Digital Payments
for a Trusted World

WORLDLINE

Open Banking API v3 for iDEAL – Implementation Guide

Status	Published
Author:	Worldline
Document date:	21 April 2023
Classification:	Confidential
Version:	1.4

Version history

Version no.	Version Date	Status	Edited by	Most important edit(s)
1.0	13-4-2022	Published	Joris Majoor	Initial Version
1.1	30-06-2022	Published	Joris Majoor	Moved Fast Checkout description to Chapter 2 Process reviewed comments
1.2	14-12-2022	Published	Joris Majoor Bhumika Dave	Rearranged Chapters Processed additional comments Amended pictures with better resolution
1.3	15-03-2023	Published	Joris Majoor Paula Pellegrini	Extended Security Chapter Updated Requests and Responses Updated Sequence Diagrams
1.4	21-04-2023	Published	Bhumika Dave Paula Pellegrini	Updated Requests and Responses

Table of contents

1 Introduction

This document describes the components from the Open Banking API version 3 which are used to initiate an iDEAL 2.0 payment. Some sections are based on the iDEAL 2.0 implementation guide from Currence, the iDEAL scheme holder.

1.1 Terminology

The terminology used in this document is based on the Payment Service Directive 2.

Term	Equivalent iDEAL term	Description
iDEAL HUB	iDEAL HUB	The iDEAL hub is a solution owned by Currence which provides a unified iDEAL experience. It's connected to the ASPSP's which provide the iDEAL 2.0 product.
Debtor token	iDEAL User Token	The token as provided by Currence to an Initiating Party to identify an iDEAL Profile, which is linked to an iDEAL Profile of the PSU on the iDEAL Hub and which is used to exchange the PSU's preferences (i.e. preferred IBAN) for the iDEAL Transactions with the Initiating Party.
PSU	Consumer	The Payment Service User (PSU) is account holder by one or more ASPSPs and allows other parties to initiate payments requests.
TPP	Acquirer	The Third Party Provider (TPP) is an intermediate between multiple Initiating Parties and ASPSPs and provides an interface used by the Initiating Party. The 'TPP solution' refers to the Worldline provided software (described in this document), which handles the routing of the iDEAL payments.
Initiating Party	Merchant / cPSP	The Initiating Party contracts the TPP for the iDEAL service, and sends an iDEAL payment request to the TPP Solution on behalf of a PSU.
ASPSP	Issuer	The Account Servicing Payment Service Provider (ASPSP); the Issuer bank which is responsible for the Consumer's account.

1.2 Open Banking API features for iDEAL payments

Standard iDEAL payments:

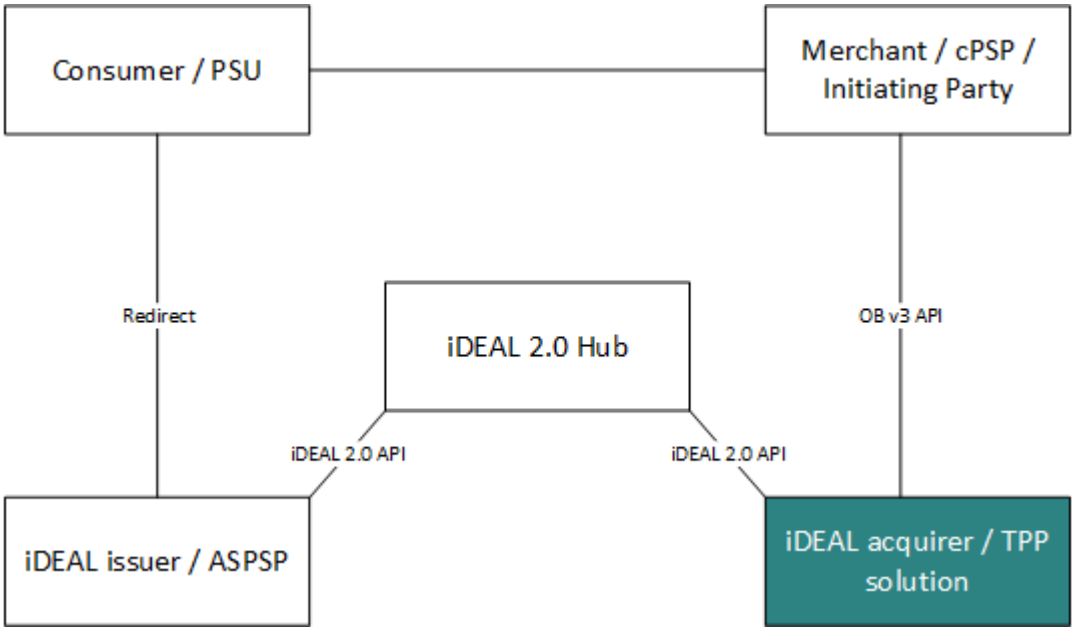
- **Payment without profile recognition** enables PSUs to pay with iDEAL without the need of a registered profile, by means of a manual bank selection or QR code, offered on an iDEAL payment page. See: sequence diagram: example of a flow without profile recognition
- **Prefill preferred bank and IBAN for iDEAL payments**: The PSU's iDEAL profile can be recognized by using the PsuId which the TPP solution can match to iDEAL debtor token, this allows the Initiating Party to immediately present the PSU with his preferred IBAN, and redirect the PSU to his preferred ASPSP (with preferred IBAN pre-selected). Upon future visits, the Initiating Party can retrieve and/or make use of the PSU's payment preferences in the iDEAL transaction. See: sequence diagram: example of a flow with profile recognition

Fast Checkout iDEAL payment:

- **Prefill address and contact details for iDEAL payments**: The iDEAL Fast Checkout function allows the PSU to share his centrally stored address and contact details with Initiating Party upon request. This means that PSUs, for example, do not have to enter their address details each time they shop at a different Initiating Party. See: sequence diagram: Fast Checkout

1.3 Ecosystem overview

The TPP Solution, marked in green, is provided by Worldline.



2 OB v3 for iDEAL - Common Elements

2.1 Timezones and Time Formats

The ISODateTime standard is used in the Open Banking v3 API.

- Datetime fields sent towards the TPP solution should have a timezone (for example: YYYY-MM-DDThh:mm:ssZ or YYYY-MM-DDThh:mm:ss+02)
- Date fields sent towards the TPP solution are always interpreted as being in UTC.

2.2 API Compatibility Policy

All APIs follow the robustness principle: "be conservative in what you do, be liberal in what you accept from others". This means the following cases MUST NOT cause an error on client or server side:

- unrecognized query parameters
- unrecognized headers
- unrecognized fields in body

2.3 Links Section

In the responses to API calls, usually a section called *Links* is included. This section contains endpoints that might be or even must be called by the Initiating Party for the next steps.

In order to inform the Initiating Party of the next request required for an authorization, the links section provides the corresponding links. If a redirection of the PSU is required, the corresponding URL for that redirection is given in the link called *RedirectUrl*.

The URLs given in the links section are provided as complete ones including the domain name and query parameters if required. An example for links with query parameters are paginated responses where the query parameter allows to address the first, next, previous or last page.

2.4 Error Codes

If a request cannot be executed successfully an error response is given. The structure of error responses is the same for all APIs. Additional to the standard HTTP Status a body is provided giving more detailed information on the error. The link is provided in the error response for future use and is currently not filled for any error response.

Mult.	Name	Comments	Type
[1..1]	Code	The Error code. The yaml file has a list of the codes that can be received	String
[1..1]	Message	The description of the error. In a human readable form.	String
[0..1]	Details	Details of the error which could be useful for a developer	String
[0..1]	Link		Group
[1..1]	+ Href	URL	String

2.5 Retries

- **Transaction initiation:** In case of receipt of a non-deterministic failure (timeout or 5xx), parties SHOULD retry the POST /payments API once.
    - When after this retry a second non-deterministic failure (timeout or 5xx) is received, this SHOULD be considered and communicated as a deterministic failure.
- **Transaction callback:** The TPP solution will attempt to perform incremental retries of a callback up to 24 hours if it does not receive a response within 8 seconds and/or receives a different response code than 200 OK, 202 ACCEPTED or 204 NO CONTENT to the initial callback. The retries follow a fixed schedule that are defined in the API specifications.

3 OB v3 for iDEAL - Security

Authentication, authorization and message integrity of requests between the Initiating Party and the TPP Solution is guaranteed through three different mechanisms.

- Transport Level Security
- Digital Signatures
- Access Tokens

3.1 Transport Level Security

On the transport level all requests to the TPP Solution, as well as all requests sent by the TPP Solution MUST be encrypted using TLS and be made over HTTPS. TLS 1.3 SHOULD be used; TLS 1.2 MAY be used. Anything below TLS 1.2 MUST NOT be used and will be refused by the TPP Solution. The TLS authentication method used is one-way, that means in requests to TPP the server authenticates itself with its certificate and in case of requests from TPP to the Initiating Party, the latter authenticates itself with its certificate.Any connection without TLS encryption, such as plain http will be refused.

3.2 Digital Signatures

3.2.1 Signing the authorization request

On the application level, the Initiating Party is authenticated and authorized by sending a digitally signed request to the Authorization API endpoint (see OB v3 for iDEAL - Authorization API). The signature validation allows to check the authenticity and integrity of the request. This is achieved by applying the "Authorization" scheme as described in https://datatracker.ietf.org/doc/html/draft-cavage-http-signatures-12 and further detailed in OB v3 for iDEAL - Authorization API.

In order to generate the string that is signed with a key, the Initiating Party must use the values of each HTTP header field in the `headers` Signature Parameter, in the order they appear in the `headers` Signature Parameter. The `headers` Signature Parameter is fixed:

```
headers=”app client id date”
```

The header field string is created by concatenating the lowercased header field name followed with an ASCII colon `:`, an ASCII space ` `, and the header field value. Leading and trailing optional whitespace (OWS) in the header field value MUST be omitted. If the value is not the last value, then an ASCII newline `\n` is appended.

Example: For the request

```
POST /authorize/token HTTP/1.1
App: IDEAL
Date: Fri, 25 Mar 2022 20:51:35 GMT
Client : idealClient
Id : 434
```

The concatenated **"String to sign"** would be:

```
app: IDEAL
client: idealClient
id: 434
date: Fri, 25 Mar 2022 20:51:35 GMT
```

The signature algorithm is fixed:

```
algorithm=”SHA256withRSA”
```

The ‘keyId’ Signature Parameter is the thumbprint of the used certificate, viewed with the SHA1 algorithm. The private key associated with `keyId` is used to generate a digital signature on the concatenated signature string applying the SHA256withRSA algorithm. The complete Authorization header looks then like this:

```
Authorization: Signature keyId=”DCAC7209573D506FC56095B8B23E8555A8F38B29”, algorithm=”SHA256withRSA”,
headers=”app client id date”,
signature=”guoLSHgl/zGRujqkDnmaWCL8kgCVnDazqkKu7nWU/uAHrS+M9eQsI8ueB4uWgxyPOnZps3vpNgkW1f4aBsdFYLS0jYeup4
yhCMN6vis2zfMKxUhZFkjELslQkit9Gwc9pqvcyH0IxUnDLbCQwkiYjf6nGbP1YNfoxVXQpfq6i6CbIXCotLfwH2kbkrnSWwAS5skZY77
+znmLDjtP3et2K94C36yPo0EEGqGkQ5xkD7owA7YxzA30xzsvkDvU3hzDzTK5wZmsgVsoyjRvMrokG0HrszUpNTwUtxflukcgs0pH7GuT
+JrIpQ55f1dpzULqxeBggnCvD9DRSuKeTakqlw==”
```

The Initiating Party must upload the used public certificate in the TPP Solution GUI so that TPP is able to validate the signature. If the signature could be validated and the sender has a valid iDEAL subscription a response containing an access token is returned. This access token can be used in all follow-up iDEAL requests until it is expired. After expiration a new access token must be requested from the Authorization API endpoint.

3.2.2 Signing requests and responses

Signing requests and responses could be enforced depending on the TPP configuration. Please check with your Acquirer whether signing is enforced or not in the following cases:

- Requests to the iDEAL Payment Initiation, Payment Status (please refer to OB v3 for iDEAL - Payment Initiation API) and Debtor Preference APIs (please refer to OB v3 for iDEAL - Debtor preference retrieval API), sent from Initiating Party to the TPP Solution
- Requests to the Notification APIs (please refer to OB v3 for iDEAL - Notification API), sent from the TPP Solution to the Initiating Party
- Responses sent back from the TPP Solution to the Initiating Party

3.2.2.1 Signing requests (iDEAL Payment Initiation, Payment Status and Debtor Preference APIs) by the Initiating Party

If signatures are enforced by the Initiating Party's Acquirer, the following headers will be mandatory in the API requests sent by the Initiating Party to the TPP solution:

- Signature
- Digest

3.2.2.1.1 The Signature header

The digital signing is done by applying the “Signature” scheme as described in https://datatracker.ietf.org/doc/html/draft-cavage-http-signatures-12. This is equivalent to the “Authorization” scheme (see Signing the authorization request) and the same procedure is followed to generate the signature and header parts but it uses the Signature header instead of the Authorization header.

In order to generate the signature string that is signed with a key, the Initiating Party must use the values of following HTTP header fields:

- Digest
- X-Request-ID
- MessageCreateDateTime

The headers used to generate the signature string also have to appear in the `headers` parameter of the Signature header, in the fixed order appearing below:

```
headers=”digest x-request-id messagecreatedatetime (request-target)”
```

To include the HTTP request target in the signature calculation, use the special `(request-target)` header field name. You can generate the header field value by concatenating the lowercased :method, an ASCII space, and the :path.

The (request-target) header field value to be used for the different APIs is:

API	(request-target) header field value
POST /payments	post /xs2a/routingservice/services/ob/pis/v3/payments
GET /payments/{paymentId}/status	get /xs2a/routingservice/services/ob/pis/v3/payments/{{paymentId}}/status*
GET /preferences/{Psuid}	get /xs2a/routingservice/services/ob/pis/v3/preferences/{{psuId}}*

**The placeholders {{paymentId}} and {{psuId}} must be substituted with the values as obtained within the respective flows.*

The signature string is created by concatenating the lowercased header field names followed with an ASCII colon `:`, an ASCII space ` `, and the header field value. Leading and trailing optional whitespace (OWS) in the header field value MUST be omitted. If the value is not the last value, then an ASCII newline `\n` is appended.

