THE Berlin GROUP

A EUROPEAN STANDARDS INITIATIVE

openFinance API Framework

Basic Operational Rules of the Framework

Version 2.0

31 October 2025

[*] 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 From Core XS2A Interface to openFinance API

With [PSD2] the European Union has published a directive on payment services in the internal market. Among others [PSD2] contains regulations on services to be operated by so-called Third Party Payment Service Providers (TPP) on behalf of a Payment Service User (PSU). These 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 Funds Service (COF) to be used by a Payment Instrument Issuing Service Provider (PIISP) TPP as defined by article 65 of [PSD2].

To implement these services (subject to PSU consent), a TPP needs to access the account of the PSU. The account is managed by another PSP called the Account Servicing Payment Service Provider (ASPSP). To support the TPP in accessing the accounts managed by an ASPSP, each ASPSP has to provide an "access to account interface" (XS2A interface). Such an interface has been defined in the Berlin Group NextGenPSD2 XS2A Framework.

The XS2A Framework is now planned to be extended to premium services. This interface is addressed in the following as **openFinance API**. This openFinance API differs from the XS2A interface in several dimensions:

- The extended services might not rely solely on PSD2 anymore.
- Other important regulatory frameworks which apply are e.g. GDPR.
- The openFinance API can address different types of **API Clients** as access clients, e.g. TPPs regulated by an NCA according to PSD2, or corporates not regulated by an NCA.
- The extended services might require contracts between the access client and the ASPSP.
- While the client identification at the openFinance API can still be based on eIDAS certificates, they do **not** necessarily need to be PSD2-*compliant eIDAS certificates*.
- The extended services might require e.g. the direct involvement of the access client's bank for KYC processes.

Note: The notions of API Client and ASPSP are used because of the technical standardisation perspective of the openFinance API. These terms are analogous to "asset broker" and "asset holder" respectively in the work of the ERPB on a SEPA API access scheme.

Note: In implementations, several ASPSPs might offer their services in an aggregation platform. Such platforms will be addressed in the openFinance API Framework as "API provider".

Note: At some places, the openFinance API Framework is still using the technical TPP notion also for the openFinance APIs. In this context, the TPP is not necessarily a licensed TPP as mentioned above, but for certain services may just be any "third party provider".

The following account access methods are covered by this framework:

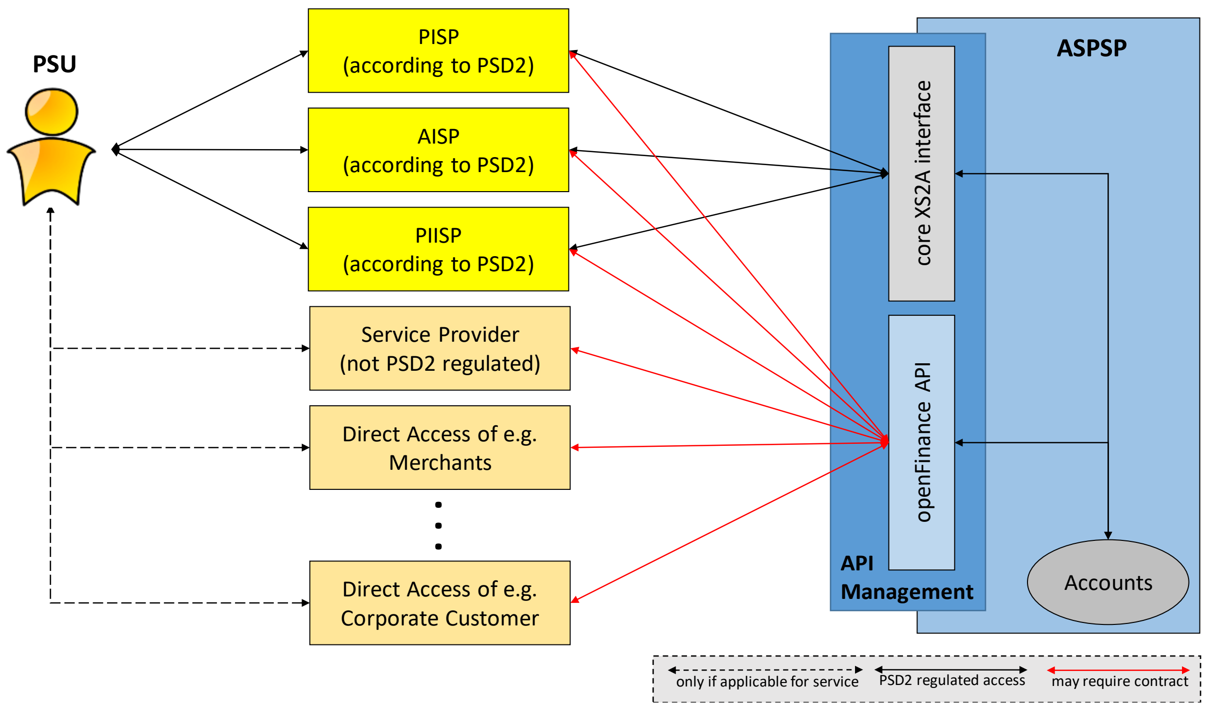
PSU
PISP (according to PSD2)
AISP (according to PSD2)
PIISP (according to PSD2)
Service Provider (not PSD2 regulated)
Direct Access of e.g. Merchants
Direct Access of e.g. Corporate Customer
ASPSP
core XS2A interface
openFinance API
API Management
Accounts
only if applicable for service
PSD2 regulated access
may require contract

Figure 1: Core XS2A interface and openFinance API

The ASPSP may restrict access to the services offered at its openFinance API and requires dedicated onboarding. The requirements for the rights to access the services offered at the openFinance API are out of scope for this document. These requirements are in detail in [oFA-OR-ADM].

In contrast to the services of the openFinance API, the ASPSP has to offer access to the services of its core XS2A API to any TPP without discrimination as long as the TPP has the necessary licence to access a service as PISP, AISP or PIISP from an NCA according to the regulations of [PSD2].

1.2 Protocol Functions supported by XS2A and openFinance API

The addressed services within the XS2A API and the openFinance API share many protocol functions. Such functions are defined in this Framework document and cover e.g.

- how to deal with error situations,
- dynamic protocol steering by hyperlinks,
- how to handle protocol preferences of the API Clients, and
- how to push resource status changes to API clients.

Such generic protocol functions are explained within this document on an abstract level and in [oFA-PFSM] in technical detail. Service specifications on the operational rules level using these functions then just refer to this openFinance API Framework documentation for better readability. The document refers in most parts to the openFinance API Framework only, instead of addressing both the XS2A API and the openFinance API. Where needed, the two APIs are addressed explicitly.

1.3 Security Measures supported by XS2A and openFinance API

Specific API protocol functions are security measures to be supported by an API. XS2A and openFinance API support different security measures and approaches for the authentication of parties, as well as for cryptographic means like encryption. These specific protocol functions are a.o.

- how to support API Client and ASPSP authentication,
- how to support message encryption,
- how to support the different SCA approaches, and
- how to proceed with an authorisation process.

Some of these security measures and approaches for PSU authentication have to be supported by an API mandatorily, for others, the support is optional for the ASPSP. Service specifications will define this on the operational rule level and refer to this document for details. As above, technical details on security measures are defined in [oFA-PFSM].

1.4 Document Structure

This document specifies the fundamental operational rules for the openFinance API Framework in the context of message exchange using the API. It must be considered in conjunction with any other Operational Rules published within the openFinance API

Framework and is further technically detailed in [oFA-PFSM]. In this document, the following topics are presented:

- Different scenarios to access the API of an ASPSP, the actors involved in this and their respective roles, and as a result of this, the different kinds of notions of an API Client in Section 2,
- Key concepts of message exchange at the API, including the dynamic protocol steering, encryption, strong customer authentication (SCA), and signing baskets in Section 3,In particular, the introduction of two very important kinds of messages: Transaction Initiation Request and Service Resource Creation Request in Sub-Section 4.3,
- Fundamental operational rules for these concepts in Section 5, and
- The Message and Data Model in Section 6.

1.5 Document History

Version	Change/Note	Approved
2.0	Initial version of the general Operational Rules document of the openFinance API framework	31 October 2025

Table 1: Document history

2 Actors and roles

2.1 Third-party processor (TPP) related scenario

In general, services offered by an ASPSP at its openFinance API may be accessed/used not only by clients registered by an NCA in the role of a TPP according to the PSD2 regulation, but by any third party. Nevertheless, for Extended Payment Initiation Services or Extended Account Information Services, this might still apply for many subservices due to the current regulation.

The PSU will never directly access the API itself in this scenario; however, the PSU’s consent is still crucial for the TPP and the ASPSP in order to execute services for the PSU. Therefore, the parties PSU – TPP – ASPSP establish a triangular relationship to ensure the PSU's consent and to execute the requested API service.

The PSU communicates with the TPP and explicitly or implicitly requests a service that the TPP shall execute on behalf of the PSU. Then, the TPP contacts the ASPSP via the API and initiates the execution of the service. A service process consists of a sequence of messages exchanged between the ASPSP and the API Client using the API. For certain steps in such a process, the PSU’s consent needs to be proven by the TPP and verified by the ASPSP. For this, direct contact is orchestrated by the ASPSP between the PSU and the ASPSP and does not involve the API (this process is initiated via a message exchange between the TPP and the PSU).

In order for a TPP to execute a service at the XS2A interface, it must inherit a corresponding role, and this must be recorded via a TPP's certificate as well. For example, a TPP may initiate payment services at the XS2A interface only if it is certified as a payment initiation service provider. The roles for payment initiation service providers and account information service providers are not exclusive, and a TPP may be one of the respective roles or both. For the extended services, this is not explicitly regulated, but still might apply to many subservices. If not specified further, the term TPP just refers to a general TPP without assigning it a certain role.

Note: An ASPSP can also act as a TPP by taking one of the respective roles. In particular, the ASPSP may offer extended services to a PSU and can access the openFinance API of other ASPSPs. Thus, the role of an ASPSP (as indicated in the related certificate) also includes the role of a TPP.

2.2 Direct access scenario

The openFinance API Framework now develops a specification to reuse the TPP – ASPSP openFinance API also as a PSU – ASPSP interface, at least for the corporate case, where the broad functionality like multi-signing etc. applies. The direct access potentially addresses all (business) services defined in the openFinance API Framework. The technical API client system in this case is either a client software hosted by the corporate itself or by e.g. an ERP cloud provider.

