THE Berlin GROUP

A EUROPEAN STANDARDS INITIATIVE

Joint Initiative on a PSD2 Compliant XS2A Interface

NextGenPSD2 XS2A Framework

Implementation Guidelines

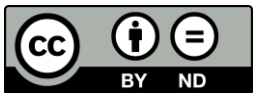
Version 1.3.14

24 January 2025

License Notice

This Specification has been prepared by the Participants of the Joint Initiative pan-European PSD2-Interface Interoperability* (hereafter: Joint Initiative). This Specification is published by the Berlin Group under the following license conditions:

- "Creative Commons Attribution-NoDerivatives 4.0 International Public License"

  This means that the Specification can be copied and redistributed in any medium or format for any purpose, even commercially, and when shared, that appropriate credit must be given, a link to the license must be provided, and indicated if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use. In addition, if you remix, transform, or build upon the Specification, you may not distribute the modified Specification.

- Implementation of certain elements of this Specification may require licenses under third party intellectual property rights, including without limitation, patent rights. The Berlin Group or any contributor to the Specification is not, and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights.
- The Specification, including technical data, may be subject to export or import regulations in different countries. Any user of the Specification agrees to comply strictly with all such regulations and acknowledges that it has the responsibility to obtain licenses to export, re-export, or import (parts of) the Specification.

---

* The 'Joint Initiative pan-European PSD2-Interface Interoperability' brings together participants of the Berlin Group with additional European banks (ASPSPs), banking associations, payment associations, payment schemes and interbank processors.

Contents

1 Introduction

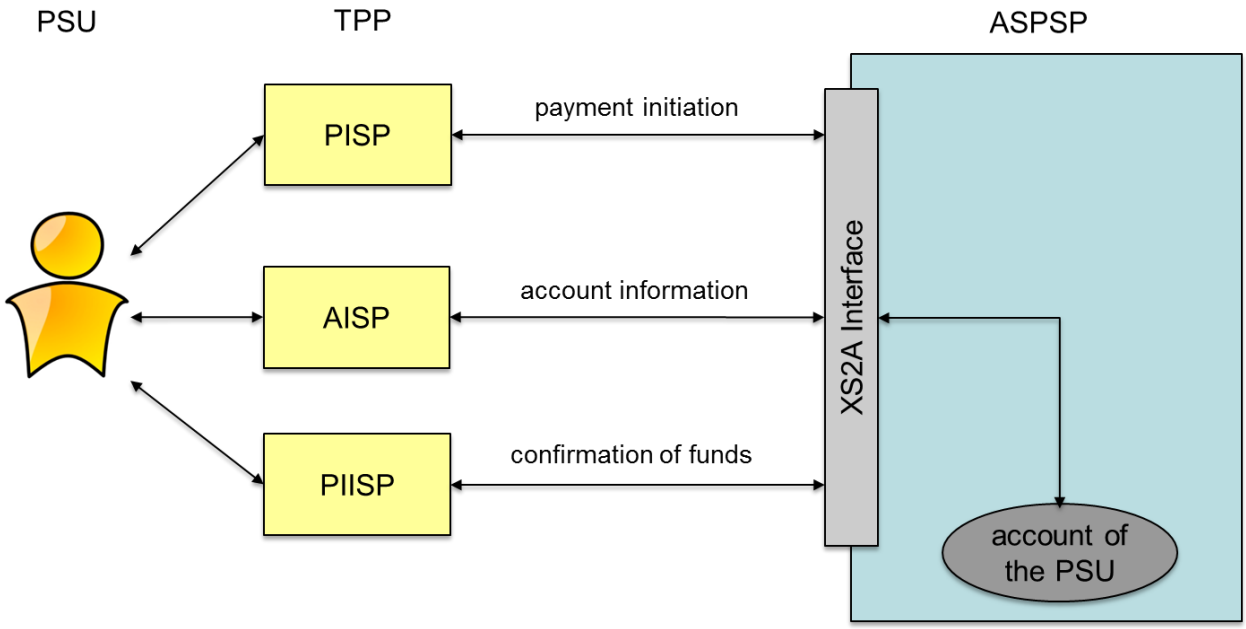
1.1 Background

With [PSD2] the European Union has published a new directive on payment services in the internal market. Member States had to adopt this directive into their national law until 13th of January 2018.

Among others [PSD2] contains regulations of new services to be operated by so called Third Party Payment Service Providers (TPP) on behalf of a Payment Service User (PSU). These new services are

- Payment Initiation Service (PIS) to be operated by a Payment Initiation Service Provider (PISP) TPP as defined by article 66 of [PSD2],
- Account Information Service (AIS) to be operated by an Account Information Service Provider (AISP) TPP as defined by article 67 of [PSD2], and
- Confirmation of the Availability of Funds service to be used by Payment Instrument Issuing Service Provider (PIISP) TPP as defined by article 65 of [PSD2].

For operating the new services a TPP needs to access the account of the PSU which is usually managed by another PSP called the Account Servicing Payment Service Provider (ASPSP). As shown in the following figure, an ASPSP has to provide an interface (called "PSD2 compliant Access to Account Interface" or short "XS2A Interface") to its systems to be used by a TPP for necessary accesses regulated by [PSD2]:



Further requirements on the implementation and usage of this interface are defined by a Regulatory Technical Standard (short RTS) from the European Banking Authority (short EBA), published in the Official Journal of the European Commission.

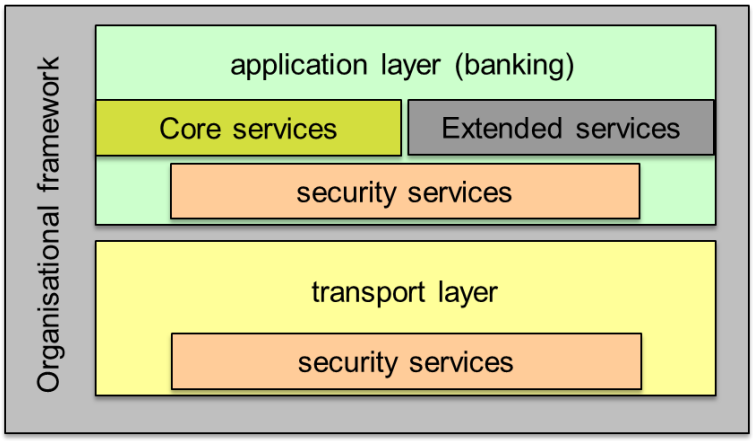
1.2 XS2A Interface Specification

This document is part of the NextGenPSD2 XS2A Specification which defines a standard for an XS2A Interface and by this reaching interoperability of the interfaces of ASPSPs at least for the core services defined by [PSD2]. An ASPSP may then use this standard as a basis for the implementation of its XS2A Interface to be compliant with PSD2.

The XS2A Interface is designed as a B2B interface between a TPP server and the ASPSP server. For time being, the protocol defined in this document is a pure client-server protocol, assuming the TPP server being the client, i.e. all API calls are initiated by the TPP. In future steps, this protocol might be extended to a server-server protocol, where also the ASPSP initiates API calls towards the TPP.

The Interoperability Framework defines operational rules, requirements on the data model and a process description in [XS2A-OR].

This document details the standard in defining messages and detailed data structures for the XS2A Interface. For the specification the two layers shown in the following figure are distinguished:



At the application layer only the core services will be specified in the first version of the framework. In addition the framework will be prepared such that the interface of an ASPSP may be extended with its own additional corporate specific services (included in the figure as "Extended services"). In future versions of this framework, some extended services will also be part of the standard. This framework documentation will point out extended services where the market need is already identified.