Example: For the payment request with the following required headers

```
POST /payments HTTP/1.1
digest: SHA-256=B/O1sG0L8+bEAqWF3aMZn3I0rx5YVi8r5cM6JHlTW7Q=
x-request-id: 1aad5e0f-02d7-aefb-61e3-6f4d3322cf71
messagecreatedatetime: 2023-03-15T10:07:26.264Z
```

The concatenated **"Signature String"** would be:

```
digest: SHA-256=B/O1sG0L8+bEAqWF3aMZn3I0rx5YVi8r5cM6JHlTW7Q=
x-request-id: 1aad5e0f-02d7-aefb-61e3-6f4d3322cf71
messagecreatedatetime: 2023-03-15T10:07:26.264Z
(request-target): post /xs2a/routingservice/services/ob/pis/v3/payments
```

The resulting signature parameter of the Signature header would be:

```
signature="N7kFLMMi/2R5hCd1gdO+GYhS70DOLMl+n8hborf42nFuu0HFjreoqU70gvxFWzgTPaWjdmNYY/7sOAUAQWudsM61Vc536X
maOGrrSxOINlH9l9QBk31xZMlJBf/+1+GtPb1BR26PYBjxKDMbN9W7PEVZLCDoObSnVLkvKbkLRWl0U8a39mDkUBu70Jw8yWusDU0g1OV
N+5YRfENPNtC2ZnVD80gxih4JoFV6f4WCcX4HXVl229veFNO5joNQyUc7qOkXUGN2g0omgN4iJxVGnzEJ9BCrNe+vK9T25LC0fwSp/W6A
9dDfuHQzMZgDJZZKpaX0Gg34i68etmi5oLrM3A=="
```

The algorithm parameter of the Signature header is fixed:

```
algorithm=”SHA256withRSA”
```

The ‘keyId’ parameter of the Signature header is the thumbprint of the used certificate, viewed with the SHA1 algorithm. The private key associated with `keyId` is used to generate a digital signature on the concatenated signature string applying the SHA256withRSA algorithm. The complete SIgnature header looks then like this:

```
Signature: keyId=”DCAC7209573D506FC56095B8B23E8555A8F38B29”, algorithm=”SHA256withRSA”, headers=”digest
x-request-id messagecreatedatetime (request-target)”,
signature="N7kFLMMi/2R5hCd1gdO+GYhS70DOLMl+n8hborf42nFuu0HFjreoqU70gvxFWzgTPaWjdmNYY/7sOAUAQWudsM61Vc536X
maOGrrSxOINlH9l9QBk31xZMlJBf/+1+GtPb1BR26PYBjxKDMbN9W7PEVZLCDoObSnVLkvKbkLRWl0U8a39mDkUBu70Jw8yWusDU0g1OV
N+5YRfENPNtC2ZnVD80gxih4JoFV6f4WCcX4HXVl229veFNO5joNQyUc7qOkXUGN2g0omgN4iJxVGnzEJ9BCrNe+vK9T25LC0fwSp/W6A
9dDfuHQzMZgDJZZKpaX0Gg34i68etmi5oLrM3A=="
```

The Initiating Party must upload the used public certificate in the TPP Solution GUI so that TPP is able to validate the signature. If the signature could be validated and the sender has a valid iDEAL subscription a successful response is returned.

3.2.2.1.2 The Digest header

Calculate the Digest header as follows:

- **Step1:** Create a SHA-256 hash of the Payload
    - Note-1: if the output is hex-encoded, please make sure to convert it to binary data (convert the hex-encoded string to a byte array)
    - Note-2: payload formatting is important. If you generate the Digest by using an unformatted JSON payload, then please make sure that it matches with an unformatted request body used in the API request
- **Step 2:** Base64-encode the SHA-256 hash
- **Step 3:** Prepend 'SHA-256=' to the resulting base64-encoded value

Example: For the following payload:

```
{
    "PaymentProduct": [
        "IDEAL"
    ],
    "CommonPaymentData": {
        "Amount": {
            "Type": "Fixed",
            "Amount": "10.00",
            "Currency": "EUR"
        },
        "RemittanceInformation": "Cookie",
        "RemittanceInformationStructured": {
            "Reference": "iDEALpurchase21"
        }
    },
    "IDEALPayments": {
        "UseDebtorToken": false,
        "FlowType": "Standard"
    }
}
```

Code Block 1 Request Payload

Step 1: The SHA-256 hash of this request body is: "07f3b5b06d0bf3e6c402a585dda3199f7234af1e58562f2be5c33a2479535bb4" (Note: this is a hex-encoded representation)

Step 2: The base64-encoded hash (Note: hex to base64 encoding): "B/O1sG0L8+bEAqWF3aMZn3I0rx5YVi8r5cM6JHlTW7Q="

Step 3: The Digest header value: "SHA-256=B/O1sG0L8+bEAqWF3aMZn3I0rx5YVi8r5cM6JHlTW7Q="

3.2.2.2 Signing Requests to the Initiating Party (notification requests) and Responses sent to the Initiating Party

If signatures are enforced by the Initiating Party's Acquirer, the following headers will be present in the API requests and responses sent by the TPP solution to the Initiating Party:

- Digest
- Signature

The digital signing performed by the TPP solution is done by applying the same “Signature” scheme as described above.

A notification request or response from TPP Solution to the Initiating Party may contain for example a header like:

```
Signature: keyId="2DOXXL7lNBNKJSMHKO2IBQC1", algorithm="SHA256withRSA", headers="digest x-request-id
messagecreatedatetime (request-target)",
signature="lulVhOhRwFs5G8+uGCh3BjJHBG540AyTCyaKhr9QE71YTtljxFt1b7i55C9QROw/zGt+iac3cBfGGRiTAfW4ta9Hn8+u7L
vyfYHFcdxBhNj7T6dgrv+MLG6aI6hw/3Cwmkz/OwESBrQJzISWf8/0bgYNuXnuPf5r7BGMhHdhIr+RNxocW6wkSOEVQfOGYazy7YsoJhV
EwNEt9gN++sw9HVfaxmwjl8MqxGNcLoVAgoGMcUOvNhjATA1MSzOj2cmw6kdi2yY2w7XiMuU9Tma0jQ3CGKhxIkD8Na2Vq1K2bs/n2DOx
xL7lNbnKjsmhkO2ibQc1+RV3pXQJ1SDSI3EW/w=="
```

In order to ensure that the contents of the sent messages are correct and have not been altered during transmission or storage the Initiating Party can validate the received signature. For the validation of the signature, the Initiating Party can use the public certificate of the TPP Solution which can be downloaded from the TPP Solution GUI.

3.3 Access Tokens

3.3.1 API calls towards the TPP solution

The access token as has been returned from the Authorization request endpoint can be used for subsequent iDEAL Payment Initiation and Debtor Preference requests towards the TPP solution. The default validity time of the access token is 60 minutes after which a new one must be requested from the Authorization endpoint. A refresh token does not exist for this type of request.

The access token must be put into the request Authorization header like for example:

```
Authorization:Bearer 4944daeae6c9115a10dafecbfad4a9c
```

With the access token the TPP Solution can validate and authorize the request.

3.3.2 API calls towards the Initiating Party (for notifications)

The access token used in requests from the TPP Solution towards the Initiating Party can be configured in the TPP Solution GUI, on the subscription page (the name of the field is: 'Notification BearerToken'). The token is static and does not have a validity time.

The access token will be placed in the notification request Authorization header

With the access token the Initiating Party can validate and authorize the request.

4 iDEAL requirements

The iDEAL scheme has some additional rules defined by Currence, the scheme owner.

4.1 Redirecting of PSU's

As a response to an iDEAL transaction initiation, the Initiating Party will receive a `RedirectUrl`, which is either an ASPSP URL (directing the PSU directly to the ASPSP domain) or a Payment Page URL (directing the PSU to the iDEAL Payment Page).

- If redirected from a browser, the redirect to the ASPSP URL or iDEAL Payment page MUST be performed from the browser window where the PSU selected iDEAL as payment method. The complete page of the InItiating Party shall be replaced by the complete iDEAL Payment page.
    - Initiating Parties MUST NOT open the redirect to the iDEAL Payment page or ASPSP URL in a new browser window.
    - Initiating Parties MUST NOT present the iDEAL Payment page or ASPSP URL embedded within its own page, as this disallows recognition of a cookie.
- If redirected from an Initiating Party app, the redirect MUST take place outside the app in the default browser of the PSU.

Initiating Parties MUST NOT redirect PSU's in a custom made in-app webview browser. Doing so will disallow for PSU's to be redirected to their mobile banking apps and will seriously breach privacy regulations.

Exceptions to the above are the use of SafariViewController for Apple iOS and Chrome Custom Tabs for Android. However be aware that these may not correspond to the PSU's default browsers, so that PSU recognition based on cookie might not work.

- During the migration period to the updated iDEAL specifications, the Initiating Party is still allowed to share a preferred ASPSP (for example when the Initiating Party also stores this preference). This feature will be discontinued after the migration period ends.

4.2 Retrieving the Status of the transaction

- If the Initiating Party has not yet received a confirmation of the payment status when the PSU opens the returnURL, the Initiating Party MUST retrieve the transaction status by calling the status API of the TPP Solution.
- The Initiating Party MUST inform the PSU of a success status of the transaction, so that the PSU is certain that the payment status was correctly received by the Initiating Party.
- The Initiating Party MUST NOT poll the status and MUST adhere to the retry scheme.

4.3 Presentation of Standard iDEAL payment on Initiating Party environment

There are some rules regarding the presentation of iDEAL on the Initiating Party's environment. The main purpose of these is to create a uniform user experience for PSU's whenever and wherever they pay with iDEAL.

- The iDEAL payment option must be presented in the list of payment options in such a way that it receives at least the same amount of attention as other payment options.
- The rules on presentation the iDEAL Payment button and the iDEAL logo can be found here: http://www.ideal.nl/en/payee/logos-banners/ .
    - In cases where the iDEAL logo cannot be displayed at the required size on a mobile device the mandatory free space of 15px around the logo may be reduced to accommodate the requirements of the mobile device.

4.4 Presentation of Fast Checkout iDEAL payment on Initiating Party environment

- To indicate to PSUs that iDEAL Checkout is offered the Initiating Party MUST show the iDEAL Checkout logo. Logo's can be found here: https://www.ideal.nl/bedrijven/logos/ . This logo may be accompanied by the term 'Snel Bestellen' to provide additional clarity on the flow the PSU will enter after selecting this option;
- Before the PSU selects iDEAL Checkout, the applicable shipping costs MUST be communicated to the PSU, to prevent that the PSU is confronted with a higher amount in the iDEAL screens than expected;
- The Initiating Party MUST confirm the datafields, received from Currence, to the PSU, preferably on the order confirmation screen.

4.5 Profile recognition via Debtor Tokens

For an enhanced user flow for returning PSU's, the Initiating Party may want to make use of Debtor Tokens. The Debtor Tokens allow the Initiating Party to present the PSU's preferred IBAN within its Initiating Party Domain, and prevent a redirect to the iDEAL Payment Page.

If an Initiating Party wishes to make use of Debtor Tokens for its PSU's, the following applies:

- The PSU MUST hold an account with the Initiating Party;
- To retrieve the preferred ASPSP and (masked) IBAN of the PSU, the Initiating Party MUST provide a Debtor Token that uniquely identifies the PSU at the Initiating Party;
- The Initiating Party MUST be able to recognize the PSU on a return visit;
- The Initiating Party MUST retrieve and display the preferred ASPSP and masked IBAN of the PSU together with the iDEAL payment button;
- The Initiating Party MUST only use Debtor Token that was received in the last transaction for that PSU

Please note that a Debtor Token can only be linked to one iDEAL profile.

5 OB v3 for iDEAL - Transaction Flow

In this chapter the transaction flows of two iDEAL payment types are described:

- Standard iDEAL payment
  - Without profile recognition
  - With profile recognition (based on debtor token)
- Fast Checkout iDEAL payment (based on browser cookie)

Additional requirements for the iDEAL payments are described in the chapter iDEAL requirements

5.1 Standard iDEAL Payment

An iDEAL transaction for a simple single payment is the basis of the iDEAL flow. The actual iDEAL transaction flow that the PSU follows, depends on where the iDEAL transaction is started at the Initiating Party (mobile app or browser, desktop browser) and where the payment authorization takes place at the ASPSP (mobile banking app, desktop browser, mobile browser). Next to this, it also depends on whether a PSU is already registered for iDEAL, and whether a registered PSU is recognized by the iDEAL Hub via a Debtor Token.

The full iDEAL transaction flow broadly follows the following steps:

1. **Shopping Process:** The PSU places an order at an Initiating Party and chooses iDEAL as payment method. If the Initiating Party makes use of Debtor Tokens, the Initiating Party shows the preferred ASPSP and IBAN of the recognized PSU alongside the iDEAL payment button.
2. **Transaction Initiation:** A transaction is created at the TPP Solution using the Post /payments API, which responds with a redirect URL to the iDEAL payment page or the ASPSP directly.
3. **Customer Recognition / ASPSP Selection:** The PSU is redirected by the Initiating Party (or CPSP) to the iDEAL payment page, where he can choose an ASPSP or scan a QR code (depending on the web/app context). Alternatively, a recognized PSU (by means of browser cookie or debtor token) may be redirected directly to his preferred ASPSP or receive a push notification from his preferred ASPSP to authorize the iDEAL transaction (decoupled flow).
4. **ASPSP Authorization:** The PSU is presented with the transaction details at the ASPSP, where he authorizes the iDEAL transaction.
5. **Transaction Confirmation & Return to Initiating Party:** When the PSU authorizes the payment, a confirmation is sent by the ASPSP to the iDEAL Hub. If applicable, the PSU may be asked to register a profile with iDEAL. The iDEAL Hub informs the TPP Solution of the payment status, who in turn informs the Initiating Party. In case the Initiating Party implemented the Post status API, they are informed of the payment status directly by the TPP Solution. In case of a redirect flow, the PSU is ultimately redirected back to the Initiating Party.

Below are two example screen flows. The top one is using a mobile device throughout. The bottom one starts on a non-mobile device and switches to a mobile device for the ASPSP Authorization.

