Berlin Group NextGen PSD2

Implementation Guidelines

PSD2 Compliance - Implementation Guidelines

Introduction

In this guide, you will find the necessary documentation to develop your application(s) with our open APIs in accordance with the PSD2 legislation of the international Berlin Group NextGenPSD2 standard, version 1.3.

Account Information Service - AIS

With our PSD2 Account Information Service APIs, you as a TPP will be able to retrieve the account information for your customer's accounts at our bank. Account information is only accessible after the customer has registered his explicit consent. The following information will be available in our APIs:

- List of authorized accounts
- Product information for each authorized account
- Balance information for each authorized account
- Transaction information for each authorized account

In order to register a consent, the customer must provide the account number(s) to the TPP, and the TPP has to send the list of authorized accounts via the consent API (cfr. 6.4.1.1 - Consent Request on Dedicated Accounts). After authentication of the customer, the TPP will receive an OAuth2 access token to retrieve the customer account information.

Note that the process described above is the mandatory process required by the NextGenPSD2 implementation. We support also the other models (cfr 6.4.1.2 - Consent Request on Account List or without Indication of Accounts):

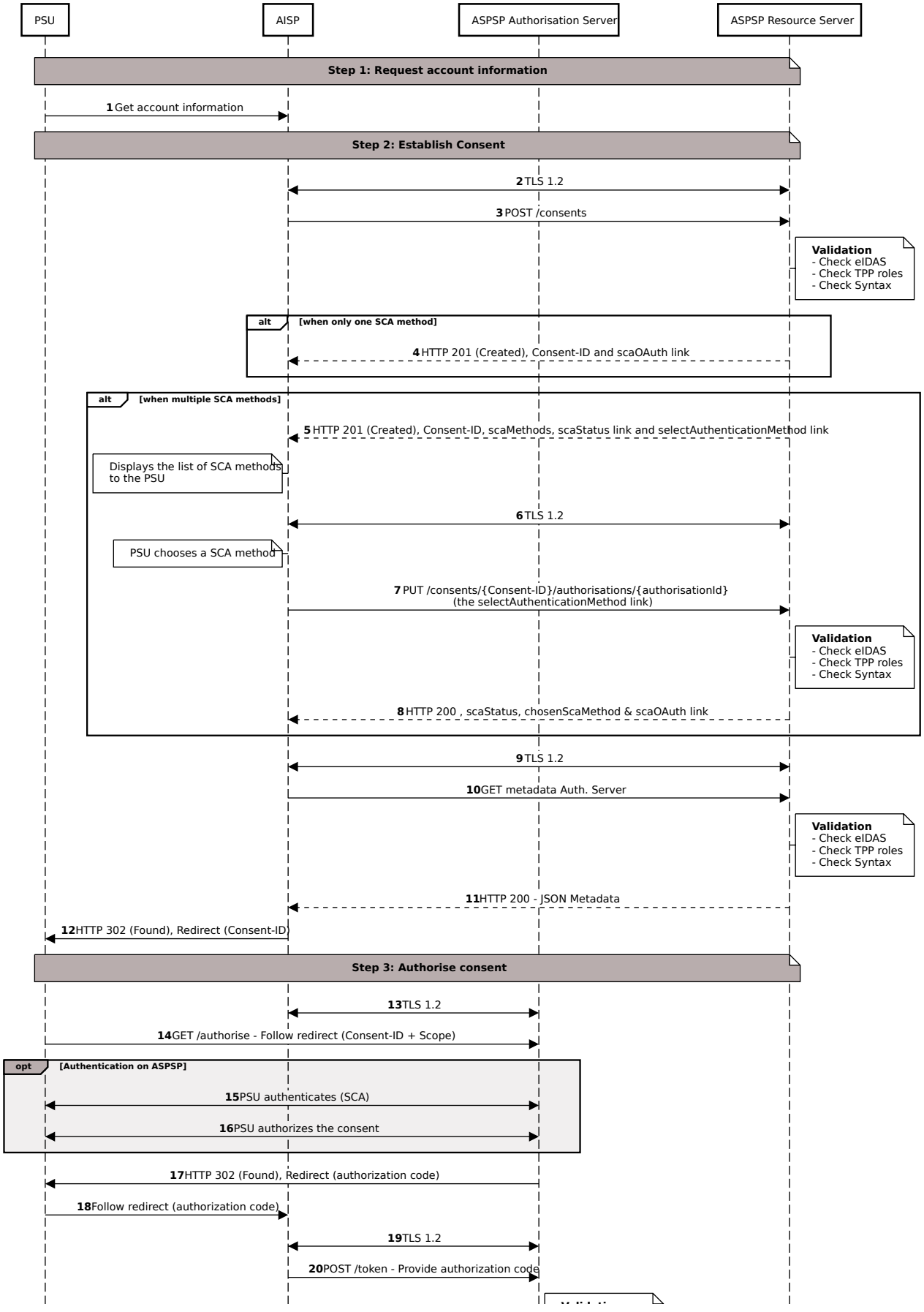
- Consent Request on Account List of Available Accounts
- Consent Request without Indication of Accounts - Bank Offered Consent
- Consent Request for Access to all Accounts for all PSD2 defined AIS - Global Consent

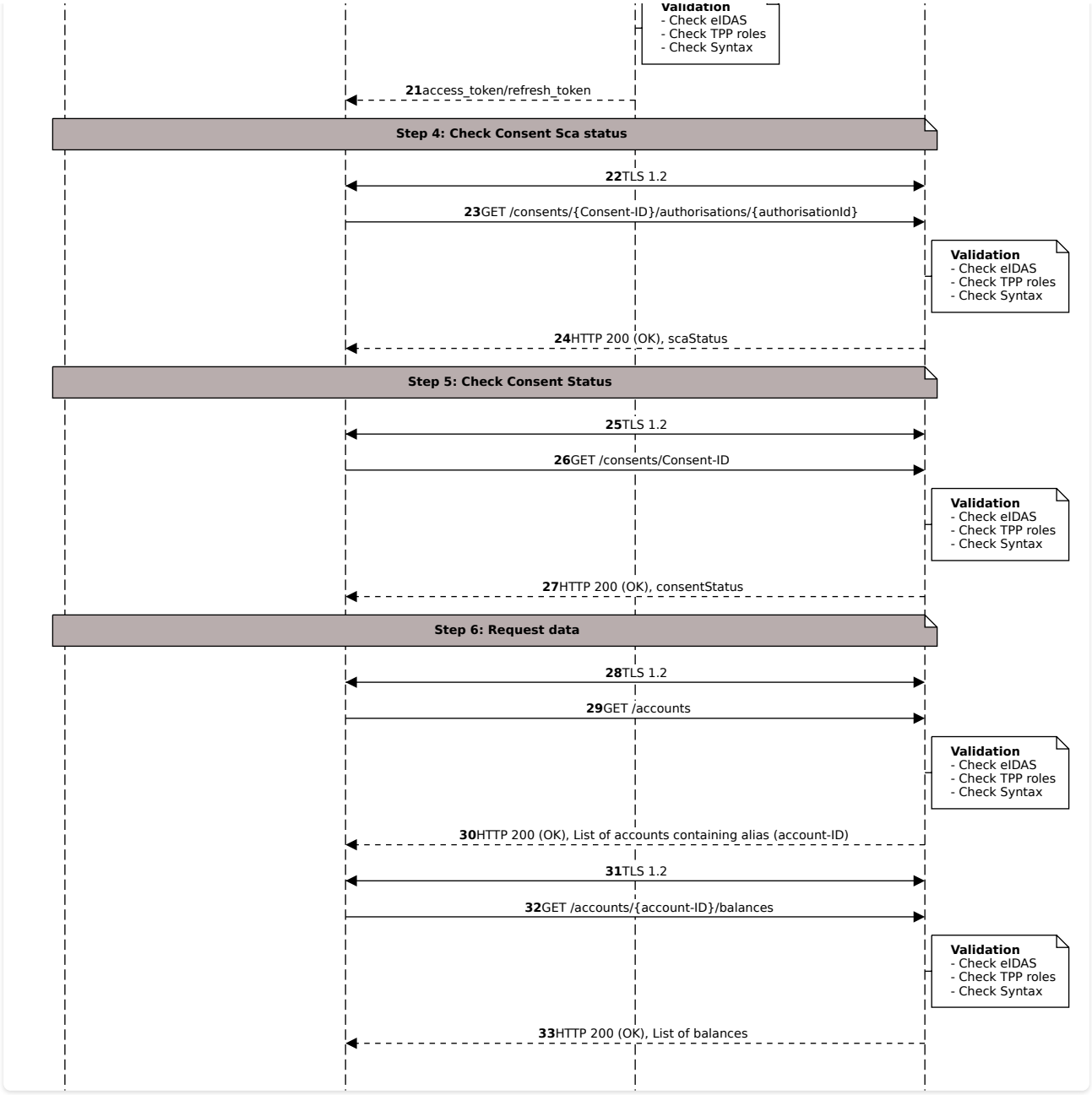
The implementation supports the Article 10 and the Article 36.5(b) related to the Regulatory Technical Standard (RTS).
This means that:

- It is **not** possible to create a consent with a validity date than exceed 90 days after the creation date.
- Only the first JWT (access_token) obtained via the consent authorisation can be used to have more that 90 days of transactions in the past. The others JWT obtained by refresh_token will allow to have only 90 days of transactions.

- Account information can be requested only 4 times in a 24-hour period if the IP-Address of the PSU is not present in the request (PSU-IP-Address). Note that 4 times is the value by default the API allow to specify another value during the intiation of the consent (see frequencyPerDay).

AIS Flow - OAuth2.0 Approach





STEP 1 - Registering the consent

The first step is to register your customers' consent. The following AIS endpoint creates a consent resource, defining access rights to dedicated accounts of a given user identified by PSU-ID:

```
POST /consents
```

> *Requirements:*
> - eIDAS QWAC certificate to allow to authenticate yourself

- apiKey linked to one of your API Portal Applications that contains "* *NextGenPSD2 - Consent Management for Accounts Information* * "
- eiDAS QSealC certificate to allow to sign your requests (see section Signing Request with QSealC - AIS/PIS/CBPII/SBS to build an HTTP signature)

The TPP must call this API to send the consent of the customer (using his eIDAS certificates). Following information is needed to capture a whole consent:

- the access information :
  - A list of authorized accounts in IBAN (or BBAN) format and the requested associated permissions (see Example - Consent Request for dedicated accounts).
    - For access sub attributes with the tags "accounts", "balances" and "transactions" are accepted. At least one tag must be present when it is used
  - **Or** sub attribute empty lists (accounts, balances or transactions) (see Example - Consent Request without indication of accounts - Bank offered consent selection).
  - **Or** requesting permission for all the account in the request using sub attributes "**availableAccounts**", "**availableAccountsWithBalance**" and "**allPsd2**" (see section 6.4.1 of the international Berlin Group NextGenPSD2). Only the value "all-accounts" is allowed to use for them (see Example - Consent Request without indicating accounts).
- the validity date of the consent
- the recurring indicator of the consent
- the daily frequency access.
- the combined service indicator can be sent but the value "true" is not supported and will be ignored by the API.

Note that if a new consent is authorized by the PSU, the old one is automatically cancelled. Only one consent can be active per PSU at any one time. So let's imagine that a PSU has a consent for an account A and wants to add a new account B, you need to create a new consent that will include both account A and B.