Using defined parameters different versions and variants of this protocol can be distinguished and implemented.

1.3 Structure of the Document

This document first outlines notations in Section 2 and requirements on the transport layer in Section 3. In Section 4, guiding principles for the definition of the XS2A interface and the API structure with API endpoints and permitted access methods are described. Section 5 then specifies in detail how a Payment Initiation Service Provider (PISP) can initiate payments within the Berlin Group XS2A. Section 6 then repeats this for the access of Account Information Service Provider (AISP) to a Payment Service User (PSU) account. The AIS and the PIS service are sharing potentially some API calls, specifically for authorising transactions directly through the TPP / ASPSP interface. These methods used within both services are specified in Section 7. Section 8 is introducing signing baskets as a new feature. These baskets allow to authorise several transactions at the same time, i.e. with one SCA. Section 9 then shortly explains how AIS and PIS services might be technically combined within one TPP / ASPSP business session.

The Confirmation of Funds Service for Payment Instrument Initiation Service Provider (PIISP) is specified in detail in Section 10. Following these chapters with functional character, Section 11 to Section 14 specify core payment structure, requirements on the optional integration of OAuth2, the usage of electronic seals for authentication on application level and general complex data structures.

Remark for Future: Please note that the Berlin Group NextGenPSD2 XS2A interface is still under constant development. Technical issues, which are already in discussion within the Berlin Group NextGenPSD2 working structure are mentioned in this document by "Remark for Future" to make the reader aware of upcoming potential changes.

1.4 Document History

Version	Change/Note	Approved
0.99	Market consultation draft of the Berlin Group XS2A Interface Framework	NextGenPSD2 Taskforce, 27 September 2017
1.0	Version 1.0 for publication. Takes into account the results of the market consultation and the final EBA-RTS on SCA and CSC.	NextGenPSD2 Taskforce, 08 February 2018
1.1	Minor release update, integrating results of convergence discussions with other API initiatives as well as some additional functionality and errata. A detailed change log will be published separately.	NextGenPSD2 Taskforce, 11 May 2018
1.2	Major release with the following major changes - support of multiple authorisations - support of payment cancellations - support of signing baskets instead of "pseudo-multi" payments as signing vehicle for multiple transactions - simplification in the /payments path: The resource will be addressable directly under /payments/paymentId even if posted on /payments/{payment-product} - the authorisation and cancellation-authorisation sub-resources have been separated from the payment, consent resp. signing basket resources and are exposed explicitly as resources in the API. In addition some minor functionality and errata have been considered. A detailed change log will be published separately.	NextGenPSD2 Taskforce, 25 July 2018
1.3	Payment Products are again involved in the path for starting authorisations, as it was	NextGenPSD2 Taskforce, 19 October 2018

Version	Change/Note	Approved
	foreseen till version 1.1 of this specification, cp. Bulletin No 001. More details on the usage for Multilevel SCA for Establish Account Information Consent Requests and Signing Baskets. Added a new endpoint for card accounts. Added a new additional variant to report additional error information following [RFC7807]. Clarifications and errata throughout the document. A change log document will be published separately.	
1.3.4	Errata on Version 1.3 Integration of Extended Services Added new headers to deal with payments which are neither rejected nor executed due to missing funds. Added a new functionality to update an authorisation resource by an additional password. A detailed change log document is published separately.	NextGenPSD2 Taskforce, 5 July 2019
1.3.5	Internal version	
1.3.6	Errata on Version 1.3.4 Integration of Extended Services on displaying account owner names and standing orders. Add new functionality on transaction confirmation as introduced by the security bulletin in Autumn 2019. Extend transaction report and payment initiation data models.	3 February 2020, NextGenPSD2 TF

Version	Change/Note	Approved
	Add new http headers e.g. for transporting TPP brand information for logging. A detailed change log document is published separately.	
1.3.7	Internal version	
1.3.8	Errata on Version 1.3.6 Integration of currency conversion fee information added to payment initiation to support conversion fee transparency requirements. Integration of a new SCA method for OTP transmission via email. Add clarifications to fulfil requirements resulting from the EBA Opinion on Obstacles from June 2020, cp. [EBA-OP2]. A detailed change log document is published separately.	30 October 2020
1.3.9	Errata on Version 1.3.8 Integration of entry details of transactions booked in batch in AIS. Integration of a flag indicating the TPP's preference for a decoupled SCA approach. New message codes added for additional error information in case of banking related errors within a http 20x response code. New bookingStatus "all" has been added as a query parameter which is optional to be supported by the ASPSP. Differentiating flag for debit or credit accounting was added to card accounts.	23 March 2021

Version	Change/Note	Approved
	A detailed change log document is published separately.	
1.3.10	Internal version	
1.3.11	Errata on Version 1.3.9. Clarifications on Signatures and oAUTH communication. Extension of Card Transactions by elements valueDate and grandTotalAmount. Extension of some detail attributes in reports by length. Added the possibility to submit standing orders with non regular execution. Clarifications on RJCT status of payment initiations in different time out scenarios.	2021-09-24
1.3.12	Errata on Version 1.3.11. Clarification on ASPSP certificates Adaptation to the new draft legal text [EBA-FR]. Additional attribute "ownerNames" to reflect accounts with multiple owners. New attribute "psuName" to reflect situations, where the psu is different from the account owner (e.g. in the context of corporate account owners). Clarification on side effects related to the situation where multiple SCA applies or where several employees would share a PSU-ID. Remark on accept headers following the Request Header definition. Clarification on debtorAccount subelement.	2022-07-01 openFinance Taskforce

Version	Change/Note	Approved
1.3.13	Errata on Version 1.3.12 Clarifications on authorisations retrieval endpoint. New optional extended status endpoint for JSON bulk payment initiation, in analogy to pain.002 messages. New attribute additionalRemittanceInformation for payment initiation. New attribute categoryPurposeCode on bulk level for JSON based payment initiation. Add an optional scope prefix "Cancel-PIS" for payment cancellation authorisation via OAuth2. Add attribute cardAcceptorName to card transaction reports. Regulatory clarification for townName usage in payment initiations. Strong recommendation not to use local time without offsets in timestamps.	2024-10-31 openFinance Taskforce
1.3.14	startDateTime and endDateTime added as new optional attributes for standing orders for instant payments in Section 5.3.4 for regulatory reasons. The bulk status attribute for payments has been changed in the extended payment status report in Section 5.5 from transactionStatus to groupStatus, the single transaction status from singleTransactionStatus to transactionStatus cp. Section 14.14. With this change, the report delivered through this endpoint is compliant to	2025-01-24 openFinance Taskforce

Version	Change/Note	Approved
	ISO20022 requirements. The related examples have been adapted as well. Minor erratum in example in Section 5.5 (V2 was used in bulk payments extended status endpoint instead of V1). Group Payment Status Code added as external ISO20022 code list in Section 14.37 to support the data type of the groupStatus introduced above. The full structured address is now supported in the payment data for describing involved parties for regulatory reasons, cp. Section 14.4. Status Reason Code added as ISO20022 external code list in Section 14.37 (Erratum). **Note:** No specific change log has been published for the above changes.	

