



openFinance API Framework

Operational Rules for Extended Services

Administrative Services

Version 1.0

31 July 2026

Berlin Group License Notice

Berlin Group Specifications are 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.
- Any OpenAPI files offered by Berlin Group are bound by “Creative Commons Attribution 4.0 International Public License” and are allowed to be changed. However, please note that the normative reference still is the Implementation Guidelines document.
- 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.
- Any right, title and interest in and to the copyright and all related rights in topic-related Scheme Rulebooks, belong to the respective Scheme Manager (amongst others, the European Payments Council AISBL - EPC).
- 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.

Contents

1	Introduction.....	1
1.1	From Core XS2A Interface to openFinance API	1
1.2	Overview: Administrative Services.....	2
1.3	Document Structure.....	5
1.4	Document History	5
2	Basics for the administrative steps	6
2.1	Different scenarios for administrative steps	6
2.2	Steps for the support of the identification of the API Client	9
2.3	Steps for signing contracts, certification and approval of the client	11
2.4	Services for information about prices and technical onboarding of an API Client	11
2.5	Basics for the price models.....	13
2.5.1	Usage of price models	13
2.5.2	Establishing a price model	15
2.5.3	Structure of the price lists.....	17
2.5.4	A priori and a posteriori predictability of the price model	23
2.5.5	Confirmation of the Price Model as part of the technical onboarding process.....	24
2.6	Steps for accessing an extended service at an openFinance API..	24
2.7	Extended Administrative Services covered by this document	25
3	API Services supported for the Administrative Services.....	26
3.1	API Service: Start of the Onboarding Process	28
3.2	API Service: Reading and confirming the Price Lists	30
3.3	API Service: Read the onboarding resource	32
3.4	API Service: Cancellation of the API Client's onboarding (by API Client).....	33
3.5	API Service: Updating the certificate of the API Client	34
4	Operational rules	35
4.1	API Client Identification	35
4.2	API Client Authorisation.....	36
4.3	Consent of the PSU.....	36

4.4	Decision about strong customer authentication	36
4.5	Onboarding process of the API Client at the ASPSP's API	36
4.6	The price lists presented during the onboarding of an API Client...	36
4.7	Fall-back price model	37
4.8	Confirmation of a price model	37
4.9	Information on the current price model and next invoicing date	37
4.10	Validity of an onboarding	38
4.11	Cancellation of the onboarding resource by an API Client	38
4.12	Cancellation of a service contract by an ASPSP	38
5	Message and data model	40
5.1	Protocol Level	40
5.2	Authorisation Data Model	40
5.3	Informative: Basic data model for different entities	40
5.3.1	ASPSP data	40
5.3.2	API Client data	41
5.3.3	Scheme data	42
5.4	Initiation of the onboarding process	42
5.4.1	Initiation Request	42
5.4.2	Initiation Response	42
5.5	Reading price lists	42
5.5.1	Reading price lists Request	42
5.5.2	Reading price lists Response	43
5.5.3	Confirmation Request	43
5.5.4	Confirmation Response	43
5.6	Reading the status of the onboarding process	43
5.6.1	Status Request	43
5.6.2	Status Response	43
5.7	Reading the onboarding resource	44
5.7.1	Reading the onboarding resource Request	44
5.7.2	Reading the onboarding resource Response	44
5.8	Cancellation of the onboarding by the API Client	44
5.8.1	Cancel Request	44

5.8.2	Cancel Response.....	44
5.9	Updating the QSealC certificate used by the API Client.....	45
5.9.1	Updating Request	45
5.9.2	Updating Response.....	45
6	Annexe	46
6.1	Glossary	46
6.2	References	48
6.3	List of figures	49
6.4	List of tables	49
6.5	Informative: Abstract model of directory services.....	50
6.5.1	Introduction	50
6.5.2	Overview: Functionality of a directory service.....	52
6.5.3	Directory Service: Add entity to directory.....	53
6.5.4	Directory Service: Update entity within directory	54
6.5.5	Directory Service: Read entity from directory	54
6.5.6	Directory Service: List of entities	55
6.5.7	Directory Service: Delete entity from directory.....	55



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 on the Availability of Funds Service (FCS) 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.

This XS2A Framework is now planned to be extended to extended 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 anymore solely on PSD2.
- 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 need to be necessarily 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" resp. in the work of the ERPB on a SEPA API access scheme.

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

The following account access methods are covered by this framework:

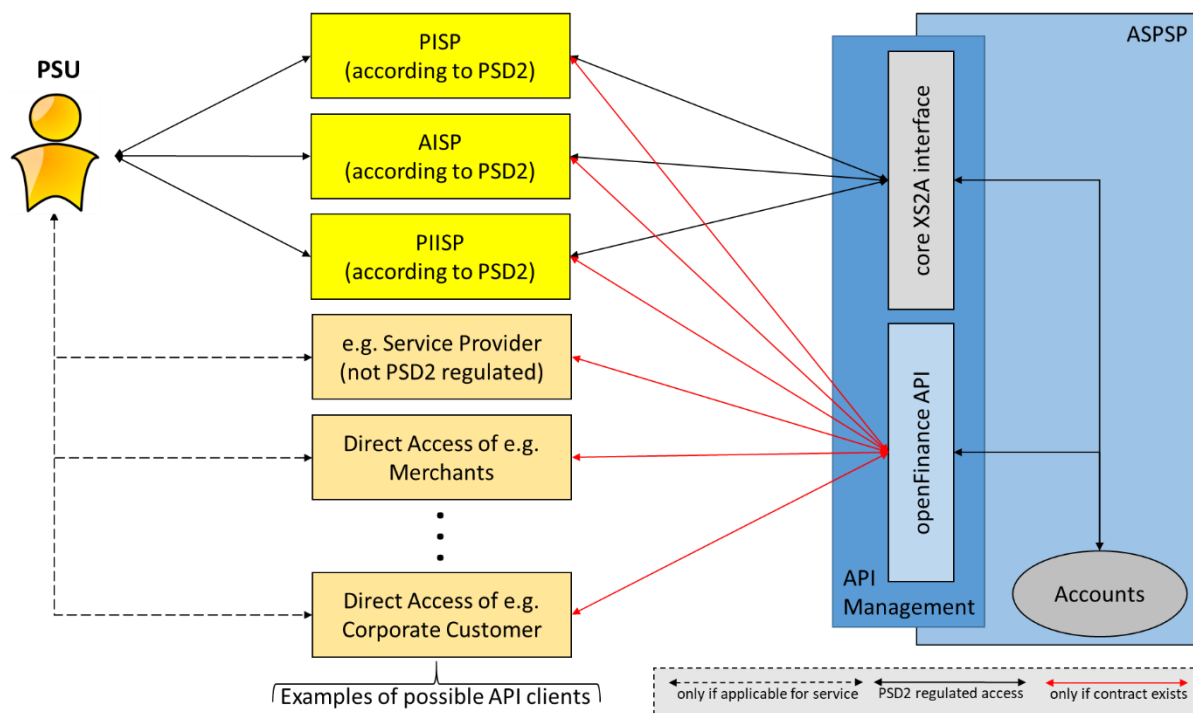


Figure 1: Core XS2A interface and openFinance API

The ASPSP may restrict the access to the services offered at its openFinance API and require dedicated onboarding. The requirements for the rights to access to services offered at the openFinance API are this operational rules document. The following section yields an overview of the administrative services.

1.2 Overview: Administrative Services

Different clients may access the openFinance API of an ASPSP for using extended services supported by the ASPSP. This holds not only for TPPs registered by a NCA according to PSD2 but also for other service provider not registered by a NCA or technical instances of an ASPSP customer directly ("direct access"). Not all service providers who want to access an openFinance API as an API Client can do this.

In order to get the right to access the openFinance API (for one or more extended services) as a client actively ("API Client"), the service provider might need to do different administrative steps. By these administrative steps the service provider changes its potential to access the openFinance API of an ASPSP step by step to become an "API Client". Administrative steps

can be for example the registration of the service provider as potential client by a scheme or by an ASPSP or the negotiation and conclusion of a contract with a scheme or a single ASPSP.

An extended service offered at the openFinance API of an ASPSP can be organised and defined by a scheme or directly by one or more ASPSP. By this, the administrative steps necessary to become an accessing "API Client" can be defined and executed on different levels, i.e. on a scheme level or on a level defined by one or more ASPSP. See section 2.1 for the different scenarios.

After a service provider has got the right to access the openFinance API of an ASPSP as an API Client for one or more extended services (by executing the necessary administrative steps first defined by a scheme or ASPSP), the API Client has to execute a technical onboarding for these extended services at the openFinance API of an ASPSP.

For this technical onboarding, administrative services supported by the openFinance API have to be used by the API Client. Only if this technical onboarding has been executed successfully, the API Client can access the openFinance API for the corresponding extended services.