Example of a request body with dedicated accounts

```
{
   "validUntil":"2019-12-12",
   "frequencyPerDay":1,
   "recurringIndicator":true,
   "access":{
      "accounts":[
         {
            "iban":"{your Iban #1}"
         },
         {
            "iban":"{your Iban #2}"
         }
      ],
      "balances":[
         {
            "iban":"{your Iban #1}"
         },
         {
            "iban":"{your Iban #2}"
         }
      ],
      "transactions":[
         {
            "iban":"{your Iban #1}"
         },
         {
            "iban":"{your Iban #2}"
         }
      ]
   }
}
```

Example of a request body without indication of accounts

```
{
   "validUntil":"2019-12-12",
   "frequencyPerDay":1,
   "recurringIndicator":true,
   "access":{
      "availableAccountsWithBalance":"all-accounts"
   }
}
```

Example of a request body without indication of accounts - Bank offered consent selection

```
{
   "validUntil":"2019-12-12",
   "frequencyPerDay":1,
   "recurringIndicator":true,
   "access" : {
       "accounts" : [ ],
       "balances" : [ ],
       "transactions" : [ ]
     }
}
```

The resulting response will contain your consentId, consentStatus and the following _links : self, status and scaStatus links. The response may contain another link under the _links attribute which is determined by the number of SCA methods available for the action of consent initiation :

- Link **scaOAuth** when a single SCA method is available.
- Link **selectAuthenticationMethod** when multiple SCA methods are available for initiation of a consent

In case of availability of multiple SCA methods, the response also contains an array object called **scaMethods**.

The response structure and the links can be summarised as below :

Response Structure

Name	Type	Required	Description
`consentStatus`	`ConsentStatus`	*Mandatory*	Status of the consent.
`consentId`	`String`	*Mandatory*	Identification of the consent resource as it is used in the API structure. Shall be contained, if a consent resource was generated.
`scaMethods`	`Array of ScaMethod`	*Conditional*	This data element might be contained, if SCA is required and if the PSU has a choice between different authentication methods. If this data element is contained, then there is also an hyperlink of type "selectAuthenticationMethods" contained in the response body. These methods shall be presented towards the PSU for selection by the TPP.

`_links`	`List of hyperlink`	*Mandatory*	A list of hyperlinks to be recognised by the TPP.

_links

Name	Type	Required	Description
`scaOAuth`	`String`	*Conditional*	In case of a SCA OAuth2 Approach, this is the link where the configuration of the Authorisation Server can be retrieved. **This link is contained only when a single SCA method is available**
`self`	`String`	*Mandatory*	The link to the consent initiation resource created by this request. This link can be used to retrieve the resource data.
`status`	`String`	*Mandatory*	The link to retrieve the transaction status of the consent initiation.
`scaStatus`	`String`	*Conditional*	The link to retrieve the scaStatus of the corresponding authorisation sub-resource. **This link is only contained, if an authorisation sub-resource has been already created.** (cfr. implicit mode)
`selectAuthenticationMethod`	`String`	*Conditional*	This is a link to a resource, where the TPP can select the applicable strong customer authentication methods for the PSU. **This link is contained only in case multiple SCA methods are available.**

Example of a consent initiation response when single SCA method

```
{
   "consentId":"5d403bd17020cc0001b2ab28",
   "consentStatus":"received",
   "_links":{
      "self":{
         "href":"/v1/consents/5d403bd17020cc0001b2ab28"
      },
      "scaOAuth":{
         "href":"https://<HOST>/.well-known/oauth-authorization-server"
      },
      "status":{
         "href":"/v1/consents/5d403bd17020cc0001b2ab28/status"
      }
   }
}
```

Example of a consent initiation response when multiple SCA method

```
{
    "consentId":"5c5c4098eb964600015ba988",
    "consentStatus":"received",
    "scaMethods":
     [
      {
          "authenticationType":"TYPE_1",
          "authenticationMethodId":"myAuthenticationID_1",
          "name":"Account Authentication",
          "explanation":"This method correspond to the TYPE_1"
      },
      {
          "authenticationType":"TYPE_2",
          "authenticationMethodId":"myAuthenticationID_2",
          "name":"Account Authentication",
          "explanation":"This method correspond to the TYPE_2"
       }
      ],
    "_links":
        {
          "self":
          {
              "href":"/v1/consents/5c5c4098eb964600015ba988"
          },
          "status":
          {
              "href":"/v1/consents/5c5c4098eb964600015ba988/status"
          },
          "scaStatus":
           {
              "href":"/v1/consents/5c5c4098eb964600015ba988/authorisations/9x3DmFS2flQj7
           },
           "selectAuthenticationMethod":
           {
              "href":"/v1/consents/5c5c4098eb964600015ba988/authorisations/9x3DmFS2flQj7
           }
        }
}
```

> **Notes:** All "href" are **relative** links. To use it you need to add the **provider** in front of it (see section 4.4).
> The provider in our case is always "HOST + /berlingroup".
> So with last example above, the complete link to use should be like that :
>
> ```
> https://host.aspsp.com/berlingroup/v1/consents/5c5c4098eb964600015ba988
> ```

STEP 2 - SCA Selection (Conditional)

In case of multiple SCA methods during consent initiation, the TPP should update the authorisation sub-resource with the **authenticationMethodId** of the SCA method selected by the PSU.

```
PUT /consents/{consentId}/authorisations/{authorisationId}
```

This URI is coming from the "_links" included in the POST /consents response by the name **selectAuthenticationMethod**.

> *Requirements:*
> - eIDAS QWAC certificate to allow to authenticate yourself
> - apiKey linked to one of your API Portal Applications that contains "NextGenPSD2 - Consent Management for Accounts Information"
> - eiDAS QSealC certificate to allow to sign your requests (see section below to build an HTTP signature)

Example of a request body

```
{
  "authenticationMethodId" : "myAuthenticationID_1"
}
```

The resulting response will contain the scaStatus, the chosenScaMethod and the scaOAuth "_links" :

Example of the response body

```
{
  "scaStatus" : "scaMethodSelected",
  "chosenScaMethod" : {
    "authenticationType" : "TYPE_1",
    "authenticationMethodId" : "myAuthenticationID"
  },
  "_links" : {
    "scaOAuth" : {
      "href" : "https://<HOST>/.well-known/oauth-authorization-server"
    }
  }
}
```

STEP 3 - Retrieve the SCA status (Optional)

The TPP has a provision to check the status of an authorisation sub-resource for a consent that is initiated and for which an authorisation has already been created.

```
GET /consents/{consentId}/authorisations/{authorisationId}
```

This URI is coming from the "_links" included in the POST /consents response by the name **scaStatus**.

> *Requirements:*
> - eIDAS QWAC certificate to allow to authenticate yourself
> - apiKey linked to one of your API Portal Applications that contains "NextGenPSD2 - Consent Management for Accounts Information"
> - eiDAS QSealC certificate to allow to sign your requests (see section below to build an HTTP signature)

Example of the response body

```
{
  "scaStatus" : "scaMethodSelected"
}
```

The possible values of the SCA status as per the international Berlin Group NextGenPSD2 standard, version 1.3 are:

- **'received'**: An authorisation or cancellation-authorisation resource has been created successfully.
- **'scaMethodSelected'**: The PSU/TPP has selected the related SCA routine. If the SCA method is chosen implicitly since only one SCA method is available, then this is the first status to be reported instead of 'received'

- **'finalised'**: The SCA routine has been finalised successfully
- **'failed'**: The SCA routine failed

STEP 4 - Customer Authorization/Authentication

The second step is to redirect your customer to the OAuth2 Authorization Server.

As a reminder, only the OAuth2 redirect approach is supported for now. The TPP must redirect the customer to our Authorization Server to perform a Strong Customer Authentication (SCA) and authorize the consent. After authorization, the customer is redirected back to the TPP with an authorization code. The TPP can use this code to retrieve an access token and a refresh token by API call. The TPP must first call the metadata URL obtained in the response of the previous step. The response will contain a json according to ***RFC 8414*** that lists description of the properties related to the Authentication Server (endpoints, scope, parameters).

```
GET /authorise
```

> *Requirements :*
> - Your client_id as a TPP corresponds to the Organization Identifier as requested by ETSI.
> - Your call-back URL where the client will be redirected to after performing SCA and authorizing the consent request.
> - The consentId obtained from the response body of the previous step.
> - code_challenge_method/code_challenge: code_challenge is the PKCE challenge according to cryptographic RFC 7636 used to prevent code injection attacks. See our API portal documentation or the NextGenPSD2 documentation for more information about these fields and how to use them.

As reminder, only the redirection is supported for now. The TPP must redirect the customer to our Authorisation Server to make a SCA and authorise the consent. After authorisation, the customer is redirected to the TPP with an authorisation code. The TPP can use this code to retrieve an access_token and/or a refresh_token.

Example of redirection URL below:

```
https://portal.test-psd2.aspsp.be/psd2/v1/berlingroup-auth/authorise?
response_type=code&
client_id=PSDBE-AIS-123456&
redirect_uri=https://azure.microsoft.com/fr-fr/overview/what-is-azure/&
code_challenge=oFTpT3qj8p4tRalSoYFBCAm_VkYY--XCIP3uh_Kf2ro&
scope=AIS:<consentId>&
state=test&
code_challenge_method=S256
```

The response to this URL will be the call-back URL with an authorization_code. You have to exchange this code via a call to the POST /token endpoint.

```
POST /token
```

> *Requirements :*
> - eIDAS QWAC certificate to allow to authenticate yourself
> - Your client_id as a TPP. Same as the previous step
> - Your call-back URL (exactly the same)
> - The grant_type with the value "authorization_code"
> - The authorization_code from the previous step.
> - code_verifier, see API Documentation for this field

Example of exchange code below:

```
POST https://portal.test-psd2.aspsp.be/psd2/v1/berlingroup-auth/token

client_id=PSDBE-AIS-123456&grant_type=authorization_code&
code=<code>&redirect_uri=https%3A%2F%2Fazure.
microsoft.com%2Ffr-fr%2Foverview%2Fwhat-is-azure%2F&
code_verifier=cGFzc3dvcmQuLy4
```

In response, you will receive an access_token and a refresh_token. The refresh_token can be used to fetch a new access_token.

Example of exchange token below:

```
POST https://portal.test-psd2.aspsp.be/psd2/v1/berlingroup-auth/token

grant_type=refresh_token&refresh_token=JWT
```

In response, you will receive the new access_token.

STEP 5 - Providing alias of authorized accounts

```
GET /accounts
```

> *Requirements :*
> - eIDAS QWAC certificate to allow to authenticate yourself
> - apiKey linked to the an Application that contains "NextGenPSD2 - Account Information"
> - eIDAS QSealC certificate to allow to sign our requests (see section below to build an HTTP signature)
> - Authorization Header with the access_token obtained in the authorisation step.

The TPP can use the access_token obtained during the customer authorisation step to retrieve the list of alias linked to all authorised accounts.

STEP 6 - Requesting account information

```
GET /accounts/{account-ID}/balances
```

```
GET /accounts/{account-ID}/transactions
```

Requirements :

- eIDAS QWAC certificate to allow to authenticate yourself
- apiKey linked to the an Application that contains "NextGenPSD2 - Account Information"
- eIDAS QSealC certificate to allow to sign our requests (see section below to build an HTTP signature)
- Authorization Header with the access_token obtained in the authorisation step.
- Alias 'account-ID' obtained in the previous step.