2 Character Sets and Notations

2.1 Character Set

The character set is UTF 8 encoded. This specification is only using the basic data elements "String", "Boolean", "ISODateTime", "ISODate", "UUID" and "Integer" (with a byte length of 32 bits) and ISO based code lists. For codes defined by ISO, a reference to the corresponding ISO standard is given in 14.37.

Max35Text, Max70Text, Max140Text Max500Text, Max1000Text are defining strings with a maximum length of 35, 70, 140, 500 and 1000 characters respectively.

ASPSPs will accept for strings at least the following character set:

a b c d e f g h i j k l m n o p q r s t u v w x y z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

0 1 2 3 4 5 6 7 8 9

/ - ? : ( ) . , ' +

Space

ASPSPs may accept further character sets for text fields like names, addresses, text. Corresponding information will be contained in the ASPSP documentation of the XS2A interface. ASPSPs might convert certain special characters of these further character sets, before forwarding e.g. submitted payment data.

2.2 Notation

2.2.1 Notation for Requests

For API request calls, query parameters, HTTP header parameters and body content parameters are specified within this specification as follows:

Attribute	Type	Condition	Description
attribute tag	type of attribute	condition	description of the semantic of the attribute and further conditions.

The following conditions may be used when describing data to be submitted by the client:

- Optional: The attribute is supported by the server, usage is optional for the client. The server may ignore the parameter if mentioned in the “Description” column of the table above.
- Conditional: The attribute is supported by the server and might be mandated by
  - the server provider in its own documentation of the support of this XS2A interface or
  - by certain rules as defined in the “Description” column of the table above.
- Mandatory: The attribute is supported by the server and shall be used by the client.
- Optional if supported by API provider: It is optional for the server to support this attribute. If the server is supporting the attribute as indicated in its own documentation of this XS2A interface, it might be used by the client optionally. If the server is not supporting the attribute, then the request is rejected when it is contained.

Remark: Please note that the conditions “Optional if supported by API provider” is used rarely in this specification.

2.2.2 Notation for Responses

For API call responses, parameters, HTTP header parameters and body content parameters are specified within this specification as follows:

Attribute	Type	Condition	Description
attribute tag	type of attribute	condition	description of the semantic of the attribute and further conditions.

The following conditions can be set on data to be provided by the server:

- Optional: The attribute is supported optionally by the server
- Conditional: The attribute is supported by the server under certain conditions as indicated in the “Description” column of the table above.
- Mandatory: The attribute is always supported by the server.

3 Transport Layer

The communication between the TPP and the ASPSP is always secured by using a TLS-connection using TLS version 1.2 or higher. For the choice of cipher suite selections, NIST recommendations on the cryptographical strength should be followed. For ASPSPs, further cipher suite requirements of their national IT security agency might apply.

This TLS-connection is set up by the TPP. It is not necessary to set up a new TLS-connection for each transaction, however the ASPSP might terminate an existing TLS-connection if required by its security setting.

The TLS-connection has to be established always including client (i.e. TPP) authentication. For this authentication the TPP has to use a qualified certificate for website authentication. This qualified certificate has to be issued by a qualified trust service provider according to the eIDAS regulation [eIDAS]. The content of the certificate has to be compliant with the requirements of [EBA-RTS]. The certificate of the TPP has to indicate all roles the TPP is authorised to use.

For establishing the TLS-connection the ASPSP has to use also a certificate for website authentication. This specification does not introduce any special requirements for these certificates of the ASPSP. But it is recommended that certificates will be used by an ASPSP, which support also the usage for client authentication (field extendedKeyUsage contains both attributes serverAuth and clientAuth).

If an ASPSP wants to support the “Resource Status Notification Service” as specified in detail in [XS2A-RSNS] its certificate for website authentication shall support also client authentication.

4 Application Layer: Guiding Principles

4.1 Location of Message Parameters

The XS2A Interface definition follows the REST service approach. This approach allows to transport message parameters at different levels:

- message parameters as part of the HTTP level (HTTP header)
- message parameters by defining the resource path (URL path information) with additional query parameters and
- message parameters as part of the HTTP body.

The content parameters in the corresponding HTTP body will be encoded either in JSON or in XML syntax. XML syntax is only used where

- an ISO 20022 based payment initiation (pain.001 message) with the corresponding payment initiation report (pain.002 message) or
- ISO 20022 based account information message (camt.052, camt.053 or camt.054 message)

is contained.

As an exception, response messages might contain plain text format in account information messages to support MT940, MT941 or MT942 message formats.

The parameters are encoded in

- in spinal-case (small letters) on path level,
- in Spinal-case (starting capital letters) on HTTP header level and
- in lowerCamelCase for query parameters and JSON based content parameters.

The following principle is applied when defining the API:

Message parameters as part of the HTTP header:

- Definition of the content syntax,
- Certificate and Signature Data where needed,
- PSU identification data (the actual data from the online banking frontend or access token),
- Protocol level data like Request Timestamps or Request/Transaction Identifiers

Message parameters as part of the path level:

- All data addressing a resource:
  - Provider identification,
  - Service identification,
  - Payment product identification,
  - Account Information subtype identification,
  - Resource ID

Query Parameters:

- Additional information needed to process the GET request for filtering information,

Message parameters as part of the HTTP body:

- Business data content,
- PSU authentication data,
- Messaging Information
- Hyperlinks to steer the full TPP – ASPSP process

4.2 Signing Messages at Application Layer

The ASPSP may require the TPP to sign request messages. This requirement shall be stated in the ASPSP documentation.

The signature shall be included in the HTTP header as defined by [signHTTP], chapter 4.

The electronic signature of the TPP has to be based on a qualified certificate for electronic seals. This qualified certificate has to be issued by a qualified trust service provider according to the eIDAS regulation [eIDAS]. The content of the certificate has to be compliant with the requirements of [EBA-RTS]. The certificate of the TPP has to indicate all roles the TPP is authorised to use.

This specification uses on a pure protocol level the following HTTP header in all HTTP requests uniformously for the support of the signature function:

Request Header

Attribute	Type	Condition	Description
Digest	String	Conditional	Is contained if and only if the "Signature" element is contained in the header of the request.
Signature	cp. Section 12	Conditional	A signature of the request by the TPP on application level. This might be mandated by ASPSP.
TPP-Signature-Certificate	String	Conditional	The certificate used for signing the request, in base64 encoding. Must be contained if a signature is contained, see above.

For a better readability, the definition of these headers is not repeated throughout this specification. If no other condition describes otherwise, then the definitions here apply to all requests.

Remark: An ASPSP will ignore signatures on application level used by the TPP if signatures are not supported by the ASPSP.

4.3 Optional Usage of OAuth2 for PSU Authentication or Authorisation

The XS2A API will allow an ASPSP to implement OAuth2 as a support for the authorisation of the PSU towards the TPP for the payment initiation and/or account information service. In this case, the TPP will be the client, the PSU the resource owner and the ASPSP will be the resource server in the abstract OAuth2 model.

This specification supports two ways of integrating OAuth2. The first support is an authentication of a PSU in a pre-step, translating this authentication into an access token to be used at the XS2A interface afterwards. This usage of OAuth2 will be referred to in this specification as "if OAuth2 has been used as PSU authentication". Further details shall be defined in the documentation of the ASPSP of this XS2A interface.

Remark: When implementing the OAuth pre-step, the requirements on e.g. registration steps or no mandatory two SCA usage in specific PIS only scenarios as defined by [EBA-OP2] should be recognised by the ASPSP.