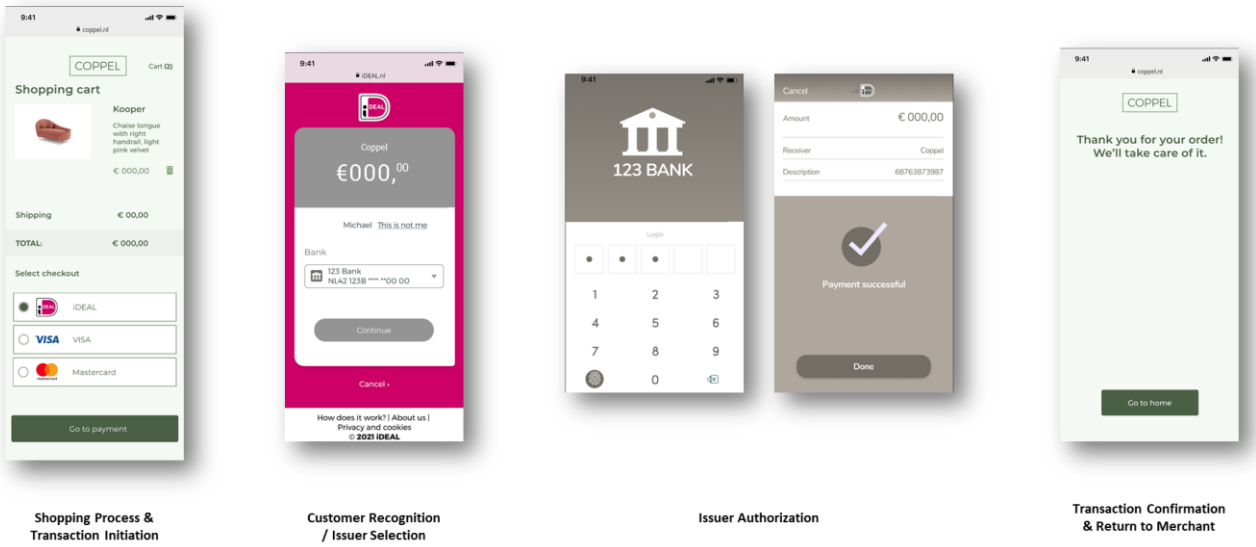


Figure 1 iDEAL transaction flow example on mobile

Figure 2 iDEAL transaction flow starts on non-mobile device (example)

For the initiation of an iDEAL payment two authentication flows are applicable; described in the table below.

Secure Customer Authentication (SCA) Approach	Indicator in the Post /payments response	Description
Redirect	**RedirectUrl**	In the Redirect Approach the browser session of the PSU is redirected from the Initiating Party to the iDEAL Hub. The iDEAL Hub will ask the PSU which ASPSP he would like to use (this preference can be stored). The browser session is then redirected to the chosen ASPSP. The ASPSP provides all the pages required for authentication of the PSU and will then ask for authorization of the payment. After that the PSU is redirected back to the Initiating Party.

Decoupled	**UseWaitingScreen** = true
	AND
	RedirectUrl
		The decoupled flow can only occur if the '`HttpHeaderUserAgent`', '`PsuId`' and '`UseDebtorToken = true`' fields are provided in the Post /payments request. The Aspsp could then decide to use the decoupled flow, in that case the Post /payments response will have '`UseWaitingScreen = true`' field. `UseWaitingScreen` is a boolean which is returned in the Post /payments response. If set to true it means the Initiating Party should display a waiting screen while the Aspsp is starting a decoupled flow with the PSU. The RedirectUrl is provided as backup. Currence set the following requirements for the waiting screen:
		• There is a <Back> button, which leads the PSU back to the previous page of the Initiating Party;
		  ○ The Initiating Party MUST make a call to retrieve the status of iDEAL payment;
		  ○ When there is no final status available, the Initiating Party MUST inform the PSU that the payment is not finalised;
		  ○ When a final status is available, the Initiating Party MUST inform the PSU about that status (payment successful, payment failed, etc.) and follow up with the applicable action;
		• There is a <Didn't receive notification> button, which leads the PSU to the iDEAL Payment Page RedirectUrl to finalise the payment;
		• A message is displayed which informs the PSU about the action that is required, for example: 'Tap the notification on your phone to pay with the following bank';
		  ○ The ASPSP name and logo are displayed;
		This page can be in the look and feel of the Initiating Party.

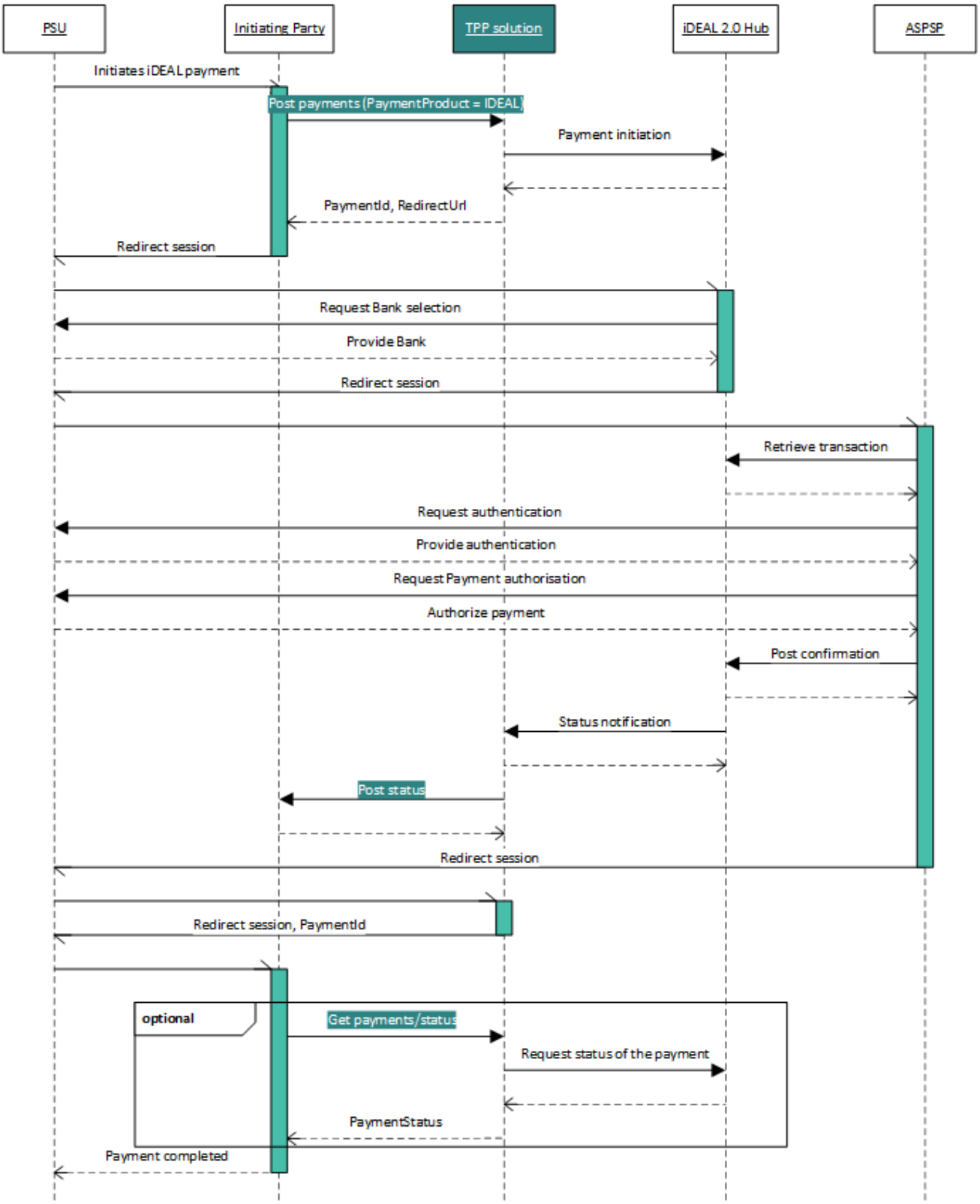
5.1.1 Sequence diagrams

The sequence diagrams in this paragraph are examples of possible flows for standard iDEAL payments with and without profile recognition, they do not encompass all the possible flows.

In the diagrams below the sequence of requests is shown. The vertical green bars indicate which party is responsible for the session of the PSU. If a party has the session a screen can be displayed. Notice that in the iDEAL 2.0 flow the Initiating Party can receive a notification when the authorisation of the payment is finished on the ASPSP side. In order to receive this notification, the Initiating Party should implement the Post status API, so this can be called by the TPP solution. The initiating party also has the option to request the status by calling the Get /payments/status API of the TPP solution.

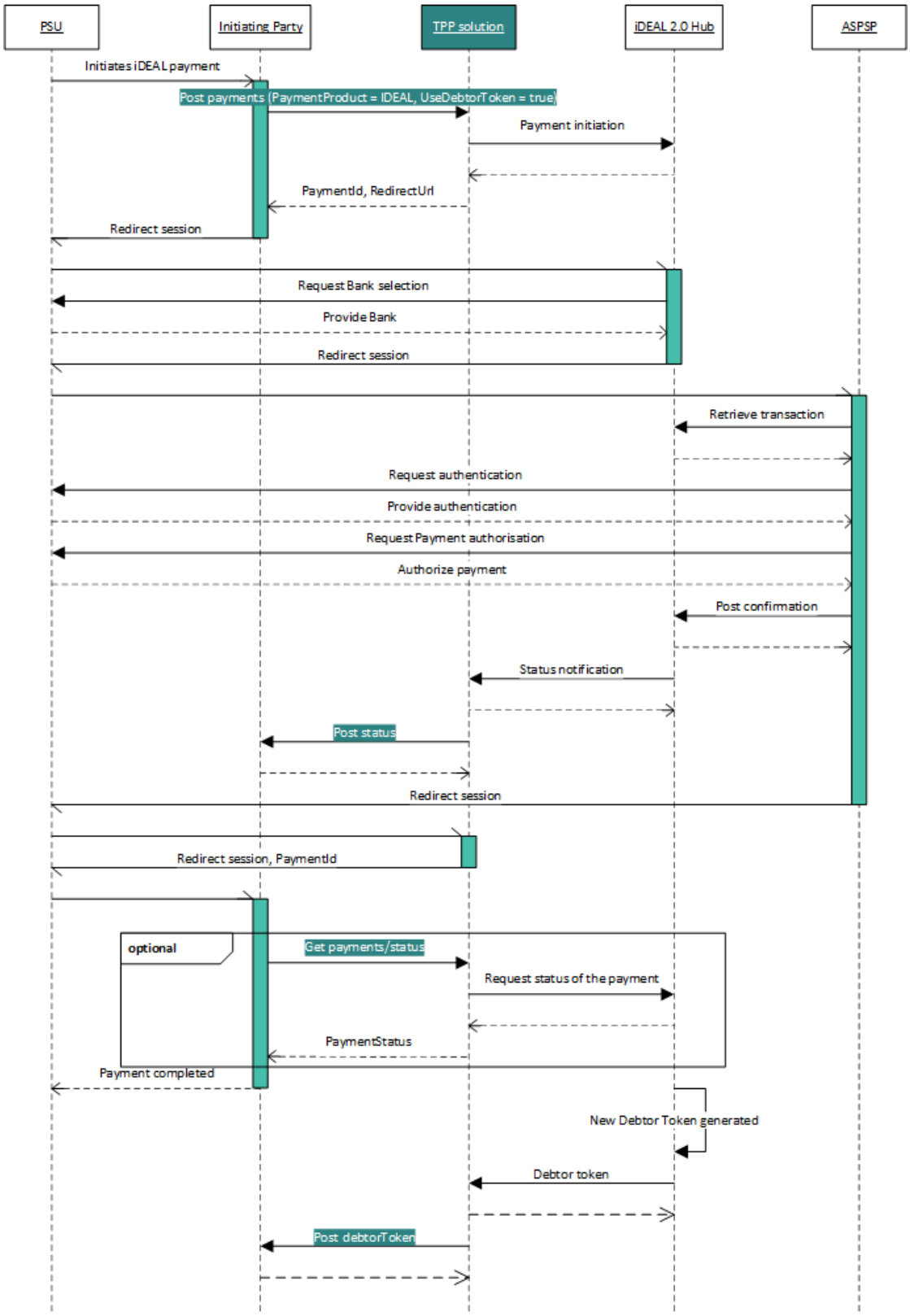
5.1.1.1 Redirect and no Debtor token

This is an example of an iDEAL payment where the PSU is not recognized by the iDEAL Hub.



5.1.1.2 Redirect with newly generated Debtor token

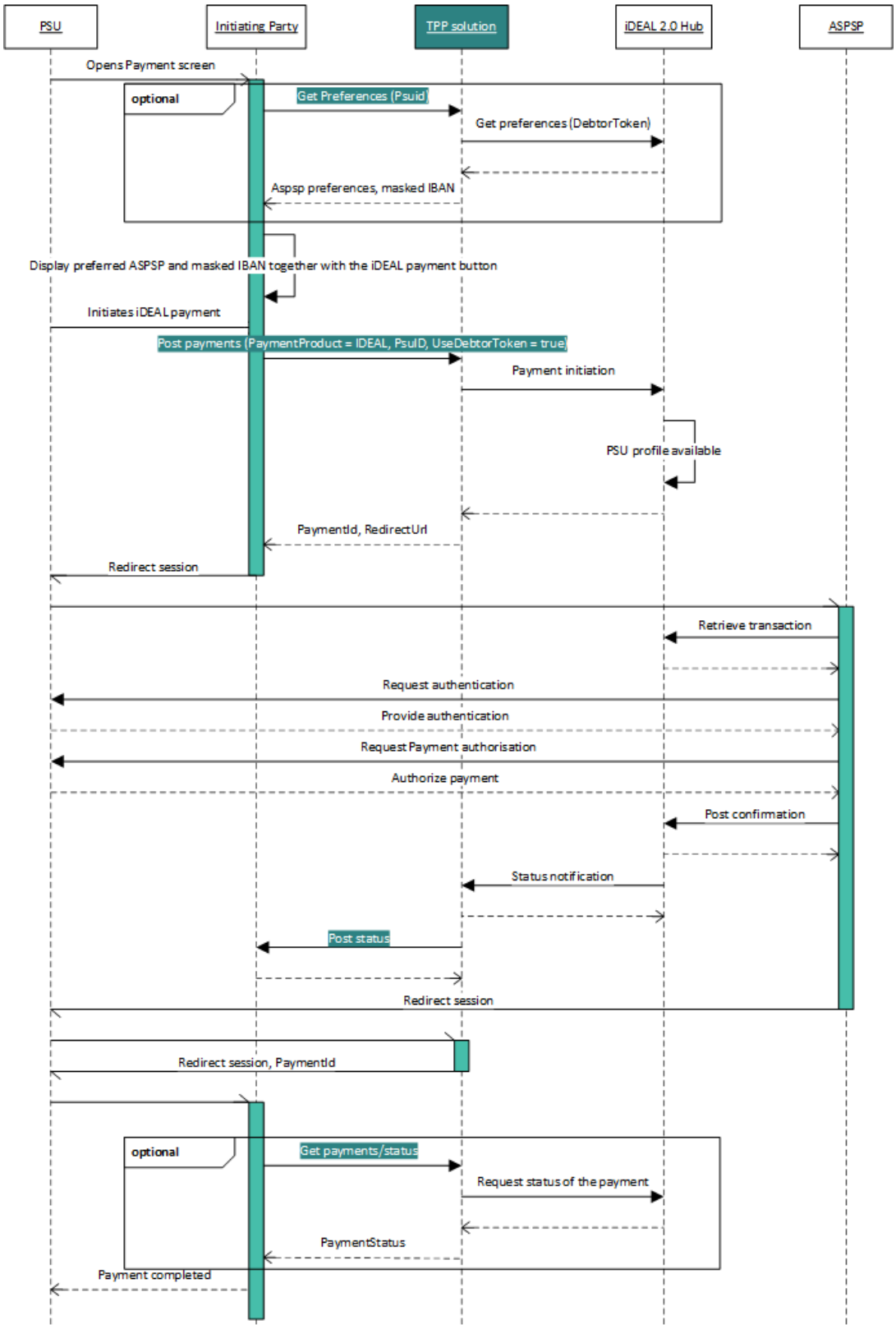
The iDEAL payment product has the option to store information about the PSU, such as the favorite bank and the IBAN. In order to use this functionality, the Initiating Party should set the field '`UseDebtorToken = true`' in a Post /payments request towards the TPP solution. If a new debtor token is generated by the iDEAL Hub the TPP solution will return, next to the Status of the payment in Post /status, a `PsuID` in Post /debtorToken. This `PsuID` can then be used in subsequent iDEAL payments request for the same PSU (again with '`UseDebtorToken`' set to true).