Example of account balance information


```
{
    "balances":
    [
        {
            "balanceAmount":
            {
                "amount":"7487.50",
                "currency":"EUR"
            },
            "balanceType":"interimAvailable",
            "referenceDate":"2019-02-07"
        },
        {
            "balanceAmount":
            {
                "amount":"7729.00",
                "currency":"EUR"
            },
            "balanceType":"interimBooked",
            "referenceDate":"2019-02-07"
        }
    ]
}
```


Example of account transaction information


```
{
  "transactions" : {
    "booked" : [ {
```

```
    "transactionAmount" : {
      "amount" : "-5343.21",
      "currency" : "EUR"
    },
    "creditorAccount" : {
      "iban" : "BE50999090049618"
    },
    "remittanceInformationUnstructured" : "JITKAZGKSRRDSNNHAOEN",
    "remittanceInformationStructured" : "",
    "creditorName" : "RLPBADOUNJXTXMIEHKIB",
    "proprietaryBankTransactionCode" : "8R5699",
    "entryReference" : "B7H31CWDOP4K1B09",
    "_links" : {
      "href" : ""
    }
  }, {
    "transactionAmount" : {
      "amount" : "5843.46",
      "currency" : "EUR"
    },
    "debtorAccount" : {
      "iban" : "BE50999090049618"
    },
    "remittanceInformationUnstructured" : "",
    "remittanceInformationStructured" : "VCUTZERMRSQPNGSMWCAG",
    "debtorName" : "ENZCASJWXSNWOOHRSGIT",
    "proprietaryBankTransactionCode" : "JF",
    "entryReference" : "B7H31CWDOQ7K1CLR",
    "_links" : {
      "href" : ""
    }
  }, {
    "transactionAmount" : {
      "amount" : "9637.31",
      "currency" : "EUR"
    },
    "debtorAccount" : {
      "iban" : "BE50999090049618"
    },
    "remittanceInformationUnstructured" : "PSQDNJPUDAUHVLAVXZIH",
    "remittanceInformationStructured" : "",
    "debtorName" : "KMUJGMCVHOEWCCCPDGLO",
    "proprietaryBankTransactionCode" : "P7MT9W",
    "entryReference" : "B7H31CWDOS4K1F13",
    "_links" : {
      "href" : ""
    }
  } ],
  "_links" : {
    "first" : {
      "href" : "/v1/accounts/5b111695400e574a608a9cbb/transactions?range=0-2"
    },
    "next" : {
      "href" : "/v1/accounts/5b111695400e574a608a9cbb/transactions?range=3-5"
     },
    "account" : {
      "href" : "/v1/accounts/5b111695400e574a608a9cbb"
    },
    "self" : {
      "href" : "/v1/accounts/5b111695400e574a608a9cbb/transactions?range=0-2"
    }
```

```
    }
  }
}
```

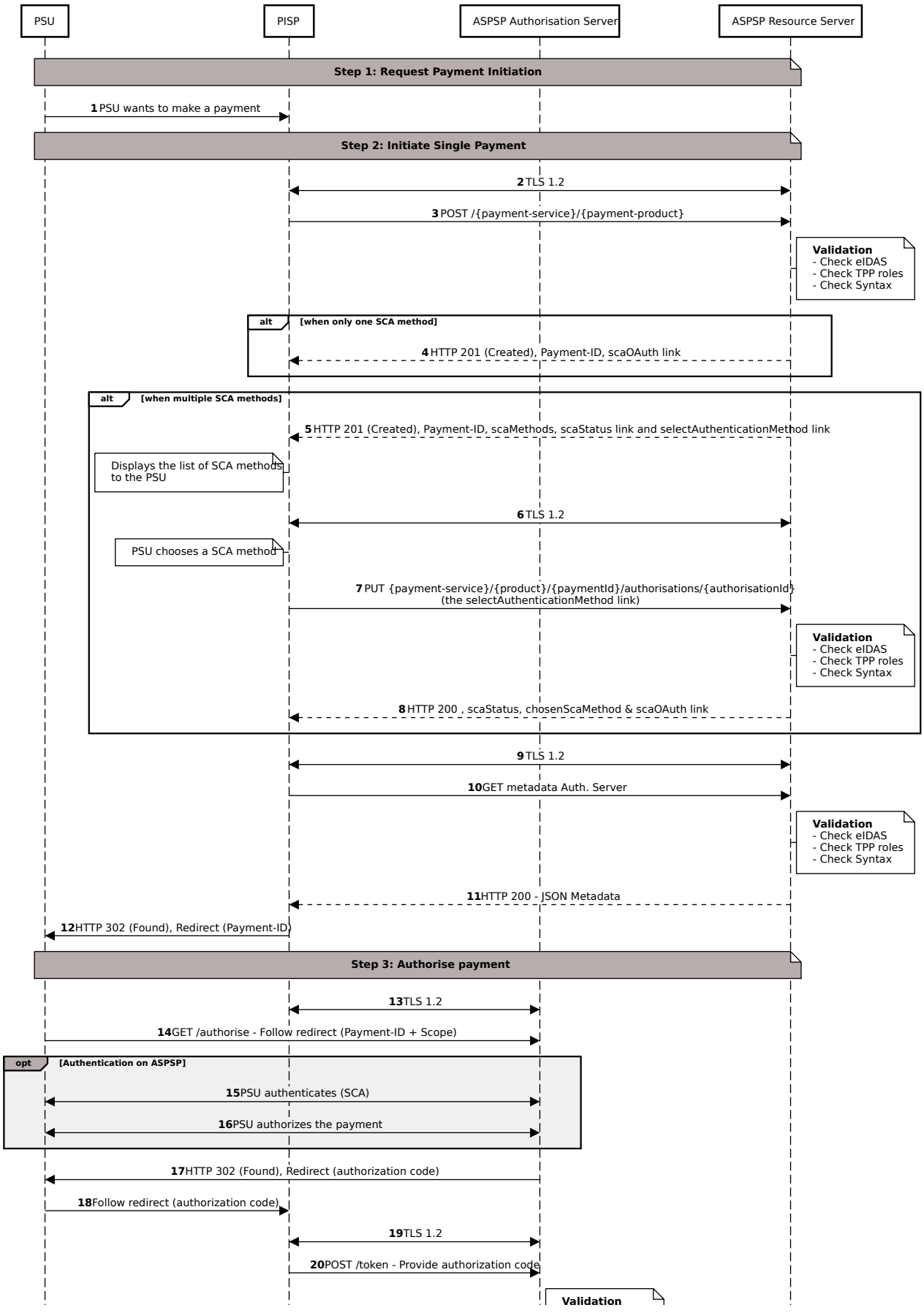
STEP 7 - Customer authorisation expired

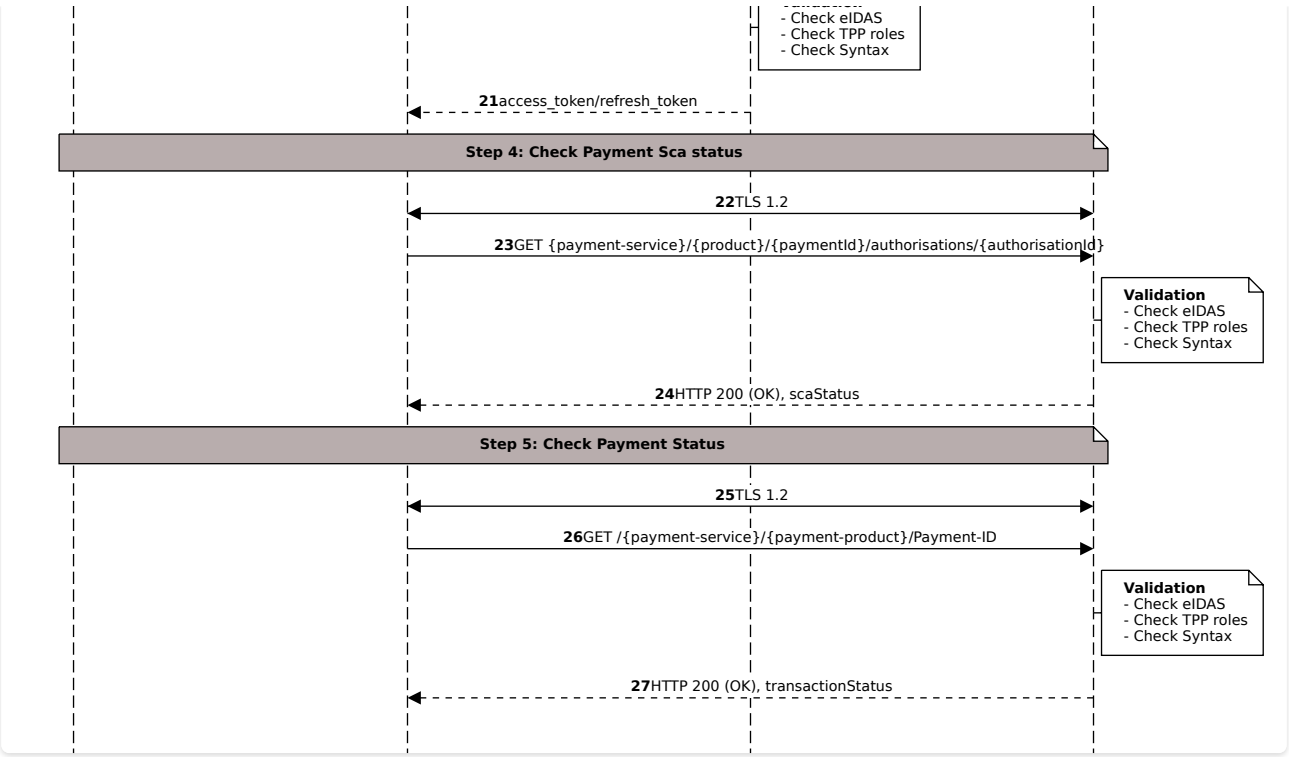
If the access_token is expired, the TPP can use the refresh_token to ask a new one. If refresh_token is expired **or** the validity date of the consent is reached the TPP has to restart the flow and ask a new consent.

Payment Initiation Service - PIS

Our PSD2 APIs enable the TPP to initiate a payment (sepa-credit-transfers, services-purchases-payments, top-up-payments, state-payments, social-security-payments). In addition to initiation, the TPP can also retrieve status, or details of a specific payment. In this sequence diagram, the {payment-service} should be "payments".

