returned. Show status for PSU and continue check for signing status.
    - In case PSU has successfully signed in the BankID application the full signing flow can in some circumstances require OTP. Then continue with step 6 below, the GUI must allow for PSU to enter OTP.
    - Signing flow complete. Then continue with step 7 below, notify about successful signing with "code" from response and the payment or payment cancellation will be completed.
    - Signing flow aborted due to some issue, final state. Show message to PSU.

6. In case of OTP
    1. TPP shows GUI where PSU can enter OTP.
    2. PSU enters OTP received in SMS.
    3. TPP sends OTP for validation.
    4. OTP validation response should be handled.
        - In case valid OTP the signing flow is complete. Then continue with step 7 below, notify about successful signing with "code" from response and the payment or payment cancellation will be completed.
        - Invalid OTP and not too many attempts. It should be possible for PSU to retry.
        - Invalid OTP and too many attempts. Signing flow aborted, final state. Show message to PSU.

7. Notify about successful signing with a received code (PATCH signing/{signingId}/code) and payment or payment cancellation will be completed (or possible fail, for example due to insufficient funds).

Note! It's not possible to restart or rerun the signing flow. Neither when it is aborted by Skandiabanken nor when cancelled by the PSU.

10.2 Decoupled endpoints

Mandatory request headers for all decoupled requests.

Name	Type	Description
Client-Id	String	The Client Id of the calling client Example : b77191d3ea614176ba9f6afc9805a320
X-Request-Id	String	ID of the request, unique to the call, as determined by the initiating party. Example : 99391c7e-ad88-49ec-a2ad-99ddcb1f7721
PSU-IP-Address	String	The forwarded IP Address header field of the HTTP request between PSU and TPP.

The authorize and idmethod endpoints have 4 additional request headers (PSU-Channel and PSU-Device-ID are mandatory, all 4 are mandatory for a web application).

Name	Type	Description
PSU-Device-ID	String	When running from an app: The identifier of the device your client is running on. This is used to uniquely identify the device and should be a value that is not tied to a single user of the device. Preferably, it should remain the same even if your app is reinstalled. When running from a web page: The identifier of the device running your client. Use a web cookie or the hash of one. This value should be unique to the user's browser and persist across sessions. Example: f1e3813ab36f114d4b0c2b3636617511467adb353ce8e5ae6c83500d932f2269
PSU-User-Agent	String	When running from an app: optional (not used) When running from a web page: The forwarded User-Agent header field of the HTTP request between PSU and TPP. Example: Mozilla/5.0 (Windows NT 10.0; WOW64; rv:54.0) Gecko/20100101 Firefox/54.0
PSU-Referring-Domain	String	When running from an app: optional (not used) When running from a web page: The fully qualified domain name from where the request originates from. Example: skandia.se
PSU-Channel	String	The channel used by the PSU. If the PSU is using a web browser on desktop or mobile phone the value should be 'Web'. If the PSU is using a mobile application the value should be 'App'. Example: WEB

The IdMethodResponse, BankIdResponse and OneTimePasswordResponse is of a specific “response type”. The model contains the union of all properties for all kind of “response types” that can be returned from the

endpoints depending on the signing state. For each response type only some of the properties in the model contains a value. The “id” property denotes the actual response type. This means TPP must analyse the response in more detail (check the “id” property) to actually know what kind of response that has been returned and to be able to do a proper action. Since same response type can be returned from several endpoints they are described in more detail in a separate chapter including response body JSON.
The ”response type” given for endpoints below is the value for the “id” property in the response.

10.2.0 Get signing methods
GET /signing/{signingId}/authorize

This is the first operation to be called in the decoupled signing flow and it returns the signing methods Skandiabanken offers.
Applicable signing methods should be shown in GUI by TPP and one should be selected by the PSU and be sent in the “Select signing method” request.

Note! TPP must analyse if application is running on PC or mobile device and show only applicable signing methods for PSU to choose.

Response example

```
{
  "availableMethods": [
    "BankIdSameDevice",
    "MobiltBankIdSameDevice",
    "MobiltBankIdOtherDevice"
  ]
}
```

10.2.1 Select signing method
POST /signing/{signingId}/idmethod

With this operation the signing method is selected and a corresponding BankID order will be created.
Depending on selected method, either BankID application start information or QR code data is returned.
Those are used by TPP to either start the BankID application or show a QR code.

Request example

```
{
  "selectedMethod": "BankIdSameDevice"
}
```

Response types
- BankId_AutoStart
- BankId_QRCode
- IdentifyAborted

10.2.2 Get signing status
GET /signing/{signingId}/bankid

With this operation the current status of the signing can be fetched.
This operation should be called every second as long as the BankID order is pending (response type BankId_QRCode or BankId_Status).

Response types
- BankId_QRCode
- BankId_Status
- Otp
- OauthCode

- IdentifyAborted

10.2.3 Verify OTP
POST /signing/{signingId}/otp

This operation verifies an OTP entered by the PSU.
OTP format is 6 digits. Range 100000-999999.

Request example

```
{
  "otpCode": 123456
}
```

Response types
- Otp (otp_invalid, correct format)
- OauthCode
- IdentifyAborted

In case an OTP with invalid format or range is sent in the request, a bad request response will be returned.