The API Client receives its specific "API-Contract-Id". Using this identifier, the API Client obtains the rights to access the openFinance APIs of an ASPSP for which the API Client completed the onboarding process.

This document will not define all the administrative steps in detail, since these administrative steps have to be implemented based on business requirements of different stakeholders and on different legal requirements, which might apply in different domestic jurisdictions. But for the technical onboarding to be done using administrative services of the openFinance API of an ASPSP it is the objective of this document that the technical execution of these administrative services as seen from the API Client as well as for the ASPSP will be as interoperable, i.e. standardised as far as possible by the Berlin Group openFinance API Framework. This will allow ASPSP and API Clients to adhere to flexible market evolution and support bilateral as well as multilaterally organised contractual situations.

A signed contract of a service provider may hold only for a single extended service or for a group of extended services. The contract may also hold only for the access to the openFinance API of a single ASPSP or for the access to the openFinance API of many different ASPSP. For example if an extended service is organised by a scheme, the conclusion of a single contract with the scheme owner might be sufficient in order to access the openFinance API (after a technical onboarding) for this extended service of all ASPSP who are participants of this scheme. The relationship between a scheme owner and a single ASPSP is out of scope for this document. But in any case the technical onboarding has to be executed for each ASPSP, the API Client wants to access.

Depending on the requirement of a scheme or an ASPSP some of the administrative steps might be combined or might be missing. Some of these steps might not have any technical impact on the openFinance API, other steps are supported technically by the openFinance

API. For administrative steps with an impact on the openFinance API of an ASPSP, special (administrative) services are defined. Currently these are

- an optional price information service supporting the retrieval of price lists indicating the fees the API Client has to pay to the ASPSP for using a service at its openFinance API. This can be used as a first step of the technical onboarding service or it can be omitted, and
- a technical onboarding service supporting the (technical) onboarding of an API Client with an ASPSP including (if necessary) the confirmation of a digital contract.

In addition, basis functionality of the API framework supports the secure identification of the API Client accessing the openFinance API.

Remark 1:

A special group of potential clients who might access the openFinance API of an ASPSP are corporate clients, who are business customers of the ASPSP. If and how a corporate client may access the openFinance API of an ASPSP will be arranged between the ASPSP and the corporate client bilaterally. This bilateral arrangement has no consequences for the possibility of the corporate client to access the openFinance API of any other ASPSP. This topic of bilateral arrangements between an ASPSP and its corporate clients is not within the scope of this document.

Remark 2:

The technical onboarding of an API Client with an ASPSP does not involve a PSU. The onboarding process takes place before any PSU is involved.

For the technical onboarding of an API Client (as well as for all further accesses) to the openFinance API of an ASPSP, it is essential that the API Client is identified securely. The secure identification of an API Client will be based on qualified certificates for website authentication (QWAC) and qualified certificates for electronic seals (QSealC) according to the eIDAS regulation [eIDAS].

These certificates will in general not contain any information about the status of the API Client according to the administrative steps shown above, i.e. the certificates will not contain any information about approvals or contacts of the API Client. To make this information available in a secure manner, it is suggested to use directory services. These directories can be operated on behalf of a scheme ("scheme directory") or on behalf of a group of ASPSP. The operation of these schemes is not standardised as part of the Berlin Group openFinance API Framework. But the annexe of this document contains the high level description of a model for such a directory, to give an example for the functionality needed.

1.3 Document Structure

This document first explains the context of the Administrative Services, i.e. different contractual access scenarios and administrative steps required for accessing an openFinance API in Section 2 – steps needed before onboarding as well as onboarding steps during onboarding. A detailed explanation on the usage of the pricelists which could be presented via the API is given in Section 2.5.

Section 3 then describes the related API services for the onboarding process in detail.

Section 4 defines operational rules that apply during the administrative services in addition to the basic operational rules of the framework.

The related messages and data model to be used within messaging are then given in an overview in Section 5.

Within the Annex in Section 6, an abstract functional model of the directory service is provided for informative reasons:

- as support for potential API access schemes to explain the interplay of onboarding processes, service contracts, scheme participation and scheme contracts, as well
- as support for future integration of the administrative services into backend systems of ASPSPs implementing the openFinance API.

1.4 Document History

Version	Change/Note	Approved
0.9	Draft for consultation	openFinance TF, 7 June 2021
0.91	Feedback of market consultation incorporated	
1.0	Final Version	07-07-2026 openFinance Taskforce

2 Basics for the administrative steps

2.1 Different scenarios for administrative steps

In the following different scenarios are distinguished for giving an overview about the administrative steps. These are:

- Scenario to get access to an extended service organised by a scheme.
- Scenario to get access to an extended service organised by an ASPSP for a client who is a new partner for the ASPSP.
- Scenario to get access to an extended service organised by an ASPSP for a client who is already a known partner of the ASPSP.

In the following figures an overview of all administrative steps is given, even if not all of these steps are defined or specified as part of the Berlin Group openFinance API Framework. As part of the figures it is indicated which steps will be handled as part of the openFinance API Framework and which have to be defined and specified outside of this framework.

The administrative steps are (slightly) different depending on the scenarios. But the differences are only relevant for the parts of the administrative steps, which are not handled by the openFinance API Framework of the Berlin Group.

The administrative steps can be summarised as follows:

- Steps for the support of the identification of the client (grey box in the figures).
- Steps for signing contracts, certification and approval of the client (blue box).
- Steps for information about prices and technical onboarding of a client (red box).
- Steps for accessing an extended service at an openFinance API (green box).

Only the steps in the red and green boxes are defined and specified by the openFinance API Framework. The steps in the red box are called administrative services and will be handled in this document of operating rules. The steps in the green box will be handled by the operating rules for the corresponding extended service.

In addition to the administrative steps also directory services are shown in the figures (blue or brown box). The directory services can be used to make results of one administrative step available to other following administrative steps. See the annexe 6.5 for an abstract model for these directory services. The directory service can be provided for example by an external Directory Service Provider or by an ASPSP itself.

Scenario 1: Extended service organised by an API access scheme

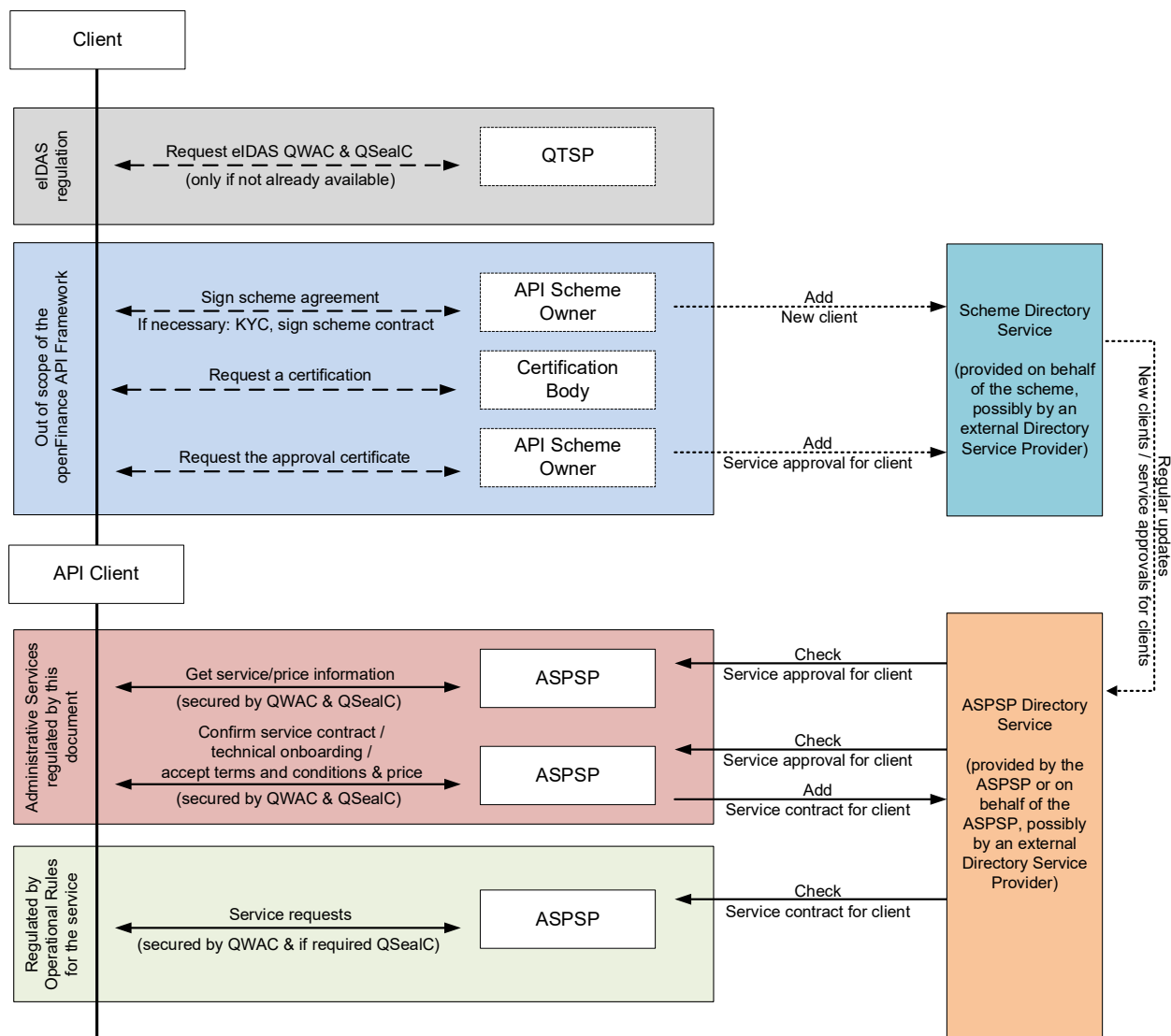


Figure 2: Scenario "services organised by a scheme"