PIS Flow - OAuth2.0 Approach





STEP 1 - Initiating a payment

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

The TPP can make different kinds of payments based on "payment-service" and the "payment-product". The "payment-service" should be **payments**. The "payment-product" currently supported are sepa-credit-transfers, services-purchases-payments, top-up-payments, state-payments, social-security-payments.

```
POST /payments/{payment-product}
```

> *Requirements:*
> - eIDAS QWAC certificate to allow to authenticate yourself
> - apiKey linked one of your API Portal Applications that contains "NextGenPSD2 - Payment Initiation "
> - eiDAS QSealC certificate to allow to sign your requests (see section below to build an HTTP signature)

The TPP must call this API to initiate a payment for the customer (using eIDAS certificates). Following information are mandatory :

- the debtor and creditor accounts
- the amount (minimum of 0,01 for services-purchases-payments and state-payments)

- the name of the creditor and the country code

Example of the request body:

```
{
    "endToEndIdentification": "END-TO-END-ID",
    "instructedAmount": {
        "currency": "EUR",
        "amount": "10.50"
    },
    "debtorAccount": {
        "iban": "{your Debtor Iban}"
    },
    "creditorAddress":{
           "country":"BE"
    },
    "creditorName" : "Dupont",
    "creditorAccount": {
        "iban": "{your Creditor Iban}"
    },
    "remittanceInformationUnstructured": "Free Communication"
}
```

Example of the request body with bban:

```
{
    "endToEndIdentification": "END-TO-END-ID",
    "instructedAmount": {
        "currency": "EUR",
        "amount": "10.50"
    },
    "debtorAccount": {
        "iban": "{your Debtor Iban}"
    },
    "creditorAddress":{
           "country":"BE"
    },
    "creditorName" : "Dupont",
    "creditorAccount": {
        "bban": "{your Creditor bban}"
    },
    "remittanceInformationUnstructured": "Free Communication"
}
```

Example of the request body for services purchases payments:

```
{
    "endToEndIdentification": "END-TO-END-ID",
    "instructedAmount": {
        "currency": "EUR",
        "amount": "12.50"
    },
    "debtorAccount": {
        "iban": "{your Debtor Iban}"
    },
    "creditorAddress":{
           "country":"BE"
    },
    "creditorName" : "Dupont",
    "creditorAccount": {
        "bban": "{your purchase reference}"
    },
    "remittanceInformationUnstructured": "76089124H;1234"
}
```

Example of the request body for state payments:

```
{
    "endToEndIdentification": "END-TO-END-ID",
    "instructedAmount": {
        "currency": "EUR",
        "amount": "12.50"
    },
    "debtorAccount": {
        "iban": "{your Debtor Iban}"
    },
    "creditorAddress":{
           "country":"BE"
    },
    "creditorName" : "Dupont",
    "creditorAccount": {
        "bban": "{your payment reference}"
    },
    "remittanceInformationUnstructured": "76089124H"
}
```

Example of the request body for social security payments:

```
{
    "endToEndIdentification": "END-TO-END-ID",
    "instructedAmount": {
        "currency": "EUR",
        "amount": "12.50"
    },
    "debtorAccount": {
        "iban": "{your Debtor Iban}"
    },
    "creditorAddress":{
           "country":"BE"
    },
    "creditorName" : "Dupont",
    "creditorAccount": {
        "bban": "{your social security number}"
    },
    "remittanceInformationUnstructured": "76089124H;08;2019;030;03;100;3"
}
```

Example of the request body for top up payments:

```
{
    "endToEndIdentification": "END-TO-END-ID",
    "instructedAmount": {
        "currency": "EUR",
        "amount": "12.50"
    },
    "debtorAccount": {
        "iban": "{your Debtor Iban}"
    },
    "creditorAddress":{
           "country":"BE"
    },
    "creditorName" : "Dupont",
    "creditorAccount": {
        "bban": "{your recharge reference}"
    },
    "remittanceInformationUnstructured": "76089124H;20839"
}
```

The resulting response will contain your paymentId and the following "_links" : self, status and scaStatus links. The response may contain another link under the "_links" attribute which is determined by the number of SCA methods available for the action of payment initiation :

- Link **scaOAuth** when a single SCA method is available.
- Link **selectAuthenticationMethod** when multiple SCA methods are available for initiation of a consent

In case of availability of multiple SCA methods, the response also contains an array object called the **scaMethods**.

The response structure and the links can be summarised as below :

Response Structure

Name	Type	Required	Description
`transactionStatus`	`Transaction Status`	*Mandatory*	Transaction status defined in ISO20022 format.
`resourceId`	`String`	*Mandatory*	Resource identification of the generated payment initiation resource.
`transactionFeeIndicator`	`boolean`	*Mandatory*	If equals 'true', the transaction will involve specific transaction cost as shown by the ASPSP in their public price list or as agreed between ASPSP and PSU. If equals 'false', the transaction will not involve additional specific transaction costs to the PSU.
`transactionFees`	`Amount`	*Conditional*	Can be used by the ASPSP to transport transaction fees relevant for the underlying payments.
`scaMethods`	`Array of ScaMethod`	*Conditional*	This data element might be contained, if SCA is required and if the PSU has a choice between different authentication methods. If this data element is contained, then there is also a hyperlink of type "selectAuthenticationMethods" contained in the response body. These methods shall be presented to the PSU for selection by the TPP.
`_links`	`Links`	*Mandatory*	A list of hyperlinks to be recognised by the **TPP** : - "self": The link to retrieve the transaction status of the payment initiation. - "scaOAuth": In case of a SCA OAuth2 Approach, the **ASPSP** is transmitting the URI where

			the configuration of the Authorisation Server can be retrieved. The configuration follows the OAuth 2.0 Authorisation Server Metadata specification.

TransactionStatus

Code	Name	ISO 20022 Definition
`ACSC`	`AcceptedSettlementCompleted`	Settlement on the debtor's account has been completed. Usage:this can be used by first agent to report to debtor that the transaction has been completed. Warning: this status is provided for transaction status reason, not for financial information. It may be used after bilateral agreement.
`ACSP`	`AcceptedSettlementInProcess`	All preceding checks such as technical validation and customer profile were successful and therefore the payment initiation has been accepted for execution.
`ACTC`	`AcceptedTechnicalValidation`	Authentication and syntactical and semantical validation are successful.
`RCVD`	`Received`	Payment initiation has been received by receiving agent.
`PDNG`	`Pending`	Payment initiation or individual transaction included in the payment initiation is pending. Further checks and status update will be performed.
`RJCT`	`Rejected`	Payment initiation or individual transaction included in the payment initiation has been rejected.
`CANC`	`Cancelled`	Payment initiation has been cancelled before execution. Remark: Change Request to ISO20022 is still needed.

Amount

Name	Type	Required	Description

currency	Currency Code	*Mandatory*	ISO 4217 code.
amount	String	*Mandatory*	The amount given with fractional digits, where fractions must be compliant to the currency definition. Up to 14 significant figures. Negative amounts are signed by minus.The decimal separator is a dot.

_links

Name	Type	Required	Description
scaOAuth	String	*Conditional*	In case of a SCA OAuth2 Approach, this is the link where the configuration of the Authorisation Server can be retrieved. **This link is contained only when a single SCA method is available**
self	String	*Mandatory*	The link to the consent initiation resource created by this request. This link can be used to retrieve the resource data.
status	String	*Mandatory*	The link to retrieve the transaction status of the consent initiation.
scaStatus	String	*Conditional*	The link to retrieve the scaStatus of the corresponding authorisation sub-resource. **This link is only contained, if an authorisation sub-resource has been already created.**
selectAuthenticationMethod	String	*Conditional*	This is a link to a resource, where the TPP can select the applicable strong customer authentication methods for the PSU. **This link is contained only in case multiple SCA methods are available.**

Example of a payment initiation response when single SCA method


```
{
    "transactionStatus":"RCVD",
    "resourceId":"5c5c370afc5c0500011bd0c2",
    "transactionFeeIndicator":true,
    "transactionFees":{
            "currency": "EUR",
            "amount": "10.50"
    },
    "_links":
    {
        "scaOAuth":
        {
            "href":"https://<HOST>/.well-known/oauth-authorization-server"
        },
        "scaStatus": {
            "href": "/v1/payments/sepa-credit-transfer/5c5c370afc5c0500011bd0c2/authorisa
        },
        "self":
        {
            "href":"/v1/payments/sepa-credit-transfers/5c5c370afc5c0500011bd0c2"
        },
        "status": {
            "href": "/v1/payments/sepa-credit-transfers/5c5c370afc5c0500011bd0c2/status"
        }
    }
}
```


Example of a payment initiation response when multiple SCA methods

```
{
    "transactionStatus":"RCVD",
    "resourceId":"5c5c370afc5c0500011bd0c2",
    "transactionFeeIndicator":true,
    "transactionFees":{
            "currency": "EUR",
            "amount": "10.50"
    },
    "scaMethods":[
      {
         "authenticationType":"M1",
         "authenticationMethodId":"myAuthenticationID_1",
         "name":"Card Reader Authentication",
         "explanation":"This method used in UCR..."
      },
      {
         "authenticationType":"platform-v3",
         "authenticationMethodId":"myAuthenticationID_2",
         "name":"Card Reader Authentication",
         "explanation":"This method used in UCR..."
      }
   ],
    "_links": {
        "scaStatus": {
            "href": "/v1/payments/sepa-credit-transfers/5c5c370afc5c0500011bd0c2/authoris
        },
        "selectAuthenticationMethod": {
            "href": "/v1/payments/sepa-credit-transfers/5c5c370afc5c0500011bd0c2/authoris
        },
        "self": {
            "href": "/v1/payments/sepa-credit-transfers/5c5c370afc5c0500011bd0c2"
        },
        "status": {
            "href": "/v1/payments/sepa-credit-transfers/5c5c370afc5c0500011bd0c2/status"
        }
    }
}
```


STEP 2 - SCA Selection (Conditional)

In case of multiple SCA methods during payment initiation the TPP should update the authorisation sub-resource with the **authenticationMethodId** of the SCA selected by the PSU.

```
PUT {payment-service}/{payment-product}/{paymentId}/authorisations/{authorisationId}
```

This URI is the same link that is received in the POST /payments/sepa-credit-transfer response by the name **selectAuthenticationMethod** when multiple SCA methods are available

> *Requirements:*
> - eIDAS QWAC certificate to allow to authenticate yourself

- apiKey linked to one of your API Portal Applications that contains "NextGenPSD2 - Payment Initiation"
- eiDAS QSealC certificate to allow to sign your requests (see section below to build an HTTP signature)

Example of the request body

```
{
  "authenticationMethodId" : "myAuthenticationID"
}
```

The resulting response will contain the scaStatus, the chosenScaMethod and the scaOAuth link.

Example of the response body

```
{
  "scaStatus" : "scaMethodSelected",
  "chosenScaMethod" : {
    "authenticationType" : "TYPE_1",
    "authenticationMethodId" : "myAuthenticationID"
  },
  "_links" : {
    "scaOAuth" : {
      "href" : "https://<HOST>/.well-known/oauth-authorization-server"
    }
  }
}
```

STEP 3 - Retrieve the SCA status (Optional)

The TPP has a provision to check the status of an authorisation sub-resource for a payment for which an authorisation has already been created.

```
GET {payment-service}/{payment-product}/{paymentId}/authorisations/{authorisationId}
```

This URI is available in the POST /payments/sepa-credit-transfers response by the name **scaStatus**.

Requirements:
- eIDAS QWAC certificate to allow to authenticate yourself
- apiKey linked to one of your API Portal Applications that contains "NextGenPSD2 - Payment Initiation"
- eiDAS QSealC certificate to allow to sign your requests (see section below to build an HTTP signature)

Example of the response body

```
{
  "scaStatus" : "scaMethodSelected"
}
```

The possible values of the SCA status as per the international Berlin Group NextGenPSD2 standard, version 1.3 are:

- **'received'**: An authorisation or cancellation-authorisation resource has been created successfully.
- **'scaMethodSelected'**: The PSU/TPP has selected the related SCA routine. If the SCA method is chosen implicitly since only one SCA method is available, then this is the first status to be reported instead of 'received'
- **'finalised'**: The SCA routine has been finalised successfully
- **'failed'**: The SCA routine failed

STEP 4 - Customer Authorisation/Authentication

This steps are exactly the same steps than the Account Information with the only change is that the resourceId is not a consentId but a paymentId. The TPP must redirect the customer to our Authorisation Server to make a Strong Customer Authentication and authorise the payment. After authorisation, the customer is redirected to the TPP with an authorisation code. The TPP can use this code to retrieve an access_token.

```
GET /authorise
```

```
POST /token
```

STEP 5 - Retrieving Payment with or without Status (Optional)

Once a payment has been submitted, the TPP may want to retrieve the payment's status or the payment details using the access_token

```
GET /{payment-service}/{payment-product}/{payment-ID}/status
```

```
GET /{payment-service}/{payment-product}/{payment-ID}
```

Requirements :

- eIDAS QWAC certificate to allow to authenticate yourself
- apiKey linked to the an Application that contains "NextGenPSD2 - Payment Initiation"
- eIDAS QSealC certificate to allow to sign our requests (see section below to build an HTTP signature)
- Authorization Header with the access_token obtained in the authorisation step (not mandatory)