Response example (400)

```
{
  "type": "https://datatracker.ietf.org/doc/html/rfc9110#section-15.5.1",
  "title": "One or more validation errors occurred",
  "detail": "OtpCode: 'Otp Code' must be greater than or equal to '100000'.",
  "code": "FORMAT_ERROR"
}
```

10.2.4 Cancel signing flow
DELETE /signing/{signingId}

This operation should be called if the PSU has cancelled the signing in the TPP GUI.
In case a pending BankID order exists, the order will be cancelled. A new BankID order cannot be created in case there is a pending order so this operation is important to call in case PSU cancels so a new order can be created immediately.

Response type
- IdentifyAborted

10.2.5 Complete payment/payment cancellation after signing
PATCH /signing/{signingId}/code

This operation completes the payment/payment cancellation after a successful signing.

Request example

```
{
  "code": "HhOnL_FSCjOKy6y3gsBXH05rXHtNU7s_JmwAAAAC"
}
```

In case of success there will be a 204/No Content response. In the case of signing basket, the response will be 200 OK with a body containing information about the status. It can be either Success or Partial success.

In case of failure a TppProblemDetails response model is returned.

Response example (200 Signing Basket)

```
{
  "status": "SUCCESS"
}
```

Response example (400)

```
{
  "type": "https://datatracker.ietf.org/doc/html/rfc9110#section-15.5.1",
  "title": "One or more validation errors occurred",
  "detail": "Payment could not be completed, insufficient funds",
  "code": "INSUFFICIENT_FUNDS"
}
```

10.3 Decoupled endpoint response types

The response models IdMethodResponse, BankIdResponse, OneTimePasswordResponse contains an “id” property denoting the actual “response type”. This chapter describes the different response types. Since same type can be returned from several endpoints they are described in detail here instead for each endpoint. Each type is described with example JSON, action to take by the TPP and what endpoint to call as next step.

10.3.0 BankId_AutoStart

The BankID application should be started with the autoStartToken given in the response. The BankID application should be started in different ways dependent on if the TPP application is a native app or a web application and if executing on iOS/Android or Desktop. The TPP is responsible to find out the proper start command and possible check and handle start errors or if the BankID app not installed. Information about how to start the BankID application can be found in BankID documentation: https://developers.bankid.com/getting-started/frontend/autostart.

Response example

```
{
  "id": "BankId_AutoStart",
  "autoStartToken": "00000000-aaaa-0000-aaaa-000000000001"
}
```

Next step: Get signing status.

10.3.1 BankId_QRCode

A QR code should be generated from qrCodeText and displayed in GUI. Some tips and recommendations can be found in BankID documentation, https://developers.bankid.com/getting-started/frontend/qr-code.

Response example

```
{
  "id": "BankId_QRCode",
  "qrCodeText": "bankid.00000000-aaaa-0000-aaaa-
000000000002.0.09e65831d6d269e50eb5e3db2ae09a35fb7d1eb98eeada3e944e5dd407770fcc"
}
```

Next step: Get signing status (after 1 second).

10.3.2 BankId_Status

The BankID signing order is pending with the given statusCode.
A status text should be displayed in GUI.
All status codes (hint codes), action and recommended status message to be displayed in GUI can be found in BankID documentation. See https://developers.bankid.com/api-references/auth--sign/collect and https://developers.bankid.com/resources/user-messages.

Responses examples

```
{
  "id": "BankId_Status",
  "statusCode": "OutstandingTransaction"
}
```

```
{
  "id": "BankId_Status",
  "statusCode": "UserSign"
}
```

Next step: Get signing status (after 1 second).

10.3.3 Otp

An SMS with a one-time password has been sent to PSU's mobile phone number.
TPP should display a field in GUI where PSU can enter the OTP (6 digits, 100000-999999).

Response example

```
{
  "id": "Otp"
}
```

Next step: Send entered OTP for verification.

Response from OTP validation endpoint in case of incorrect OTP (valid OTP format) contains a statusCode with otp_invalid:

```
{
  "id": "Otp",
  "statusCode": "otp_invalid"
}
```

PSU should be able to enter OTP again and it should be sent to the OTP validation endpoint.

10.3.4 OauthCode

When the PSU has successfully signed in the BankID application and other possible checks, such as OTP verification, are fulfilled, the signing is complete.

Response example

```
{
  "id": "OauthCode",
  "code": "kYmW0O0ST6w4BWQ9H2TjJqIo3JDuntfIEUkAAAAC"
}
```

Next step: Notify about successful signing (PATCH code). The code from the response should be included in the request. The payment/payment cancellation will then be completed.

10.3.5 IdentifyAborted

The signing flow has been aborted. The response is a 200/OK with a response body that contains the reason and a message (reasonDescription) in Swedish that can be shown for the PSU.
This is a final state. Calling an operation in this state results in an error.