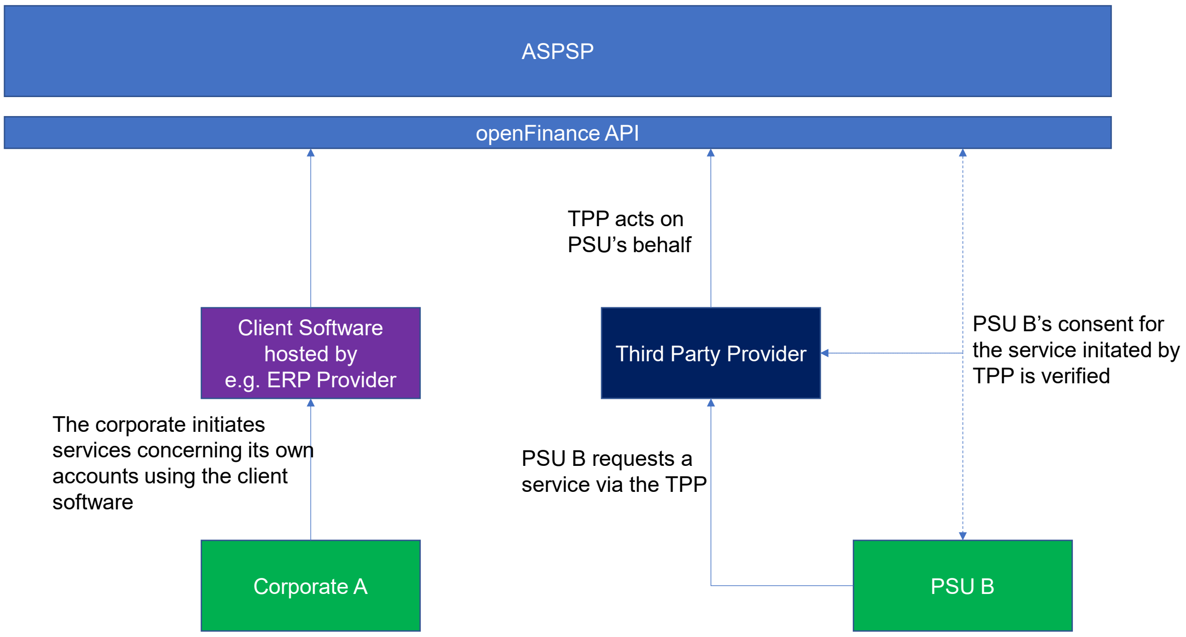
ASPSP
openFinance API
TPP acts on PSU's behalf
Client Software hosted by e.g. ERP Provider
Third Party Provider
PSU B's consent for the service initated by TPP is verified
The corporate initiates services concerning its own accounts using the client software
PSU B requests a service via the TPP
Corporate A
PSU B

Figure 2: Difference of usage of a Client Software by Corporate A and usage of a TPP acting on behalf of PSU B

An ERP cloud provider allows a corporate client to access the openFinance API and act on it directly. The corporate is the immediate actor and driving force in all interactions at the API. A TPP acts on behalf of the PSU but as a third party, meaning it may initiate services requested by the PSU but the PSU does not access the API itself. All interactions at the API are managed by the TPP. The TPP may also act on its own without the PSU's immediate involvement, e.g. to receive the status of a previously established resource or to cancel a transaction initiation. As the PSU is not directly involved at the service transactions at the API, the PSU's consent must be verified explicitly.

2.3 Notion of API Client

The use of the openFinance API enables the issuance of premium services at the ASPSP's API within already well-established standards. Contrary to the XS2A interface, the openFinance API is not regulated via the [PSD2] or a similar directive. In the TPP-related scenario, the ASPSP may link access to its openFinance API to certain conditions. Therefore, actors accessing the API are not necessarily regulated TPPs but general clients to the API. With the direct access scenario, this concept is expanded even further, allowing the PSU to directly access the API itself. Strictly speaking, the PSU – technical API client – ASPSP triangle still exists in some way, since the PSU initiates the requests itself using a client software or software as a service via a cloud system, which functions as a technical API client, to gain access to the API. In this situation, the division between PSU and API client is not as clear/strict as it is in the first case, but still makes sense to clearly separate actors and roles clearly.

In the following, we will use the term "**API Client**" as a generalisation from both – the TPP in a TPP scenario or a technical API client in the direct access scenario, since in most cases, the services offered via the openFinance API Framework can be potentially addressed in both scenarios. The notion of API Client allows for subsuming both scenarios in one wording.

3 API Services supported at the openFinance API on Framework Level

The following table lists the services currently supported at the openFinance API that are introduced in this document:

API Services	Service Type	Technical Functionalities	PSU directly involved
Grouping API Services into a Signing Basket	SGNB	Creating a signing basket resource with a corresponding basket-id that bundles transactions, which are already initiated but not yet authorised The bundled transactions can still be addressed individually and managed without using the basket structure Using the basket-id, the API Client may initiate the authorisation of the signing basket, leading to the authorisation for all of its entries	Yes
		Retrieve information which transactions are grouped together in the signing basket resource	No
		Retrieve the status of the signing basket	No
		Cancel the signing basket resource, this only effects the signing basket structure and not the transactions grouped bundled inside it	No

Table 2: API services at the openFinance API

4 Key concepts of the openFinance API

This section contains an overview of the key concepts of an openFinance API implemented according to the specification of the Joint Initiative. For the detailed specification, please refer to the documents [oFA-PFSM] and the implementation guidelines of the respective services.

4.1 Layers of the interface

There is a specification between two kinds of layers at the openFinance API: the transport layer and the application layer. Their usage depends on the kind of message which is exchanged.

The services of the API are specified by the exchange of defining messages and data elements between the API Client and the ASPSP. This whole exchange at the API defines the application layer.

On the transport layer, the technical exchange of messages takes place. The communication is always secured by a TLS-connection.

Among other things, the Implementation Guidelines of the organisational framework [oFA-PFSM] details the rules and processes for the exchange management and for the further development on the technical level.

4.2 Messages

Messages are the basic building blocks for the execution of processes at the API. A message is uniquely defined by its set of parameters (i.e. data elements) which are transported. These data elements determine the service explicitly and therefore a message refers to a unique service action.

It is distinguished between request messages and response messages. In most cases[2], a request message is sent by an API Client to the ASPSP as an HTTPS request addressing the openFinance API of the ASPSP. After executing an incoming request message, the ASPSP will send the corresponding HTTPS response message to the API Client.

[2] The openFinance API Framework also allows the exchange the roles of server and client for pushing some information from the ASPSP to the API Client. This is in full analogy and will be addressed specifically in the definition of related push functions or services.

For the implementation of an API, each request message will be executed using either POST, GET, PUT or DELETE HTTP methods. When specifying the related HTTP methods in detail, the following technical levels are distinguished, which are explained in detail in [oFA-PFSM]:

- Path parameters:

  Are addressing specific products of a related service and the service resources, see below.

- Query parameters:

  Are mainly used when applying filtering in reading data from the openFinance API.

- Header parameters:

  Are mainly used for steering the protocol, by e.g. dealing with API Client preferences for SCA processing, subscribing dynamically to status push functions etc.

- Payload (message body):

  Mainly the actual service business content to be transported, as well as some information on the data resources and metadata in the ASPSP in response messages.

4.3 Message sequences

In most cases, an API service consists of a sequence of messages exchanged between the API Client and the ASPSP. Often, the first message sent by the API Client to the ASPSP is particularly relevant and leads to the generation of a resource with related resource identification, which enables the API Client to address the newly created resource later on and to continue the execution of the corresponding service. Depending on the kind of service, there are two distinct types of messages that are used as initial start: **Transaction Initiation Requests** and **Service Resource Creation Requests**. These messages come with a lot of service-independent parameters and handling throughout the openFinance API Framework. These definitions are then used to simplify the service definitions and concentrate on service -specific definitions in related service documentations.

Transaction Initiation Request and Response

The Transaction Initiation Request refers to a service initiation request, where the PSU needs to authorise/provide consent with the addressed service by applying potentially Strong Customer Authentication via the openFinance API. Examples are a Payment Initiation

Request, an Establish Consent Request, an Establish Signing Basket Request, or an Initiate Subscription Request, for a specific transaction service.

Besides transporting the related business information, the Transaction Initiation Request and Response messages contain specific parameters, which also contain context data and data steering the usage of openFinance API Framework functions, specifically around complex integration of applying SCA. The specific header parameters of a Transaction Initiation Request and the header and body parameters of the corresponding Transaction Initiation Response are listed in the table below. The usage of certain header parameters in the request might lead to the usage of specific header parameters in the response.

In general, after receiving a Transaction Initiation Request, the ASPSP creates a resource together with a related resource identification. Besides this data, the corresponding Transaction Initiation Response message contains many hyperlinks to indicate the possible next steps to be taken to authorise the transaction. These hyperlinks are independent of the actual service initiated. They are defined in a general context and described in detail in [oFA-PFSM]. These links may lead to initiate the authorisation process with the PSU, the option for the API Client to select the authorisation method, the transaction initiation resource itself so that the API Client can retrieve the resource data, the option to retrieve the status of this initiated transaction, and/or to the option for the API Client to check the SCA status of the corresponding resource. Thus, the following steps of the service can then be mediated and controlled by the API Client at the API. This is further instantiated to service definitions explicitly only via OpenAPI files.

In this way, the Transaction Initiation Request is particular for a message exchange at the API. It unites the start of an API service, the creation of a corresponding resource (e.g. a payment resource) and the hyperlinks allowing the API Client to steer the next service steps dynamically in one message.

Note: In this context, the notion "transaction" is used as a meta description for a complex service interaction with the API, which needs to be authorised by the PSU, normally by applying SCA. This document also uses the term transaction to describe a banking transaction in the context of AIS services. This document only uses the term transaction when the meaning is apparent from the context.

Service Resource Creation Request and Response

The notion of a Service Resource Creation Request refers to a service request, where a service resource is created, but only some restricted API interaction features are supported for this resource at the openFinance API, for example a push resource status function. This notion is used especially when no SCA can be applied via the openFinance API for the addressed service. Examples of Service Resource Creation Requests include a Document Submission Request or a Request to Pay Request.

The Service Resource Creation Request/Response exchange contains specific header and body parameters, leading to the creation of a resource, and allowing the API Client to access the resource to receive and exchange information of the ASPSP using the resource (i.e. like a push resource status function). The table below yields an overview of the respective parameters used in the request and response messages.

In general, after receiving a Service Resource Creation Request, the ASPSP creates a resource together with a related resource identification. In addition to this data, the corresponding Service Resource Creation Response message contains some hyperlinks. Examples of these hyperlinks are hyperlinks to the resource itself or to the status of the related resource, so that the API Client can receive further information and might mediate or control services based on this information.

Due to its nature, the PSU may not be involved to give its consent via the API at any point in order for the Service Resource Creation Request to be processed.

Note: The addressed API services normally still require a consent between the PSU and a TPP in the TPP scenario. This consent is referred to as a mandatory feature in PSD2 compliance services. In premium APIs, this consent might be handled in a pre-step with or without the involvement of the ASPSP.

The following table gives an overview of the request and response headers and body parameters of these two kinds of messages, which are specific to these types of messages and are available independently of the service. Depending on the type of message, a complete overview of the parameters can be found in Section 6.2 and Section 6.3, respectively.

Type of message	Request Header	Response Header	Response Body
Transaction Initiation	• PSU Identification Data, • PSU Corporate Identification Data, • PSU Context Data, • SCA Approach Preferences, • Redirect URI, • Client Notification Data, • Client Brand Information	• ASPSP SCA Approach • ASPSP Notification Data	• Service resource identification, • Service resource status, • PSU message information, • API Client message information, • Available SCA methods, • SCA challenge data, • Hyperlinks that lead to next possible steps and can be used by the API Client, e.g. o Redirect URL ASPSP, o To initiate the authorisation, o To the created resource, o To the resource status, o To the SCA status of the resource
Service Resource Creation	• Client Notification Data, • Client Brand Information	• ASPSP Notification Data	• Service resource identification, • Service resource status, • PSU message information, • API Client message information, • Hyperlinks that lead to next possible steps and can be used by the API Client, e.g. o To the created resource, o To the resource status

Table 3: Header and Body parameters of Transaction Initiation and Service Resource Creation messages

The possible links listed in the response's body may vary depending on the service. In general, the response headers may vary depending on the headers used in the original request, and may also depend on the actual implementation, specifically on the SCA approaches supported, cf. 4.8.

4.4 Dynamic protocol steering by hyperlinks

The services defined within the openFinance API Framework require several requests from the API Client towards the ASPSP. When a Transaction Initiation Request or a Service Resource Creation Request is issued, the ASPSP generates a resource presentation, and the corresponding response message contains a link to the created resource.