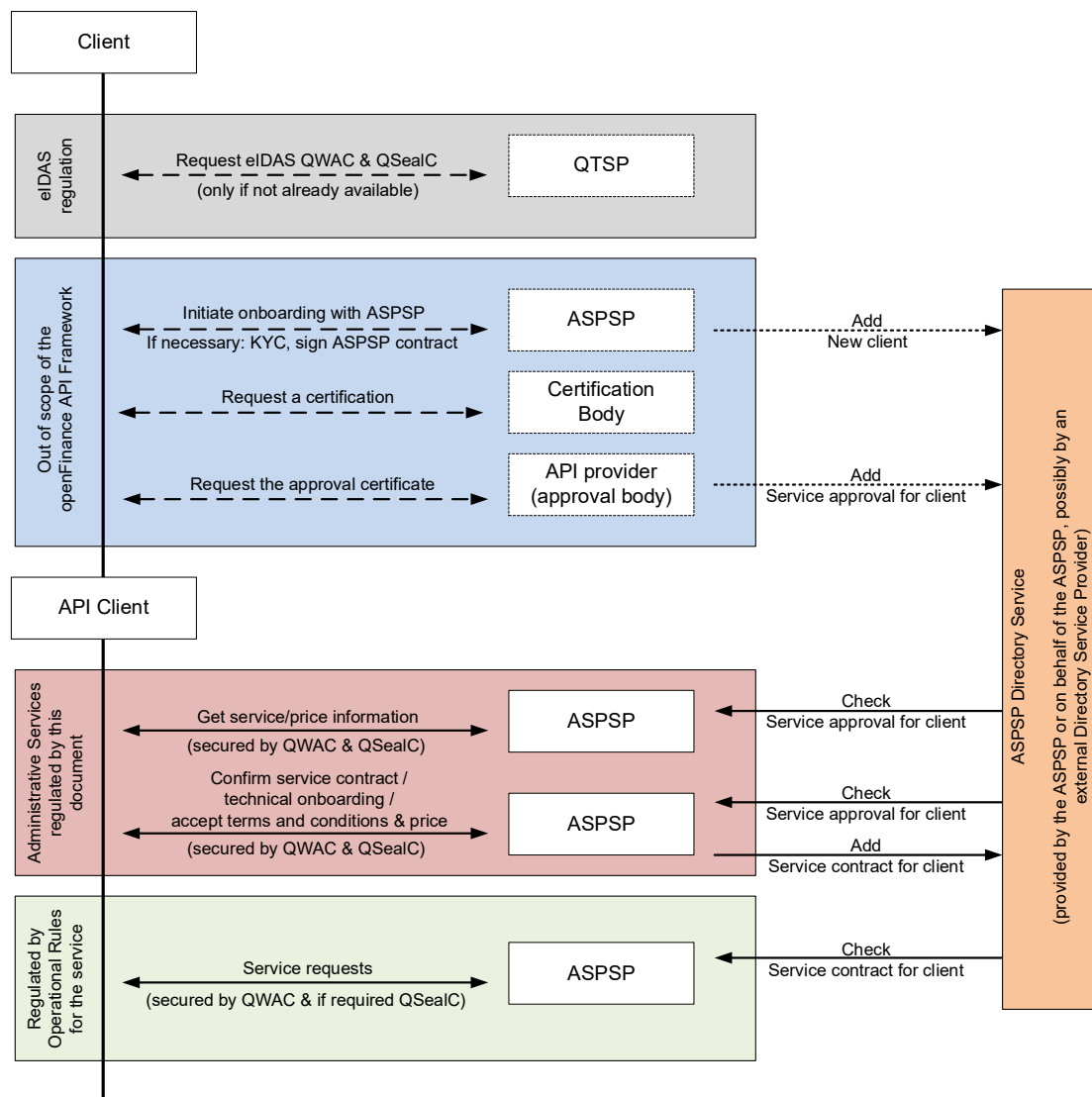
Scenario 2:**Extended service organised by an ASPSP, client is a new partner for the ASPSP**

Figure 3: Scenario "services organised by ASPSP – client is new partner"

Scenario 3:

Extended service organised by an ASPSP, client is a partner already known by the ASPSP

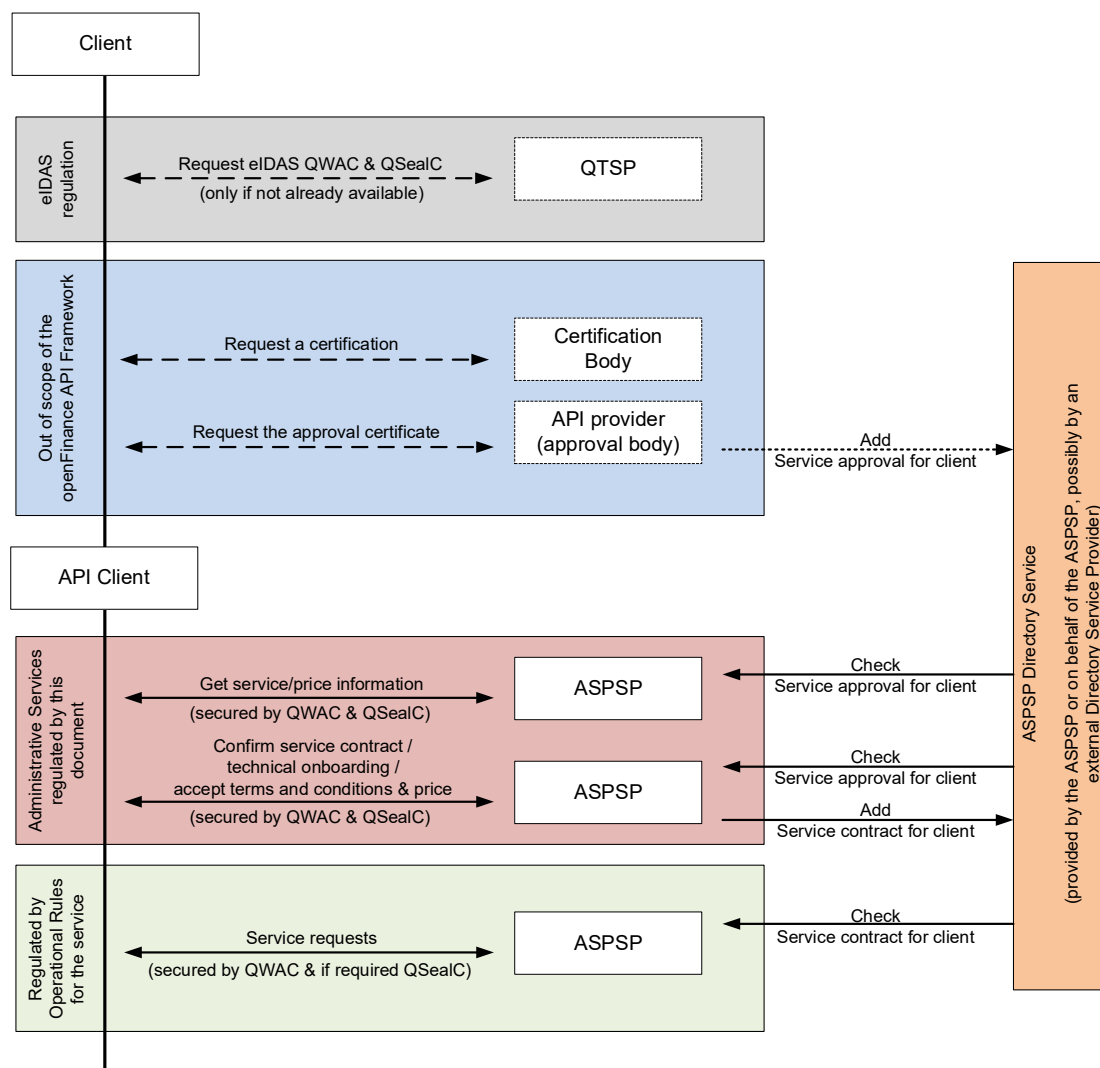


Figure 4: Scenario "services organised by ASPSP – client is a known partner"

2.2 Steps for the support of the identification of the API Client

As a basic requirements an API Client accessing the openFinance API of an ASPSP has to identify himself in a secure manner. As for the core XS2A interface this identification shall be based on the usage of qualified certificates for website authentication (QWAC) and qualified certificates for electronic seals (QSealC) according to the eIDAS regulation [eIDAS].