5.1.1.3 Redirect with existing Debtor token

The TPP solution also offers a separate API which can be used to retrieve the preferences of the PSU, the Get /preferences API.

In the sequence diagram below the Get /preferences api is the optional step in the beginning. This flow is using the Debtor Token to reduce the number of steps for the PSU to complete the payment (no ASPSP selection required in the iDEAL Hub).



5.1.1.4 Decoupled with existing Debtor token

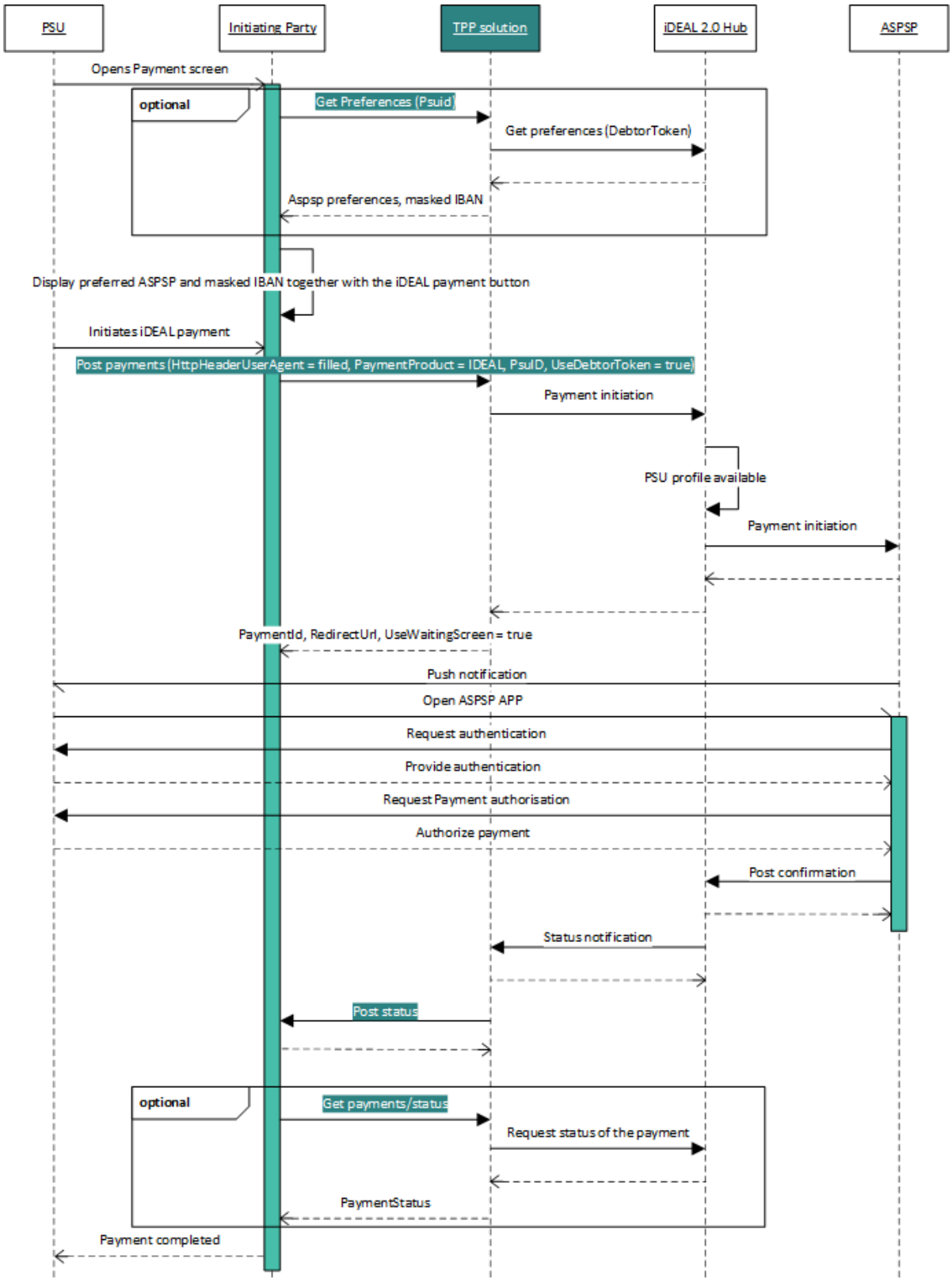
This sequence diagram shows a possible decoupled flow, but is otherwise the same as the 'Redirect with existing debtor token' flow.

In a decoupled flow another device is used to authenticate and authorize the payment, this can be seen in the sequence diagram below by the fact that 2 sessions (green vertical bars) are active at the same time. The PSU doesn't leave the session with the Initiating Party, in the meantime a session is started on another device (for example a smartphone) which handles the interaction between the PSU and the ASPSP. The decoupled flow can potentially speed up the authentication process when bio-metrics are used in the ASPSP's app on the smartphone.

If the following fields are provided in the Post /payments request a decoupled flow can be initiated by the ASPSP:

- `HttpHeaderUserAgent`
- `PsuId`
- `UseDebtorToken = true`

In case of a decoupled flow the Post /payments response will have '`UseWaitingScreen = true`' field.



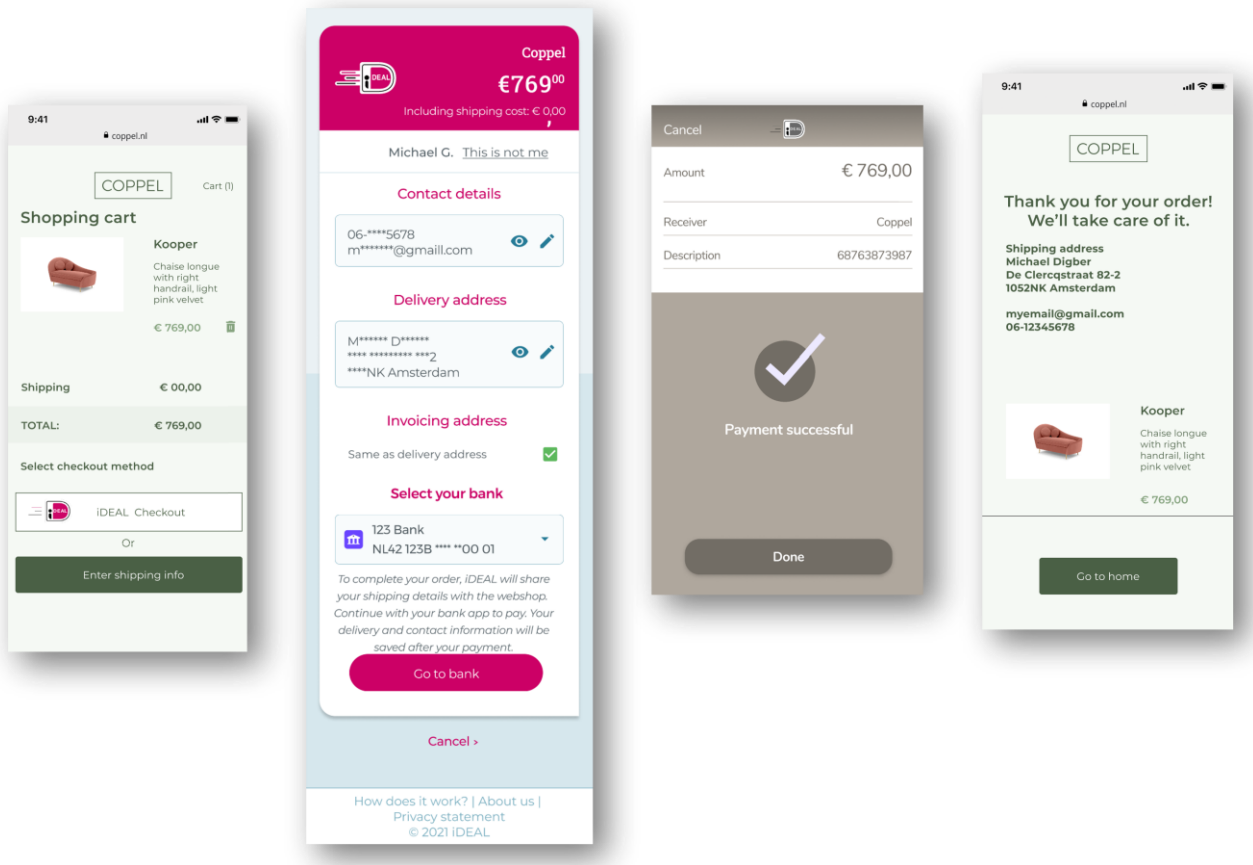
5.2 Fast Checkout iDEAL Payment

An iDEAL payment with iDEAL Checkout allows a PSU to centrally manage his shipping, invoice and contact details in its iDEAL profile and provide these to an Initiating Party as part of the iDEAL payment.

In an iDEAL payment with iDEAL Checkout, a PSU is redirected to the iDEAL Payment Page. If the PSU is recognized by a cookie and an iDEAL profile is detected, the shipping address preferences are shown to him on the iDEAL payment page. The PSU then confirms its preferred contact details and shipping address on the iDEAL payment page, after which he proceeds with the transaction through either a redirect to the ASPSP, a QR code scan, or a push notification to its mobile banking app. After the PSU has authorized the payment at the ASPSP, he is redirected back to the Initiating Party. Upon authorization, the Initiating Party receives confirmation of the payment (similar to a regular iDEAL transaction), which will additionally include the chosen contact details and shipping address of the PSU.

Currently only Dutch shipping addresses are allowed for PSUs

Below an example screen flow of a Fast Checkout iDEAL Payment.



5.2.1 Initiating an iDEAL Checkout transaction with the Open Banking v3 API

An iDEAL Checkout transaction can be initiated by indicating this in the `FlowType` field of the Post /payments. A breakdown of the transaction amount needs to be added to this call, which will be shown to the PSU during the iDEAL Checkout transaction flow alongside its shipping details.

5.2.1.1 Initiating an iDEAL Checkout transaction

- To mark a transaction as iDEAL Checkout, the field `FlowType` MUST be set to `FastCheckout`
- To initiate an iDEAL Checkout transaction, the `AmountBreakdown` group MUST be included in the POST /payments call.
    - This group MUST include both an OrderAmount and a ShippingCost. If no shipping costs apply, the field must be filled with 0.
- The sum of the `OrderAmount` and `ShippingCost` MUST be equal to the Amount field, which represents the total transaction amount to be payed by the PSU.
- To specify the desired iDEAL Checkout data to be received, the `ExpectedCheckoutData` group can be set. Only those data fields that are set to `true` will be included in the callback.

In a iDEAL Checkout transaction, the response of the Post /payments call will always include a URL to the payment page in the `Links.RedirectUrl` field.

5.2.1.2 Shipping amount and shipping/invoice address restrictions

- When initiating an iDEAL Checkout transaction, the Initiating Party MUST specify the desired iDEAL Checkout data to be received, which can be included in the payment initiation via the `ExpectedCheckoutData` group
    - The Initiating Party MUST comply with the General Data Protection Regulation (GDPR). The Initiating Party SHOULD only request the personal data which is needed to fullfill the agreement with the PSU;
- The Initiating Party MUST provide both an `OrderAmount` and `ShippingCost`.
- The PSU can only add Shipping and Invoice addresses in The Netherlands. Any other non-NL addresses are currently not allowed. Initiating Party's therefore can trust to receive only NL addresses.
- It is (currently) not possible to dynamically define or alter the shipping amount based on the PSU's chosen shipping address. The shipping amount can only be provided one time in the transaction initiation. The shipping amount can also be 0.
- In case the PSU cannot or does not want to provide its iDEAL Checkout data, the transaction is canceled. This means that if an iDEAL Checkout transaction is initiated, a successful and payed transaction will always include the iDEAL Checkout data.

5.2.2 Selection/registration of iDEAL Checkout data by PSU

After initiating an iDEAL Checkout, the PSU is redirected to the iDEAL Payment Page. Here, the amount breakdown in order amount and shipping amount is shown, next to the regular payment details.

- If the PSU is recognized by a browser cookie, the known shipping address preferences from its iDEAL profile are shown, in addition to the regular profile preferences. Note that only the data fields indicated in `ExpectedCheckoutData` will be shown to the PSU (this applies to all iDEAL Checkout flows).

- Profile information presented will be partly masked and can only be fully shown if the PSU authenticates via its ASPSP.
- If a registered PSU does not have any shipping address registered with its profile yet, it will first be asked to provide an address before continuing with the iDEAL Checkout transaction.
- If the PSU is not recognized, it will first be asked whether the PSU has a registered iDEAL profile. In case of an existing profile, the PSU will be asked to select its ASPSP and is redirected to that ASPSP for authentication. After successful authentication, the PSU will return to the iDEAL payment page to view its (unmasked) profile and iDEAL Checkout data.
- At any time during the iDEAL Checkout flow, the PSU may choose to change its preferred address for the transaction or add a new address. This is done on the iDEAL payment page.
- If the PSU is not recognized and is a new PSU, the PSU is asked to register an iDEAL profile, including data for iDEAL Checkout.