Additionally, the ASPSP can add a list of hyperlinks, so that all succeeding requests within the services may be initiated via a hyperlink. These hyperlinks follow the URI references as defined in [RFC3986]. It is possible that a single link or multiple links are sent back to the API Client than specified in the related call. The openFinance API Framework defines the semantics of the hyperlinks to a large extent.

The hyperlinks may lead to the necessary next steps of the service and are sometimes generic. For example, in the case of a Transaction Initiation Response, they can lead to the following processes, independent of the actual service:

- Starting the authorisation process of the PSU (with and without password encryption),
- Updating the authentication of the PSU (again with and without password encryption),
- Selecting the authorisation method (selected by the API Client or by the PSU),
- Getting status information about the transaction or about the SCA, and
- The resource created by the request, to be used for
- Receiving resource data of the newly created resource.

The Service Resource Creation Response must include at least hyperlinks to the last two bullet points, i.e. a hyperlink to the newly-created service resource itself and a hyperlink to the status of this resource, in order to retrieve status information.

4.5 Error Handling

In general, the same rules regarding the presence of header elements apply to both positive and negative response messages of any kind. In the case that an error occurred before functional processing took place, this might be omitted. The ASPSP may send additional error information to the API Client within a request/response exchange, sending HTTP 4xx and 5xx codes or (in some cases) 2xx response codes. For this, together with the HTTP code, the ASPSP can send additional and more granular error information via a "Message Code" as defined in [oFA - DD].

At the openFinance API, the ASPSP has the option to send an error message that includes a free-text field, where the error context or actions to be taken by the API Client are described. It might contain a link leading the API Client to the next step in order to avoid further errors.

4.6 Identification of the API Client

Only after an API Client has identified itself at the API, it may access the API of an ASPSP. The identification process depends on the access scenario, among other things.

4.6.1 Identification of the API Client in a TPP-related scenario

Due to the current regulation, for many premium services at the openFinance API (e.g. Extended Payment Initiation Services), some parts of the PSD2 regulation still apply as it does for the core services at the XS2A interface.

In general, the premium services offered through the openFinance API are not necessarily regulated by the [PSD2]. Accessing TPPs are identified similarly, but without mandating PSD2-specific requirements. The ASPSP may mandate access to its openFinance API itself by demanding different/additional requirements (for example, by maintaining a directory of all allowed TPPs on its openFinance API). The technical details of the identification process of the TPP (at both interfaces) are explained in [oFA-PFSM].

In the context of the identification of a TPP at any API it is distinguished between two levels:

- Identification of the TPP at the transport layer, and
- Identification of the TPP at the application layer.

The TPP must always be identified at the transport layer. At the transport layer, the TPP is identified by means of the client authentication which is part of the TLS-connection setup between the TPP and the ASPSP.

The ASPSP can additionally define in its API that a further identification of the TPP at the application layer is necessary. In this case, the TPP needs to sign all messages to be sent to the ASPSP at the application layer to authenticate itself. The electronic seal (i.e. the electronic signature) and the certificate of the TPP have to be sent to the ASPSP as part of these messages.

Clarification: If a TPP uses further technical service provider(s) to implement its accesses to the API of an ASPSP, the setup of the TLS-connection with the ASPSP and the electronic seal shall be generated by the TPP and the certificate of the TPP shall be sent to the ASPSP. The certificate of a technical service provider(s) may not be used.

The ASPSP will reject the service if the API Client cannot be identified correctly at the API and/or, in the case of a TPP, if the certificate used by the TPP to identify itself does not list the role required for this service or is not listed in related service directories respectively.

4.6.2 Identification of the API Client in a direct access scenario

In a direct access scenario, the requirements discussed above may not apply. The ASPSP can decide freely if and, if so, which requirements shall be fulfilled and which certificates shall be used for accessing its API and does not have to follow any predetermined regulations. At the same time, different technical scenarios have to be supported.

For the Request to Pay Services, the direct access is applicable specifically for the interface RTP-Originator and the Originator-RTPSP. This case is technically easy since no authorisation process, i.e. no SCA, is involved in the API as such. For these services, the direct access of a PSU to its ASPSP functioning as its Receiver-RTPSP is already defined by the PUSH AIS services.

4.6.2.1 Client TLS Certificate

Corporate connections need to support the direct connections of corporates and connections via a cloud service, e.g. hosted by an ERP provider. This should be reflected in the usage of client certificates for TLS:

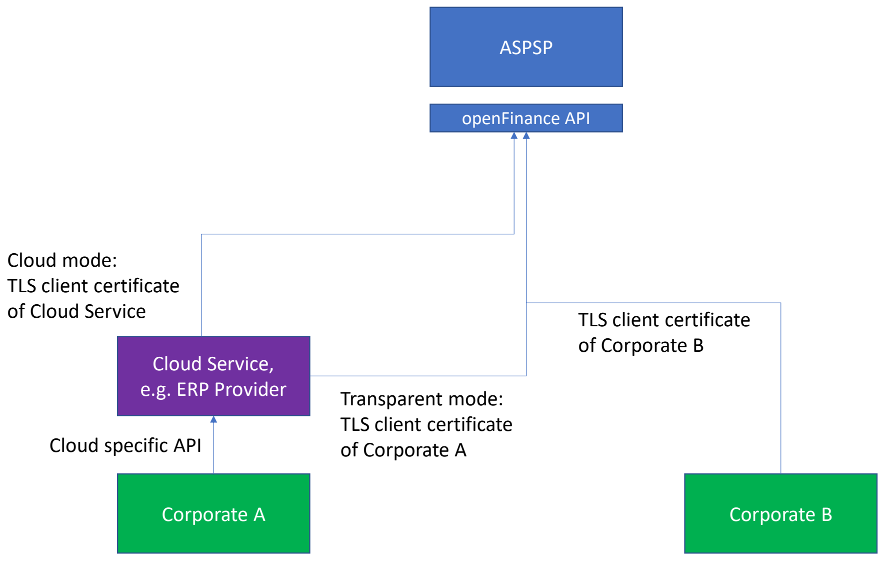

Figure 3: Usage of client TLS certificates

In a cloud service scenario, it should be possible to identify the cloud API client by a client certificate of the cloud provider as well as a client certificate of the corporate. In the latter case, the cloud provider is working in a "transparent mode".

Note: Alternative solutions allowing also Bearer tokens as client identifications, which might be used e.g. in an SME scenario, might be covered in future in [oFA-IG-UAM].

4.6.2.2 Corporate Seal for Message Signing

Sealing of HTTP messages where required will always be performed with a corporate seal (i.e. using a certificate with the corporate entity as certificate holder), and **not** by a seal of the cloud provider. So, this sealing also provides corporate identification.

See section 6 of [oFA-PFSM] for technical details of sealing HTTP messages.

Remark: A corporate can allow its cloud provider to access and use its certificate and the corresponding key pair so that the cloud provider can seal HTTP messages on behalf of the corporate. Rights and obligations in dealing with the certificate and the key pair, in particular with regard to their security, as well as the protocols to be used, must be regulated bilaterally between the corporate and the cloud provider. This is not part of this document.

4.7 Encryption

In general, parts of the body or the complete body of a message exchanged between the API client and the ASPSP can be encrypted. The API Client and the ASPSP must follow the rules listed in the table underneath for the implementation of encryption. Among other factors, these rules also depend on the type of message. Via the API access scheme, the ASPSP might demand further requirements for the encryption of messages.

Type of message	API Client	ASPSP
Request message send by the API Client to the ASPSP	• Shall support the encryption of (parts of) the body, if the addressed ASPSP supports it • If the ASPSP requests it, the API Client shall encrypt (parts of) the body	• May support the decryption of (parts of) the body
Response message send by the ASPSP to the API Client	• May support the decryption of (parts of) the body	• May support the encryption of (parts of) the body • Only if the API Client accepts encrypted (parts of) the body, the ASPSP shall encrypt (parts of) body
Request message send by the ASPSP to the API Client (for push based services)	• May support the decryption of (parts of) the body	• May support the encryption of (parts of) the body • If the API Client accepts encrypted (parts of) the body, the ASPSP shall encrypt (parts of) the body
Response message send by the API Client to the ASPSP (for push based services)	• Currently not applicable	• Currently not applicable

Table 4: Rules for encryption required by the API Client and the ASPSP

Further details on the encryption process like how to address encryption profiles, etc. are specified in [oFA-PFSM].

4.8 Strong customer authentication

Transaction Initiation Requests are used for services which require Strong Customer Authentication (SCA) of the PSU at the openFinance API as part of the service, e.g.:

- Payment initiation request,
- Cancellation initiation request (depending on the regulations for this kind of requests at the user interfaces offered by the ASPSP to the PSU directly),
- Establish account information consent request.

Due to current regulations, it might occur that for the authorisation of a premium service, the PSD2 regulations still apply, as they do for the core services. The requirements for the application of SCA and possible exemptions are defined in Article 97 of [PSD2] and Chapter III of [RTS].

The same requirements might be demanded by the ASPSP itself for some/all extended API services, for example, to ensure the ASPSP's own risk management concerning the respective service. It may also be offered by the ASPSP towards the API Client as a service to guarantee a certain level of security for the API Client.

For each individual request, the ASPSP has to decide whether SCA needs to be executed. This decision has to be compliant with the requirements defined by [PSD2] and [RTS] for all services of the openFinance API Framework which fall under these regulations.

If a SCA is necessary, then it has to be decided on

- the procedure and
- the personalised security credentials

to be used by the PSU. If multiple SCA procedures are available for the PSU, the ASPSP shall offer these procedures to the API Client/PSU to choose from.

The framework of the openFinance API and of the XS2A interface is supporting the following SCA approaches for the API Client interaction:

- Redirect approach,
- OAuth2 approach,
- Decoupled approach,
- Embedded approach, and

- ASPSP Channel approach.

In a direct access scenario, corporates often use digital signatures as authentication measure within the authorisation process: Transactions need to be signed by one or more persons of a corporate within the corporate IT systems, depending on the related signing rights of the involved persons, before the transactions are submitted to and further processed by the corporate ASPSP.

Thus, a new SCA approach is introduced for the openFinance API Framework, which is called

- Digital Signature SCA approach.

For this additional SCA approach, the SCA interaction is replaced by a signed request (API request with signed body).

The ASPSP decides which of these SCA approaches are supported by its API implementation. The API Client can indicate in the Transaction Initiation Request which SCA approaches the API Client prefers in the form of a list, going from most preferred to least preferred, respectively not listed. In this context, the OAuth2 approach is also seen as a technically redirect-based SCA approach. The ASPSP may consider this indication when deciding on the SCA approach to be performed.

The openFinance API documentation of the ASPSP will provide the API Client with the necessary information. If SCA is necessary as part of an API service, the API Client has to use one of the approaches supported by the ASPSP.

Note: SCA of one or more PSUs for a single API service is supported at the openFinance API.

Note: Currently, only the SCA as performed by the ASPSP is directly supported as part of the service at the openFinance API. SCA methods offered by the API Client towards the PSU are not yet explicitly supported by the openFinance API. Future releases may support this approach for SCA ("delegated approach") at the openFinance API.

Note: The new Digital Signature SCA Approach is introduced as part of the corporate direct access scenario. Of course, this Digital Signature SCA Approach could also be used in any other API Client scenario.

4.8.1 SCA using the redirect approach

For the redirect approach, the individual steps of the SCA are not executed at the API, but directly between the PSU and the ASPSP. In this case, the PSU is redirected to a web interface of the ASPSP for authentication. Depending on the device used, the PSU may also be redirected to a special authentication app of the ASPSP (see the next section).