Remark:

The eIDAS certificate infrastructure is used for this to allow a homogeneous infrastructure for the issuance/usage and verification of certificates used to secure all API accesses (i.e. XS2A API and openFinance API).

The infrastructure for defining the structure of the certificates and for issuing the qualified certificates is not regulated or specified by the openFinance API Framework of the Berlin Group. Instead all trust service provider which are qualified as a QTSP according to the eIDAS regulation can issue the certificates needed by clients to access the openFinance API of an ASPSP.

Each client has to get its QWAC and QSealC from a QTSP of the clients own choice. If a client has been registered by a NCA as a TPP, this client may use its QWAC and QSealC according to [TS 119 495] also at the openFinance API of an ASPSP. If a client has not been registered as a TPP by a NCA, the QWAC and QSealC of this client may defer from some of the requirements of [TS 119 495].

The conditions (legally, organisational, technical) for issuing these eIDAS compliant qualified certificates are regulated by the QTSP. If an API Client has got an eIDAS compliant qualified certificate from a QTSP this certificate has to be accepted by all ASPSP for the identification of the API Client at the openFinance API.

An ASPSP has to be prepared to accept eIDAS compliant qualified certificates from an API Client at its openFinance API, regardless which QTSP has issued these certificates. See section 4.1.

Remark:

If the access of a corporate client to the openFinance API of an ASPSP has been arranged bilaterally between the corporate client and the ASPSP, requirements for the certificates to be used for the identification of the corporate client can be determined by the ASPSP. For example the ASPSP can issue these certificates for its own corporate clients on its own. In this case the corporate client can use this certificate only for accessing the openFinance API of this ASPSP.

The QWAC and the QSealC will not contain any information about

- the approval of the client by a scheme or an ASPSP or
- information about contracts the client has concluded with a scheme or an ASPSP.

For this reason the QWAC and QSealC will be sufficient only for the secure identification of the client accessing the openFinance API of an ASPSP. But it will not be sufficient to verify if the client has got the right (approval/contract) to access the openFinance API for a given extended service as an API Client.

Since the certificates (QWAC and QSealC) are only used for secure identification of an API Client, an API Client can also use (if available) different certificates for its identification at the openFinance API of an ASPSP, as long as the API Client is identified uniquely by these certificates.

As for the XS2A interface the basic identification of an API Client is done at protocol level based on the QWAC of the client. Each access to the openFinance API of an ASPSP has to be secured at the protocol level (i.e. TLS connection with client authentication based on the QWAC).

For the administrative services it is necessary to have in addition a secure identification/authentication of the API Client at application level as a proof of origin. This is done by securing the http message by an electronic seal based on the QSealC of the API Client. The generation of this electronic seal is based on the JSON Web Signature Profile with the restrictions according to [OBE-JWS].

For some extended services it might also be necessary to have in addition a secure identification/authentication of the API Client at application level. If this is the case this will be defined by the operational rules for these extended services.

2.3 Steps for signing contracts, certification and approval of the client

The definition and specification of these steps are out of the scope of the openFinance API Framework of the Berlin Group. Details have to be determined by scheme owner, certification bodies and/or ASPSPs depending on the organisation of the extended service to be defined.

2.4 Services for information about prices and technical onboarding of an API Client

Prior to the step to use these administrative services the API Client has signed already a general agreement with the scheme owner or with one or more ASPSP for the right to access a set of extended services. By certification and approval it has been shown that the API Client has implemented its technical infrastructure to access the extended services correctly. The next steps are the technical onboarding of the API Client and an optional reading and confirming of the price conditions. This will be described in this subsection and the detailed in the rest of this operational rules document.

In the following different variants of the onboarding process are distinguished:

- One-step onboarding process without presentation and confirmation of (price) conditions at the API

The onboarding process consists only of one simple message exchange sent by the API Client. The ASPSP verifies the identity of the API Client, the message seal used and if a service contract is available and the onboarding process is completed (either successfully or the client gets rejected).

- Two-step onboarding process with confirmation of (price) conditions at the API:

The onboarding process consists of three message exchanges between the API Client and the ASPSP, initiated by the client.

- The API Client starts the onboarding process and an onboarding resource is created which allows the client to access the price conditions.
- Then the API Client reads the price lists. These might be specific to the API Client or be a general fall-back price model of the ASPSP or API access scheme.
- The API Client confirms the price model presented via the onboarding resource and signs the price lists. This confirmation marks the end of the technical onboarding process of the API Client towards the API. In case, the API Client does not send a confirmation, the onboarding process remains unfinished.

For all these onboarding steps, the API Client must use its QWAC to identify itself at the message layer.

It will be decided by a scheme if the one-step process or the two-step process has to be used by an API Client for onboarding to an openFinance API of an ASPSP as participant of the scheme. If no scheme is involved, this will be decided by the ASPSP.

Both services will be supported by the openFinance API of an ASPSP. The necessary API calls and data structures are specified in [oFA-IG-Adm].

After a successful onboarding is executed, the ASPSP will immediately provide a unique identifier for the API Client: the **API-Contract-Id**. The API Client shall use this identifier when accessing the ASPSP's APIs in the future, e.g. to initiate extended services at the API.

The price lists will contain information about the fees the API Client has to pay for accessing an extended service at the openFinance API of an ASPSP. The fees will be in general different between different ASPSP. Only fees for the concrete usage of an extended service are contained in the price lists. Further dues to be paid by an API Client as for example (scheme) membership fees, shares in the costs for centralised infrastructures, etc. have to be defined as part of the contract to be agreed on before between the API Client and the scheme owner or one or more ASPSP. The same holds potentially also for special agreements for a refund of fees or for a discount of fees for example in the case a given limit of fees has been reached within a year.

It might be possible that for some extended services not the API Client has to pay a fee to the ASPSP, but instead the ASPSP has to pay a commission to the API Client. This could be for example the case for the placement of a loan or for the purchase of securities. These commissions will also not be part of the price lists but have to be defined as part of the contract to be agreed on before between the API Client and the scheme owner or one or more ASPSP.

A detailed explanation on the price models that are supported in the openFinance data structure is given in Section 2.5. The actual invoicing and settlement of the fees are out of the scope of this document. This has to be defined by the scheme or the ASPSP.

Remark:

This technical onboarding of an API Client to the openFinance API of an ASPSP for an extended service shall not be confused with an activation of a PSU for the services of the API Client. Activation of a PSU is out of scope of this document.

The access to these two special administrative services will not only be secured by a QWAC at the transport level, but will be secured by an additional electronic seal based on the QSealC of the API Client. By this the ASPSP gets a (legal) proof of origin according to the eIDAS regulation for these requests. By this the identity of the requested API Client is secured.

Before responding to the requests of these administrative services, the ASPSP might do the following steps:

- The ASPSP might check the status of the QSealC with the QTSP who issued this certificate.
- The ASPSP might get more information about the status of the service approval of the API Client from a directory service.

Depending of the result of these checks the ASPSP might restrict the access of an API Client to these administrative services at its openFinance API.

It is up to the internal business and security management rules of the ASPSP to decide to which extend these checks have to be executed and how the results are interpreted.

After the successful onboarding the API Client can access the openFinance API of that ASPSP for subsequent service requests for that extended service. The technical onboarding of an API Client for an extended service at the openFinance API of a given ASPSP has no side effects to the openFinance APIs of other ASPSP.

2.5 Basics for the price models

2.5.1 Usage of price models

The price information service at the openFinance API is an instrument of the Framework for API Clients as well as for ASPSPs to communicate its API prices for the usage of certain premium API services. Furthermore, it allows the ASPSP to update its pricing models and to inform the API Client on these changes while the current API pricing still applies. The price lists will contain information about the fees and prices the API Client has to pay for executing an extended service and for the general access to the ASPSP's AP.

The API Client retrieves price lists yielding an overview of all possible costs that apply for the API Client when requesting a service at the API of the ASPSP. The fees depend on the ASPSP as well as on the API Client.

It might be possible that for some extended services instead of the API Client paying a fee to the ASPSP, the ASPSP has to pay a commission to the API Client. This could be for example the case for the placement of a loan or for the purchase of securities. These commissions may be part of the price lists, too. They must be specified as negative amounts for the API Client.