The second option to integrate OAuth2 is an integration as an OAuth2 SCA Approach to be used for authorisation of payment initiations and consents. In both services, PIS and AIS, OAuth2 will in this option be used in an integrated way, by using the following steps:

Integrated OAuth in the Use Case SCA for PIS:

1.) The payment data is posted to the corresponding payment initiation endpoint of the XS2A API.
2.) The OAuth2 protocol is used with the "Authorisation Code Grant" flow to get the consent on the payment authorised by the PSU, while using the "scope" attribute in OAuth2 to refer to the data from Step 1.).
3.) The corresponding payment is then automatically initiated by the ASPSP after a successful authorisation by the PSU.

Integrated OAuth in the Use Case SCA for AIS:

1.) The AIS consent data is posted to the consents endpoint of the XS2A API.
2.) The OAuth2 protocol is used with the "Authorisation Code Grant" flow to get the consent on the payment resp. the AIS access authorised by the PSU, while using the "scope" attribute in OAuth2 to refer to the data from Step 1.).
3.) The TPP can use the access token received during the OAuth2 protocol to access the /accounts endpoint for authorised account information for the validity period of the authorised consent resp. the validity period of the technical access token.

For Step 2.), details are described in Section 13.

When using OAuth2, the XS2A API calls will work with an access token instead of using the PSU credentials.

4.4 XS2A Interface API Structure

The XS2A Interface is resource oriented. Resources can be addressed under the API endpoints

```
https://{provider}/v1/{service}{?query-parameters}
```

using additional content parameters {parameters}

where

- {provider} is the host and path of the XS2A API, which is not further mentioned. The host or path may contain release version information of the ASPSP.
- v1 is denoting the final version 1.3.x of the Berlin Group XS2A interface Implementation Guidelines.

  Remark for Future: The handling of implementation and specification release version information is planned to be adapted in a more standardized way in future versions of the specification.
- {service} has the values consents, payments, bulk-payments, periodic-payments, accounts, card-accounts, signing-baskets or funds-confirmations, eventually extended by more information on product types and request scope
- {?query-parameters} are parameters detailing GET based access methods, e.g. for filtering content data
- {parameters} are content attributes defined in JSON or XML encoding according to the following
  - XML encoding appears only when ISO 20022 pain.001 messages are transported when demanded by the ASPSP for the corresponding payment product
  - all other request bodies are encoded in JSON

The structure of the request/response is described according to the following categories

- Path: Attributes encoded in the Path, e.g. "payments/sepa-credit-transfers" for {resource}
- Query Parameters: Attributes added to the path after the "?" sign as process steering flags or filtering attributes for GET access methods. Query parameters of type Boolean shall always be used in a form query-parameter=true or query-parameter=false.
- Header: Attributes encoded in the HTTP header of request or response

- Request: Attributes within the content parameter set of the request
- Response: Attributes within the content parameter set of the response, defined in XML, text or JSON:
  - XML encoding appears only, when camt.052, camt.053 or camt.054 messages (reports, notifications or account statements) or pain.002 payment status messages are transported. pain.002 messages will only be delivered for the GET Status Request, and only in cases where the payment initiation was performed by using pain.001 messages.
  - Text encoding appears only, when MT940, MT941 or MT942 messages (reports, notifications or account statements) are transported.
  - All other response bodies are encoded in JSON.

The HTTP response codes which might be used in this XS2A interface are specified in Section 14.11. This is not repeated for every API call definition.

> **Remark:** For JSON based responses, this specification defines body attributes which are responded from ASPSP to TPP following POST or PUT API calls. The ASPSP is free to return the whole addressed resource within the response, following usual REST methodologies.

4.5 Multicurrency Accounts

Definition: A multicurrency account is an account which is a collection of different sub-accounts which are all addressed by the same account identifier like an IBAN by e.g. payment initiating parties. The sub-accounts are legally different accounts and they all differ in their currency, balances and transactions. An account identifier like an IBAN together with a currency always addresses uniquely a sub-account of a multicurrency account.

This specification supports to address multicurrency accounts either on collection or on sub-account level. The currency data attribute in the corresponding data structure "Account Reference" allows to build structures like

```
{"iban": "DE40100100103307118608"}
```

or

```
{"iban": "DE40100100103307118608",
 "currency": "EUR"}
```

If the underlying account is a multicurrency account, then

- the first reference is referring to the collection of all sub-accounts addressable by this IBAN, and
- the second reference is referring to the euro sub-account only.

This interface specification is acting on sub-accounts of multicurrency accounts in exactly the same way as on regular accounts. This applies to payment initiation as well as to account information.

Remark: The multi-currency account product is in use in some markets in Europe, e.g. in Online-Banking products within the Belgium market. The support of this functionality in the XS2A API is only applicable in these markets.

4.6 Authorisation Endpoints

The NextGenPSD2 API is supporting dedicated authorisation endpoints for payment initiation transactions and establish consent transactions in order to handle transaction authorisation by PSUs. These authorisation endpoints are supported from version 1.2 of this specification for supporting the following new features in a common structured way

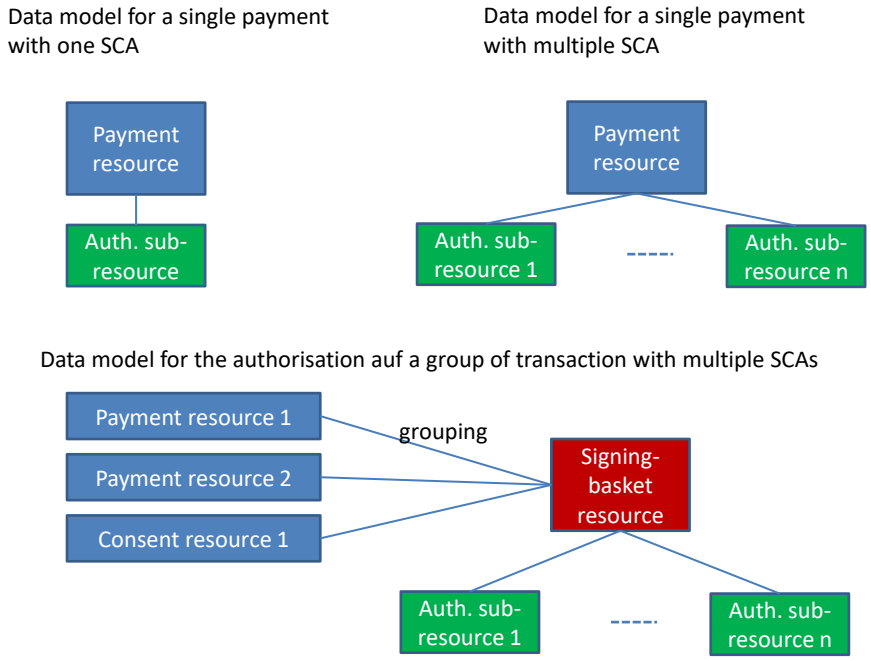
- multiple level SCA, where a transaction needs an authorisation by more than one PSU, e.g. in a corporate context,
- signing of a group of transactions with one SCA, as it is offered by ASPSPs today in online banking,
- signing of a group of transactions with multi-level SCA, where this group of transactions need an authorisation by more than one PSU, e.g. in a corporate context.

To support this, the resources resulting from the submission of payment data or consent data are separated from authorisation (sub-)resources. A payment which needs to be signed *n* times then will end up in a payment resource with *n* SCA (sub-)resources in a normal successful process.