Once the PSU has been redirected to the ASPSP (app or web interface), the SCA of the PSU is executed step by step and directly between the ASPSP and the PSU. After completion of the SCA, the PSU is redirected back to the API Client. The following figure shows the (much simplified) top-level information flow for a payment initiation service with SCA based on the redirect approach as an example:

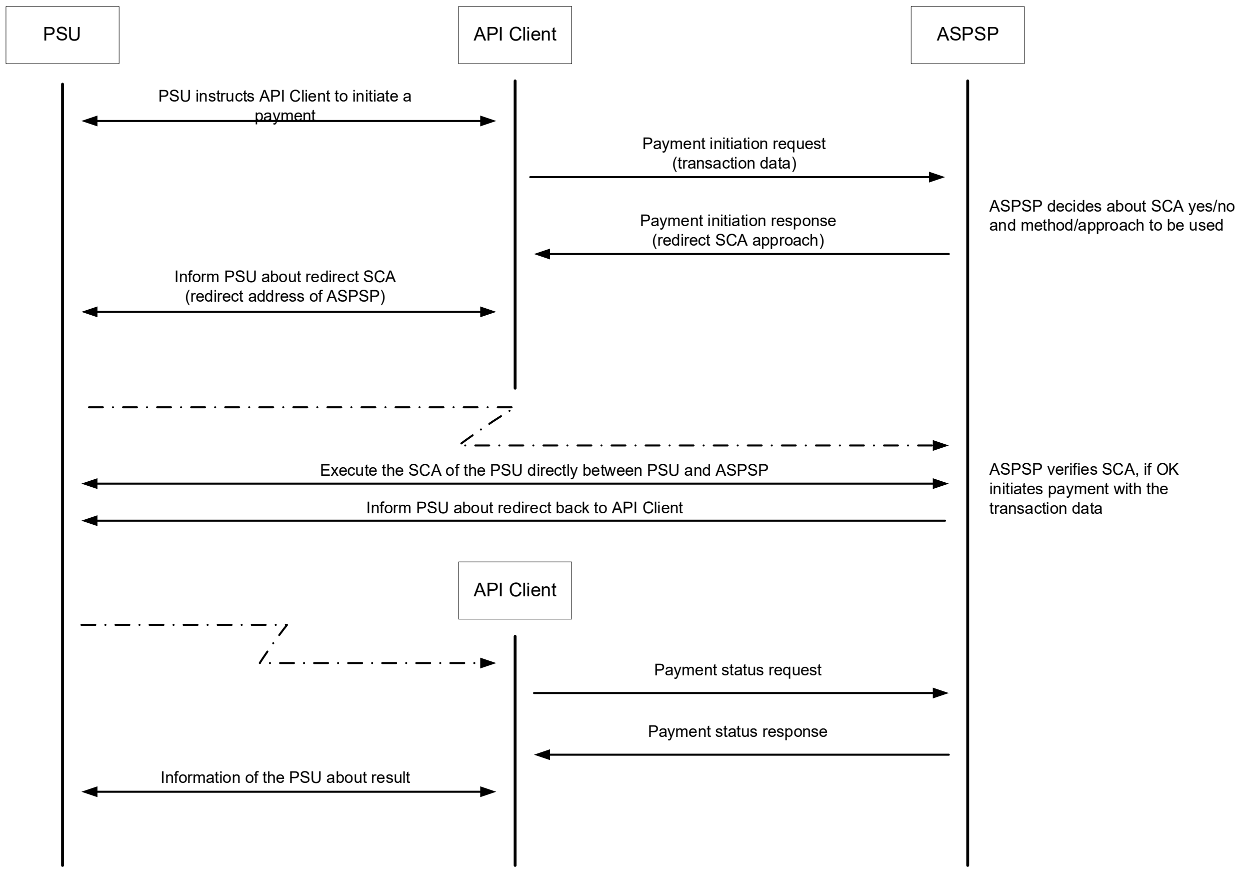
PSU
API Client
ASPSP
PSU instructs API Client to initiate a payment
Payment initiation request (transaction data)
ASPSP decides about SCA yes/no and method/approach to be used
Payment initiation response (redirect SCA approach)
Inform PSU about redirect SCA (redirect address of ASPSP)
Execute the SCA of the PSU directly between PSU and ASPSP
ASPSP verifies SCA, if OK initiates payment with the transaction data
Inform PSU about redirect back to API Client
API Client
Payment status request
Payment status response
Information of the PSU about result

Figure 4: Redirect approach for SCA

When applying the redirect approach, the API Client does not need detailed information about the individual steps of the SCA of the PSU. The redirect approach therefore allows the API Client to avoid the implementation of the different SCA methods at its PSU – API Client interface.

4.8.2 SCA using the integrated OAuth2 approach

The information flow of the OAuth2 approach to SCA is similar to that of the redirect approach. The difference is that the redirection to the authentication server of the ASPSP is embedded into the OAuth2 protocol, where the "scope" attribute of the OAuth authorisation request is linked to the created transaction resource. The access to the interface to retrieve the actual account data is then performed by using the access token delivered by the ASPSP's authentication server. The following figure shows the (much simplified) top-level information flow for a payment initiation service with SCA based on the OAuth2 approach as an example:

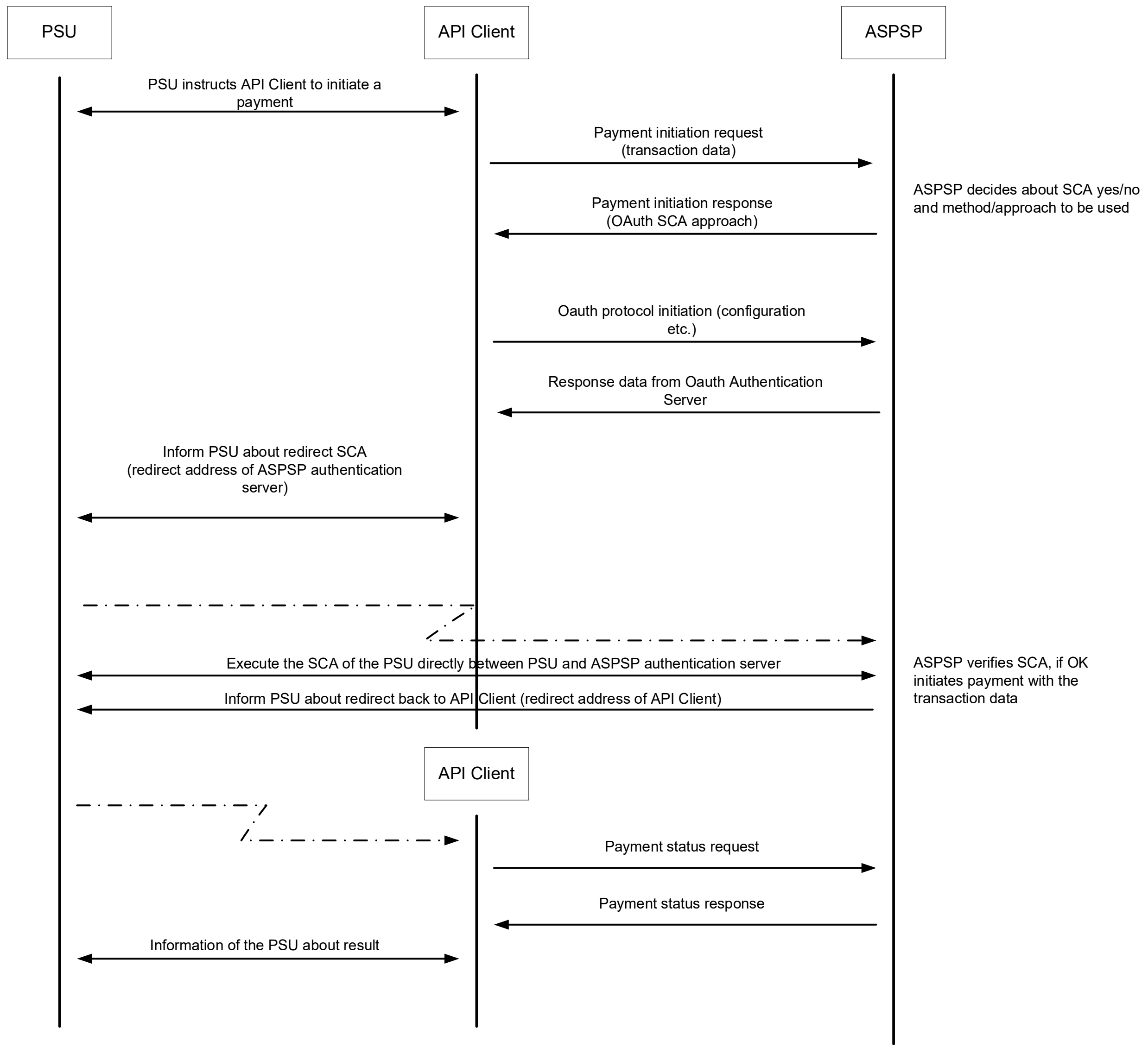
PSU
API Client
ASPSP
PSU instructs API Client to initiate a payment
Payment initiation request (transaction data)
ASPSP decides about SCA yes/no and method/approach to be used
Payment initiation response (OAuth SCA approach)
Oauth protocol initiation (configuration etc.)
Response data from Oauth Authentication Server
Inform PSU about redirect SCA (redirect address of ASPSP authentication server)
Execute the SCA of the PSU directly between PSU and ASPSP authentication server
ASPSP verifies SCA, if OK initiates payment with the transaction data
Inform PSU about redirect back to API Client (redirect address of API Client)
API Client
Payment status request
Payment status response
Information of the PSU about result

Figure 5: OAuth approach for SCA

When applying the OAuth SCA approach, the API Client does not need detailed information about the individual steps of the SCA of the PSU. The OAuth approach therefore allows the API Client to avoid the implementation of the different SCA methods at its PSU – API Client interface.

4.8.3 SCA using the decoupled approach

The information flow of the decoupled approach to SCA is similar to that of the redirect approach. The difference is that the ASPSP asks the PSU to authenticate e.g. by sending a push notification with payment service details to a dedicated mobile app or via any other application or device which is independent of the online banking frontend. In contrast to the redirection flow, there is no impact on the PSU/API Client interface during the technical processing. The following figure shows the (much simplified) top-level information flow for a payment initiation service with SCA based on the decoupled approach as an example:

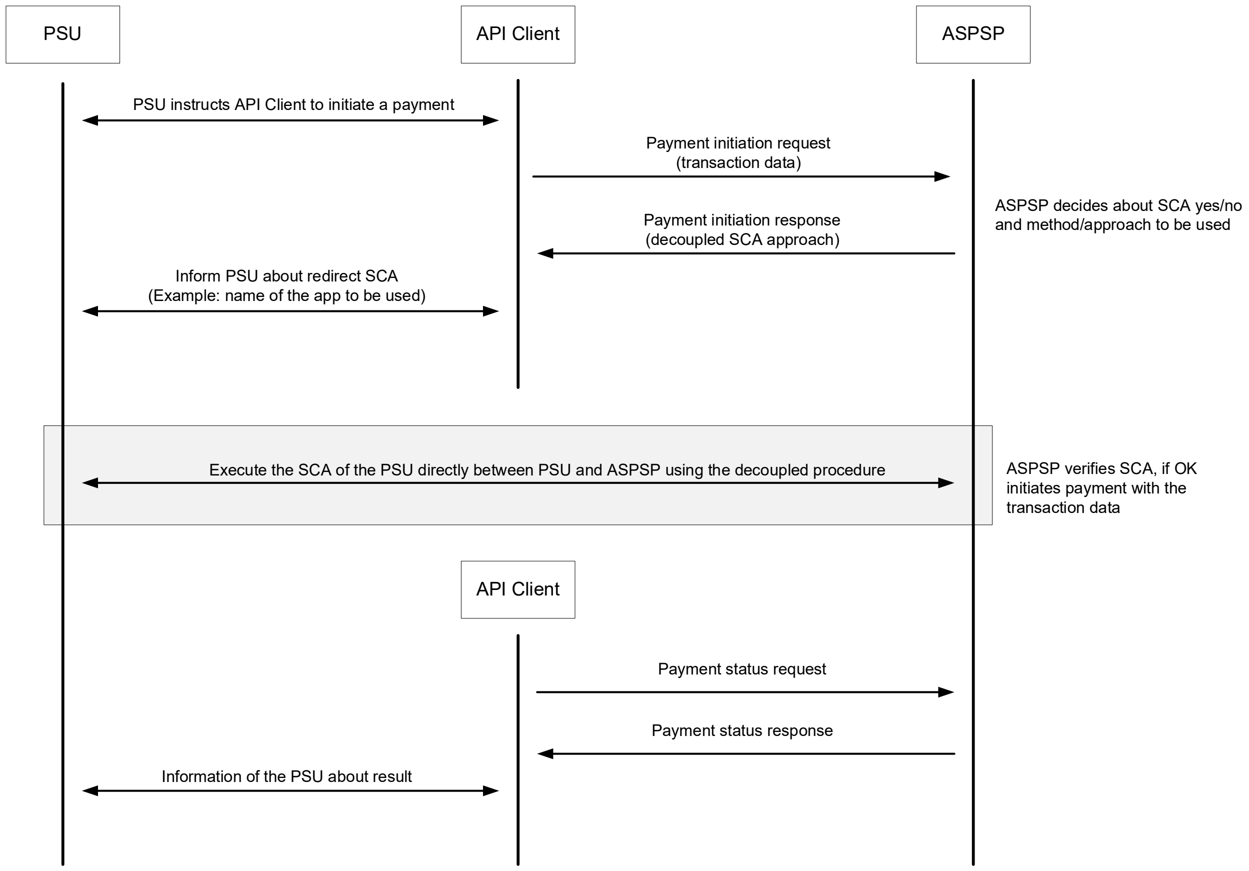
PSU
API Client
ASPSP
PSU instructs API Client to initiate a payment
Payment initiation request (transaction data)
ASPSP decides about SCA yes/no and method/approach to be used
Payment initiation response (decoupled SCA approach)
Inform PSU about redirect SCA (Example: name of the app to be used)
Execute the SCA of the PSU directly between PSU and ASPSP using the decoupled procedure
ASPSP verifies SCA, if OK initiates payment with the transaction data
API Client
Payment status request
Payment status response
Information of the PSU about result

Figure 6: Decoupled approach for SCA

Just as for the redirect approach, the API Client does not need to have detailed knowledge about the individual steps of the SCA when applying the decoupled approach. The decoupled approach therefore allows the API Client to avoid the implementation of the different SCA methods at its PSU – API Client interface.

4.8.4 SCA using the embedded approach

When applying the embedded approach, the SCA of the PSU is executed entirely as part of the request/response exchange at the openFinance API. The following figure shows the (much simplified) top-level information flow for a payment initiation service with SCA based on the embedded approach. The basis for this example is a SCA method based on a static password of the PSU (as an element of knowledge) and a dynamic one-time password (OTP) calculated by the PSU, for example using a smart card (as an element of possession).

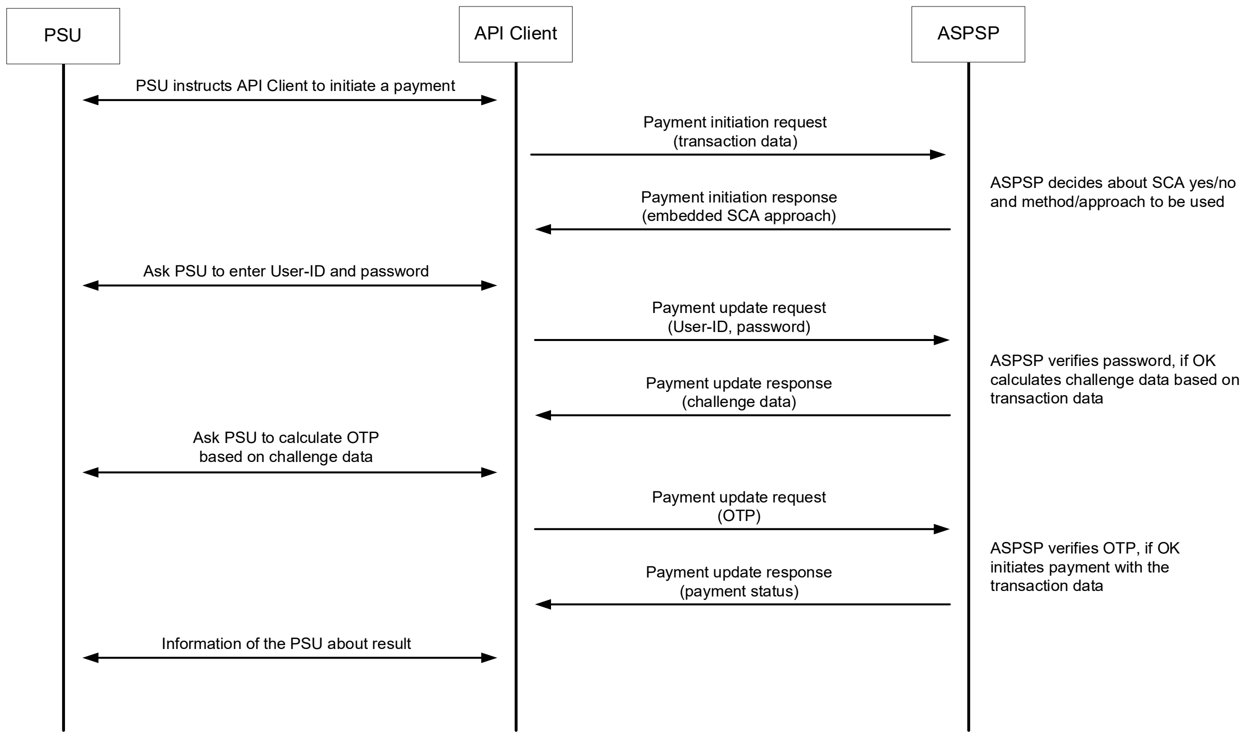
PSU
API Client
ASPSP
PSU instructs API Client to initiate a payment
Payment initiation request (transaction data)
ASPSP decides about SCA yes/no and method/approach to be used
Payment initiation response (embedded SCA approach)
Ask PSU to enter User-ID and password
Payment update request (User-ID, password)
ASPSP verifies password, if OK calculates challenge data based on transaction data
Payment update response (challenge data)
Ask PSU to calculate OTP based on challenge data
Payment update request (OTP)
ASPSP verifies OTP, if OK initiates payment with the transaction data
Payment update response (payment status)
Information of the PSU about result

Figure 7: Embedded approach for SCA

Again, the interface PSU – API Client is not covered by this document. When using the embedded approach, the API Client has to know how to inform the PSU about the different steps of the SCA (in contrast to the redirect or decoupled approach). In particular, the API Client has to provide a means of displaying the challenge data to the PSU, as the PSU needs this data to calculate the OTP. Depending on the ASPSP, there is a large number of possible methods for this, for example the display of an animated graphic or the display of a black and white or coloured matrix code. The openFinance API documentation of the respective ASPSP will contain a specification of the steps of the SCA methods. The challenge data sent by the ASPSP will contain an identification of the SCA method to be used.

4.8.5 SCA using the ASPSP Channel approach

The ASPSP might offer the API Client to authorise an API service not within the current ASPSP/API Client session, but instead later via an online channel of the ASPSP. Then the ASPSP uses its online channel to let the PSU authorise the requested action. The following figure shows the (much simplified) top-level information flow where the approach for SCA based on the ASPSP Channel is used for initiating an (electronic) direct debit mandate:

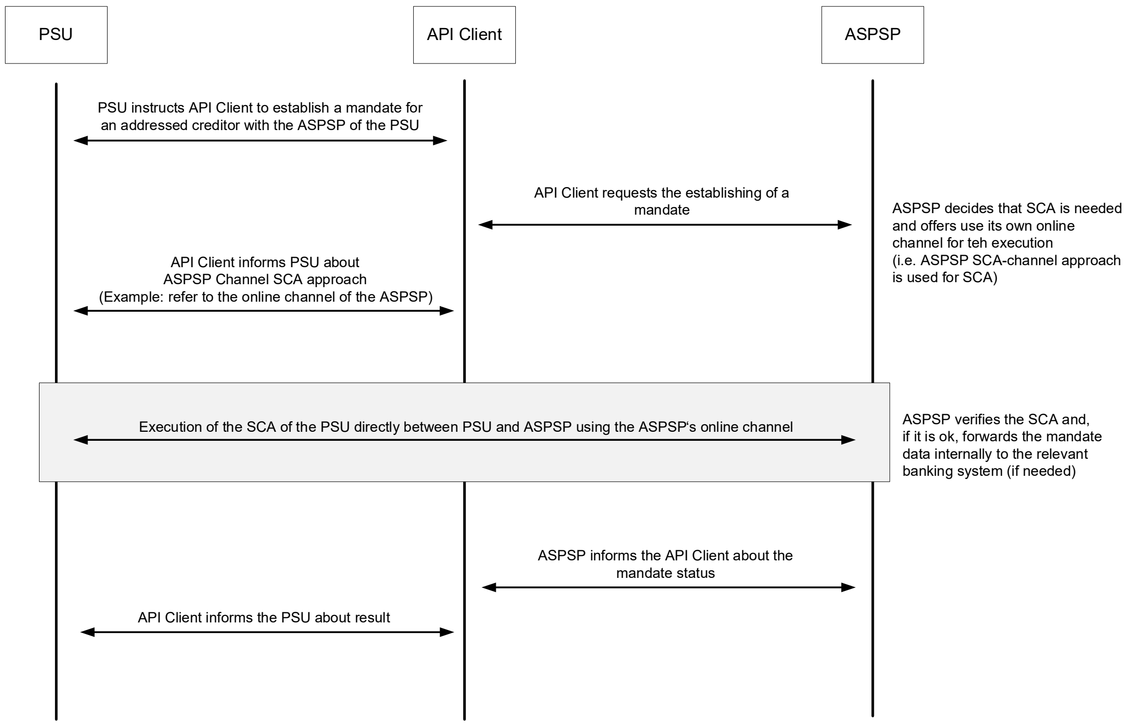
PSU
API Client
ASPSP
PSU instructs API Client to establish a mandate for an addressed creditor with the ASPSP of the PSU
API Client requests the establishing of a mandate
ASPSP decides that SCA is needed and offers use its own online channel for teh execution (i.e. ASPSP SCA-channel approach is used for SCA)
API Client informs PSU about ASPSP Channel SCA approach (Example: refer to the online channel of the ASPSP)
Execution of the SCA of the PSU directly between PSU and ASPSP using the ASPSP's online channel
ASPSP verifies the SCA and, if it is ok, forwards the mandate data internally to the relevant banking system (if needed)
ASPSP informs the API Client about the mandate status
API Client informs the PSU about result

Figure 8: ASPSP channel approach for SCA

In this case, the API Client does not need to have any knowledge about the individual steps of the SCA. The ASPSP Channel SCA approach therefore allows the API Client to avoid the implementation of the different SCA methods at its PSU – API Client interface.

4.8.6 SCA using the digital signature SCA approach

Instead of having a distinct authorisation phase during or after the transaction initiation where the ASPSP receives the PSU's consent, this SCA approach relies on the usage of certificates which have been submitted to the ASPSP and activated in a process including the ASPSP and the PSU beforehand. Then these certificates can be used to digitally sign request messages, which yields the SCA of the PSU. For details about a certificate management via the openFinance API, see [oFA-OR-UAM].