In general, the pricing of API services is left to the corresponding API access scheme or to the ASPSP, thus, it is expected for the prices to vary between different ASPSPs/access schemes. Additionally, prices for the API usage of a specific ASPSP might also vary between different API Clients. The next invoicing date shall be presented together with the price lists whenever a (potential) API Client is requesting to read the price lists.

It is possible to implement client-specific price models that were previously agreed on between the API Clients and the ASPSP e.g. via bilateral contracts instead of a standard price model relying on the ASPSP's fall-back prices. In the case of client-specific price model, the ASPSP must ensure that only the corresponding API Client can access the information on the price model. This contract containing the agreement on the client-specific price models must include regulations, in the case a given limit in the contract has been reached within a pre-defined period.

Fall-back price lists may also be distributed by an ASPSP as part of its Discovery API (which is out of scope of the Administrative Services). It is the ASPSP's responsibility to consider any requirements of the current legal situation when making price lists available to other parties and an ASPSP of course has to consider any requirements of national or European law.



2.5.2 Establishing a price model

It is possible to think of vastly different price models that are used by different ASPSP or by the same ASPSP for different API Clients to calculate prices. Some examples for the different variants for pricing the API usage are the following:

- **Prices per API call** that apply for each call that is addressed to the respective API/APIs. These prices and price points only apply when an API call was issued by the API Client to the API. The price might depend on the usage (or value) of a parameter used within the API call.

It might occur that multiple instances of prices per API call apply to the same API call. Then the sum of these partial prices yield the overall price for this API call.

The ASPSP might limit the possible outcome of this sum by defining a minimum or maximum price for an API Call. If the sum of all partial prices of an API call is smaller than the minimum price resp. bigger than the maximum price, the minimum resp. maximum price is used instead of the price calculated by the sum when pricing the respective API call.

The ASPSP has the option to additionally establish that a given pricing for an API Call yields the price of an API service which consists of (potentially) multiple API calls. Then, this indicates that the pricing does apply for the overall service usage.

Furthermore, the ASPSP can also set a price for an API call but restricts that the price only applies when a later authorisation was completed (if needed) for the requested transaction, e.g. the pricing of a payment initiation request only applies if the related authorisation is completed afterwards and the payment actually was executed.

- **Flat fee prices** that apply generally for a given time period and do not depend on any API calls. These prices yield a fixed price for the usage of the (specified) API(s).

Flat fees are set for a fixed predetermined period, e.g. each month, and function as an entry fee for the API Client to access the respective API/APIs during that period. These prices will always apply even if the API Client has not issued any API call addressed to the respective API/APIs during the fee period.

They can be defined for an overall service category, defined by its service type, or be defined on the level of a specific service endpoint to establish a flat fee for specific service functions of the related service type.

- **Prices depending on the API usage of the API Client:** the price model that applies for a certain API Client might also depend on factors that are independent of the (singular) API call or API access but depend on the API Client's overall API usage during a certain predetermined time period.

The ASPSP might apply different pricing models depending on how many calls are issued by the API Client or how much volume was initiated or for how many different PSU's the API Client has initiated services at the API over the course of a fixed period (e.g. a month).

When an API Client accepts the conditions presented in one or multiple extended services, the API Client signs the lists and includes them in its response to the ASPSP and thus, completing the second step of the two-step technical onboarding for this extended service to the openFinance API of that ASPSP. At the end of this process, the API Client will retrieve a so-called **API-Contract-Id**, which enables access to business services APIs and which is bound to the price list agreed. Only then the API Client may execute the services offered at the ASPSP's APIs.

Remark: The technical onboarding of an API Client does not involve a PSU. The onboarding process must be completed before a potential PSU might initiate a service using the API Client.



2.5.3 Structure of the price lists

In order to present its price lists to a possible API Client, the ASPSP first needs to develop a JSON data structure that yields the corresponding price structure of the ASPSP's price model.

In some cases, the price model might be pre-defined such that the ASPSP only needs to fill out the respective price points, e.g. if the API is defined by a scheme, the price structure might be fixed by the scheme as well but the ASPSP might set the price amount/percentage of a certain price.

The ASPSP may change the prices for its services either unilateral or bilateral negotiated with an API Client. Each price list will indicate the starting date and the end date of its validity. The next invoicing date shall be presented in the response data as well, when an API Client requests to read the price lists.

The complete price model of an ASPSP containing all the prices that might occur at the API is a complex, multilayered structure. For a complete overview of the structure please refer to [oFA-IG-Adm] and [oFA-DaD]. The following diagram shows a very top-level overview how the pricing model structure is represented at the API:

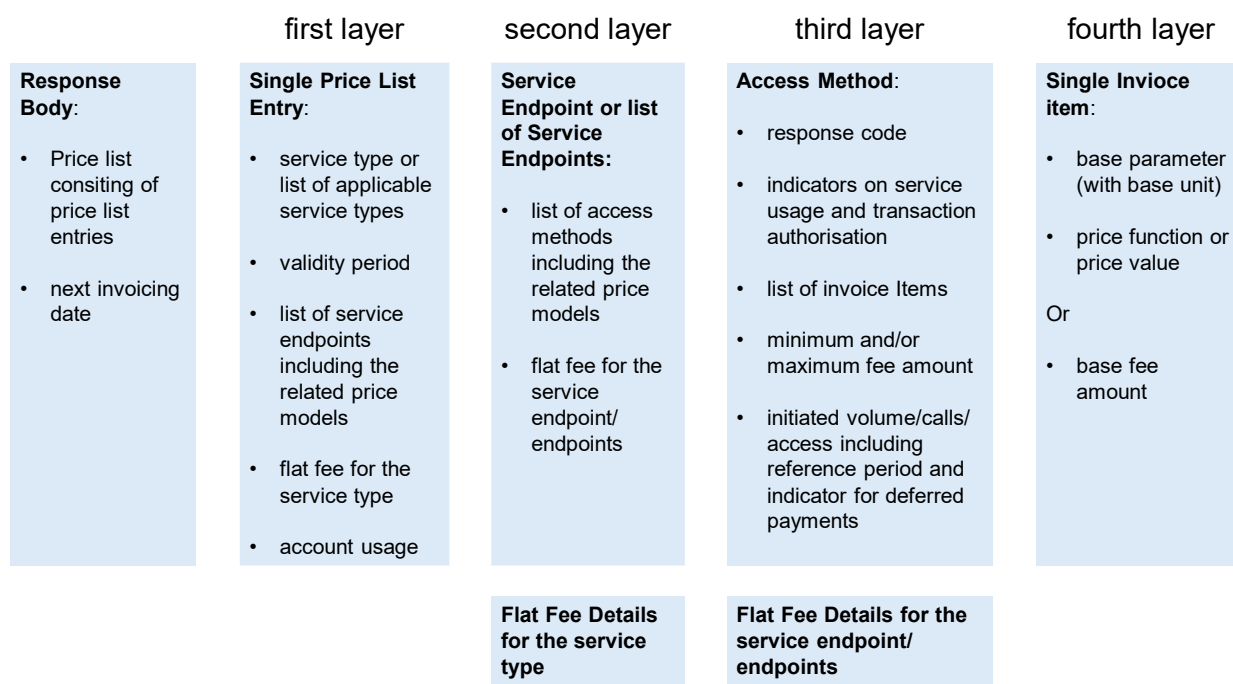


Figure 5: layers of response presenting the price list

In the following, each layer together with its necessary and optional structures is explained in detail.

The Response Body

The price model is defined as a list consisting of (potentially multiple) price list entries.

The ASPSP might use different price list entries per service type and per account usage and per validity period. It is also possible to bundle several service types together into one price list entry. Then each of the applicable service types must be given explicitly.

The ASPSP can optionally impose different pricing models depending on account usage, to differentiate prices depending if a service is initiated for personal private accounts or for corporate accounts. If no account usage is specified, the price model does not differ between different account usages.

The next invoicing date shall be given by the ASPSP within the response as well.

First layer: single price list entry

Each entry of the price list must always convey the information of the dedicated service type or service types, its validity period and the related account usage, if used. There is at most one entry for each combination of these three parameters. Furthermore, the given validity periods of price list entries of the same services type (and account usage, if specified) may not overlap.

A single price list contains an overview of all service endpoints related to the specified service type(s) to convey the specific price model. The related endpoints might be given separately or be summarised by referring to a parent node of the endpoints. This then includes all endpoints which obtain this as parent node.

Additionally, the price list might also specify if different prices are used depending on the account usage and an overall service description can be included.

The ASPSP may include service fee's e.g. for the general usage of services of the addressed service type(s) at the API. Such generic flat fees must be included in this layer as well. If flat fee does depend on the endpoint or access method, these kind of fees must be added in the following layer. The structure of the flat fee entry is described in detail in Section 2.5.3.1.