5.2.3 Shipping data in transaction callback

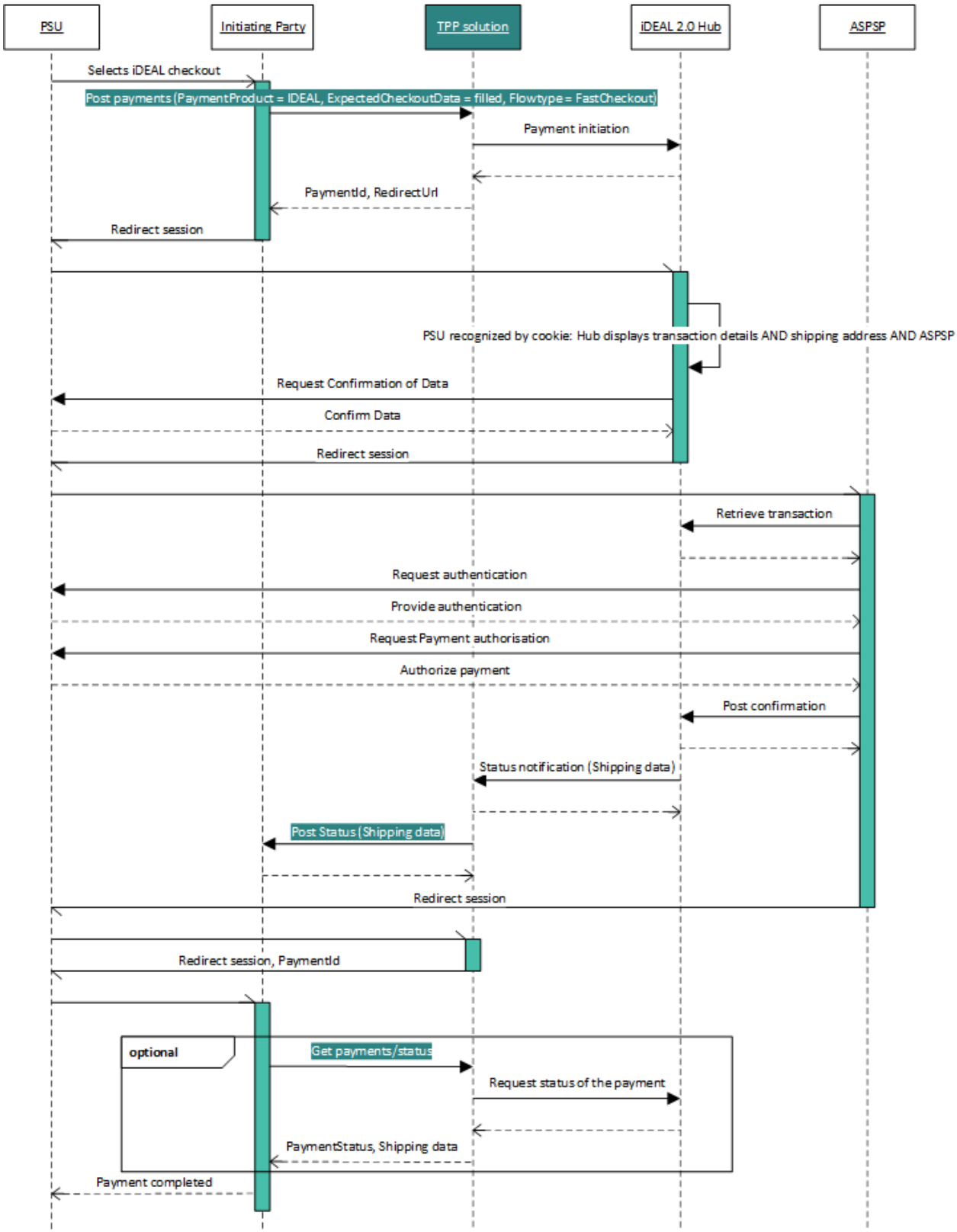
After confirming the selected delivery details and preferred IBAN, the PSU is redirected to the ASPSP for authenticating and authorizing the payment. This is done at the ASPSP, similar to a regular transaction.

If the PSU was not registered with iDEAL yet, a registration flow was initiated at the iDEAL payment page.

After the PSU has confirmed the transaction payment at the ASPSP, the PSU's iDEAL Checkout data (like shipping address) is added to the Post /status callback alongside the other payment details. The iDEAL Checkout data can be found in the DebtorInformation group. Note that only data fields that were indicated in the ExpectedCheckoutData will be included in the callback.

5.2.4 Sequence diagram

This iDEAL Fast Checkout flow is based on de the recognition of the PSU by the iDEAL 2.0 Hub based on a browser cookie. In the sequence diagram below the PSU is recognized. This flow will simplify the checkout screen of the Initiating Party; shipping, invoice and contact details are all handled by the iDEAL Hub. The shipping data is provided in the Post /status request or the Get /payments/status response after the payment was finalized at the ASPSP.



6 OB v3 for iDEAL - Authorization API

The following two steps have to be performed for each service to facilitate a secure connection between the Initiating Party and the TPP Solution.

1. Upload certificate
2. Retrieve token

6.1 Upload certificate

The Initiating Party uploads his public certificate in the TPP Solution GUI.

6.2 Retrieve token

Endpoint: POST /token

This API retrieves the token which is used in the communication between the Initiating Party and the TPP Solution. The Initiating Party is using his private key to sign the request. In the response he will receive an access token from the TPP Solution. This token is used in all subsequent API calls towards the TPP Solution.

Request

Location	Name	Comments	Type
FormData	**grant_type** *required*	To be set to 'client_credentials'	String

Header	**Authorization**
	required
		The signature. It contains the header attributes 'app', 'client', 'id' and 'date' signed with the private key of the client.
		The signature will be used to sign the authorization request with the private key which corresponds to the certificate provided for the onboarding.
		Structure
		`Signature keyId="<thumbprint of certificate>", algorithm="SHA256withRSA", headers="app client id date", signature="<signature>"`
		Example
		`Signature keyId="58AF4EC5ADD4C4A3F28D3AEFF60656B2F2xxxxxx", algorithm="SHA256withRSA", headers="app client id date", signature="Abczym2rZF…r5qcvgmA=="`
		Generating rules
		The signature must be created over a String where app, client, id and date are concatenated with the following rules:
		• The keyId is the thumbprint of the certificate, viewed with the SHA1 algorithm.
		• Create the header field string by concatenating the lowercased header field name followed with an ASCII colon `:`, an ASCII space ` `, and the header field value. Leading and trailing optional whitespace (OWS) in the header field value MUST be omitted (as specified in RFC7230 [RFC7230], Section 3.2.4 [7]). If value is not the last value then append an ASCII newline `\n`.
		More details can be found here: https://datatracker.ietf.org/doc/draft-ietf-httpbis-message-signatures/
			String

Header	**App** *required*	The name of the service. Only `IDEAL` is allowed.	String
Header	**Client** *required*	The name of the client. This name is provided to the Initiating Party during onboarding. The name of the client is created by the TPP Solution.	String
Header	**Id** *required*	The combination of Initiating Party ID and sub Id. For example if Initiating Party ID is 433 and the sub ID is 5 the ID will be 433:5 IP=433, subId=5 -> 433:5 IP=434, no subId -> 434	String
Header	**Date** *required*	Should be filled with the current date. The following date formats are supported: 1. EEE MMM dd HH:mm:ss zzz yyyy 2. ISO DATE: for example 2011-12-03T10:15:30+01:00 3. RFC 1123: for example Tue, 3 Jun 2008 11:05:30 GMT	Date

Response

Location	Name	Comments	Type
[1..1]	access_token	Token to be used in further API calls	String
[1..1]	token_type	Type of the token: Bearer	String
[1..1]	expires_in	Expiration time in seconds	Integer

6.2.1 Example: Authorization

Request

```
    Address: https://localhost:8443/xs2a/routingservice/services/authorize/token
    HttpMethod: POST

    Headers: {App=IDEAL, Accept=application/json, Date=2022-03-25T09:41:31.256Z, Authorization=Signature
KeyID="8D0F688AD3E6C2D4D5FB99FE129F2A2E3B496AF7", algorithm="SHA256withRSA", headers="app client id
date", signature=
"kAIepMoo6CRTWz9CLUFcpZj8eNQTdjXq6V8+kdk/9M1GmVud2CVrP1NMNTEiXgKzBlFQQlhv1iaFhMVOLVq7u8aEV4eeoNxjTLDK+lk4
zkjCBjeOyXtr32dtfjsvytlzhXw7KJizgOGd+m4Gh9xtSjY0I5QM/p+znKZsJCVKNSUUBZndAxIudsxy2Srp/yzexmvWpsoAvWIZzwtDS
03h4PjGTGKloXz6KyC+/I+GSBjw9M3GATUMMVrrgTKoR8oI0Xcr9v7ZTr3KpT1d1/LrcxQ82o2kq0+4ECVoJdVRezr2oZRmZ5hTHTIHeh
MNkASnuDqzDaQxQvMInUTg8tFKGA==", Id=000784, Client=Worldline}

    grant_type=client_credentials
```