It is also possible for a corporate and an ASPSP to bilaterally agree on a certificate beforehand which then can be used for signing messages and yielding the corresponding strong customer authentication for corresponding request messages.

There exist two variants of the digital signature SCA approach:

- Variant A: a certificate personalised to an employee (PSU) is used to sign the body as SCA mechanism, and
- Variant B: a corporate seal is used to sign the body as SCA mechanism, but where personal information about the signing employee might still be transported via separate attributes.

The technical approach to transport the signatures for a signed API request not directly in the payload is defined in [oFA-PFSM]. The reason behind the chosen architecture is that a posting of digital signatures would anyway needed to support "distributed" signatures, where several signatures (by different persons) are needed and some or all of them can only be applied later, when the transaction resource as such has already been created.

NOTE: The Digital Signature SCA Approach should not be mixed up with (http-) message signing. Both elements come with JWS signatures (at different places). The digital signature generated with a certificate assigned to an employee is contained in the body (or posted to an authorisation resource) and is a transport of SCA information. The digital signature for message signing generated always with a corporate seal is contained in the http headers, cp. [oFA-PFSM] for details.

4.8.6.1 Scenario A: Signed Request with Personal Certificates

In this scenario, every PSU (employee) with the related signing rights needs to obtain a dedicated certificate, provided by a certification authority accepted by the ASPSP.

Of course, it is possible that the ASPSP provides the certificates needed by the employees of a corporate, if the ASPSP supports also the services of a certification authority.

Certificates of the employees of a corporate provided by other certification authority need some registration and activation with the ASPSP. For this certificate management functions are supported by the openFinance API Framework, for details about these processes see [oFA-OR-UAM].

After the activation of the certificate, the PSU (the employee who also is the certificate holder) may use it to sign the body of a request.

The following figure shows the (much simplified) top-level information flow for a payment initiation service with SCA based on digital signing as an example:

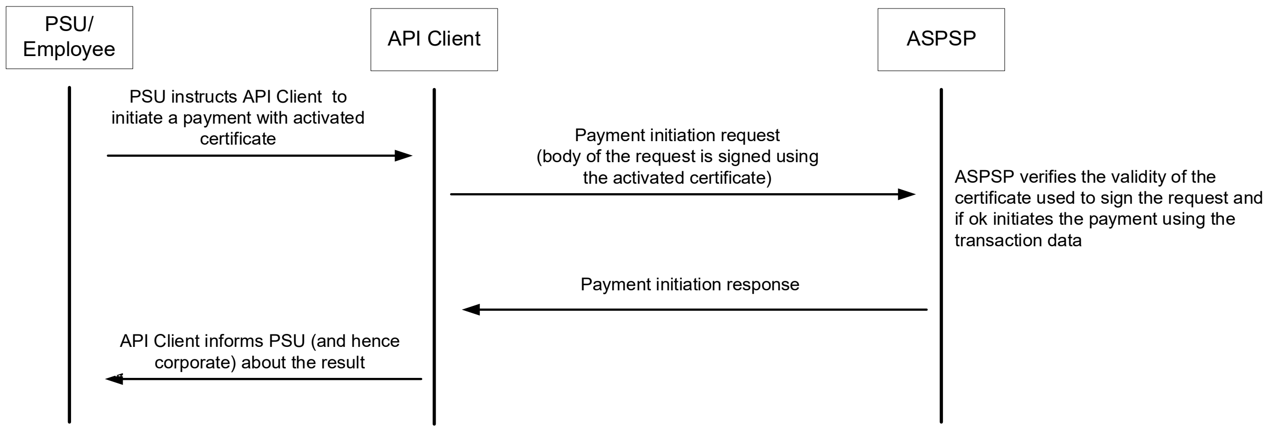
PSU/ Employee
API Client
ASPSP
PSU instructs API Client to initiate a payment with activated certificate
Payment initiation request (body of the request is signed using the activated certificate)
ASPSP verifies the validity of the certificate used to sign the request and if ok initiates the payment using the transaction data
Payment initiation response
API Client informs PSU (and hence corporate) about the result

Figure 9: payment initiation with signed body using the digital signature approach for SCA

As the PSU and the rights of the PSU have been verified during the process of establishing and activating the certificate beforehand, no further verification of the PSU is needed for the transaction initiation process.

4.8.6.2 Scenario B: Signed Request with a Corporate Seal

From a pure technical point of view, the openFinance Framework could also support the usage of corporate seals for signed requests instead of personalized certificates, while still providing information about the signatory/signatories in related header attributes. It is left to the ASPSP to decide whether it accepts this kind of signed request as a form of strong customer authentication or if it does not support this approach.

Details on conditions, liabilities and legal aspects have to be agreed bilaterally between the corporate and the ASPSP.

Some details are covered in Annex B of [oFA-OR-UAM] since the technical framework is independent of the certificate nature.

4.9 OAuth2 as a pre-step for PSU authentication

The openFinance API supports, in addition to the use of OAuth2 as an integrated SCA approach as described in section 4.8.2, also the use of OAuth2 as a pre-step to deliver an access token for a PSU without already dealing with functional consent.

This access token in this case encapsulates the PSU identification as such and can be used in all SCA approaches for PIS or AIS services. There are no further restrictions set to OAuth2 protocol configuration from an openFinance API perspective. ASPSPs will define the details of the OAuth2 usage as a pre-step in the mandated API documentation.

4.10 Multiple SCAs of more than one PSU

For some accounts, more than one PSU has to give its consent before accessing this account. In this case, SCA has to be executed by more than one PSU following up a Transaction Initiation Request.

If the account is accessed by an API Client, the API Client can use the API to execute the necessary SCA for this account. For this, the steps for executing SCA for a single PSU have to be repeated for the SCA of each PSU required for accessing this account.

If the SCA for a PSU has been finished correctly, the status of the resource corresponding to this API service will be changed. The API Client can access the resource to get the current status of the resource. If SCA of further PSU is required, this is indicated by a corresponding status, e.g. for payment initiations, the status PATC (`PartiallyAcceptedTechnicalCorrect`) is used. The Transaction Initiation Request will only be executed further if the SCA of all PSUs required for this account has been executed correctly.

The API Client (in cooperation with the PSU) accessing an account has to know all PSUs whose SCA is required for accessing this account. No information about this is given by the ASPSP at the openFinance API.

4.11 Signing Baskets

The signing basket is an API Service which allows to bundle several services, which are already initiated but not yet authorised. The idea behind this is to apply one single SCA which then provides the consent for all entries of the signing basket simultaneously. The SCA is not part of the service itself, the service only leads to creating the basket structure and does not include the execution of the services grouped together. The ASPSP might limit the signing basket to certain service types, i.e. only for payment services.

The support of this API Service at the XS2A interface is at the discretion of the ASPSP.

The API Client can use services according to this API Service to group other services into a signing basket. The following holds for the services to be grouped:

- The service may be any service according to an API Service of the payment initiation service.
- The service may be a service of the API Service "Establish account information consent" of the account information service.
- The service shall not be already fully authorised, i.e. at least the authorisation of one PSU shall be missing.

The following holds for the API Service "Grouping services to signing baskets":

- The PSU has to initiate this service at the PSU – API Client interface.
- The ASPSP will reject the service if the API Client cannot be identified at the API.
- In the case of a TPP at the XS2A interface: The ASPSP will reject the service if the TPP does not have the necessary roles corresponding to the services to be grouped.

The initiation of the services to be grouped has to be done by the API Client using the attribute `Client-Explicit-Authorisation-Preferred`. As result of the initiation the API Client will get a `paymentID` or a `consentID`. These IDs will be used as input for the initiation of the grouping service.

As result of the grouping service the API Client gets a `basketID`. This `basketID` can be used by an API Client in the following to start an authorisation process. In this case the following authorisation process affects all services contained in the basket. If a strong customer authentication is part of this authorisation process, it is up to the ASPSP to ensure that the SCA complies with all requirements of [RTS] for all services included in the basket.

A service contained in a signing basket can also be further processed individually as well, i.e. the resource corresponding to this individual service can still be accessed at the API. The completion, revocation or cancellation of this service is possible without relying on the structure of the signing basket.

The cancellation of a signing basket only affects the structure of the signing basket and not elements grouped together with the signing basket, i.e. the cancellation of the signing basket does not lead to the cancellation of its contained transactions.

4.12 Pushing resource status changes to API Clients

The resource status notification function allows the ASPSP to push status changes of a resource created by the API Client within the openFinance API Framework. Prior to this, the API Client needs to register (i.e. consent) for receiving such notification messages for every resource where such a function is envisaged. This registration is prompted by the API Client by including certain header parameters within the service initiation request. It may happen that the resource status push function is not supported for the related API Client by the ASPSP. Then the ASPSP ignores the related header fields.

The API Client can request to receive push notification messages about the following:

- notifications about every change of the SCA status for all related authorisation processes,
- notifications about all changes of resource status attributes, and
- only one notification on the last resource status as available in the XS2A interface or the openFinance API.

The ASPSP's response message includes an indicator whether the resource status push function is supported for this request, is supported in general but not for the current request or if it is not supported in general. If the ASPSP supports pushing the resource status, it will include a list for which constants the ASPSP will push notification messages.

5 Operational rules

This section summarises the operational rules to be observed by each API Client accessing the openFinance API and each ASPSP providing an API. Not all of these rules are enforced by technical means of the openFinance API.

The rules are given in alphabetical order. The order does not represent an order of importance.

5.1 Coding of business data

For a payment initiation service, the ASPSP may support the coding of the payment data using XML, JSON or both. If XML is supported, the payment data has to be coded as pain.001 message and the service status response will be coded as pain.002 message. If JSON is supported, the payment data has to be transferred as a JSON structure to be defined in the respective service specifications and in [oFA-PFSM].

For the reading of account information, the ASPSP shall support one or more of the following coding of the account data to be delivered:

- camt.05x,
- JSON,
- MT94x.

  Note: MT94x is still supported by the openFinance API but will be deprecated in the near future.

The JSON structure to be used for transferring the account information will be defined by the Implementation Guidelines of account information related services like [oFA-IG-XAIS], [oFA-IG-PAIS] and [oFA-IG-Com] and in [oFA - DD] respectively.

In any case, it is up to the ASPSP to decide what kind of coding will be supported for service data at its API. In many cases, specifically for new premium services like Mandate APIs, Document APIs etc., only JSON coding is supported. Generally, the ASPSP will inform the API Client about its decision as part of the API documentation.

5.2 Currency of API services

The specification of the openFinance API does not make any provisions concerning the currency of services, if the currency is an important service attribute at all.

5.3 Signing messages on the application layer

Signing the HTTP message will authenticate the API Client sending the request message to the openFinance API of the ASPSP.

The support of this security measure is optional for the ASPSP and mandatory for the API Client. If the ASPSP requests signing a HTTP request message at the application level, the API Client shall sign the HTTP request message at application level. HTTP response messages are not signed by the ASPSP, if not stated otherwise explicitly in the ASPSP documentation.

For push-based services, signing of HTTP request messages at the application level is also possible. The ASPSP or the related API access scheme decides if it signs the message or not.

At the XS2A interface, in order to sign an HTTP request message at the application level, the API Client has to use a QSealC, which has to be issued by a qualified trust service provider according to [eIDAS]. At the openFinance API, the requirements for signing HTTP request messages by the API Client will be defined by the ASPSP. If an API Client uses a QSealC, which allows the API Client to sign messages at the XS2A interface of the ASPSP, it can also be used to sign messages are the openFinance API of the ASPSP.