Second layer: service endpoint(s)

Each endpoint that is addressed for a service function which is priced by the ASPSP must be included in this list. As detailed before the related endpoint can be given explicitly or is included implicitly by listing a related parent endpoint of the specific endpoint.

A list of access methods for the specified endpoint(s) shall be included to provide the price model depending on the specific API Call. If the entry yields the price model for multiple entries, then the price model given for an access method applies to all endpoints that can be addressed via this access method.

Additionally, flat fees that only apply for the addressed endpoint(s) must be included here. The usage of flat fees is optional again. If used, the structure of a flat fee entry is described in detail in Section 2.5.3.1.

Third layer: access method

Once the service type and related endpoint(s) are determined, on this layer each entry yields the price model for an access method to the endpoint(s) for the service type usage. This is used to determine the price points depending on the specific values and parameters of the API call.

The ASPSP might define different price models for pricing API calls depending on the API call's response code, as well as if the transaction was authorised afterwards or not (if authorisation is needed for the API service). Additionally, the ASPSP might want to price a service usage for which (potentially) multiple API calls must be issued instead of a single API call. In that case, this must be indicated within this layer by clarifying that the pricing applies *per usage* of the service.

If the response code and/or the indicators for a later authorisation resp. for a service usage are not included, it is assumed that the given prices of the API Call only applies to the given API call and the price will be issued only in the positive case, i.e. if the request was issued successfully and a later authorisation was given (if needed).

An entry on the third layer yields a **list of invoice items**. This allows to define partial prices depending on the (parameters and values used within each) API call. These partial prices might be

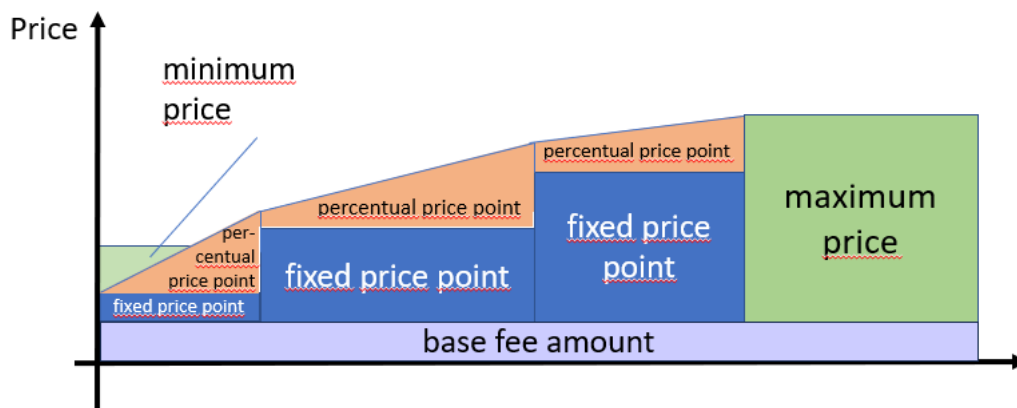
- total prices given by fixed price values, or
- percentual prices depending on values given within the API call given as price functions within the price model, or
- a base fee amount that shall occur independently on any parameters/values of the API Call (i.e. this yields a fixed "price per call").

For price values and price functions, a staggered pricing can be implemented. This means that depending on other values/parameters used within the API call, different amounts for the price value resp. different percentage values could be used when determining the price point.

All price points listed in the list of invoice items that apply for the call will be summed up to obtain the total price. The definition of these partial prices within the list of invoice items is described in more detail in Section 2.5.3.2.

The ASPSP may decide to give a lower and/or upper bound for the total price of an API call given by this sum of the invoice items. Such bounds can be given optionally by parameters yielding the minimum and maximum fee amount.

The following shows a top-level overview of the pricing of an API call that relies on the usage of a base fee amount, partial price points consisting of a percentual price and a fixed price value (both staggering systems depending on values of attributes used in the API Call), and bound by a minimum and maximum price:



Additionally, the APSPS may mandate different prices depending on the API Client's total API usage during a certain period, cf. Section 2.5.3.3.

For each tuple consisting of an access method, response code, indicators on a later authorisation and specification(s) of the API Client's API usage does the price list obtain a different entry in the third layer. It is the ASPSP's responsibility to ensure its price model is defined unambiguously and without overlapping areas of applicability.

2.5.3.1 Flat Fee

The flat fees on each layer are defined to apply on this layer. Each flat fee is defined for the overall usage of the related services specified by the given service type or by the given endpoint. A flat fee does not depend on the API calls issued via the API and a flat fee does apply even if the API was not addressed at all by the API Client. The fee is defined by its frequency describing how often the price of the fee is issued and the price of the flat fee.

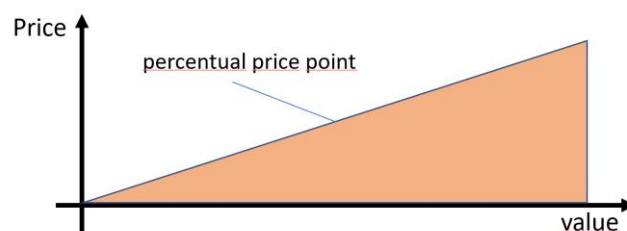
2.5.3.2 Price elements used in the invoice items

The list of invoice items allows the ASPSP to define partial prices for an issued API Call depending on the different attributes and values used within the specific API call and potentially use a tiered pricing for this.

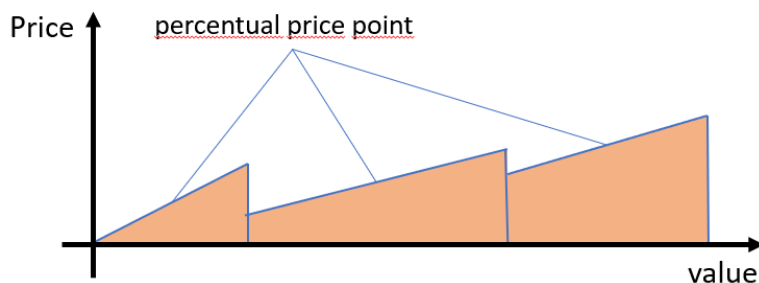
A partial price is given by a base parameter used within the call and which defines the staggering system of pricing (e.g. for deferred payments, the expiry date might lead to higher prices if it covers a longer time span as it would for a shorter time span). If no such tiered pricing is needed, this is indicated and it is assumed the following defined partial price always applies. The unit of the base parameter is included if needed to clarify the following staggering system. Partial prices can be defined percentual or by a total price value. An attribute used as

a base parameter in a pricing model may be used multiple times as such within a list of invoice items.

If the price is defined percentual based on a percentage value given by a parameter within the API call, the price is given as a **price function**. The price function includes the percentage reference parameter (e.g. for deferred payments the authorised amount might be used as reference allowing a gradual pricing depending on the amount) and the percentage value which shall be applied to the percentage reference parameter to calculate the partial price.



If in addition, a staggered price system applies, the price function must determine the interval resp. units of the base parameter for which a certain percentage value shall be used. This allows the ASPSP to establish the usage of different percentage values for different intervals/units of the base parameter (e.g. for deferred payments, the percentage value used might be higher if the requested reservation covers a longer timespan).

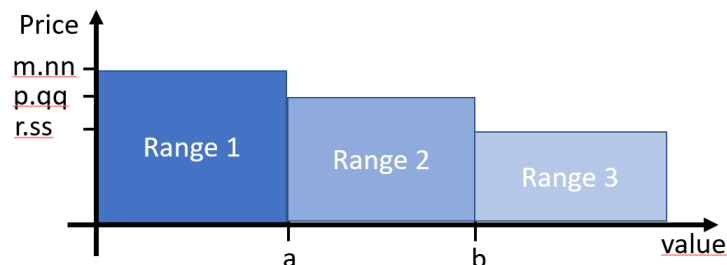


If the price is defined by a total value, the price is given as a **price value**.



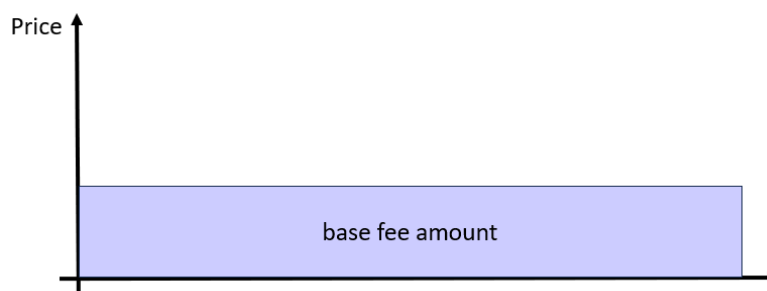
If a tiered pricing model is used, this defines different price values depending in the base parameter (e.g. for recurring payments, if the base parameter is set to be the frequency, then an API call requesting a weekly recurring payment might lead to a higher price than to a

monthly recurring payment). Then, the price value must yield different entries for prices depending on the given intervals resp. units of the base price.