Response

```
Content-Type: application/json;charset=UTF-8
    ResponseCode: 200

    Headers: {X-Request-ID=23eeed2d-f163-43c5-94b1-eeadcbb393e3, MessageCreateDateTime=2022-03-
25T09:41:31.819Z, Date=Fri, 25 Mar 2022 09:41:31 GMT, Content-Type=application/json;charset=UTF-8}
    Payload: {
    "access_token": "abb5468b4845dffff9cccd7c950e529",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

7 OB v3 for iDEAL - Payment Initiation API

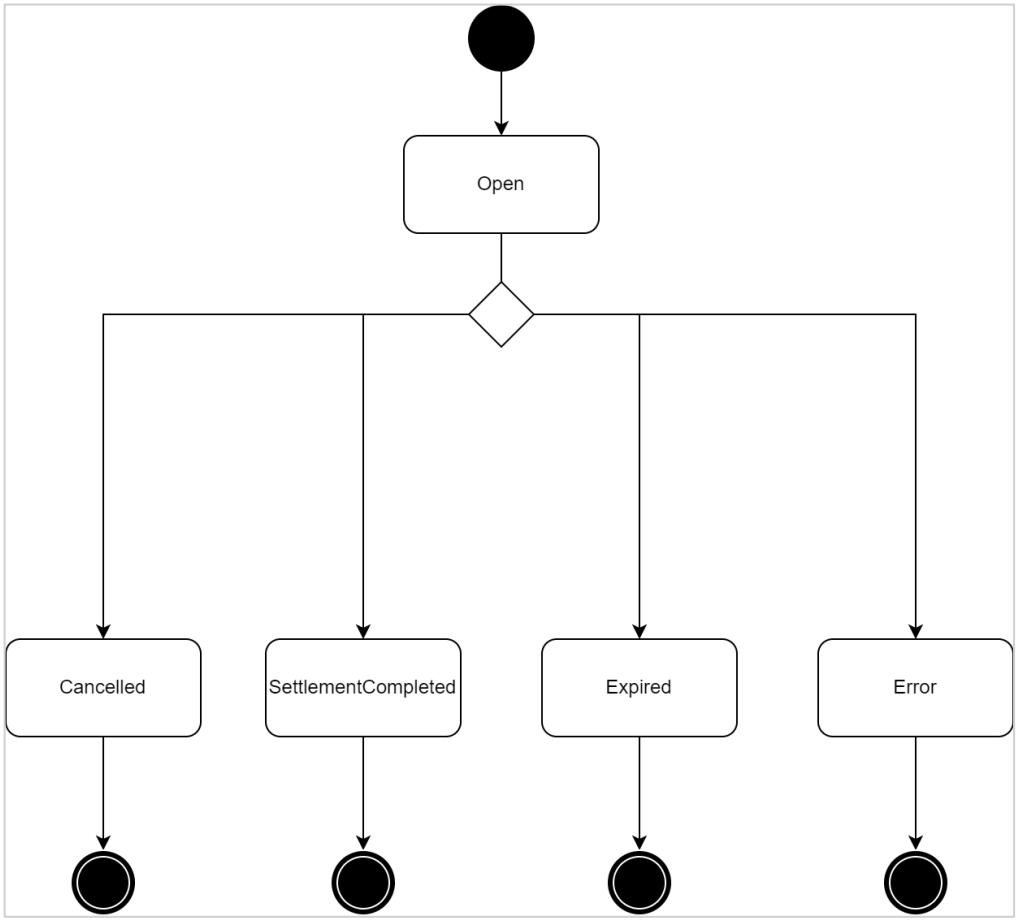
The iDEAL payment api uses a subset of the available open banking API's. In this chapter only the api's needed for iDEAL payment initiation are described, furthermore the fields applicable to an iDEAL payment are marked purple (optional) and orange (mandatory).

7.1 Payment Status

Payments have different status depending on the actual state of processing:

- Open
- SettlementCompleted
- Cancelled
- Expired
- Error

In the following picture the possible iDEAL status are shown in an activity diagram:



7.2 Payment Initiation Endpoints

The base URL for the payment initiation service API is: **/xs2a/routingservice/services/ob/pis/v3**

7.2.1 POST Payments

Endpoint: POST /payments

This endpoint is used by the Initiating Party to set up payment initiation for an ASPSP.

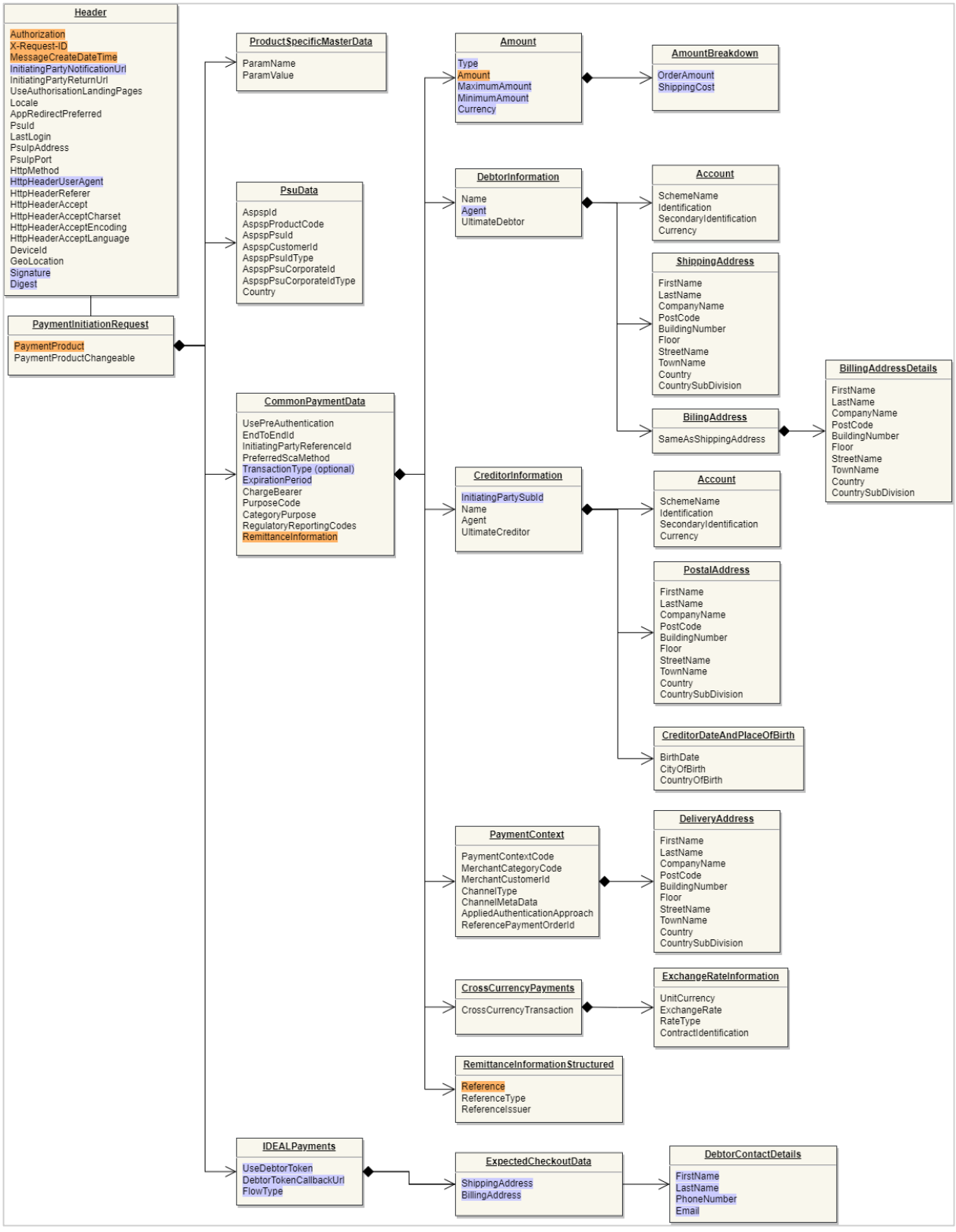
- The purple fields are applicable to IDEAL, those fields will be used towards the IDEAL Hub.
- The orange fields are mandatory for an IDEAL payment.

More details about the fields can be found in the yaml specification.

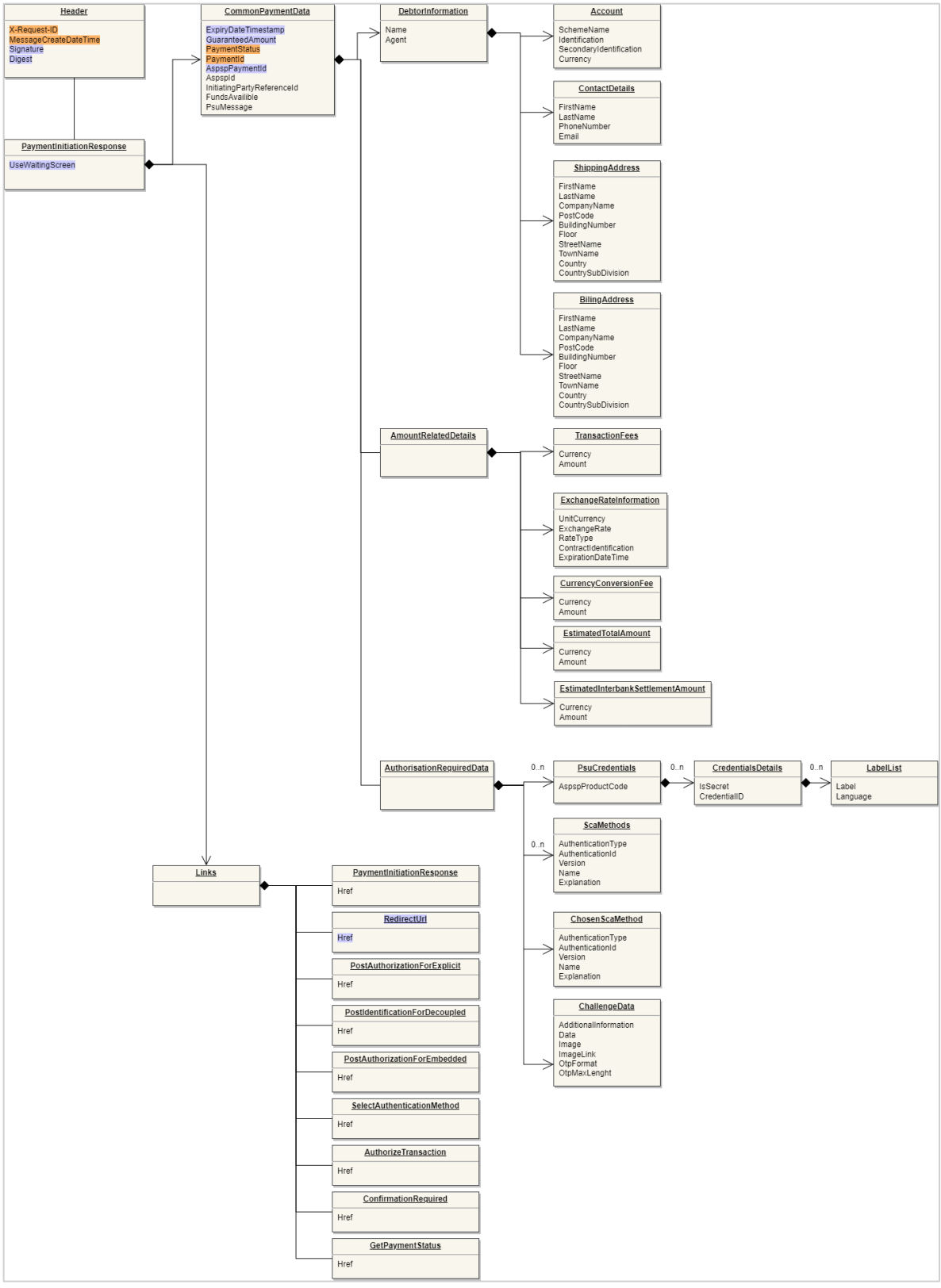
During a prolonged period the Initiating Party is allowed to use the `CommonPaymentData.DebtorInformation.Agent` field to pass a BIC which identifies the ASPSP. This is done to facilitate a smooth migration from iDEAL to iDEAL 2.0.
When the `CommonPaymentData.DebtorInformation.Agent` field is filled the response will always have a `RedirectUrl` which redirects directly to the ASPSP.

- This feature is only available if FlowType = standard and no iDEAL Debtor token is used.
- This feature will be discontinued and is only meant to be used during the migration period.

7.2.1.1 Request



7.2.1.2 Response



7.2.1.3 Example: Standard iDEAL payment without Debtor token

Request (Signature-related fields "Digest" and "Signature" are conditionally mandatory):

```
Address: https://localhost:8443/xs2a/routingservice/services/ob/pis/v3/payments
    HttpMethod: POST
    Headers: {X-Request-ID=27965d49-b27b-38a9-6f7a-2211cb5c0c78, Authorization=Bearer
69eb307c55eb91b4a4144ec8a2d8562d, MessageCreateDateTime=2022-03-22T14:02:13.184Z, Digest=SHA-
256=DZ0QKHaRy/MdbWEI62wuE7xJMrfJJBTUWF8irHIHTGw=,
Signature=keyId="8D0F688AD3E6C2D4D5FB99FE129F2A2E3B496AF7",algorithm="SHA256withRSA",headers="digest x-
request-id messagecreatedatetime (request-
target)",signature="DK7sQCW56EMXHOw/rdW3RlrtM2U/QL2Hy5g5Fsq+oZ0qspiDerzYL0FRu47EdGWT9qy7n7R3ay6iJHVCUrgFv
VXS806E3Clgh5u7+J8/xgUeEPRglA6jeNaLYErKC+xIklJrmpKODDkD407IgA6Y9LhixFisSfawR1pDs2vwOsB1wnujrwBcKFzYf66/px
hW06NZu98VM0pkLm5OFp82G0x4vHfOPAoo46xcSXd/IbQV0DYZewosQIaI5qAclPvmusO9uXDx90uqgDL4E8fLW4yHEaraW7yDlCx54bc
raU4ubo+wqDE8NJMKxEVreI/Se9YY73mvAMnOkon1jZHsAQ==",
    Payload: {
    "PaymentProduct": ["IDEAL"],
    "CommonPaymentData": {
        "Amount": {
            "Type": "Fixed",
            "Amount": "20.45",
            "Currency": "EUR"
        },
        "RemittanceInformation": "Cookie",
        "RemittanceInformationStructured": {"Reference": "iDEALpurchase21"}
    },
    "IDEALPayments": {
        "UseDebtorToken": false,
        "FlowType": "Standard"
    }
}
```

Response (Signature-related fields "Digest" and "Signature" are conditionally present):

```
ResponseCode: 201

    Headers: {X-Request-ID=0e1b8b7d-2438-4bb0-acf5-d68f967c8b25, MessageCreateDateTime=2022-03-
22T14:02:13.694Z, Digest=SHA-256=gJNKvdO+dlRlWyevVFfDeEAAJDtmw41A3pQOyNX3K6o=,
Signature=keyId="3EBEF6033C00730D9C6DA05165A3CAA1F31036FB",algorithm="rsa-
sha256",headers="messagecreatedatetime x-request-id
digest",signature="NMQImrvrg664n3rBCF9s8fQLvZVH/zPsHfzzxJ671BWz6fvjtn81Arv6TPd8N6/nLLNNjY23YubI33tYDjyEf1
vq1wF3vq/gFqmF2s66dxHFy7Gd9mOEdpxc0BYCwpG2ozntRvnrIv2z9SXFlkljc6OT+mqLo+vCB8Gn6frdrwGgZqY6uHOEh3rPvYJ/nWT
HQkfxzx/U624wqtuoI8Va4kIXM2pPBCPo4AAq2hFXtj4Okh5vFmRlRiIM9CFgfO+5k1lMC3YzrmE8aB9CyFoJoT3e5pSKNePL+hunGlJG
6XPg92Hg/gpS06PyvCtHA5585ONkQv+oNhFdpfdtob1qBQ=="}
    Payload: {
    "CommonPaymentData": {
        "ExpiryDateTimestamp": "2023-03-14T11:46:58.579Z",
        "PaymentStatus": "Open",
        "PaymentId": "19897",
        "AspspPaymentId": "0001475439353173"
    },
    "Links": {
        "RedirectUrl": {
            "Href": "https://worldline.com"
        },
        "GetPaymentStatus": {
            "Href": "https://localhost:8443/xs2a/routingservice/services/ob/pis/v3/payments/19897/status"
        }
    },
    "UseWaitingScreen": false
}
```

7.2.1.4 Example: iDEAL Payment with Fast Checkout

Request (Signature-related fields "Digest" and "Signature" are conditionally mandatory):

```
Address: https://localhost:8443/xs2a/routingservice/services/ob/pis/v3/payments
    HttpMethod: POST
    Headers: {X-Request-ID=2a773a8d-f7ca-ad5f-f98b-eb42b3a4f327, Authorization=Bearer
69eb307c55eb91b4a4144ec8a2d8562d, MessageCreateDateTime=2022-03-22T14:46:03.235Z,  Digest=SHA-
256=gJNKvdO+dlRlWyevVFfDeEAAJDtmw41A3pQOyNX3K6o=,
Signature=keyId="3EBEF6033C00730D9C6DA05165A3CAA1F31036FB",algorithm="rsa-sha256",headers="digest x-
request-id messagecreatedatetime (request-
target)",signature="NMQImrvrg664n3rBCF9s8fQLvZVH/zPsHfzzxJ671BWz6fvjtn81Arv6TPd8N6/nLLNNjY23YubI33tYDjyEf
1vq1wF3vq/gFqmF2s66dxHFy7Gd9mOEdpxc0BYCwpG2oztntRvnrIv2z9SXFlKljc6OT+mqLo+vCB8Gn6frdrwGgZqY6uHOEh3rPvYJ/nW
THQkfxzx/U624wqtuoI8Va4kIXM2pPBCPo4AAq2hFXtj4Okh5vFmRlRiIM9CFgfO+5k1lMC3YzrmE8aB9CyFoJoT3e5pSKNePL+hunGlJ
G6XPg92Hg/gpS06PyvCtHA5585ONkQv+oNhFdpfdtob1qBQ==",signature="NMQImrvrg664n3rBCF9s8fQLvZVH/zPsHfzzxJ671BWz6fvjtn81Arv6TPd8N6/nLLNNjY23YubI33tYDjyEf1vq1wF3vq/gFqmF2s66dxHFy7Gd9mOEdpxc0BYCwpG2oztntRvnrIv2z9SXFlKljc6OT+mqLo+vCB8Gn6frdrwGgZqY6uHOEh3rPvYJ/nWTHQkfxzx/U624wqtuoI8Va4kIXM2pPBCPo4AAq2hFXtj4Okh5vFmRlRiIM9CFgfO+5k1lMC3YzrmE8aB9CyFoJoT3e5pSKNePL+hunGlJG6XPg92Hg/gpS06PyvCtHA5585ONkQv+oNhFdpfdtob1qBQ==",signature="NMQImrvrg664n3rBCF9s8fQLvZVH/zPsHfzzxJ671BWz6fvjtn81Arv6TPd8N6/nLLNNjY23YubI33tYDjyEf1vq1wF3vq/gFqmF2s66dxHFy7Gd9mOEdpxc0BYCwpG2oztntRvnrIv2z9SXFlKljc6OT+mqLo+vCB8Gn6frdrwGgZqY6uHOEh3rPvYJ/nWTHQkfxzx/U624wqtuoI8Va4kIXM2pPBCPo4AAq2hFXtj4Okh5vFmRlRiIM9CFgfO+5k1lMC3YzrmE8aB9CyFoJoT3e5pSKNePL+hunGlJG6XPg92Hg/gpS06PyvCtHA5585ONkQv+oNhFdpfdtob1qBQ=="}
    Payload: {
    "PaymentProduct": ["IDEAL"],
    "CommonPaymentData": {
        "Amount": {
            "Type": "Fixed",
            "Amount": "33.45",
            "Currency": "EUR",
        "AmountBreakdown": {
                "OrderAmount": "23.45",
                "ShippingCost": "10.00"
            }
        },
        "RemittanceInformation": "Cookie",
        "RemittanceInformationStructured": {"Reference": "iDEALpurchase21"}
    },
    "IDEALPayments": {
        "UseDebtorToken": false,
        "FlowType": "FastCheckout",
        "ExpectedCheckoutData": {
            "DebtorContactDetails": {
                "FirstName": true,
                "LastName": true,
                "PhoneNumber": true,
                "Email": true
            },
            "ShippingAddress": true
            "BillingAddress": true
        }
    }
}
```

Response (Signature-related fields "Digest" and "Signature" are conditionally present):