Category	Reason	Description
BankID	BankID_AlreadyInProgress	BankID signing order could not be created due to a BankID order for this user is already in progress. See BankID documentation, https://developers.bankid.com/api-references/errors.
	BankID_ExpiredTransaction, BankID_CertificateErr, BankID_UserCancel, etc	BankID signing failed. See BankID documentation for an exhaustive list, descriptions and action to take, https://developers.bankid.com/api-references/auth--sign/collect.
	BankID_QRTimeout	QR code wasn't scanned within time limit.
OTP	Otp_SecureMobileNumberMissing	No mobile phone number for OTP registered for PSU. PSU needs to contact Skandia's customer service.
	Otp_MaxAttemptsExceeded	Invalid OTP given too many times.
PIN	Policy_Pin_Change	PSU needs to log in to www.skandia.se and change PIN.
Miscellaneous	Cancel	Confirmation response to successful cancel request.
	Unknown_Reason	Technical error. Contact Skandia's customer service if the error persists.

Response examples
BankID


```
{
  "id": "IdentifyAborted",
  "reason": "BankID_AlreadyInProgress",
  "reasonDescription": "En identifiering eller underskrift för det här personnumret är redan påbörjad. Försök igen."
}
```

```
{
  "id": "IdentifyAborted",
  "reason": "BankID_CertificateErr",
  "reasonDescription": "Det BankID du försöker använda är för gammalt eller spärrat. Använd ett annat BankID eller hämta ett nytt."
}
```

```
{
  "id": "IdentifyAborted",
  "reason": "BankID_UserCancel",
  "reasonDescription": "Åtgärden avbruten."
}
```

```
{
  "id": "IdentifyAborted",
  "reason": "BankID_QRTimeout",
  "reasonDescription": "Giltighetstiden för QR-koden för att starta BankID har gått ut."
}
```

OTP


```
{
  "id": "IdentifyAborted",
  "reason": "Otp_SecureMobileNumberMissing",
  "reasonDescription": "Vi har inget mobilnummer för engångskoder till dig. Kontakta Skandias kundservice för att registrera ditt mobilnummer så att vi kan skicka SMS med engångskoder till dig."
}
```


PIN


```
{
  "id": "IdentifyAborted",
  "reason": "Policy_Pin_Change",
  "reasonDescription": "Du behöver byta din PIN-kod. Det kan du göra på https://www.skandia.se/"
}
```


Miscellaneous


```
{
  "id": "IdentifyAborted",
  "reason": "Cancel",
  "reasonDescription": "Identiferingen/signeringen avbröts."
}
```

```
{
  "id": "IdentifyAborted",
  "reason": "Unknown_Reason",
  "reasonDescription": "Ett tekniskt fel har uppstått. Kontakta Skandias kundservice om felet kvarstår."
}
```

11 Fallback Solution

Skandia allows TPPs to make use of the fallback solution (online bank) when the special interface does not meet all the requirements or if functions are temporarily out of order or have slow response times. You do not have to wait for information from us about incidents or deficiencies, you can use the backup solution when deficiencies occur. As a TPP, you are asked to report your use of the backup solution to us. Send an e-mail to openbanking@skandia.se. The backup solution consists of the bank's customer interface via so-called screen scraping or overlay.

The fallback solution will let you continue to provide services for account information and payment initiation to customers without any interruption. You can identify yourself in the fallback solution by the usage of signed headers, using the “Signing HTTP messages”-standard that is defined in “draft-cavage-http-signatures-10” by the HTTP Networking Group (https://tools.ietf.org/html/draft-cavage-http-signatures-10).

The solution requires you to provide an extra set of http headers to the first request towards login.skandia.se. The headers include:

- **Tpp-Signature-Certificate**, your eIDAS QSEAL certificate that is to be used for signing the message. The certificate should be Base64-encoded in DER format (PEM format without the encapsulations BEGIN CERTIFICATE and END CERTIFICATE).
- **Signature**, a signature of the request as described in “draft-cavage-http-signatures-10”.
  - *keyId* is always assumed to be the certificate provided in header *Tpp-Signature-Certificate*.
  - *algorithm* is used to specify the digital signature algorithm that was used when generating the signature. Allowed algorithms include SHA-256 and SHA-512.
  - *headers* should include at least ‘(request-target)’, ‘host’ and ‘date’.
  - *signature* contains the actual signature in base64 encoding.

Example:

GET /?client_id=i_web_individual_short&redirect_uri=https://secure.skandia.se/overview/signin-Skandia&response_type=… HTTP/1.1
Host: login.skandia.se
Date: Sat, 14 Sep 2019 00:00:00 GMT
Tpp-Signature-Certificate: MIIHoDCCBoigAwIBAgIKBIlyT7rSlPwDBDANBgkqhkiG9w0BAQ….
Signature: keyId="tpp-signature-certificate",algorithm="rsa-sha256",headers="(request-target) host date", signature="Base64(RSA-SHA256(**signing_string**))"

where **signing_string** is (\n is the ASCII-value for newline):
(request-target): get
/?client_id=i_web_individual_short&redirect_uri=https://secure.skandia.se/overview/signin-Skandia&response_type=…\n
host: login.skandia.se\n
date: Sat, 14 Sep 2019 00:00:00 GMT

If you have questions about this don’t hesitate to contact us at openbanking@skandia.se.