5.4 Signing message bodies

The body of a request or response message may be signed by one or more responsible entities. The party that sends the request/response is responsible for signing the body, who exactly is responsible depends on the single use case (i.e. for a request message as part of a push-based service the body may be signed by a responsible department or one or more responsible employees of the ASPSP).

Signing the body of a request message authenticates the PSU. If a request message sent by an API Client to the openFinance API of an ASPSP is part of a payment service, contains payment data, and its body is signed by one or more PSUs, then this will be called a **signed payment request**. If the creation of the corresponding signatures is compliant with the requirements of [RTS] for a signed payment request, further SCA procedures involving the PSU will not be necessary.

It is up to the ASPSP to decide if it supports signed bodies for messages. For some use cases an ASPSP might mandate that the body of a request message sent by an API Client has to be signed by one or more PSUs. There are no specific requirements defined for the quality of the process to generate the signature over the body of the HTTP message. It is possible that the ASPSP will define further requirements for this.

5.5 Consent of the PSU

An API Client may execute a service at the API of an ASPSP if it has the necessary consent of the PSU. The consent will be authorised by the PSU towards the ASPSP by performing a SCA procedure. In cases of exemptions, alternative authentication procedures might be agreed upon between PSU and ASPSP and might be integrated into the consent authorisation flow.

An ASPSP might reject a Transaction Initiation Request at the openFinance API if the consent of the PSU cannot be proven. This depends on the specific service and is defined in the respective service specification.

5.6 Validity of an API service

One API service may consist of several request/response interactions between API Client and ASPSP. Between these request/response interactions some time is needed e.g. for PSU SCA interaction.

During the first request/response interaction, the ASPSP generates a new resource representing this initiated resource. There exists a validity time of this resource which may depend on the kind of resource/API service and the ASPSP. Details are specified further in the respective service documents. During this validity time, the API Client may address the resource of the ASPSP by means of requests permitted by this framework.

5.7 Decision about Strong Customer Authentication

The ASPSP has to decide

- if SCA has to be executed as part of a service at the API,
- which method and personalised credentials have to be used for SCA, where the PSU will be involved in a selection process if several SCA procedures are available, and
- which approach has to be used for executing SCA, possibly but not necessarily considering the redirection preferences of the API Client.

The API Client must follow the decision of the ASPSP.

5.8 Revocation/cancellation of payment initiations

This framework allows the revocation of all future dated payments by the PSU directly between the PSU and ASPSP. This applies e.g. to single future-dated payments, future-dated bulk payments and to recurring payments initiated by standing orders.

A future-dated payment initiated by an API Client at the openFinance API can also be cancelled before execution by the PSU by directly accessing the ASPSP.

A recurring payment initiated by an API Client at the openFinance API can also be cancelled before execution by the PSU by directly accessing the ASPSP. It is also possible that the PSU changes parameters of the recurring payment, such as frequency and validity, or terminates the recurring payment finally by directly accessing the ASPSP.

In general, this framework allows the revocation of consent for services that need an authorisation via the PSU. For this, the status of the respective resource first needs to be retrieved and verified whether or not the revocation consent is (still) possible for the resource. In the case that this is possible, the consent is changed and this change is recorded by the ASPSP.

6 Message and data model

In the following, an abstract data model is presented for the usage of the openFinance API. This model is further refined in the respective operational rules and implementation guidelines of the respective services and in particular in [oFA-PFSM].

6.1 Protocol Level

The following data elements are used independently of the semantics of the related messages, building an abstract basic protocol level.

6.2 Transaction Initiation related data model

In the following, a minimum set of requirements for the Transaction Initiation Request is defined. The requirements are independent of the encoding. Additional requirements may apply depending on the specific service addressed within the request. These further requirements are specified in the operational rules of the respective services.

6.2.1 Transaction Initiation Request Data on Protocol Level

The following protocol level data is defined for Transaction Initiation request messages:

- Request Identification (Mandatory)

  This is a unique ID generated by the API Client to identify individual request messages. It is used to solve e.g. operational issues.

- Request Timestamp (Mandatory)

  Each request of a service contains a standard HTTP timestamp indicating the current processing time of the request.

- Access Token from the optional OAuth2 pre-step approach (conditional)

  The bearer token is only contained if an OAuth2-based authentication was performed in a pre-step or an OAuth2-based SCA was performed in a preceding AIS service in the same session.

- Further signature related data (conditional)

  In the case that the API Client includes a signature, the hash value is calculated over the content of the body of the HTTP request message; if the message does not contain a body, the digest header must contain the hash of an empty byte list.

- API Client Electronic Signature (conditional)

  The electronic signature is generated by the API Client following the definition in [oFA-PFSM] Section 6.2.2.2. The key to be used shall be the private key corresponding to the certificate of the API Client. Only to be delivered if mandated by the ASPSP for the case that the request message has to be signed by the API Client.

- PSU Identification and type (conditional)

  Client ID of the PSU in the ASPSP client interface might be mandated in the ASPSP's documentation. In the case that OAuth2-based authentication was performed in a pre-step, it might be contained. The type of the PSU identification is needed in scenarios where the PSUs have several PSU-IDs as access possibility. In this case, the mean and use are defined in the ASPSP's documentation.

- PSU Corporate Identification and Type (conditional)

  Client ID of the Corporate in the ASPSP client interface, might be mandated in the ASPSP's documentation. In the context of a corporate, it might occur that both a PSU Corporate ID and a PSU ID (e.g. of an employee of the corporate) are needed for a service execution

- PSU Context Data (used for the ASPSP's risk management, if not included in the message, the ASPSP will take this into account in its risk management)

  - IP Address PSU (mandatory for an initiation request, else optional)

  - PSU Device and Application Software Information (operating system, browser etc.) (optional),

  - GEO Location PSU (optional)

- SCA Approach Preferences (optional)

  A list from the most preferred SCA approach to the least preferred approach. Not all possible approaches must be listed; not listed approaches are interpreted as less preferred than the last entry in the list without specific order. The list might be ignored by the ASPSP.

- Redirect URI (conditional)

  URI of the API Client, where the transaction flow shall be redirected to after a Redirect. This is mandated for the Redirect SCA Approach.

- Client Notification Data (optional)

  URI for the endpoint of the API Client to which the ASPSP shall send the status of the resource. The possible status of the resources that are communicated are notification on every change of the SCA, notification on all changes of the resource status attributes, and/or a single notification on the last resource status as available in the XS2A interface or openFinance API.

- Client Brand Information (optional)

  Informs the ASPSP about the brand used by the API Client towards the PSU. This information is intended for logging entries to enhance communication between ASPSP and PSU or ASPSP and API Client.

Further technical data like resource IDs and interface versions etc. are determined by the API structure and defined in detail in the [oFA-PFSM].

6.2.2 Transaction Initiation Response Data on Protocol Level

The following data is defined for Transaction Initiation Response messages.

- Response Code (mandatory)

  The response code is a HTTP code on transport level indicating a correct processing, session re-routing to another site or processing errors. In [oFA-PFSM], a complete set of processing error codes is defined in detail.

- Request Identification (Mandatory)

  This is a unique ID generated by the API Client to identify individual request messages. It is used to solve e.g. operational issues.

- ASPSP SCA approach (conditional)

  The SCA approach that is chosen by the ASPSP

- ASPSP Notification data (conditional)

  Indicator if the ASPSP supports resource status push function for the created resource (in general and in this case) and if so, for which status constants it will be provided.

- Resource ID (mandatory)

  This is a unique ID generated by the ASPSP after the first service-related request of an API Client was received, and the ASPSP created a corresponding resource. This resource ID is transmitted to the API Client within the first response. The token shall be used by the API Client for all subsequent requests within a service lifecycle. The ASPSP is free to define the token and its security features.

- Service Resource Status (mandatory)

  This is the status of a service on the ASPSP side. This service status refers to a created resource like a payment initiation or a consent resource. This service status is used in response messages only when a service status is explicitly requested by the API Client or when the whole resource is created or addressed.

- PSU Message Information (optional)

  This data element contains a message, which is to be displayed to the PSU by the API Client.

- API Client Message Information (optional)

  This data element contains a message code and optionally open text information from the ASPSP to the API Client. It can be used by the ASPSP e.g. in exception situations to deliver detailed error information and additional information to the API Client.

- Available SCA methods (conditional)

  Might be contained if several authentication methods are available

- Challenge data (conditional, if only one authentication method is available and if the embedded SCA approach is chosen by the ASPSP)

- Hyperlinks (mandatory)

  for possible next service steps, the choice of hyperlinks depends a.o. on decisions of the ASPSP when processing the request, e.g.

  - Redirect URL ASPSP (conditional, only in case of Redirect Approach)
  - Hyperlink to authorise a transaction of the requested service, i.e. with different semantics like "startAuthorisationWithPsuIdentification", "startAuthorisationWithPsuAuthentication", "startAuthorisationWithEncryptedPsuAuthentication", "startAuthorisationWithTransactionAuthorisation", "startAuthorisationWithAuthenticationMethodSelection".
  - Hyperlink to receive the status of a corresponding resource, i.e. "status".
  - Hyperlink to the corresponding authorisation sub-resources like "authoriseTransaction"

6.3 Service Resource Creation related data model

6.3.1 Service Resource Creation Request Data on Protocol Level

The following protocol level data is defined for Service Resource Creation Request messages:

- Request Identification (Mandatory)

  This is a unique ID generated by the API Client to identify individual request messages. It is used to solve e.g. operational issues.

- Request Timestamp (Mandatory)

  Each request of a service contains a standard HTTP timestamp indicating the current processing time of the request.

- Access Token from the optional OAuth2 pre-step approach (conditional)

  The bearer token is only contained if an OAuth2-based authentication was performed in a pre-step.

- API Client Electronic Signature (conditional)

  The electronic signature is generated by the API Client following the definition in [oFA-PFSM] Section 6.2.2.2. The key to be used shall be the private key corresponding to the certificate of the API Client. Only to be delivered if mandated by the ASPSP for the case that the request message has to be signed by the API Client.

- Further signature related data (conditional)

  In the case that the API Client includes a signature, the hash value is calculated over the content of the body of the HTTP request message; if the message does not contain a body, the digest header must contain the hash of an empty byte list.

- Client Notification Data (optional)

  URI for the endpoint of the API Client to which the ASPSP shall send the status of the resource. The possible status of the resources that are communicated is notification on all changes of the resource status attributes and/or a single notification on the last resource status as available in the XS2A interface or openFinance API.

- Client Brand Information (optional)

  Informs the ASPSP about the brand used by the API Client towards the PSU. This information is intended for logging entries to enhance communication between ASPSP and PSU or ASPSP and API Client.

Further technical data like resource IDs and interface versions etc. are determined by the API structure and defined in detail in the [oFA-PFSM].

6.3.2 Service Resource Creation Response Data on Protocol Level

The following data is defined for Service Resource Creation Response messages.

- Response Code (mandatory)

  The response code is a HTTP code on transport level indicating a correct processing, session re-routing to another site or processing errors. In [oFA-PFSM], a complete set of processing error codes is defined in detail.

- ASPSP Notification data (conditional)

  Indicator if the ASPSP supports resource status push function for the created resource (in general and in this case), and if so, for which status constants it will be provided.

- Request Identification (Mandatory)

  This is a unique ID generated by the API Client to identify individual request messages. It is used to solve e.g. operational issues.

- Resource ID (mandatory)

  This is a unique ID generated by the ASPSP after the first service-related request of an API Client was received, and the ASPSP created a corresponding resource. This resource ID is transmitted to the API Client within the first response. The token shall be used by the API Client for all subsequent requests within a service lifecycle. The ASPSP is free to define the token and its security features.