Remark: This new resource structure also applies to the authorisation of individual transactions, which is a major change to the data model supported in version 1.0 and 1.1 of this specification. Nevertheless, the optimised integration of the authorisation process into the payment initiation or establish consent process will still be supported, cp. the paragraph at the end of this section.

The optional function of grouping several transactions for one common authorisation process is supported by the signing-baskets endpoint, which might be offered by the ASPSP. If this function is offered by an ASPSP, the TPP can first submit payment and consent data without starting the authorisation. After having grouped the related payment and consent resources by using a grouping command through the signing-baskets endpoint, the authorisation then can be started by authorising this basket content. This results in a basket resource with the corresponding authorisation sub-resource.

The following picture gives an overview on the abstract data model for the different scenarios:



Remark: When offering the signing basket function, the ASPSP might restrict the grouping e.g.

- to payments as such,
- to individual payments,
- to the same payment product.

This restriction on groupings will then be detailed in the ASPSPs documentation.

Note: The grouping of transaction is only a "signing vehicle", bundling authorisation processes for the grouped transactions. The authorisation rules for transactions can be very complex in a corporate context. The signing basket gets the status of being fully authorised as soon as all grouped transactions have been successfully authorised by the applied SCA mechanism. A transaction with less authorisation requirements might then be authorised earlier than the

whole signing basket and also already processed. In addition, single transactions of the signing basket could be authorised with additional SCAs directly on transaction level, depending on the implementations of the ASPSPs – the signing basket is a non-exclusive mechanism to bundle authorisations. Current implementations of this functionality differ in Europe, specifically in a corporate context. For this reason, more complex functionality as DELETE processes on partially authorised signing baskets are not supported yet.

Remark for Future: The upcoming versions of the specification might implement more advanced functionality of the signing basket function and cancellation processes around it.

Optimisation process for the submission of single payments

The general model introduced above requires the TPP to start two sub-processes when initiating a payment, creating a signing basket or submitting a consent. In a payment initiation of a sepa credit transfer this would result in

```
POST /payments/sepa-credit-transfers {payment data}
```

which is generating the payment resource and returns paymentId as a resource identification.

```
POST /payments/sepa-credit-transfers/paymentId/authorisations
```

is then starting the authorisation process with creating an authorisation sub-resource and returning an authorisationId for addressing this sub-resource in the following.

Applying this requirement to all authorisations of transactions e.g. in the Redirect SCA Approach would significantly augment the calls on the resulting API. For this reason, this specification still enables the ASPSP to directly start e.g. a Redirect SCA processing after the submission of a payment or a consent, if no other data from the TPP has to be submitted anyhow. In this case, the ASPSP will create the related authorisation sub-resources automatically and will give access to these sub-resources to the TPP by returning corresponding hyperlinks, cp. Section 4.15. As a consequence, the authorisation status would still result by submitting the command

```
GET /payments/sepa-credit-
transfers/paymentId/authorisations/authorisationId,
```

where the authorisation resource with identification authorisationId has been created by the ASPSP implicitly.

4.7 Payment Cancellation Endpoints

Starting from version 1.2, this specification is supporting the cancellation of payment initiations by PISPs. This process is divided into two steps

1. DELETE the corresponding resource.

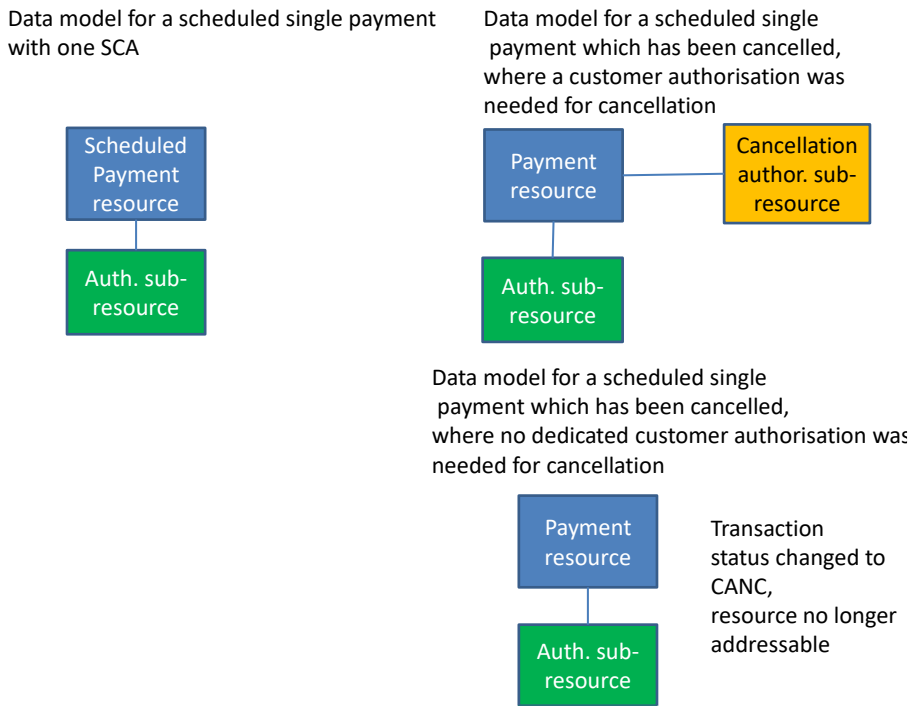
2. Start an authorisation process for the cancellation by the PSU where needed by submitting a

   `POST payments/sepa-credit-transfers/paymentId/cancellation-authorisations`

   command.

The second step might be omitted, where a dedicated authorisation of the cancellation is not foreseen by the ASPSP. The need to authorise the cancellation will be communicated by sending corresponding hyperlinks to the TPPs, cp. Section 4.15.

In the two-step approach, this cancellation process will be handled by cancellation-authorisation sub-resources in analogy to the actual authorisations. The authorisation sub resources will stay unchanged. The following picture shows the changes on resource level in case of a scheduled payment:



The corresponding original authorisation sub-resources stay unchanged.

For transactions, where a multilevel SCA is needed for authorisation, also a multilevel SCA might be needed for cancellation, depending on ASPSP role management. In equivalence to authorisation, the model would then be extended by more cancellation sub-resources.

4.8 Requirements on PSU Context Data

The following data elements are forwarding information about the PSU-TPP interface and are enhancing the risk management procedures of the ASPSP. It is strongly recommended to send these data elements in all request messages within the payment initiation or consent initiation

transaction flow, i.e. flows with a PSU authentication involved. The further definitions of request parameters within the related sections are not repeating the definition of these elements for the matter of better readability. The only exception is where conditions other than "optional" apply on specific request messages, e.g. for the PSU IP Address. More details are provided in the data overview in within Section 5.2 or Section 6.2.

Attribute	Format	Condition	Description
PSU-IP-Address	String	Optional	The forwarded IP Address header field consists of the corresponding HTTP request IP Address field between PSU and TPP.
PSU-IP-Port	String	Optional	The forwarded IP Port header field consists of the corresponding HTTP request IP Port field between PSU and TPP, if available.
PSU-Accept	String	Optional	The forwarded IP Accept header fields consist of the corresponding HTTP request Accept header fields between PSU and TPP, if available.
PSU-Accept-Charset	String	Optional	see above
PSU-Accept-Encoding	String	Optional	see above
PSU-Accept-Language	String	Optional	see above
PSU-User-Agent	String	Optional	The forwarded Agent header field of the HTTP request between PSU and TPP, if available.
PSU-Http-Method	String	Optional	HTTP method used at the PSU – TPP interface, if available. Valid values are: • GET • POST • PUT • PATCH • DELETE
PSU-Device-ID	String	Optional	UUID (Universally Unique Identifier) for a device, which is used by the PSU, if available.