Example of a payment status

```
{
   "transactionStatus":"ACSP"
}
```

Example of payment detail when {payment-service} is "payment"

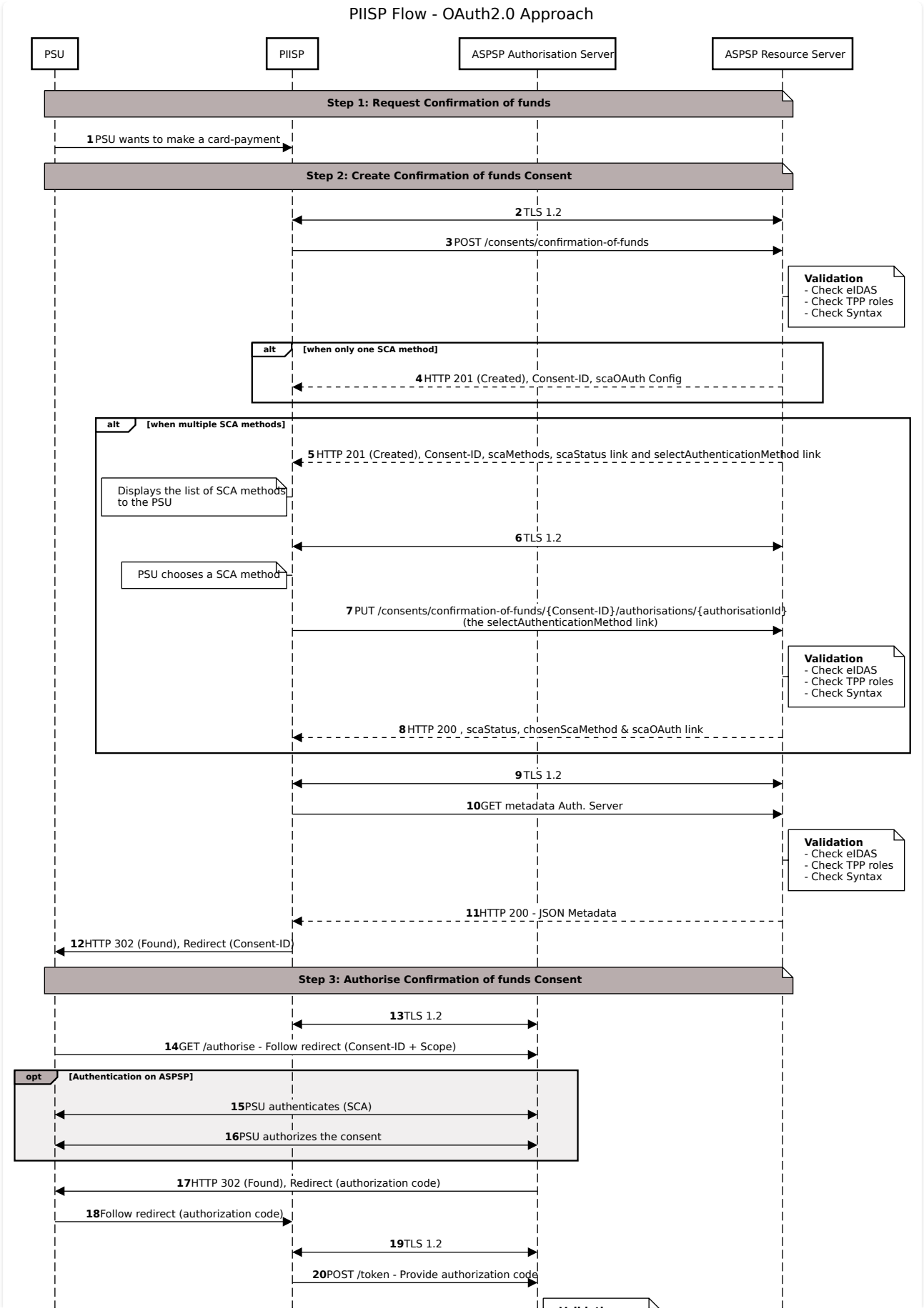
```
{
    "transactionStatus":"ACSP",
    "endToEndIdentification":"EndToEndIdentification",
    "debtorAccount":
    {
        "iban":"FR7630004015240000096421772",
        "currency":"EUR"
    },
    "instructedAmount":
    {
        "amount":"100.00",
        "currency":"EUR"
    },
    "creditorAccount":
    {
        "iban":"FR7630006000011234567890189",
        "currency":"EUR"
    },
    "creditorName":"nom1",
    "requestedExecutionDate":"2018-09-09",
    "remittanceInformationStructured":
    {
        "reference":"000000000101"
    }
}
```

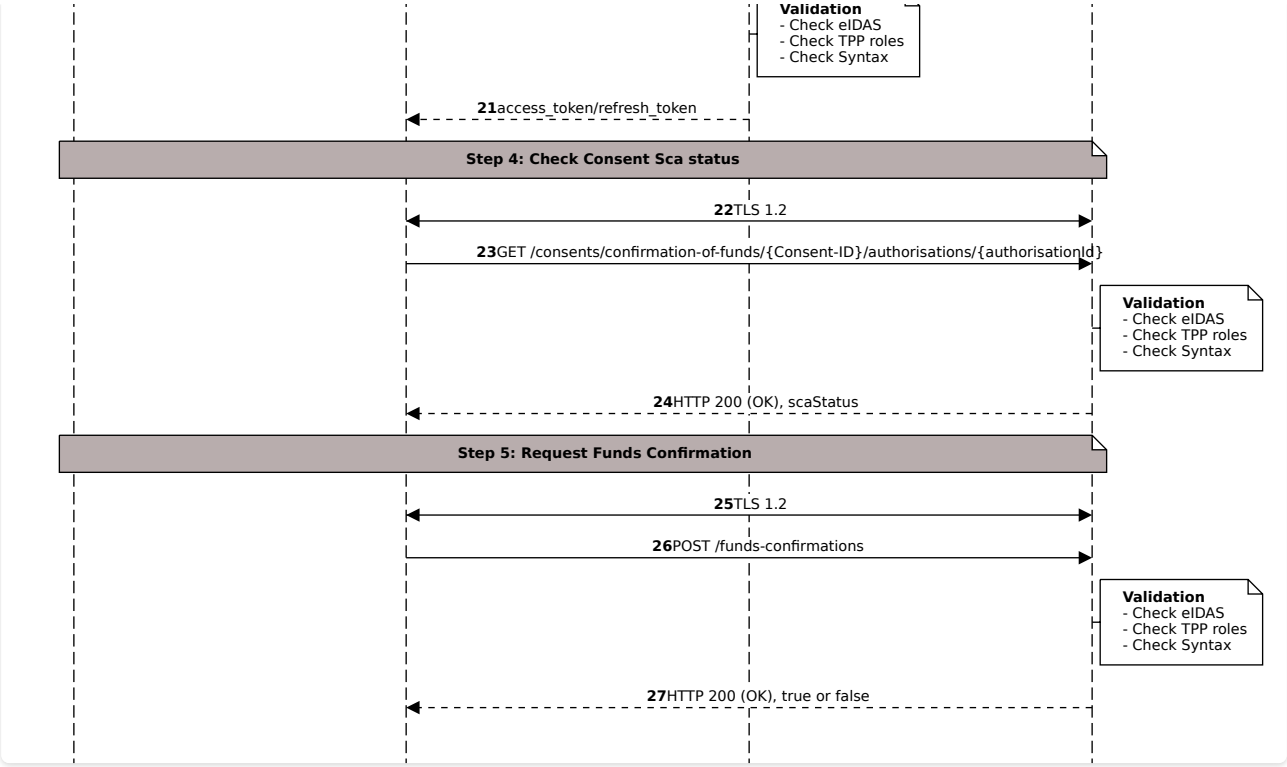
Funds Confirmation Service - CBPII/PIIS

With our PSD2 Confirmation of Funds Service APIs you as a TPP will have the possibility to request a confirmation of funds for your customers accounts at our bank. Confirmation of

Funds is only accessible after the customer has registered his explicit consent for a confirmation of funds.

In order to register a confirmation of funds consent, the customer must provide the account number (in IBAN format) to the TPP, and the TPP has to send this account via the consent API (cfr. NextGenPSD2 standard: "Consent Request on Dedicated Accounts"). After authentication of the customer, the TPP will receive an OAuth2 access token to make confirmation of funds requests.


PIISP Flow - OAuth2.0 Approach
PSU
PIISP
ASPSP Authorisation Server
ASPSP Resource Server
Step 1: Request Confirmation of funds
1 PSU wants to make a card-payment
Step 2: Create Confirmation of funds Consent
2 TLS 1.2
3 POST /consents/confirmation-of-funds
Validation
- Check eIDAS
- Check TPP roles
- Check Syntax
alt
[when only one SCA method]
4 HTTP 201 (Created), Consent-ID, scaOAuth Config
alt
[when multiple SCA methods]
5 HTTP 201 (Created), Consent-ID, scaMethods, scaStatus link and selectAuthenticationMethod link
Displays the list of SCA methods to the PSU
6 TLS 1.2
PSU chooses a SCA method
7 PUT /consents/confirmation-of-funds/{Consent-ID}/authorisations/{authorisationId} (the selectAuthenticationMethod link)
Validation
- Check eIDAS
- Check TPP roles
- Check Syntax
8 HTTP 200 , scaStatus, chosenScaMethod & scaOAuth link
9 TLS 1.2
10 GET metadata Auth. Server
Validation
- Check eIDAS
- Check TPP roles
- Check Syntax
11 HTTP 200 - JSON Metadata
12 HTTP 302 (Found), Redirect (Consent-ID)
Step 3: Authorise Confirmation of funds Consent
13 TLS 1.2
14 GET /authorise - Follow redirect (Consent-ID + Scope)
opt
[Authentication on ASPSP]
15 PSU authenticates (SCA)
16 PSU authorizes the consent
17 HTTP 302 (Found), Redirect (authorization code)
18 Follow redirect (authorization code)
19 TLS 1.2
20 POST /token - Provide authorization code



STEP 1 - Registering the confirmation of funds consent

The first step is to register your customers' consent by using the following CBPII endpoint:

```
POST /consents/confirmation-of-funds**
```

> *Requirements:
> - eIDAS QWAC certificate to allow to authenticate yourself
> - apiKey linked one of your API Portal Applications that contains "NextGenPSD2 - Consent Management for Confirmation of Funds"
> - eiDAS QSealC certificate to allow to sign your requests (see Signing Request with QSealC - AIS/PIS/CBPII/SBS section below to build an HTTP signature)

The TPP must call this API to send the confirmation of funds consent of the customer (using eIDAS certificates). Following information is needed to capture a whole consent:

- the authorized account in IBAN format
- the validity date of the consent

Example of the request body:

```
{
  "cardNumber": "{cardnumber}",
  "account":{
      "iban": "{Your iban}"
  },
  "validUntil": "2019-12-01"
}
```

The resulting response will contain your consentId and the following "_links" : self, status and scaStatus links. The response may contain another link under the "_links" attribute which is determined by the number of SCA methods available for the action of consent initiation :

- Link **scaOAuth** when a single SCA method is available.
- Link **selectAuthenticationMethod** when multiple SCA methods are available for initiation of a consent

In case of availability of multiple SCA methods, the response also contains an array object called the **scaMethods**.