```
ResponseCode: 201

    Headers: { X-Request-ID=c3c2c78f-6588-40b8-b15f-685d0d987277, MessageCreateDateTime=2022-03-
22T14:46:03.668Z, Date=Tue, 22 Mar 2022 14:46:03 GMT, Digest=SHA-
256=gJNKvdO+dlRlWyevVFfDeEAAJDtmw41A3pQOyNX3K6o=,
Signature=keyId="3EBEF6033C00730D9C6DA05165A3CAA1F31036FB",algorithm="rsa-
sha256",headers="messagecreatedatetime x-request-id
digest",signature="NMQImrvrg664n3rBCF9s8fQLvZVH/zPsHfzzxJ671BWz6fvjtn81Arv6TPd8N6/nLLNNjY23YubI33tYDjyEf1
vq1wF3vq/gFqmF2s66dxHFy7Gd9mOEdpxc0BYCwpG2ozntRvnrIv2z9SXFlKljc6OT+mqLo+vCB8Gn6frdrwGgZqY6uHOEh3rPvYJ/nWT
HQkfxzx/U624wqtuoI8Va4kIXM2pPBCPo4AAq2hFXtj4Okh5vFmRlRiIM9CFgfO+5k1lMC3YzrmE8aB9CyFoJoT3e5pSKNePL+hunGlJG
6XPg92Hg/gpS06PyvCtHA5585ONkQv+oNhFdpfdtob1qBQ=="}
    Payload: {
    "CommonPaymentData": {
        "ExpiryDateTimestamp": "2023-03-14T11:46:58.579Z",
        "PaymentStatus": "Open",
        "PaymentId": "19897",
        "AspspPaymentId": "0001475439353173"
    },
    "Links": {
        "RedirectUrl": {
            "Href": "https://worldline.com"
        },
        "GetPaymentStatus": {
            "Href": "https://localhost:8443/xs2a/routingservice/services/ob/pis/v3/payments/19897/status"
        }
    },
    "UseWaitingScreen": false
}
```

7.2.2 GET Payment Status

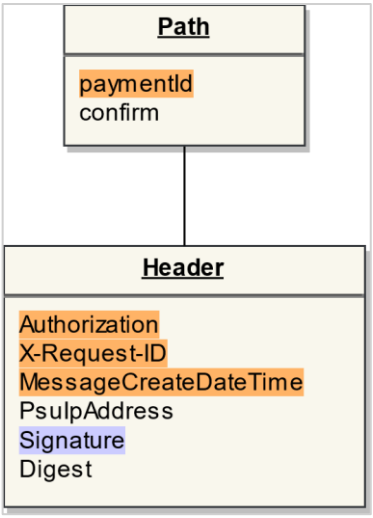
Endpoint: GET /payments/{paymentId}/status

This endpoint is used to retrieve the status of a payment.

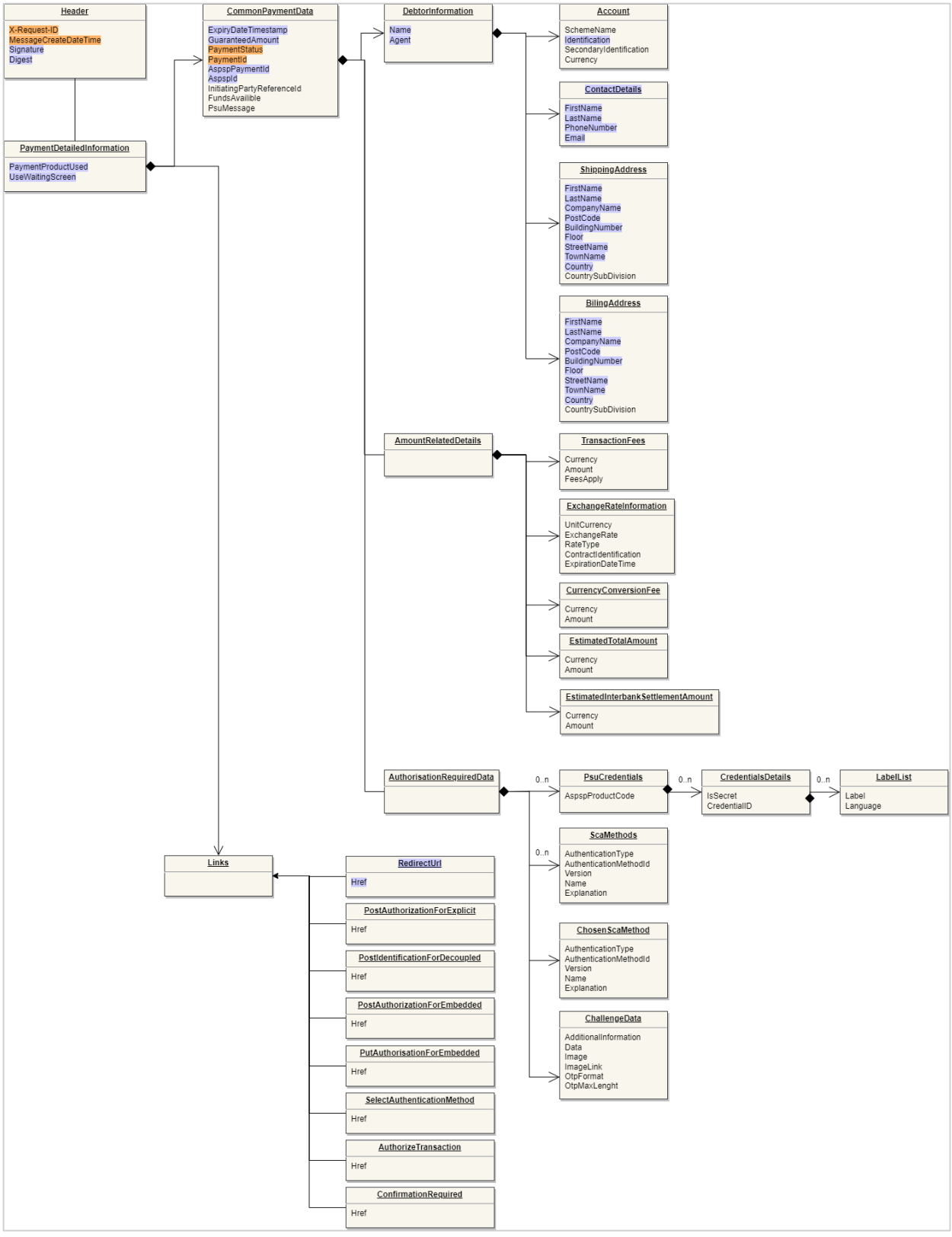
- The purple fields are applicable to IDEAL, those fields will be used towards the IDEAL Hub.
- The orange fields are mandatory for an IDEAL payment.

More details about the fields can be found in the yaml specification.

7.2.2.1 Request



7.2.2.2 Response



7.2.2.3 Example: iDEAL payment status

Request (Signature-related fields "Digest" and "Signature" are conditionally mandatory):

```
Address: https://localhost:8443/xs2a/routingservice/services/ob/pis/v3/payments/11376380/status
    HttpMethod: GET

    Headers: {Accept=application/json, X-Request-ID=de875f27-5eb7-2b3d-bb88-86e0fe0cdf65,
Authorization=Bearer ec464ba833a955952646cbc834f4e6b, MessageCreateDateTime=2022-03-24T14:48:16.143Z,
Digest=SHA-256=gJNKvdO+dlRlWyevVFfDeEAAJDtmw41A3pQOyNX3K6o=,
Signature=keyId="3EBEF6033C00730D9C6DA05165A3CAA1F31036FB",algorithm="rsa-sha256",headers="digest x-
request-id messagecreatedatetime (request-
target)",signature="NMQImrvrg664n3rBCF9s8fQLvZVH/zPsHfzzxJ671BWz6fvjtn81Arv6TPd8N6/nLLNNjY23YubI33tYDjyEf
1vq1wF3vq/gFqmF2s66dxHFy7Gd9mOEdpxc0BYCwpG2ozntRvnrIv2z9SXFlKljc6OT+mqLo+vCB8Gn6frdrwGgZqY6uHOEh3rPvYJ/nW
THQkfxzx/U624wqtuoI8Va4kIXM2pPBCPo4AAq2hFXtj4Okh5vFmRlRiIM9CFgfO+5k1lMC3YzrmE8aB9CyFoJoT3e5pSKNePL+hunGlJ
G6XPg92Hg/gpS06PyvCtHA5585ONkQv+oNhFdpfdtob1qBQ=="}
 }
```

Response (Signature-related fields "Digest" and "Signature" are conditionally present):

```
ResponseCode: 200

    Headers: { X-Request-ID=283c264a-3212-4240-a039-01b229fa854f, MessageCreateDateTime=2022-03-
24T14:48:16.541Z, Digest=SHA-256=gJNKvdO+dlRlWyevVFfDeEAAJDtmw41A3pQOyNX3K6o=,
Signature=keyId="3EBEF6033C00730D9C6DA05165A3CAA1F31036FB",algorithm="rsa-
sha256",headers="messagecreatedatetime x-request-id
digest",signature="NMQImrvrg664n3rBCF9s8fQLvZVH/zPsHfzzxJ671BWz6fvjtn81Arv6TPd8N6/nLLNNjY23YubI33tYDjyEf1
vq1wF3vq/gFqmF2s66dxHFy7Gd9mOEdpxc0BYCwpG2ozntRvnrIv2z9SXFlKljc6OT+mqLo+vCB8Gn6frdrwGgZqY6uHOEh3rPvYJ/nWT
HQkfxzx/U624wqtuoI8Va4kIXM2pPBCPo4AAq2hFXtj4Okh5vFmRlRiIM9CFgfO+5k1lMC3YzrmE8aB9CyFoJoT3e5pSKNePL+hunGlJG
6XPg92Hg/gpS06PyvCtHA5585ONkQv+oNhFdpfdtob1qBQ=="}
    Payload: {
    "PaymentProductUsed": "IDEAL",
    "CommonPaymentData": {
        "PaymentStatus": "SettlementCompleted",
        "PaymentId": "19900",
        "AspspPaymentId": "0001893221189524",
        "AspspId": "10002",
        "DebtorInformation": {
            "Name": "Edsger Wybe Dijkstra - Callback",
            "Agent": "RABONL2UXXX",
            "Account": {
                "SchemeName": "IBAN",
                "Identification": "NL44RABO0123456789"
            }
        }
    }
}
```

8 OB v3 for iDEAL - Notification API

The notification APIs described in this chapter needs to be implemented on the Initiating Party side, if the Initiating Party decides to use them. The TPP solution will post notifications to these endpoints. For the iDEAL product the post status notification is part of the product, a value-added service is not required (because the notification is part of the iDEAL scheme and the TPP solution doesn't have to-do additional polling).

The fields marked in purple are applicable to the iDEAL payment product.

8.1 POST Status

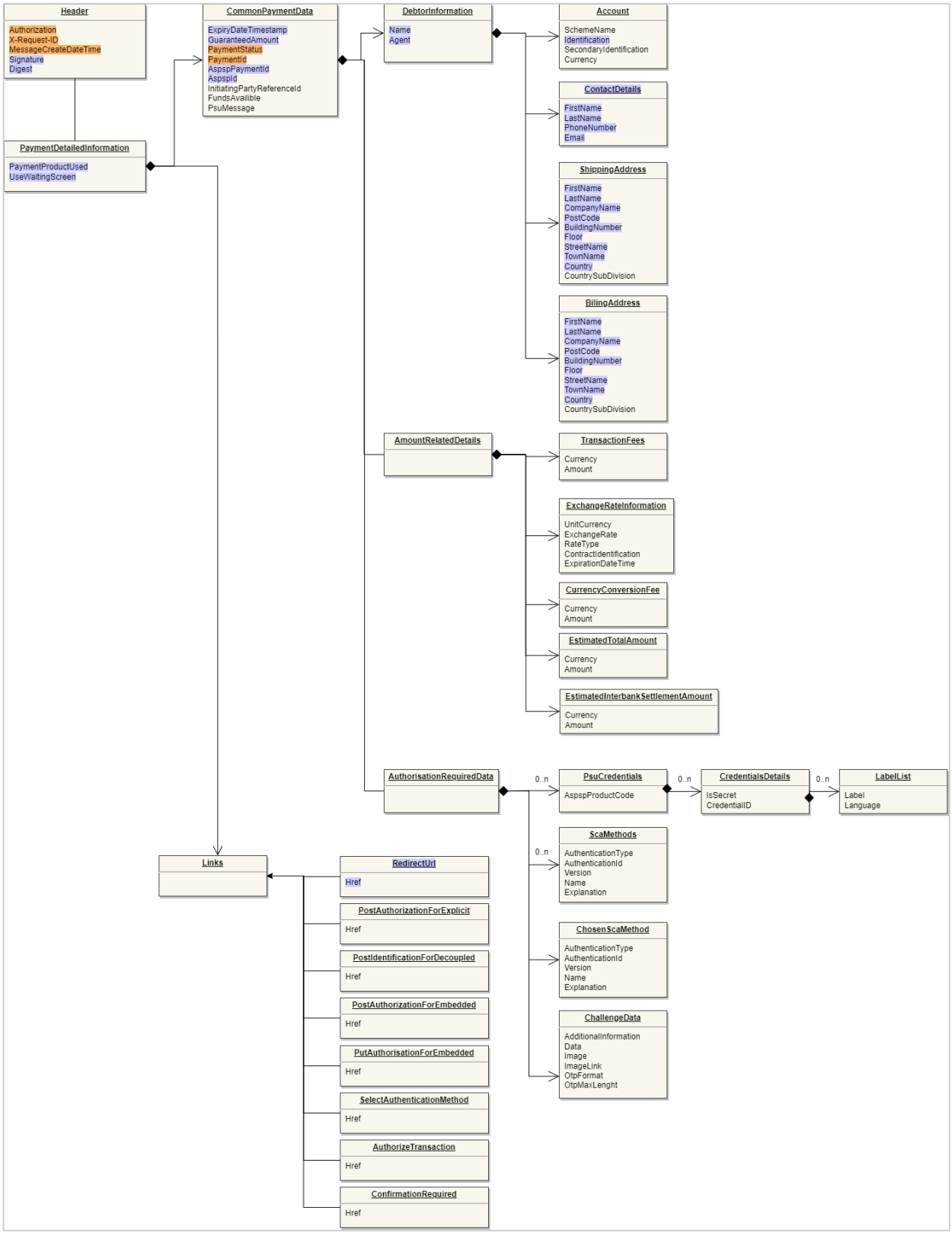
Endpoint: POST /status

This API will notify the initiating party about the status of the payment.

- The purple fields are applicable to IDEAL.
- The orange fields are mandatory for an IDEAL payment.

More details about the fields can be found in the yaml specification.

8.1.1 Request



8.1.2 Response

HTML 204 (No Content)

Header
X-Request-ID MessageCreateDateTime

8.1.3 Example: Notification for Standard iDEAL payment

Request (Signature-related fields "Digest" and "Signature" are conditionally present):