Attribute	Format	Condition	Description
			UUID identifies either a device or a device dependant application installation. In case of an installation identification this ID need to be unaltered until removal from device.
PSU-Geo-Location	Geo Location	Optional	The forwarded Geo Location of the corresponding HTTP request between PSU and TPP if available.

Note: Information about the PSU/TPP interface might be used by the ASPSP as input for his fraud detection and risk management systems. Some ASPSPs use this information also to exclude some authentication methods (for example some ASPSPs do not allow to receive an OTP by SMS on the same smartphone used also for the transaction itself). In addition the ASPSP might need to receive specific device related information to be able to support an optimised app-2-app redirection procedure for the TPP. For these reasons it is highly recommended that a TPP includes all of this information into the related request messages. Missing information may result in an assessment of the user device as not useable for the authentication method or in a classification of the current transaction as a "higher risk transaction" e.g. due to session attacs. By this the probability of a rejection of that transaction due to the result of fraud detection and/or risk management might be increased, cp [XS2A-SecB] for details.

4.9 Requirements on TPP Identification

[PSD2] is mandating the identification of TPPs by PSD2 related eIDAS certificates. The specific certificates to be used within the PSD2 context are specified within [ETSI PSD2]. The requirements defined in [XS2A-OR] yield a TPP identification by the QWAC and/or QSEAL certificate used by the TPP.

The TPP is noted in the eIDAS certificate by its legal name. Still, the TPP might use brand names towards the PSU, which are differing from the legal name strongly. Thus, it might be beneficial for the TPP if the ASPSP is able to use also TPP brand names towards the PSU in all PSU related processes like SCA. Specific brand names of the TPP could be entered into the certificate field Organisation Unit (marked with the tag “OU”). ASPSP may ignore entries in this field.

Remark: The usage of the certificate field “OU” by the TPP will lead to the usage of several certificates if the TPP intends to separate different TPP brands in processing.

Note: The usage of more than one certificate by the TPP, differing e.g. by different OU entries does not mean that the consent management is treating these different certificates as different entities. By default of this framework, the legal owner of the certificate is the counterparty for access managements through consent tokens and not the brand cited in the OU field.

4.10 Requirements on TPP URIs

The TPP can provide several URIs to the ASPSP as parameters for succeeding protocol steps. For security reasons, it should be ensured that these URIs are secured by the TPP eIDAS QWAC used for identification of the TPP. The following applies:

URIs which are provided by TPPs in TPP-Redirect-URI or TPP-Nok-Redirect-URI should comply with the domain secured by the eIDAS QWAC certificate of the TPP in the field CN or SubjectAltName of the certificate. Please note that in case of example-TPP.com as certificate entry TPP-Redirect-URI like

- www.example-TPP.com/xs2a-client/v1/ASPSPidentifcation/mytransaction-id or
- redirections.example-TPP.com/xs2a-client/v1/ASPSPidentifcation/mytransaction-id

would be compliant.

Wildcard definitions shall be taken into account for compliance checks by the ASPSP.

Remark for Future: ASPSPs in future may reject requests, if the provided URIs do not comply. This is not yet valid for the current version of the specification.

Remark for Future: For migration reasons, this specification mandates the TPP to keep the TPP-Redirect-URI used within all authorisation processes for a specific transaction during the lifecycle of this transaction constant. This might be removed in the next version of the specification.

Remark for Future: The restrictions on URIs will apply to TPP-URIs used within future Push Services of the ASPSP.

4.11 API Access Methods

The following tables gives an overview on the HTTP access methods supported by the API endpoints and by resources created through this API.

Conditions in the following tables

It is further defined, whether this method support is mandated for the ASPSP by this specification or whether it is an optional feature for the ASPSP. Please note that this condition is given relative to the parent node of the path, i.e. the condition e.g. on a method on /v1/consents/{consentId} applies only if the endpoint /v1/consents is supported at all.

Please note that all methods submitted by a TPP, which are addressing dynamically created resources in this API, may only apply to resources which have been created by the same TPP before.

Examples

Please further note, that sections are referred in the Description's column. These sections provide example for all related access methods.

4.11.1 Payments Endpoints

Endpoints/Resources	Method	Condition	Description
payments/{payment-product}	POST	Mandatory	Create a payment initiation resource addressable under {paymentId} with all data relevant for the corresponding payment product. This is the first step in the API to initiate the related payment. Section 5.3.1 and 5.3.2
payments/{payment-product}/{paymentId}	GET	Mandatory	Read the details of an initiated payment. Section 5.6
payments/{payment-product}/{paymentId}/status	GET	Mandatory	Read the transaction status of the payment Section 5.4

Endpoints/Resources	Method	Condition	Description
bulk-payments/{payment-product}	POST	Optional	Create a bulk payment initiation resource addressable under {paymentId} with all data relevant for the corresponding payment product. This is the first step in the API to initiate the related bulk payment. Section 5.3.3
bulk-payments/{payment-product}/{paymentId}	GET	Mandatory	Read the details of an initiated bulk payment. Section 5.6
bulk-payments/{payment-product}/{paymentId}/status	GET	Mandatory	Read the transaction status of the bulk payment Section 5.4
bulk-payments/{payment-product}/{paymentId}/extended-status	GET	Optional	Read the extended transaction status of the bulk payment. The response contains an array with status information for every rejected transaction. This is an analog structure to pain.002 for JSON based bulk payments. Section 5.5
periodic-payments/{payment-product}	POST	Optional	Create a standing order initiation resource for recurrent i.e. periodic payments addressable under {paymentId} with all data relevant for the corresponding payment product and the execution of the standing order. This is the first step in the API to initiate the related recurring/periodic payment. Section 5.3.4

Endpoints/Resources	Method	Condition	Description
periodic-payments/{payment-product}/{paymentId}	GET	Mandatory	Read the details of an initiated standing order for recurring/periodic payments. Section 5.6
periodic-payments/{payment-product}/{paymentId}/status	GET	Mandatory	Read the transaction status of the standing order for recurring/periodic payments. Section 5.4
{payment-service}/{payment-product}/{paymentId}/authorisations	POST	Mandatory	Create an authorisation sub-resource and start the authorisation process, might in addition transmit authentication and authorisation related data. This method is iterated n times for a n times SCA authorisation in a corporate context, each creating an own authorisation sub-endpoint for the corresponding PSU authorising the transaction. The ASPSP might make the usage of this access method unnecessary in case of only one SCA process needed, since the related authorisation resource might be automatically created by the ASPSP after the submission of the payment data with the first POST payments/{payment-product} call. Section 7.1
{payment-service}/{payment-product}/{paymentId}/authorisations	GET	Mandatory	Read a list of all authorisation sub-resources IDs which have been created. Section 7.4