The response structure and the links can be summarised as below :

Response Structure

Name	Type	Required	Description
`consentStatus`	`ConsentStatus`	*Mandatory*	Status of the consent.
`consentId`	`String`	*Mandatory*	Identification of the consent resource as it is used in the API structure. Shall be contained, if a consent resource was generated.
`scaMethods`	`Array of ScaMethod`	*Conditional*	This data element might be contained, if SCA is required and if the PSU has a choice between different authentication methods. If this data element is contained, then there is also an hyperlink of type "selectAuthenticationMethods" contained in the response body. These methods shall be presented towards the PSU for selection by the TPP.
`_links`	`List of hyperlink`	*Mandatory*	A list of hyperlinks to be recognised by the TPP.

_links

Name	Type	Required	Description
`scaOAuth`	`String`	*Conditional*	In case of a SCA OAuth2 Approach, this is the link where the configuration of the Authorisation Server can be retrieved. **This link is contained only when a single SCA method is available**
`self`	`String`	*Mandatory*	The link to the consent initiation resource created by this request. This link can be used to retrieve the resource data.
`status`	`String`	*Mandatory*	The link to retrieve the transaction status of the consent initiation.
`scaStatus`	`String`	*Conditional*	The link to retrieve the scaStatus of the corresponding authorisation sub-resource. **This link is only contained, if an authorisation sub-resource has been already created.**
`selectAuthenticationMethod`	`String`	*Conditional*	This is a link to a resource, where the TPP can select the applicable strong customer authentication methods for the PSU. **This link is contained only in case multiple SCA methods are available.**

**_Example of a confirmation-of-funds consent initiation response when

single SCA method_**

```
{
   "consentId":"5c5c534feb964600015ba991",
   "consentStatus":"received",
   "_links": {
            "scaStatus": {
                "href": "/v1/consents/confirmation-of-funds/5c5c534feb964600015ba991/auth
            },
            "scaOAuth": {
                "href": "https://<HOST>/.well-known/oauth-authorization-server"
            },
            "self": {
                "href": "/v1/consents/confirmation-of-funds/5c5c534feb964600015ba991"
            },
            "status": {
                "href": "/v1/consents/confirmation-of-funds/5c5c534feb964600015ba991/statu
            }
        }
}
```


**_Example of a confirmation-of-funds consent initiation response when

multiple SCA methods_**


```
{
   "consentId":"5c5c534feb964600015ba991",
   "consentStatus":"received",
   "_links": {
            "scaStatus": {
                "href": "/v1/consents/confirmation-of-funds/5c5c534feb964600015ba991/auth
            },
            "selectAuthenticationMethod": {
                "href": "/v1/consents/confirmation-of-funds/5b2a57af999caafd37ab1fe7/auth
            },
            "self": {
                "href": "/v1/consents/confirmation-of-funds/5c5c534feb964600015ba991"
            },
            "status": {
                "href": "/v1/consents/confirmation-of-funds/5c5c534feb964600015ba991/statu
            }
   }
}
```


STEP 2 - SCA Selection (Conditional)

In case of multiple SCA methods during confirmation-of-funds consent initiation the TPP should update the confirmation-of-funds consent sub-resource with the **authenticationMethodId** of the SCA selected by the PSU.

```
PUT /consents/confirmation-of-funds/{consentId}/authorisations/{authorisationId}**
```

This URI is the same link that is received in the POST /consents/confirmation-of-funds response by the name **selectAuthenticationMethod** when multiple SCA methods are available

Requirements:

- eIDAS QWAC certificate to allow to authenticate yourself
- apiKey linked to one of your API Portal Applications that contains "NextGenPSD2 - Consent Management for Confirmation of Funds"
- eiDAS QSealC certificate to allow to sign your requests (see section below to build an HTTP signature)

Example of the request body

```
{
  "authenticationMethodId" : "myAuthenticationID"
}
```

The resulting response will contain the scaStatus, chosenScaMethod object and the scaOAuth link

Example of the response body

```
{
  "scaStatus" : "scaMethodSelected",
  "chosenScaMethod" : {
    "authenticationType" : "TYPE_1",
    "authenticationMethodId" : "myAuthenticationID"
  },
  "_links" : {
    "scaOAuth" : {
      "href" : "https://<HOST>/.well-known/oauth-authorization-server"
    }
  }
}
```

STEP 3 - Retrieve SCA status (Optional)

The TPP has a provision to check the status of an authorisation sub-resource for a consent that is initiated and for which an authorisation has already been created.

```
GET /consents/{consentId}/authorisations/{authorisationId}**
```

This URI is available in the POST /consents response by the name **scaStatus**.

Requirements:

- eIDAS QWAC certificate to allow to authenticate yourself
- apiKey linked to one of your API Portal Applications that contains "NextGenPSD2 - Consent Management for Confirmation of Funds"

- eiDAS QSealC certificate to allow to sign your requests (see section below to build an HTTP signature)

Example of the response body

```
{
  "scaStatus" : "scaMethodSelected"
}
```

The possible values of the SCA status as per the international Berlin Group NextGenPSD2 standard, version 1.3 are:

- **'received'**: An authorisation or cancellation-authorisation resource has been created successfully.
- **'scaMethodSelected'**: The PSU/TPP has selected the related SCA routine. If the SCA method is chosen implicitly since only one SCA method is available, then this is the first status to be reported instead of 'received'
- **'finalised'**: The SCA routine has been finalised successfully
- **'failed'**: The SCA routine failed

STEP 4 - Customer Authorisation/Authentication

This steps are exactly the same steps than the Account Information. The TPP must redirect the customer to our Authorisation Server to make a Strong Customer Authentication and authorise the confirmation of funds consent. After authorisation, the customer is redirected to the TPP with an authorisation code. The TPP can use this code to retrieve an access_token and a refresh_token.

GET /authorise

```
POST /token
```

STEP 5 - Request confirmation of funds

The TPP must use the access_token obtained during the customer authorisation step to request the confirmation of funds (with the authorized account and an amount).

```
POST /funds-confirmation
```

Requirements:

- eIDAS QWAC certificate to allow to authenticate yourself
- apiKey linked one of your API Portal Applications that contains "NextGenPSD2 - Confirmation of funds"
- eiDAS QSealC certificate to allow to sign your requests (see section below to build an HTTP signature)

The TPP must call this API to send a request of confirmation of funds (using eIDAS certificates). Following information is needed to capture a whole consent:

- the authorized account in IBAN format (same than in the consent)
- the amount and the currency

Example of the request body:

```
{
 "cardNumber": "{cardnumber}",
  "account":{
      "iban":"{Your iban}"
  },
  "instructedAmount": {
      "amount" : "100.00",
      "currency" : "EUR"
  }
}
```

STEP 6 - Customer authorisation expired

If the access_token is expired, the TPP can use the refresh_token to ask a new one. If refresh_token is expired or the validity date of the consent is reached the TPP has to restart the flow and ask a new consent.