```
Address: https://checkout.company.com/transaction/webhook/91FA6EEC30844FAAB5/v3/notification/status
  HttpMethod: POST
  Headers: {Authorization=Bearer 123456789, X-Request-ID=c1452392-6c3f-4365-93f8-40558f61ac36,
MessageCreateDateTime=2023-03-15T11:51:24.185+01:00, Digest=SHA-
256=0hq1mKzxB1yyc6+hut2bEX7ps+nWyWb2pgQb6AhfhfM=,
Signature=keyId="3EBEF6033C00730D9C6DA05165A3CAA1F31036FB",algorithm="rsa-sha256",headers="digest x-
request-id messagecreatedatetime (request-
target)",signature="uYgovoK+ibAE7+MzJEKrApDUAgWfUv7RQK22zAxWHCdKCuG4d0HgqpDSqcGlKmP2IMFsC787zDU3oqKeeIIVX
R72uZBiOnm0/84UL9e7LVDHDLQsRbfDnmvgX/4xQvdwROmyqh8kkcXTf/48zY0wo2n9iDspCbgTn1DEqAqtAlwunIpea8eYA3FQc+pV2p
x77wVP7l+9mTxexzLSmum61wWbqE4ESJn0K37gXY54229ZtCnNSlu9rsvjQ5xmDf1e6MvMLBOblXHIReN2t8IH85VGK7mpi8T7JeKb8rI
G8qDbQ5TD3BmIS1+RspI95FldLCKLH91/KNrxsgPsrC2QgQ==", Content-Type=application/json}
  Payload: {
  "PaymentProductUsed" : "IDEAL",
  "CommonPaymentData" : {
    "GuaranteedAmount" : "10.00",
    "PaymentStatus" : "SettlementCompleted",
    "PaymentId" : "19928",
    "AspspPaymentId" : "0001070883053837",
    "AspspId" : "10002",
    "DebtorInformation" : {
      "Name" : "Edsger Wybe Dijkstra - Callback",
      "Agent" : "ABNANL2AXXX",
      "Account" : {
        "SchemeName" : "IBAN",
        "Identification" : "NL44RABO0123456789",
        "Currency" : "EUR"
      },
  "UseWaitingScreen" : false
}
```

Response:

```
ResponseCode: 204
```

8.1.4 Example: Notification for iDEAL Payment with Fast Checkout

Request (Signature-related fields "Digest" and "Signature" are conditionally present):

```
Address: https://checkout.company.com/transaction/webhook/91FA6EEC30844FAAB5/v3/notification/status
  HttpMethod: POST
  Content-Type: application/json
  Headers: {Authorization=Bearer 123456789, X-Request-ID=c1452392-6c3f-4365-93f8-40558f61ac36,
MessageCreateDateTime=2023-03-15T11:51:24.185+01:00, Digest=SHA-
256=0hq1mKzxB1yyc6+hut2bEX7ps+nWyWb2pgQb6AhfhfM=,
Signature=keyId="3EBEF6033C00730D9C6DA05165A3CAA1F31036FB",algorithm="rsa-sha256",headers="digest x-
request-id messagecreatedatetime (request-
target)",signature="uYgovoK+ibAE7+MzJEKrApDUAgWfUv7RQK22zAxWHCdKCuG4d0HgqpDSqcGlKmP2IMFsC787zDU3oqKeeIIVX
R72uZBiOnm0/84UL9e7LVDHDLQsRbfDnmvgX/4xQvdwROmyqh8kkcXTf/48zY0wo2n9iDspCbgTn1DEqAqtAlwunIpea8eYA3FQc+pV2p
x77wVP7l+9mTxexzLSmum61wWbqE4ESJn0K37gXY54229ZtCnNSlu9rsvjQ5xmDf1e6MvMLBOblXHIReN2t8IH85VGK7mpi8T7JeKb8rI
G8qDbQ5TD3BmIS1+RspI95FldLCKLH91/KNrxsgPsrC2QgQ==", Content-Type=application/json}
  Payload: {
  "PaymentProductUsed" : "IDEAL",
  "CommonPaymentData" : {
    "GuaranteedAmount" : "10.00",
    "PaymentStatus" : "SettlementCompleted",
    "PaymentId" : "19928",
    "AspspPaymentId" : "0001070883053837",
    "AspspId" : "RABONL2UXXX",
    "DebtorInformation" : {
      "Name" : "Edsger Wybe Dijkstra - Callback",
      "Agent" : "ABNANL2AXXX",
      "Account" : {
        "SchemeName" : "IBAN",
        "Identification" : "NL44RABO0123456789",
        "Currency" : "EUR"
      },
      "ContactDetails" : {
        "FirstName" : "Edsger",
        "LastName" : "Dijkstra",
        "PhoneNumber" : "+31612345678",
        "Email" : "edsger@domain.nl"
      },
    "ShippingAddress" : {
        "FirstName" : "Edsger",
        "LastName" : "Dijkstra",
        "PostCode" : "52066",
        "Country" : "NL"
        },
    "BillingAddress" : {
        "FirstName" : "Edsger",
        "LastName" : "Dijkstra",
        "PostCode" : "52066",
        "Country" : "NL"
        }
    }
  },
  "UseWaitingScreen" : false
}
```

Response:

```
ResponseCode: 204
```

8.2 POST Debtor token

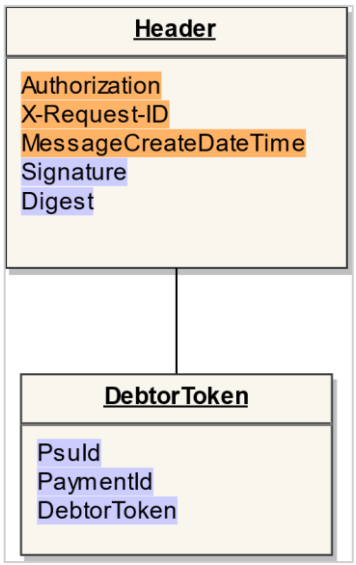
Endpoint: POST /debtorToken

This API will provide a debtor token update to the Initiating party.

- The purple fields are applicable to IDEAL.
- The orange fields are mandatory for an IDEAL payment.

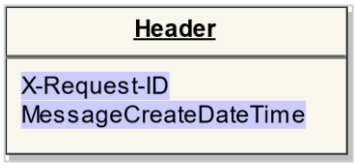
More details about the fields can be found in the yaml specification.

8.2.1 Request



8.2.2 Response

HTML 204 (No Content)



8.2.3 Example: Debtor Token Notification for Standard iDEAL payment

Request (Signature-related fields "Digest" and "Signature" are conditionally present):

```
Address: https://checkout.company.com/transaction/webhook/91FA6EEC30844FAAB5/v3/notification/status
  HttpMethod: POST
  Content-Type: application/json
  Headers: {Authorization=Bearer 123456789, X-Request-ID=c1452392-6c3f-4365-93f8-40558f61ac36,
MessageCreateDateTime=2023-03-15T11:51:24.185+01:00, Digest=SHA-
256=0hq1mKzxB1yyc6+hut2bEX7ps+nWyWb2pgQb6AhfhfM=,
Signature=keyId="3EBEF6033C00730D9C6DA05165A3CAA1F31036FB",algorithm="rsa-
sha256",headers="messagecreatedatetime x-request-id
digest",signature="uYgovoK+ibAE7+MzJEKrApDUAgWfUv7RQK22zAxWHCdKCuG4d0HgqpDSqcGlKmP2IMFsC787zDU3oqKeeIIVXR
72uZBiOnm0/84UL9e7LVDHDLQsRbfDnmvgX/4xQvdwROmyqh8kkcXTf/48zY0wo2n9iDspCbgTn1DEqAqtAlwunIpea8eYA3FQc+pV2px
77wVP7l+9mTxexzLSmum61wWbqE4ESJn0K37gXY54229ZtCnNSlu9rsvjQ5xmDf1e6MvMLBOblXHIReN2t8IH85VGK7mpi8T7JeKb8rIG
8qDbQ5TD3BmIS1+RspI95FldLCKLH91/KNrxsgPsrC2QgQ==", Content-Type=application/json}
  Payload: {
  "PsuId": "TestOSZ",
  "PaymentId": "12345",
  "DebtorToken": "absjrfergd"
 }
```

Response:

```
ResponseCode: 204
```

9 Get Preferences

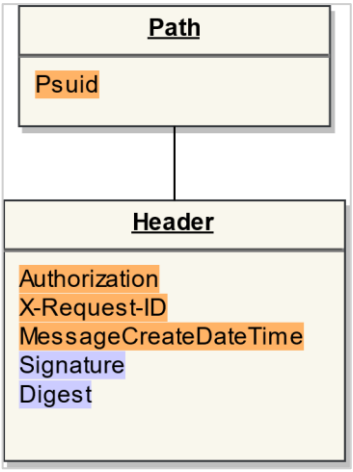
Endpoint: Get /preferences/{Psuid}

This API can be used to retrieve the iDEAL preferences of a PSU. This will only work after an initial payment by this PSU was successfully completed.

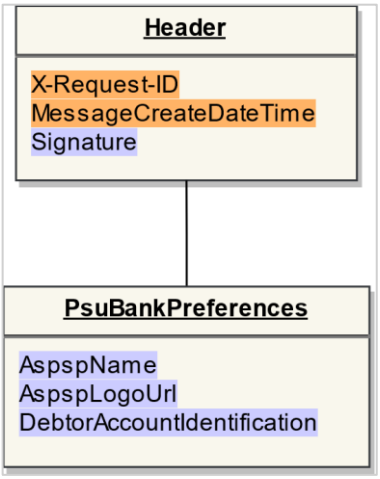
- The purple fields are applicable to IDEAL, those fields will be used towards the IDEAL Hub.
- The orange fields are mandatory for an IDEAL payment.

More details about the fields can be found in the yaml specification.

9.1 Request



9.2 Response



9.3 Example:

Request (Signature-related fields "Digest" and "Signature" are conditionally mandatory):

```
Address: https://localhost:8443/xs2a/routingservice/services/ob/pis/v3/preferences/896587495-51254-85475893254
    HttpMethod: GET
    Headers: {Accept=application/json, AspspId=10002, Digest=SHA-256=47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU=, X-Request-ID=08cc7007-f3c4-6586-4449-8dbf7a92740b, Authorization=Bearer 788b75b7ea27116c74d4a271404c5f49074fbc786ae67cf771a3a44a02a387d8, MessageCreateDateTime=2023-03-15T13:16:40.448Z, Signature=keyId="8D0F688AD3E6C2D4D5FB99FE129F2A2E3B496AF7",algorithm="SHA256withRSA",headers="digest x-request-id messagecreatedatetime (request-target)",signature="hjSmi6wuf/FTDyJ5W7F7QnzDj2X1KVjYuy5hbig+gF6JBhxBfJtZT4utnBgw6OG9dPyqB0EYHPJby786k+q44kl+pWMPJWAJjNHeXyNufNDENF3GGsRsnqBjwrNp9575vsIeimG34A6x47Ap1/01Nmsqk4NzGr0F0HAVyIbeX0Q9oKd6MJHBaswkv9nqaQTajIpmdesAA5D6eNTSfz4oBOCgk2FlJpUAdaAg9Nd1uZuXV+Jeug2coU+jc5FZUQXyZRaV7hiE5UtBIQVxpml6aydFZwCD6OaRJrGvluebVprYc1bqSsiMmS4Pk+qDJshSk5cWWhnp/bVsUFIIK+NU3g=="}
```

Response (Signature-related fields "Digest" and "Signature" are conditionally present):

```
ResponseCode: 200

    Headers: {X-Request-ID=2db64c57-fc84-4998-a83c-aa33ff7fbd9b, MessageCreateDateTime=2023-03-15T13:16:40.898Z, Digest=SHA-256=bd7zfJ14uHY5YwvcXpqr78Df5jUpb0Z64styUET3afI=, Signature=keyId="3EBEF6033C00730D9C6DA05165A3CAA1F31036FB",algorithm="rsa-sha256",headers="messagecreatedatetime x-request-id digest",signature="kBpt9O2Auydz0398VFDf0M9lVWcGVI8CsNJbpvAcXxtmTnlO9cH2MlqZQqlMHGjDqcOMmjGeELaQLAft/R1p8HQoyIMJRxh9PtauyKTnHqLavLj6bJr4BoEUeQpE+xfXaR3tebVWz6zr+c7guHEqDjfNKaoKs5HjQIMm/qjKaptU7zkTCIiFsTWh0tmy/h+biI3MJzxqwpZcGdlOCgJ0LI1squZaQXmgA+AsC+uOomBgyhBMPRxMiFIyYPKnA3Ev+UL7UEm3F49B4d/AWYyO2E6B+9p470eA4Ippc4PHeBGvdNgX6zKERsDhCM8ZTmWlOeBh3i5QWvpJF/5Kv6qYHA==",  Date=Wed, 15 Mar 2023 13:16:40 GMT}
    Payload: {
    "AspspName": "IssuerName",
    "AspspLogoUrl": "https://checkout.company.com/login",
    "DebtorAccountIdentification": "NL44RABO******6789"
}
```