Endpoints/Resources	Method	Condition	Description
{payment-service}/{payment-product}/{paymentId}/authorisations/{authorisationId}	PUT	Mandatory for Embedded SCA Approach, Conditional for other approaches	Update data on the authorisation resource if needed. It may authorise a payment within the Embedded SCA Approach where needed. Independently from the SCA Approach it supports e.g. the selection of the authentication method and a non-SCA PSU authentication. Section 7.2 and Section 7.3
{payment-service}/{payment-product}/{paymentId}/authorisations/{authorisationId}	GET	Mandatory	Read the SCA status of the authorisation. Section 7.5
{payment-service}/{payment-product}/{paymentId}	DELETE	Optional	Cancels the addressed payment with resource identification paymentId if applicable to the payment-service, payment-product and received in product related timelines (e.g. before end of business day for scheduled payments of the last business day before the scheduled execution day). The response to this DELETE command will tell the TPP whether the • access method was rejected • access method was successful, or • access method is generally applicable, but further authorisation processes are needed. Section 5.7

Endpoints/Resources	Method	Condition	Description
{payment-service}/{payment-product}/{paymentId}/cancellation-authorisations	POST	Optional	Starts the authorisation of the cancellation of the addressed payment with resource identification paymentId if mandated by the ASPSP (i.e. the DELETE access method is not sufficient) and if applicable to the payment-service, and received in product related timelines (e.g. before end of business day for scheduled payments of the last business day before the scheduled execution day). Section 7.1
{payment-service}{payment-product}/{paymentId}/cancellation-authorisations	GET	Optional	Retrieve a list of all created cancellation authorisation sub-resources. If the POST command on this endpoint is supported, then also this GET method needs to be supported. Section 5.8
{payment-service}/{payment-product}/{paymentId}/cancellation-authorisations/{authorisationId}	PUT	Mandatory for Embedded SCA Approach, Conditional for other approaches	Update data on the cancellation authorisation resource if needed. It may authorise a cancellation of the payment within the Embedded SCA Approach where needed. Independently from the SCA Approach it supports e.g. the selection of the authentication method and a non-SCA PSU authentication. Section 7.2 and Section 7.3
{payment-service}/{payment-product}/{paymentId}/cancellation-authorisations/{authorisationId}	GET	Mandatory	Read the SCA status of the cancellation authorisation. Section 7.5

4.11.2 Accounts Endpoint

Endpoints/Resources	Method	Condition	Description
accounts	GET	Mandatory	Read all identifiers of the accounts, to which an account access has been granted to through the /consents endpoint by the PSU. In addition, relevant information about the accounts and hyperlinks to corresponding account information resources are provided if a related consent has been already granted. **Remark:** Note that the /consents endpoint optionally offers to grant an access on **all available** payment accounts of a PSU. In this case, this endpoint will deliver the information about all available payment accounts of the PSU at this ASPSP. Section 6.5.1
accounts?withBalance	GET	Optional	Read the identifiers of the available payment account together with booking balance information, depending on the consent granted Section 6.5.1
accounts/{account-id}	GET	Mandatory	Give detailed information about the addressed account. Section 6.5.2
accounts/{account-id}?withBalance	GET	Optional	Give detailed information about the addressed account together with balance information Section 6.5.2
accounts/{account-id}/balances	GET	Mandatory	Give detailed balance information about the addressed account Section 6.5.3

Endpoints/Resources	Method	Condition	Description
accounts/{account-id}/transactions	GET	Mandatory	Read transaction reports or transaction lists of a given account. For a given account, additional parameters are e.g. the attributes "dateFrom" and "dateTo". The ASPSP might add balance information, if transaction lists without balances are not supported. Section 6.5.4
accounts/{account-id}/transactions?withBalance	GET	Optional	Read transaction reports or transaction lists of a given account, depending on the steering parameter "bookingStatus" together with balances. Section 6.5.4
accounts/{account-id}/transactions/{transactionId}	GET	Optional	Read transaction details of an addressed transaction. Section 6.5.5

Remark: Note that the {account-id} parameters can be tokenised by the ASPSP such that the actual account numbers like IBANs or PANs are not part of the path definitions of the API for data protection reasons. This tokenisation is managed by the ASPSP.

4.11.3 Card-accounts Endpoint

This endpoint delivers credit card account related account information, where the account is used to reconcile credit card transactions with the PSU. This endpoint is not directly related to credit cards as such, but the financial account behind the related cards.

Remark: The access methods to card accounts are less detailed compared to access methods to accounts due to the reduced functionality compared to generic payment accounts.

Endpoints/Resources	Method	Condition	Description
card-accounts	GET	Optional	Read all identifiers of the card accounts, to which an account access has been granted to through the /consents endpoint by the PSU. In addition, relevant information about the card accounts and hyperlinks to corresponding account information resources are provided if a related consent has been already granted. Section 6.6.1
card-accounts/{account-id}	GET	Optional	Give detailed information about the addressed card account. Section 6.6.2 **Remark for Future**: This endpoint might be made mandatory for future versions of the specification.
card-accounts/{account-id}/balances	GET	Optional	Give detailed balance information about the addressed card account. Section 6.6.3 **Remark for Future**: This endpoint might be made mandatory for future versions of the specification.
card-accounts/{account-id}/transactions	GET	Mandatory	Read transaction reports or transaction lists related to a given card account. For a given card account, additional parameters are

Endpoints/Resources	Method	Condition	Description
			e.g. the attributes "dateFrom" and "dateTo". Section 6.6.4

Remark: Note that the {card-account-id} parameters can be tokenised by the ASPSP such that the actual card account or card number like IBANs or PANs are not part of the path definitions of the API for data protection reasons. This tokenisation is managed by the ASPSP.

4.11.4 Consents Endpoint

Endpoints/Resources	Method	Condition	Description
consents	POST	Mandatory	Create a consent resource, defining access rights to dedicated accounts of a given PSU-ID. These accounts are addressed explicitly in the method as parameters as a core function. Section 6.3.1
consents	POST	Optional	As an option, an ASPSP might optionally accept a specific access right on the access on all psd2 related services for all available accounts. As another option an ASPSP might optionally also accept a command, where only access rights are inserted without mentioning the addressed account. The relation to accounts is then handled afterwards between PSU and ASPSP. This option is not supported for the Embedded SCA Approach. As a last option, an ASPSP might in addition accept a command with access rights • to see the list of available payment accounts or

Endpoints/Resources	Method	Condition	Description
			• to see the list of available payment accounts with balances. Section 6.3.1
consents/{consentId}	GET	Mandatory	Reads the exact definition of the given consent resource {consentId} including the validity status Section 6.3.3
	DELETE	Mandatory	Terminate the addressed consent. Section 6.4
consents/{consentId}/status	GET	Mandatory	Read the consent status of the addressed consent resource. Section 6.3.2
consents/{consentId}/authorisations	POST	Mandatory	Create an authorisation sub-resource and start the authorisation process, might in addition transmit authentication and authorisation related data. The ASPSP might make the usage of this access method unnecessary, since the related authorisation resource will be automatically created by the ASPSP after the submission of the consent data with the first POST consents call. Section 7.1
consents/{consentId}/authorisations	GET	Mandatory	Read a list of all authorisation sub-resources IDs which have been created. Section 7.4

Endpoints/Resources	Method	Condition	Description
consents/{consentId}/authorisations/{authorisationId}	PUT	Mandatory for Embedded SCA Approach, Conditional for other approaches	Update data on the authorisation resource if needed. It may authorise a consent within the Embedded SCA Approach where needed. Independently from the SCA Approach it supports e.g. the selection of the authentication method and a non-SCA PSU authentication. Section 7.2 and Section 7.3
consents/{consentId}/authorisations/{authorisationId}	GET	Mandatory	Read the SCA status of the authorisation. Section 7.5