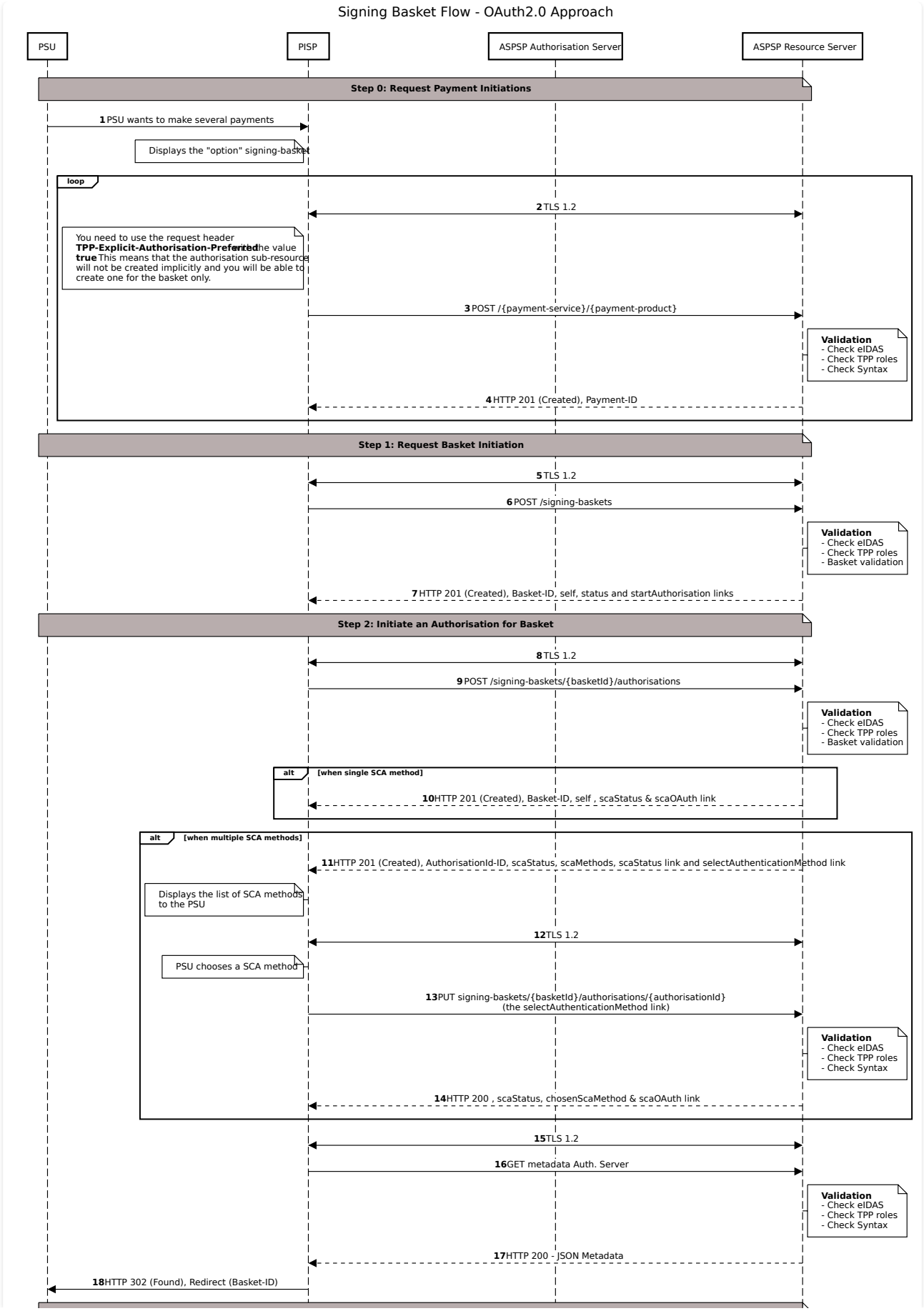
Signing Basket Service - SBM

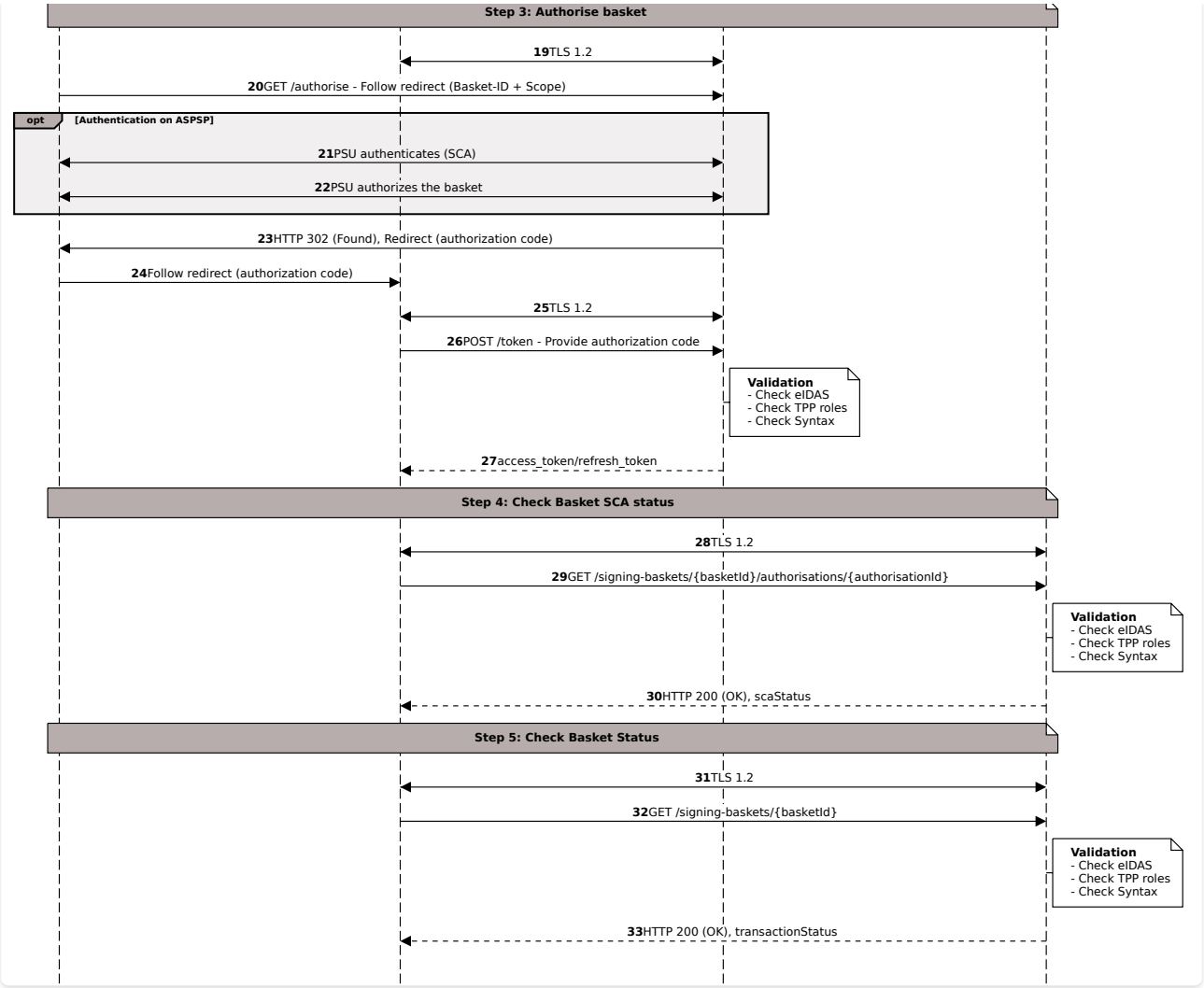
With our PSD2 APIs you as a TPP will have the possibility to sign several resources (only payments for now) one time (only one SCA will be required for all resources).

The TPP can propose to the customer to initiate several payments, when all required payments are initiated, the TPP will have the possibility to group them and use our API to create a basket that will include all the payment resources.
Once the basket resource is created, the TPP will receive the resource identifier and as it is done fot the authorisation of a consent or a payment, the TPP must redirect the customer to

the authentication server to perform a SCA.
The cancellation of a basket is also managed if no authorisations have already been initiated.

To be able to put a payment in the signing basket, the payment must be created with a new header **TPP-Explicit-Authorisation-Preferred**. Indeed, this means that the payment will not be signed with an "implicit" authorisation.
The authorisation will have to be create by the TPP by using the endpoint "POST /authorisations".


Signing Basket Flow - OAuth2.0 Approach
PSU
PISP
ASPSP Authorisation Server
ASPSP Resource Server
Step 0: Request Payment Initiations
1 PSU wants to make several payments
Displays the "option" signing-basket
loop
2 TLS 1.2
You need to use the request header TPP-Explicit-Authorisation-Preferred with the value true. This means that the authorisation sub-resource will not be created implicitly and you will be able to create one for the basket only.
3 POST /{payment-service}/{payment-product}
Validation
- Check eIDAS
- Check TPP roles
- Check Syntax
4 HTTP 201 (Created), Payment-ID
Step 1: Request Basket Initiation
5 TLS 1.2
6 POST /signing-baskets
Validation
- Check eIDAS
- Check TPP roles
- Basket validation
7 HTTP 201 (Created), Basket-ID, self, status and startAuthorisation links
Step 2: Initiate an Authorisation for Basket
8 TLS 1.2
9 POST /signing-baskets/{basketId}/authorisations
Validation
- Check eIDAS
- Check TPP roles
- Basket validation
alt [when single SCA method]
10 HTTP 201 (Created), Basket-ID, self , scaStatus & scaOAuth link
alt [when multiple SCA methods]
11 HTTP 201 (Created), AuthorisationId-ID, scaStatus, scaMethods, scaStatus link and selectAuthenticationMethod link
Displays the list of SCA methods to the PSU
12 TLS 1.2
PSU chooses a SCA method
13 PUT signing-baskets/{basketId}/authorisations/{authorisationId} (the selectAuthenticationMethod link)
Validation
- Check eIDAS
- Check TPP roles
- Check Syntax
14 HTTP 200 , scaStatus, chosenScaMethod & scaOAuth link
15 TLS 1.2
16 GET metadata Auth. Server
Validation
- Check eIDAS
- Check TPP roles
- Check Syntax
17 HTTP 200 - JSON Metadata
18 HTTP 302 (Found), Redirect (Basket-ID)



STEP 1 - Initiating a basket

The first step is to initiate the basket with a group of payment's unique identifier. The payments eligible for signing-basket should be initiated with the header **tpp-explicit-authorisation-preferred** as true to indicate the explicit authorisation of that payment. If this header is not present or set to false then that payment would be considered for implicit authorisation and would not be part of a basket. The following endpoint creates a basket resource for authorising several payments with one SCA.

```
POST /signing-baskets
```

Requirements:

- eIDAS QWAC certificate to allow to authenticate yourself
- apiKey linked to the an Application that contains "NextGenPSD2 - Signing Basket Service "

- eiDAS QSealC certificate to allow to sign your requests (see section Signing Request with QSealC - AIS/PIS/CBPII/SBS to build an HTTP signature)

The TPP must call this API to initiate the basket (using eIDAS certificates). Please note that the only resources authorized are the payment resources.
Following information is mandatory:

- paymentIds - the list of resource ids to be authorised

Example of the request body:

```
{
   "paymentIds":[
      "123qwert456789",
      "12345qwert7899"
   ]
}
```

The resulting response will contain the transactionStatus, basketId and the following "_links" : self, startAuthorisation and status.

The response structure and the links can be summarised as below :

Response Structure

Name	Type	Required	Description
`transactionStatus`	`TransactionStatus`	*Mandatory*	Transaction status defined in ISO20022 format.
`basketId`	`String`	*Mandatory*	Resource identification of the generated payment initiation resource.
`_links`	`Links`	*Mandatory*	A list of hyperlinks to be recognised by the **TPP**.

TransactionStatus

Code	Name	ISO 20022 Definition
`ACTC`	`AcceptedTechnicalValidation`	Authentication and syntactical and semantical validation are successful.

RCVD	Received	Payment initiation has been received by receiving agent.
RJCT	Rejected	Payment initiation or individual transaction included in the payment initiation has been rejected.
CANC	Cancelled	The basket resource has been cancelled before authorisation.

Links

Name	Description
self	The link to the basket initiation resource created by this request. This link can be used to retrieve the resource data
startAuthorisation	The link to use to create the sub-resource authorisation.
status	The link to retrieve the status of the transaction resource.

Example of a basket initiation response:


```
{
    "transactionStatus":"RCVD",
    "basketId":"5c5c370afc5c0500011bd0c2",
    "_links":
    {
        "startAuthorisation": {
            "href": "/v1/signing-baskets/5c5c370afc5c0500011bd0c2/authorisations"
        },
        "self":
        {
            "href":"/v1/signing-baskets/5c5c370afc5c0500011bd0c2"
        },
        "status": {
            "href": "/v1/signing-baskets/5c5c370afc5c0500011bd0c2/status"
        }
    }
}
```


STEP 2 - Initiating an authorisation for basket

This endpoint creates an authorisation sub-resource and allow to start the authorisation process of a basket.

```
POST /signing-baskets/{basketId}/authorisations
```

Requirements:

- eIDAS QWAC certificate to allow to authenticate yourself
- apiKey linked to the an Application that contains "NextGenPSD2 - Signing Basket Service"
- eiDAS QSealC certificate to allow to sign your requests (see section Signing Request with QSealC - AIS/PIS/CBPII/SBS to build an HTTP signature)

Response Structure

Name	Type	Required	Description
`scaStatus`	`String`	*Mandatory*	Status of the authorisation.
`authorisationId`	`String`	*Mandatory*	ID of the corresponding authorisation.
`scaMethods`	`Array of ScaMethod`	*Conditional*	This data element might be contained, if SCA is required and if the PSU has a choice between different authentication methods. If this data element is contained, then there is also a hyperlink of type "selectAuthenticationMethods" contained in the response body. These methods shall be presented to the PSU for selection by the TPP.
`_links`	`Links`	*Optional*	A list of hyperlinks to be recognised by the **TPP**.

Links

Name	Description
`scaStatus`	The link to retrieve the scaStatus of the corresponding authorisation sub-resource.
`scaOAuth`	In case of a SCA OAuth2 Approach, this is the link where the configuration of the Authorisation Server can be retrieved. **This link is contained only when a single SCA method is available**

Example of a basket authorisation initiation response

```
{
  "authorisationId": "5e85662602c0190001bad81f",
  "scaStatus":"received",
      "scaMethods":[
        {
           "authenticationType":"TYPE_1",
           "authenticationMethodId":"myAuthenticationID_1",
           "name":"Card Reader Authentication",
           "explanation":"This method correspond to the TYPE_1"
        },
        {
           "authenticationType":"TYPE_2",
           "authenticationMethodId":"myAuthenticationID_2",
           "name":"Card Reader Authentication",
           "explanation":"This method correspond to the TYPE_2"
        }
     ],
      "_links": {
          "scaStatus": {
              "href": "\/v1\/signing-baskets\/5e856624816cec000149de68\/authorisations\/5
          },
          "selectAuthenticationMethod": {
              "href": "\/v1\/signing-baskets\/5e856624816cec000149de68\/authorisations\/5
          }
      }
}
```


STEP 3 - SCA Selection (Conditional)

In case of multiple SCA methods during basket initiation, the TPP should update the authorisation sub-resource with the **authenticationMethodId** of the SCA method selected by the PSU.

```
PUT /signing-baskets/{basketId}/authorisations/{authorisationId}
```

This URI is coming from the "_links" included in the POST /signing-baskets response by the name **startAuthorisation**.

> *Requirements:*
> - eIDAS QWAC certificate to allow to authenticate yourself
> - apiKey linked to one of your API Portal Applications that contains "NextGenPSD2 - Consent Management for Accounts Information"
> - eiDAS QSealC certificate to allow to sign your requests (see section below to build an HTTP signature)

Example of a request body

```
{
  "authenticationMethodId" : "myAuthenticationID_1"
}
```

The resulting response will contain the scaStatus, the chosenScaMethod and the scaOAuth "_links" :

Example of the response body

```
{
  "scaStatus" : "scaMethodSelected",
  "chosenScaMethod" : {
    "authenticationType" : "TYPE_1",
    "authenticationMethodId" : "myAuthenticationID"
  },
  "_links" : {
    "scaOAuth" : {
      "href" : "https://<HOST>/.well-known/oauth-authorization-server"
    }
  }
}
```

STEP 4 - Retrieve SCA status (Optional)

The TPP has a provision to check the status of an authorisation sub-resource for a basket that is initiated and for which an authorisation has already been created.

```
GET /signing-baskets/{basketId}/authorisations/{authorisationId}
```

This URI is available in the POST /signing-baskets/{basketId}/authorisations response by the name **scaStatus**.