- Service Resource Status (mandatory)

  This is the status of a service on the ASPSP side. This service status refers to a created resource like a payment initiation or a consent resource. This service status is used in response messages only when a service status is explicitly requested by the API Client or when the whole resource is created or addressed.

- PSU Message Information (optional)

  This data element contains a message, which is to be displayed to the PSU by the API Client.

- API Client Message Information (optional)

  This data element contains a message code and optionally open text information from the ASPSP to the API Client. It can be used by the ASPSP e.g. in exception situations to deliver detailed error information and additional information to the API Client.

- Hyperlinks (mandatory)

  for possible next service steps, the choice of hyperlinks depends a.o. on decisions of the ASPSP when processing the request, e.g.

  - Hyperlink "self" to the created resource, used to retrieve the resource data,
  - Hyperlink "status" to receive the status of a corresponding resource.

6.4 Authorisation Data Model

For services that mandate authorisation such as payment initiation, establish consent or signing baskets, a uniform authorisation data model is used. To enable such a uniform authorisation, to each service to be authorised, an authorisation sub-resource is associated. In cases where several authorisations are needed, this sub-resource is repeated.

6.4.1 Start Authorisation Request

This request is used to create an authorisation sub-resource after the submission of payment initiations, consent data or grouping a signing basket.

The sort of data is directly addressed by semantic hyperlinks contained in the related response message of the preceding request. The following data can be addressed:

- PSU Identification and type (if not yet submitted within the Payment Initiation Request but mandated by a prior start authorisation hyperlink) (conditional)

  Client ID of the PSU in the ASPSP client interface, might be mandated in the ASPSP's documentation. In the case that OAuth2-based authentication was performed in a pre-step, it might be contained. The type of the PSU identification is needed in scenarios where the PSUs have several PSU-IDs as access possibility. In this case, the mean and use are defined in the ASPSP's documentation.

- PSU Corporate Identification and Type (if mandated by a prior start authorisation hyperlink, if not yet submitted within the Payment Initiation Request, if the PSU is corporate and if mandated by parameters published by the ASPSP) (conditional)

- PSU Password (conditional)

- SCA Approach Preferences (optional)

  A list from the most preferred SCA approach to the least preferred approach. Not all possible approaches must be listed; not listed approaches are interpreted as less preferred than the last entry in the list without specific order. The list might be ignored by the ASPSP.

- Redirect URI Client (conditional)

  Only mandated if the Redirect Preferred Indicator equals true or if this Indicator is not contained. This data element defines a URL to which the ASPSP shall redirect the PSU browser session once the SCA on bank websites is performed

- Client Explicit Authorisation Preferred Indicator (optional)

  With this indicator, the API Client can explicitly set its priority to create an authorisation sub-resource. This should be used by the API Client in cases where the corresponding initiated service is later added to a signing basket.

- Client Notification URI, Client Notification Content Preference (optional)

  Yielding the Endpoint of the Client API to which the status of the resource should be sent, and which kind of content the Client would prefer to be informed about respectively

- Authentication Method Choice (conditional, if ASPSP supports several SCA methods for the corresponding PSU)

- Proprietary Data (conditional, if correspondingly documented by the ASPSP)

6.4.2 Start Authorisation Response

- Available Authentication Methods, if the ASPSP supports several SCA methods for the corresponding PSU (conditional)

- SCA approach the ASPSP on (conditional)

  This data element must be contained if the SCA approach is already fixed, and then yields the SCA approach that the ASPSP determines. Possible values are: embedded, decoupled, redirect and ASPSP channel. The OAuth2 approach is subsumed by redirect.

- Challenge Data (conditional, used only for the Embedded SCA Approach and depending on the risk management of the ASPSP, and in rare cases also for the first authentication of the PSU via a password)

  The challenge data contains

  - challenge data if required by the SCA method,

  - formatting information for the PSU Authentication Data within the Service Authorisation Request, additional information for the background of the SCA usage or in rare cases also for password usage (e.g. to type in certain digits of a secret)

6.4.3 Update Data Request

The authorisation sub-resource created by the request in Section 6.4.1 might need to be updated with additional data, depending on the authorisation process. This is then indicated by semantic hyperlinks contained in the response message of the preceding request. The following data can be addressed:

- PSU Identification and type (if not yet submitted within the Payment Initiation Request but mandated by a prior start authorisation hyperlink) (conditional)

  Client ID of the PSU in the ASPSP client interface, might be mandated in the ASPSP's documentation. In the case that OAuth2-based authentication was performed in a pre-step, it might be contained. The type of the PSU identification, it is needed in scenarios where the PSUs have several PSU-IDs as access possibility. In this case, the mean and use are defined in the ASPSP's documentation.

- PSU Corporate Identification and Type (if mandated by a prior start authorisation hyperlink, if not yet submitted within the Payment Initiation Request, if the PSU is corporate and if mandated by parameters published by the ASPSP) (conditional)

- PSU Password (conditional)

- SCA Approach Preferences (optional)

  A list from the most preferred SCA approach to the least preferred approach. Not all possible approaches must be listed; not listed approaches are interpreted as less preferred than the last entry in the list without specific order. The list might be ignored by the ASPSP.

- SCA Preferred Indicator (optional)

  With this indicator, the API Client can set its priority for a re-direct based SCA Approach (Redirect SCA Approach or OAuth2 SCA Approach) vs. a SCA Approach without a re-direction to a bank site (Embedded SCA Approach or Decoupled SCA Approach, depending on the authentication method).

- Redirect URI Client (conditional)

  Only mandated if the Redirect Preferred Indicator equals true or if this Indicator is not contained. This data element defines a URL to which the ASPSP shall redirect the PSU browser session once the SCA on bank websites is performed

- Client Authorisation Preferred Indicator (optional)

  With this indicator, the API Client can explicitly set its priority to create an authorisation sub-resource. This should be used by the API Client in cases where the corresponding initiated service is later added to a signing basket.

- Client Notification URI, Client Notification Content Preference (optional)

  Yielding the Endpoint of the Client API to which the status of the resource should be sent, and which kind of content the Client would prefer to be informed about respectively

- Authentication Method Choice (conditional, if ASPSP supports several SCA methods for the corresponding PSU)

- Proprietary Data (conditional, if correspondingly documented by the ASPSP)

6.4.4 Update PSU Data Response

- Available Authentication Methods, if the ASPSP supports several SCA methods for the corresponding PSU

- SCA approach the ASPSP on (conditional)

  This data element must be contained if the SCA approach is already fixed and then yields the SCA approach the ASPSP determines. Possible values are: embedded, decoupled, redirect and ASPSP channel. The OAuth2 approach is subsumed by redirect.

- Challenge Data (conditional, used only for the Embedded SCA Approach and depending on the risk management of the ASPSP, and in rare cases also for the first authentication of the PSU via a password)

  The challenge data contains

  - challenge data if required by the SCA method,

  - formatting information for the PSU Authentication Data within the Service Authorisation Request,

  - additional information for the background of the SCA usage, or in rare cases, also for password usage (e.g. to type in certain digits of a secret)

6.4.5 Service Authorisation Request

This message is used only in case of the Embedded SCA Approach and when an authorisation of the initiated payment by a SCA method is needed.

- PSU Authentication Data (mandatory)

  Data submitted by the PSU to authorise the service when performing a SCA method. This is e.g. an OTP or an electronic signature.

6.4.6 Service Authorisation Response

This message has no additional data elements.

6.5 Signing basket related data model

The Grouping Service Request is a Transaction Initiation Request in the sense of this document. In addition to the elements listed in Section 6.2 the following data attributes are also supported.

6.5.1 Grouping Service Request

- List of payment resource identifications (optional)

  All payments referred to in this list by payment identifications are part of the signing basket to be created,

- List of consent resource identifications (optional)

  All consents referred to in this list by consent resource identifications are part of the signing basket to be created,

- List of subscription resource identifications (optional)

  All subscriptions referred to in this list by subscription resource identifications are part of the signing basket to be created,

- List of subscription entry resource identifications (optional)

  All subscription entries referred to in this list by subscription entry resource identifications are part of the signing basket to be created,

- List of mandate resource identifications (optional)

  All mandates referred to in this list by mandate resource identifications are part of the signing basket to be created.

6.5.2 Grouping Service Response

- signing basket identification (mandatory)

  A resource identification of the generated signing basket resource.

7 Annex

7.1 Glossary

AIS	Account Information Service according to article 4 (16) of [PSD2] and as regulated by article 67 of [PSD2].
AISP	Payment service provider offering an AIS to its customer. See article 4 (19) of [PSD2].
ASPSP	Account Servicing Payment Service Provider providing and maintain a payment account for a payer. See article 4 (17) of [PSD2].
PIISP	Payment Instrument Issuer Service Provider according to article 4 (14) and 45) of [PSD2]. A PIISP can use the service "Confirmation on the availability of funds" as regulated by article 65 of [PSD2].
PIS	Payment Initiation Service according to article 4 (15) of [PSD2] and as regulated by article 66 of [PSD2].
PISP	Payment service provider offering a PIS to its customer. See article 4 (18) of [PSD2].
PSP	Payment service provider according to article 4 (11) of [PSD2].
PSU	Payment Service User according to article 4 (10) of [PSD2].
QTSP	Qualified Trust Service Provider, e. g. a trust centre issuing qualified certificates

SCA

Strong Customer Authentication – authentication procedure based on two factors compliant with the requirements of [PSD2] and [RTS].

TPP

Third Party Provider – generic term for AISP/PIISP/PISP.

TSP/QTSP

Trust Service Provider according to [eIDAS]. Within the context of the XS2A interface specification only qualified TSPs (QTSPs) according to section 3 of [eIDAS] issuing qualified certificates for electronic seals and/or qualified certificates for website authentication which are compliant with the requirements of [RTS] are relevant.

XS2A interface

Access to account interface – interface provided by an ASPSP to TPP for accessing accounts.

7.2 References

[eIDAS] Regulation (EU) No 910/2014 of the European Parliament and of the Council on Electronic Identification and Trust Services for Electronic Transactions in the Internal Market, 23 July 2014, published 28 August 2014

[oFA - DD] openFinance API Framework, Data Dictionary for V2.x, Version 2.3, 31 October 2025

[oFA-IG-XAIS] openFinance API Framework, Implementation Guidelines, Extended Account Information Service, Version 2.2, 17 April 2025

[oFA-IG-PAIS] openFinance API Framework, Implementation Guidelines, Push AIS Services. Version 2.1, 31 July 2024

[oFA-IG-Com] openFinance API Framework, Implementation Guidelines, Compliance Services, Version 2.4, 31 October 2025

[oFA-IG-UAM] openFinance API Framework, Implementation Guidelines, Certificate, User and Account Management, early draft, not available yet

[oFA-OR-UAM] openFinance API Framework, Operational Rules, Certificate, User and Account Management Services, Version 0.91, not published yet

[oFA-OR-ADM]	openFinance API Framework, Operational Rules, Admin Services, Draft, August 2021
[oFA-PFSM]	openFinance API Framework, Implementation Guidelines for Protocol Functions and Security Measures, Version 2.3, 31 October 2025
[PSD2]	Directive (EU) 2015/2366 of the European Parliament and of the Council on Payment Services in the Internal Market, published 25 November 2016
[RFC3986]	T. Berners-Lee, R. Fielding and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", RFC 3986, January 2005, https://tools.ietf.org/html/rfc3986
[RTS]	Commission Delegated Regulation (EU) 2018/389 of 27 November 2017 supplementing Directive 2015/2366 of the European Parliament and of the Council with regard to Regulatory Technical Standards for Strong Customer Authentication and Common and Secure Open Standards of Communication, L69/23, Official Journal of the European Union, 13.03.2018

7.3 List of figures

7.4 List of tables