4.11.5 Signing-baskets Endpoint

Endpoints/Resources	Method	Condition	Description
signing-baskets	POST	Optional	Create a signing basket resource for authorising several transactions with one SCA method. The resource identifications of these transactions are contained in the payload of this access method Section 8.1
signing-baskets/{basketId}	GET	Optional	Retrieve the signing basket content Section 8.2
	DELETE	Optional	Delete the signing basket structure as long as no authorisation has yet been applied. The underying transactions are not affected by this deletion. Section 8.5
signing-baskets/{basketId}/status	GET	Optional	Read the status of the signing basket

Endpoints/Resources	Method	Condition	Description
			Section 8.3
signing-baskets/{basketId}/authorisations	POST	Mandatory	Create an authorisation sub-resource and start the authorisation process, might in addition transmit authentication and authorisation related data. The ASPSP might make the usage of this access method unnecessary, since the related authorisation resource will be automatically created by the ASPSP after the submission of the basket data with the first POST consents call. Section 7.1
signing-baskets/{basketId}/authorisations	GET	Mandatory	Read a list of all authorisation sub-resources IDs which have been created. Section 7.4
signing-baskets/{basketId}/authorisations/{authorisationId}	PUT	Mandatory for Embedded SCA Approach, Conditional for other approaches	Update data on the authorisation resource if needed. It may authorise all transactions in the addressed signing basket within the Embedded SCA Approach where needed. Independently from the SCA Approach it supports e.g. the selection of the authentication method and a non-SCA PSU authentication. Section 7.2 and Section 7.3
signing-baskets/{basketId}/authorisations/{authorisationId}	GET	Mandatory	Read the SCA status of the authorisation. Section 7.5

Remark: The signing basket as such is not deletable after a first authorisation has been applied. Nevertheless, single transactions might be cancelled on an individual basis on the XS2A interface.

4.11.6 Funds-Confirmations Endpoint

Endpoints/Resources	Method	Condition	Description
funds-confirmations	POST	Mandatory	Checks whether a specific amount is available at point of time of the request on an account linked to a given tuple card issuer(TPP)/card number, or addressed by IBAN and TPP respectively Section 10.2

Remark for Future: The PUT HTTP methods might be adapted to technical PATCH methods in a future version of the specification. A corresponding decision will reflect current market practices and the work in ISO TC68/SC9/WG2 on Financial API services.

4.12 HTTP Response Codes

The HTTP response code is communicating the success or failure of a TPP request message, cp. [RFC7231]. The 4XX HTTP response codes should only be given if the current request cannot be fulfilled, e.g. a payment initiation cannot be posted or account transactions cannot be retrieved. A request to get the status of an existing payment or a consent usually returns HTTP response code 200 since the actual request to retrieve the status succeeded, regardless if that payment or consent state is set to failure or not.

This specification supports the following HTTP response codes:

Status Code	Description
200 OK	PUT, GET Response Codes This return code is permitted if a request was repeated due to a time-out. The response in that might be either a 200 or 201 code depending on the ASPSP implementation. The POST for a Funds request will also return 200 since it does not create a new resource. DELETE Response Code where a payment resource has been cancelled successfully and no further cancellation authorisation is required.
201 Created	POST response code where Payment Initiation or Consent Request was correctly performed.
202 Accepted	DELETE response code, where a payment resource can be cancelled in general, but where a cancellation authorisation is needed in addition.
204 No Content	DELETE response code where a consent resource was successfully deleted. The code indicates that the request was performed, but no content was returned.
400 Bad Request	Validation error occurred. This code will cover malformed syntax in request or incorrect data in payload.
401 Unauthorized	The TPP or the PSU is not correctly authorized to perform the request. Retry the request with correct authentication information.
403 Forbidden	Returned if the resource that was referenced in the path exists but cannot be accessed by the TPP or the PSU. This code should only be used for non-sensitive id references as it will reveal that the resource exists even though it cannot be accessed.

Status Code	Description
404 Not found	Returned if the resource or endpoint that was referenced in the path does not exist or cannot be referenced by the TPP or the PSU. When in doubt if a specific id in the path is sensitive or not, use the HTTP response code 404 instead of the HTTP response code 403.
405 Method Not Allowed	This code is only sent when the HTTP method (PUT, POST, DELETE, GET etc.) is not supported on a specific endpoint. It has nothing to do with the consent, payment or account information data model. DELETE Response code in case of cancellation of a payment initiation, where the payment initiation cannot be cancelled due to legal or other operational reasons.
406 Not Acceptable	The ASPSP cannot generate the content that the TPP specified in the Accept header.
408 Request Timeout	The server is still working correctly, but an individual request has timed out.
409 Conflict	The request could not be completed due to a conflict with the current state of the target resource.
415 Unsupported Media Type	The TPP has supplied a media type which the ASPSP does not support.
429 Too Many Requests	The TPP has exceeded the number of requests allowed by the consent or by the RTS.
500 Internal Server Error	Internal server error occurred.
503 Service Unavailable	The ASPSP server is currently unavailable. Generally, this is a temporary state.

4.13 Responses in Error Cases

In order to achieve a better readability, the main part of the document describes responses in case of a positive processing result only. The following section gives specific rules for the case of a negative processing result.

4.13.1 Header

In general, the same rules regarding the presence of header elements apply for both positive and negative responses. An exception is made for cases, where an error occurred before functional processing. Examples for such error cases are general server errors (typically with 50x http response code) and – depending on the implementation – the validation of the certificate. In those error cases, the ASPSP may omit the specified headers. However, when functional processing has already taken place, the ASPSP is still requirted to include the mandated (and if applicable conditional) headers also in a negative response.

4.13.2 Body

All descriptions of body elements in the following document only apply to cases with a positive response.. The related attributes need also be provided in the body, where possible and applicable. In addition, the API shall offer additional error information as described in Section 4.13.3, when the API server is technically able to provide it.

4.13.3 Additional Error Information

If necessary, the ASPSP **might** communicate additional error information to the TPP within a request/response dialogue which results in 4xx or 5xx HTTP response codes, in some exemptions also for HTTP response code 2xx. This specification offers two possibilities for ASPSPs to communicate additional error information. The ASPSP might choose one of the solutions. Note that the major additional error information is the detailed error code which is of type "Message Code" as defined in Section 14.11 is used in both variants of additional error information.

In cases, where no message code is defined for an HTTP response code in Section 14.11, the additional error information is not used, since the messageCode is a mandatory subfield. In this case, the HTTP code gives sufficient information about the error situation.

4.13.3.1 NextGenPSD2 Specific Solution

The NextGenPSD2 XS2A specification offers a proprietary way to transport additional error information. In this solution, the additional error information is sent to the TPP using the data element tppMessageInformation with the attribute category set to "ERROR". The attribute "code" indicates the error, cp. Section 14.11 and if applicable the path of the element of the request message which provoked this error message. It will further offer a free text field to describe the error context or actions to be taken to the TPP.

Usually, a tppMessageInformation accompanies a negative response. However, there are cases where the ASPSP sends a positive response but still includes a tppMessageInformation. This might occur, when the TPP sends a status request and the request itselves is technically accepted but the requested status indicates some kind of banking processing issue/error or the requirement of additional action by the TPP or the PSU. In the same way, the requirement of additional actions can be indicated when (generally) accepting a payment initiation.