> *Requirements:*
> - eIDAS QWAC certificate to allow to authenticate yourself
> - apiKey linked to the an Application that contains "NextGenPSD2 - Signing Basket Service"
> - eiDAS QSealC certificate to allow to sign your requests (see section Signing Request with QSealC - AIS/PIS/CBPII/SBS to build an HTTP signature)

Example of the response body

```
{
  "scaStatus" : "scaMethodSelected"
}
```

The possible values of the SCA status as per the international Berlin Group NextGenPSD2 standard, version 1.3 are:

- **'scaMethodSelected'**: The PSU/TPP has selected the related SCA routine. If the SCA method is chosen implicitly since only one SCA method is available, then this is the first status to be reported instead of 'received'
- **'finalised'**: The SCA routine has been finalised successfully
- **'failed'**: The SCA routine failed

STEP 5 - Customer Authorisation/Authentication

This steps are exactly the same steps as for the Account Information with the only change of the resourceId which in this case is a basketId. The TPP must redirect the customer to our Authorisation Server to make a Strong Customer Authentication and authorise the basket. After authorisation, the customer is redirected to the TPP with an authorisation code. The TPP can use this code to retrieve an access_token.

```
GET /authorise
```

```
POST /token
```

STEP 6 - Retrieving a Basket with or without Status (Optional)

Once a Basket has been submitted, the TPP may want to retrieve the basket's status or the basket details using the access_token

```
GET /signing-baskets/{basketId}
```

```
GET /signing-baskets/{basketId}/status
```

> *Requirements:*
> - eIDAS QWAC certificate to allow to authenticate yourself
> - apiKey linked to the an Application that contains "NextGenPSD2 - Signing Basket Service"
> - eiDAS QSealC certificate to allow to sign your requests (see section Signing Request with QSealC - AIS/PIS/CBPII/SBS to build an HTTP signature)

Example of a basket detail

```
{
   "payments":[
      "5dc40c408bdeea00014df65c",
      "5dc40c418bdeea00014df65d",
      "5dc40c428bdeea00014df65e"
   ],
   "transactionStatus":"ACTC",
    "_links":
    {
        "self": {
            "href":"/v1/signing-baskets/5dc40c4991af8c00012dd294"
        },
        "status": {
            "href": "/v1/signing-baskets/5dc40c4991af8c00012dd294/status"
        }
    }
}
```


Example of a basket status


```
{
   "transactionStatus":"ACTC"
}
```


STEP 7 - Cancelling a Basket (Optional)

This endpoint allows the TPP to cancel a basket by deleting a basket object (with the resourceId).

The response for a basket cancellation request will be HTTP 204 (NO CONTENT) provided the basket is in a cancellable state.

```
DELETE /signing-baskets/{basketId}
```

Note: The signing basket as such is not deletable after an authorisation has been applied. Nevertheless, single transactions might be cancelled on an individual basis.

> *Requirements:*
> - eIDAS QWAC certificate to allow to authenticate yourself
> - apiKey linked to the an Application that contains "NextGenPSD2 - Signing Basket Service"
> - eiDAS QSealC certificate to allow to sign your requests (see section Signing Request with QSealC - AIS/PIS/CBPII/SBS to build an HTTP signature)

Signing Request with QSealC - AIS/PIS/CBPII/SBS

With our PSD2 APIs the TPP needs to sign the request and send the tpp-certificate in header, which will ensure the data integrity of the request.

As part of signature model, TPP needs to send 3 specific headers:

- **Digest**: It contains a Hash of the message body. SHA-256 and SHA-512 are the supported hash algorithms that can be used to calculate the Digest.

  Digest can be computed by taking the byte array of json body and then creating a message digest of it using SHA-256 or SHA-512 algorithm, then convert in Base64 encoding value.

  Note: that for the GET and DELETE endpoints, the digest will be equals to a byte array of 0 size.

- **Tpp-Signature-Certificate**: The certificate used for signing the request, in pem encoding. The certificate can be put with or without the "BEGIN" and "END".

- **Signature**: A signature of the request by the TPP on application level. Signature can be computed by concatenated key=value of following sub values : keyId, algorithm, headers (digest and x-request-id), signature.

Elements of the "Signature" Header

Element	Type	Required	Description
`keyId`	`String`	*Mandatory*	Serial Number of the TPP's certificate included in the "TPP-Signature-Certificate" header of the request. It shall be formatted as follows: keyId="SN=XXX,CA=YYYYYYYYYYYYYYY" where "XXX" is the serial number of the certificate in hexadecimal coding given in the "TPP-Signature-Certificate" Header and "YYYYYYYYYYYYYYY" is the full Distinguished Name of the Certification Authority having produced this certificate. Please respect the format defined in RFC2253 for SN and CA.
`algorithm`	`String`	*Mandatory*	The algorithm must identify the same algorithm for the signature as presented in the certificate (Element "TPP-Signature-Certificate") of this Request. It must identify SHA-256 or SHA-512 as Hash algorithm.
`headers`	`String`	*Mandatory*	Must include "digest" header and "x-request-id" header. **No other entries may be included.**

`signature`	`String`	*Mandatory*	The "signature" parameter is a base64 encoded digital signature, as described in RFC 4648, Section 4. The client uses the "algorithm" and "headers" signature parameters to form a canonicalised "signing string". This "signing string" is then signed with the key associated with "keyId" and the algorithm corresponding to "algorithm". The "signature" parameter is then set to the base64 encoding of the signature.

signature = sign(canonical form of headers list (key: value))) with algorithm specified in algorithm header and with TPP private key. For example

```
digest: SHA-256=D8XFTnfEij+pTZ0zt088csVDVEvqytJjxkbpOom8RiU=
x-request-id: 99391c7e-ad88-49ec-a2ad-99ddcb1f7721)
```

Note: Private key used for signing the request shouldn't contain any new line and it should be used as one single string.

Example of create consent in AIS without signed request:

```
POST https://<HOST>/berlingroup/v1/consents HTTP/1.1
Connection: keep-alive
x-request-id: 4e73b197-fa18-4d29-9c81-00538d4088ca
apiKey: 3d6e4c19-dca6-4c85-a507-3b9f98be9111
content-type: application/json
psu-ip-address: 192.168.0.6
Content-Length: 359
Host: <HOST>
User-Agent: Apache-HttpClient/4.5.7 (Java/1.8.0_161)
```

```
{
  "validUntil" : "2019-08-02",
  "frequencyPerDay" : 1,
  "recurringIndicator" : false,
  "access" : {
    "accounts" : [ {
      "iban" : "BE63999001487608"
    } ],
    "balances" : [ {
      "iban" : "BE63999001487608"
    } ],
    "transactions" : [ {
      "iban" : "BE63999001487608"
    } ]
  }
}
```

Example of create consent in AIS with signed request:

```
POST https://<HOST>/berlingroup/v1/consents HTTP/1.1
Connection: keep-alive
x-request-id: 4e73b197-fa18-4d29-9c81-00538d4088ca
apiKey: 3d6e4c19-dca6-4c85-a507-3b9f98be9111
content-type: application/json
digest: SHA-256=z18RGk+oEX/b32Kkkh3tNMD4KvKh0hnnUinyNQRlLvY=
signature: keyId="SN=4145E06EFDA52DC2,CA=EMAILADDRESS=rootCA@root.com, OU=DxP, O=SBS, L=
psu-ip-address: 192.168.0.6
tpp-signature-certificate: -----BEGIN CERTIFICATE-----
MIIERzCCAy+gAwIBAgIIQUXgbv2lLcIwDQYJKoZIhvcNAQELBQAwaTELMAkGA1UE
BhMCRlIxDjAMBgNVBAgTBVBhcmlzMQ4wDAYDVQQHEwVQYXJpczEMMAoGA1UEChMD
U0JTMQwwCgYDVQQLEwNEeFAxHjAcBgkqhkiG9w0BCQEWD3Jvb3RDQUByb290LmNv
bTAeFw0xOTAyMTQxMzU4MDBaFw0yODEyMTkxODAwMDBaMIGSMQswCQYDVQQGEwJG
UjEOMAwGA1UECBMFUGFyaXMxDjAMBgNVBAcTBVBhcmlzMQwwCgYDVQQKEwNTQlMx
DDAKBgNVBAsTA0R4UDEMMAoGA1UEAxMDU0JTMR4wHAYJKoZIhvcNAQkBFg9yb290
Q0FAcm9vdC5jb20xGTAXBgNVBGETEFBTREJFLUFJUy0xMjM0NTYwggEiMA0GCSqG
SIb3DQEBAQUAA4IBDwAwggEKAoIBAQCyB1NOqlS7X/XOZyoMPXHhDG8c8VyVC6sC
M6dW2rTkQEr4fm57fi3bdPjMTBRFB+dRNcERyWUSWiE85LW9UWcMIORltL59utio
1dySGUV+DRKffLAoe2jyhq5iWsU6WvK3kVcqLddXMfgdSB+qgYhQqCK6mpOilhdG
BE6nFbF4M4+sTGL/0ncW4eGqAIsk6VvCEjuXtLabeE0Ji2BSekQnYMgu0qEXkwcg
TPkMw1topWz3VNAgN0aGAOMAN0piFBGayoLVGqoLd3eD4Xk/EzFkt3/XBYZ1TSaF
1jevtwrscrbnIrajfg1R36a5gJAUZpweT3RvpIVsYDBdCSUbCPsjAgMBAAGjgcgw
gcUwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQUQHK9dnN045oBeO2Be73LDTJU
VlswCwYDVR0PBAQDAgEGMBQGA1UdIAQNMAswCQYHBACL7EABATA9BggrBgEFBQcB
AwQxMC8wLQYGBACBmCcCMCMwEzARBgcEAIGYJwEDDAZQU1BfQUkMBEF1dGgMBkZy
YW5jZTARBglghkgBhvhCAQEEBAMCAAcwHgYJYIZIAYb4QgENBBEWD3hjYSBjZXJ0
aWZpY2F0ZTANBgkqhkiG9w0BAQsFAAOCAQEAgPKZSlIVmemajqUe290ecKQ3HLd4
KruOMEqpJoIOPRF9zIXe9RQb7qwbYpP0tweL4tA8eEdkwz7dxOwno/75GeCb5vVo
1GDbeOecgIyEMjEPyxoTtuo1qr07GIcVXmx/aOMLoZXAbKq1Tg2SU+876z1YlBs8
4n4hloJeWpkbKOTQkzfmTWW1AtLrBnU7KFhKdC1s5g/EyVz063lscSJNg07/i+n2
LSvrOptLOogZ92EcuVB6t7/3tmp6lIptMxoTNwZee8mqDI7ogUxcZUxGpAJzG6Ub
2qmL9KvMFjKTMhfyuTG8xx+4EN+VsduIxe/7/0nM5zQFwEZvPX1CQizAzQ==
-----END CERTIFICATE-----
Content-Length: 359
Host: <HOST>
User-Agent: Apache-HttpClient/4.5.7 (Java/1.8.0_161)
```


```
{
  "validUntil" : "2019-08-02",
  "frequencyPerDay" : 1,
  "recurringIndicator" : true,
  "access" : {
    "accounts" : [ {
      "iban" : "BE63999001487608"
    } ],
    "balances" : [ {
      "iban" : "BE63999001487608"
    } ],
    "transactions" : [ {
      "iban" : "BE63999001487608"
    } ]
  }
}
```