The ASPSP can include the name of the invoice item to each price value entry, to connect the given price points with the invoice which will be issued to the API Client later.

Furthermore, in addition to these two types of calculating a partial price of an API call, the ASPSP might include a basic price called **base fee amount** which applies for each API call without any further prerequisites on the parameters or values of parameters used in the call. This base fee might be included in the list of invoice items at most once per API call. Besides the price amount, the related invoice item number might be included as well.



Note: The ASPSP can include the name of the invoice item to each entry to connect the given price points with the invoice which will be issued to the API Client later.

2.5.3.3 Prices depending on the API usage of the API Client

Depending on the API Client's API usage during a certain period (e.g. the billing period), the ASPSP might mandate different pricing rules. Then, these are represented by multiple entries on the third layer of the price model. For each entry the applicable restrictions to the API usage of the API Client must be specified. The following variants might lead to different pricing models:

- the initiated volume at the addressed API during the predefined period, or
- the number of previously initiated API calls during the predefined period, or

- the number of PSU's for which the API Client has initiated the API call for during the pre-defined period.

Whenever such a differentiation takes place, the ASPSP must include the reference period to this layer as well and if multiple of these differentiations are used, they must relate to the same pre-defined period and the API Client must fulfil all the conditions of the parameter in order for the price entry to apply to him.

Note: There are two options how the ASPSP does apply such conditions on the API Client's API usage (if different price models are wanted depending on this). Within the pre-defined period, the price model (i.e. the related access method entry) might be decided on

- the current API usage of the API Client:

This implies any API call is priced directly using the price model that applies to the API Client's API usage from the beginning of the reference period up until the request was issued.

Even if the API Client's API usage implies the usage of a different price model at some later point during the same reference period, only the API calls that are issued afterwards will be priced using the new pricing model. The prices of past API calls will not be changed because of this. Thus, a dynamic pricing throughout the reference period is applied.

- the total API usage of the API Client during the reference period:

By this pricing, all API calls may be priced only after the related period has passed. The pricing is determined according to the total usage of the API Client during that period. This implies the price of an API Call can only be determined after the related reference period is finished and not during. Thus, a delayed static pricing is applied.

The ASPSP has to indicate which variant it chooses for its price model. The ASPSP should use the same variant for all entries of the price list.

2.5.4 A priori and a posteriori predictability of the price model

The given price model yields highly versatile options when establishing a price model for the API usage. The model allows the mapping of interactions at the API directly onto the price model for which a variety of pricing systems can be applied: a broad variety of calculation approaches, enabling anatomic calculations, where each attribute of an API call can be studied and priced separately, up to a very generalised approach, where all price points are covered by general flat fees for the API usage.

The price model structure is very similar to the general API structure of the openFinance Framework. It can be used very extensively to describe prices for specific API calls depending on multiple factors given by the API Call itself and service flat fees. Additional parameters like the API Client's API usage, the account usage of the requested account and the requesting

API Client can be considered with the model as well and different pricing rules can be presented within the model.

In case the API Client's API usage leads to different prices at the API, the ASPSP may decide to determine the prices immediately based on the API Client's API usage until the related API request was issued or to determine all prices a posteriori a certain reference period is exceeded.

However, the price of an API call can only be determined a priori if it does not depend on any external factors (like the API Client's API usage or the response code) or process that would be initiated afterwards (e.g. a successful transaction authorisation of a transaction initiation request).

Note: The pricing of business service which a priori could consist of multiple API calls is more complex. As such a business service might be finished after the first API call or after multiple API Calls are exchanged depending on the response data, a simple mapping of the service onto the API interactions is not possible. The ASPSP should yield all general price points within the price entry of the first API call that is to be issued for the service and use a respective indicator to stress that this price entry does apply per service usage and the prices cover (potentially more) API Calls related to the service usage.

2.5.5 Confirmation of the Price Model as part of the technical onboarding process

The procedure how an API Client can select the price information service supporting the retrieval of price lists is detailed in Section 3.2. It is part of the two-step approach of the technical onboarding process. Receiving the API Client's confirmation on the price lists is part of the necessary second step in order to complete the onboarding of the API Client to the openFinance APIs of the ASPSP, in case where a one-step approach is not sufficient. Then, the price model is included in the service contract and can be accessed via the onboarding resource.

After the onboarding process, the API Client may execute the respective services that were agreed on during this process (i.e. in particular the services where the corresponding price models were agreed on by the API Client).

2.6 Steps for accessing an extended service at an openFinance API

The operational rule for these steps are within the scope of the openFinance API Framework of the Berlin Group, but not within the scope of this document. Details will be determined by the operational rules document relevant for the given extended service.

2.7 Extended Administrative Services covered by this document

The following sub services of the Administrative Steps and Services are addressed in this document within the openFinance Framework:

Service addressed as	Usage
Technical onboarding service	Service which supports - the technical onboarding of an API Client for an extended service at the openFinance API, following a one-step or a two-step approach, - for the two-step approach: the reading and confirmation of a digital service contract (including the price model), and price information service are supported.
Price information service	Sub-service of the onboarding service (when the two-step approach is used) which supports the retrieval of the ASPSP's price model that would apply for the client, if accepted.
Cancellation of a service contract	Services which support the cancellation of a service contract by the API Client.

Table 1: Administrative services

3 API Services supported for the Administrative Services

The current version of the Administrative Services supports the following API services. The **Service Type** is indicating a bundling of several API services to a service family. This document introduces the service type

- Administrative Services (XAS)

The following table gives an overview of the API services:

API Services	Service Type	Technical Functionalities	PSU involved
Initiation of the onboarding process	XAS	Creates an onboarding resource Start of the onboarding procedure of the API Client, either following a one-step process or a two-step process	No
Reading price lists	XAS	In the two-step approach of the onboarding process: Read one or more price lists valid for using the corresponding extended services by the given API Client	No
Confirmation of price lists	XAS	In the two-step approach of the onboarding process: Confirm the price model presented beforehand	No
Read the status of the onboarding resource	XAS	Get status of the onboarding resource	No
Read the onboarding resource	XAS	Read the full onboarding resource	No
Cancellation of the onboarding resource by the API Client	XAS	Cancel the onboarding resource	No
Information about a new certificate (used for the http signature)	XAS	Update of the certificates used by the API Client for message signing, i.e. update of the signature recorded in the onboarding resource	No

Table 2: API Services of the Administrative Services

In addition, the openFinance API will support technical use cases within the RESTful API.

Remark: PSUs are never involved in any of these administrative services (neither directly nor indirectly). The API Client is using these administrative services always on its own behalf, never on behalf of a PSU. For this reason the term "consent of a PSU" is not relevant for these administrative services.

The access of the API Client to the openFinance API of an ASPSP will be secured by eIDAS compliant qualified certificates. The ASPSP has to be prepared to accept eIDAS compliant qualified certificates from an API Client at its openFinance API, regardless which QTSP has issued these certificates. See section 4.1.



3.1 API Service: Start of the Onboarding Process

Before accepting a contract for a given service the API Client has to get all necessary information about the service to be contracted at the openFinance API of an ASPSP. In the following it is assumed that the necessary information is divided in two blocks, i.e. legal contract details and pricing information.

The following figure shows only the very top level information flow:

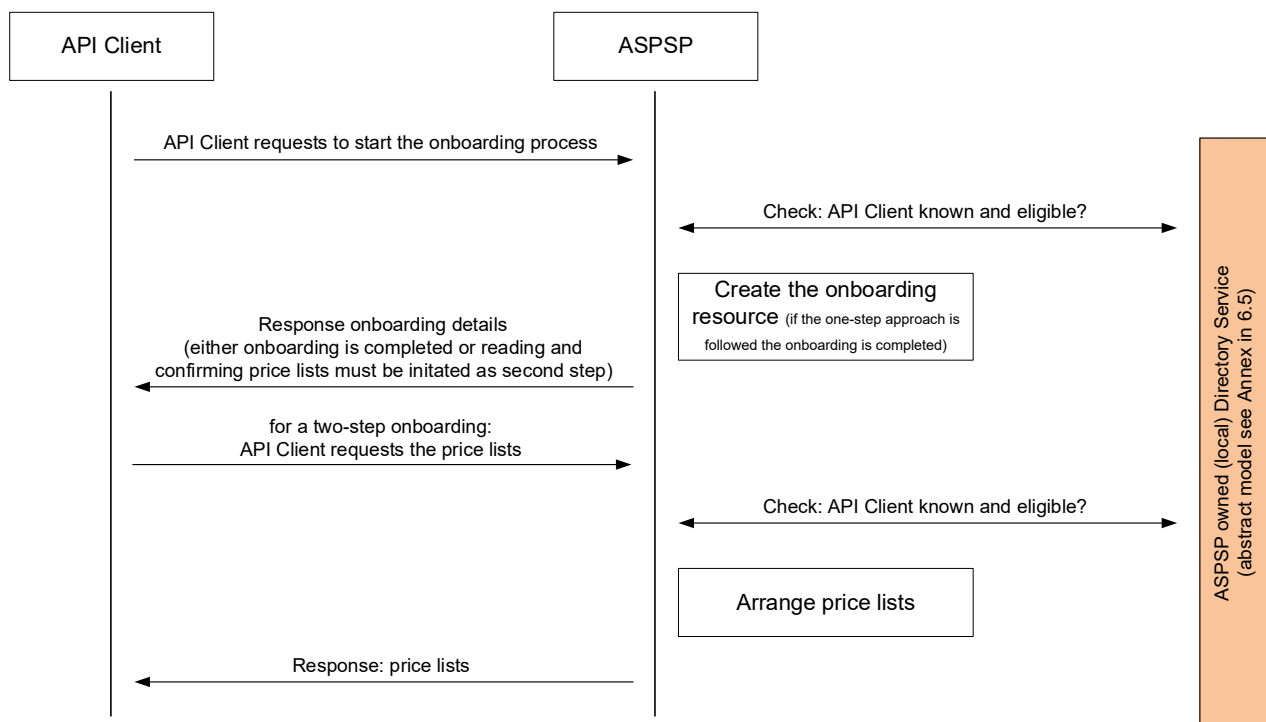


Figure 6: API Service: Start of the onboarding

The API Client sends a request message to the onboarding endpoint of the openFinance API of the ASPSP.

The ASPSP creates an onboarding resource with a corresponding identifier and checks the identity of the client, the certificates used in the request message and if a service contract is already established with the client. The authentication process of an API Client is explained in detail in [oFA-PFSM].

It is decided by a scheme if a one-step approach or a two-step approach is used for the onboarding process. In case no scheme is involved, the ASPSP decides on the kind of approach.

If the verification of the client is successful, in the one-step approach the status of the onboarding resource is set to valid and the ASPSP includes the API-Contract-Id in the response.

In the two-step approach, the onboarding resource status is now set to received. Then, the client must read and confirm the price conditions, before the onboarding process is completed. Thus, the API Client requests to read the price lists and the ASPSP must present the price model that applies exactly to the client.

Remark: For the two-step approach, the necessity of the second check of the directory service depends on the local implementation of the openFinance API.

Reading price lists can be repeated by the API Client at any time. The response of the ASPSP may contain more than one price list, if besides the current price list a new price list already exists.

The figure above shows only the "pull version" for requesting price lists, i.e. the API Client sends a request for a price list and the ASPSP sends the price list as part of the response. If a contract has already been established between an ASPSP and an API Client it is also possible that the ASPSP "pushes" a new price list to the API Client as soon as the current price list has been changed, if ASPSP and API Client both support the push service function, see [oFA-OR-FW] for details.

The ASPSP may change the prices for its services either unilateral or bilateral negotiated with an API Client. Each price list will indicate the starting date and the end date of its validity.

Restrictions about the next possible date for adapting new prices for a service should be part of the contract the API Client accepts for using the service.

3.2 API Service: Reading and confirming the Price Lists

For this API service two different variants have to be distinguished:

- The API Client accepts the price lists for the requested services offered by the ASPSP for the first time.
- The API Client has completed the onboarding already with the ASPSP but some details/prices have been changed. In this case the existing price lists have to be adapted and renewed.

Variant: First-time onboarding

As soon as an API Client has got all information about a service offered at the openFinance API by an ASPSP, the API Client can accept the conditions and perform the technical onboarding to the openFinance API for this service. In the following it is assumed that this is done in two steps.

First the API Client requests the technical onboarding for a given service. The ASPSP checks this request. If it is OK the ASPSP will send the applicable service price list to the API Client. In the second step the API Client checks the service price list for API usage. If OK the API Client will sign the price list and sends it back to the ASPSP. If verified by the ASPSP successfully the technical onboarding of the API Client has been finished successfully. The first step is detailed in Section 3.1. This section focuses on the completion of the second step.

The following figure shows only the very top level information flow:

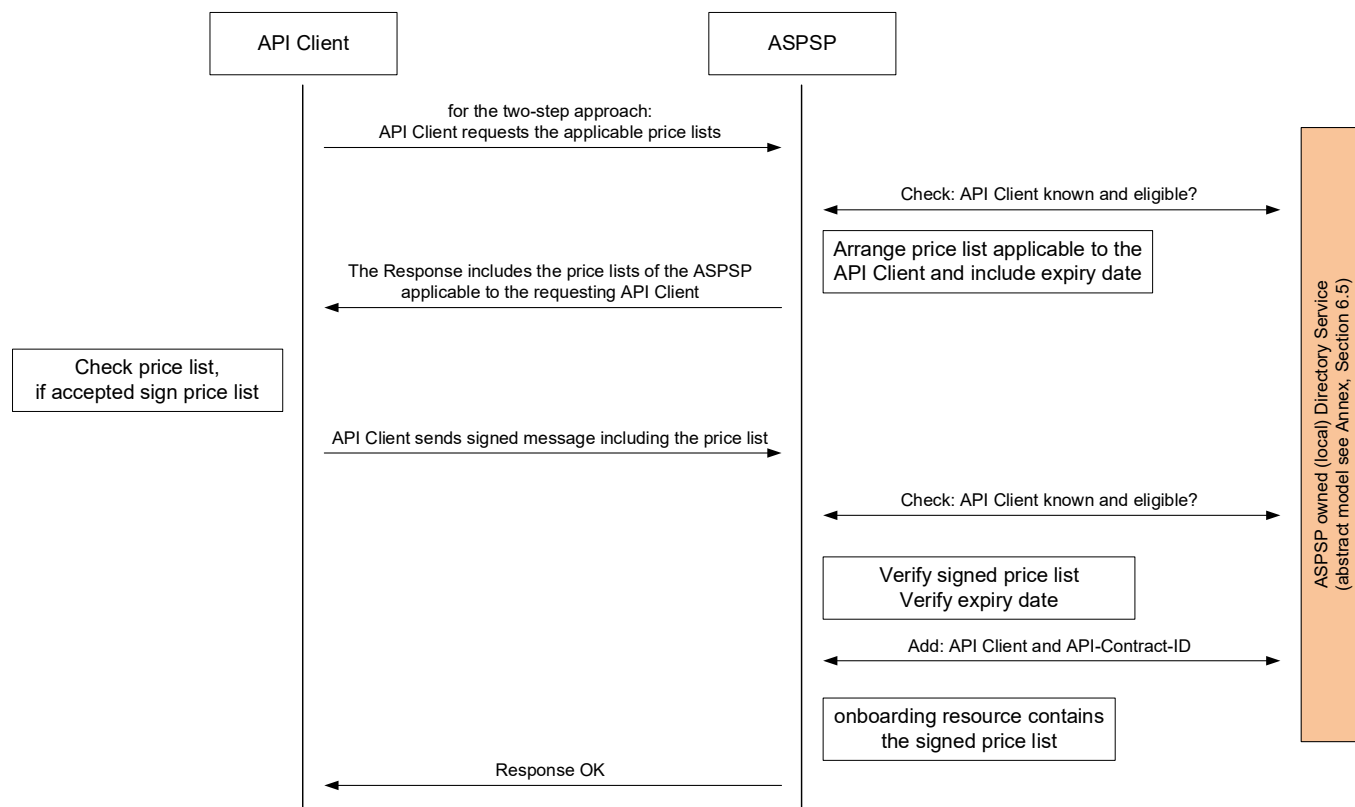


Figure 7: API Service: Accept the price lists – first time

Remark:

If the client accepts the conditions of the price list, the API Client returns the unchanged price lists to the ASPSP. This API Call must be a signed HTTP message using the QSealC of the API Client. The signing follows the HTTP message signing as specified in [oFA-PFSM].

The price list arranged by the ASPSP shall contain an expiration date of that price list. If the API Client returns the signed price lists after this expiration date the onboarding will be rejected by the ASPSP.

If the confirmation of the API Client is verified by the ASPSP, the technical onboarding of the API Client has been finished successfully. The onboarding status is set to valid and together with the expiration date and the unique API-Contract-Id included in the ASPSP's response. This shall be used by the API Client for further accesses to the corresponding services at the openFinance API of the ASPSP.

The API Client can read the onboarding resource including the price lists at any time by requesting to retrieve this resource.

3.3 API Service: Read the onboarding resource

The API Client can read all information about its onboarding using a GET on the corresponding resource. The following figure shows only the very top level information flow:

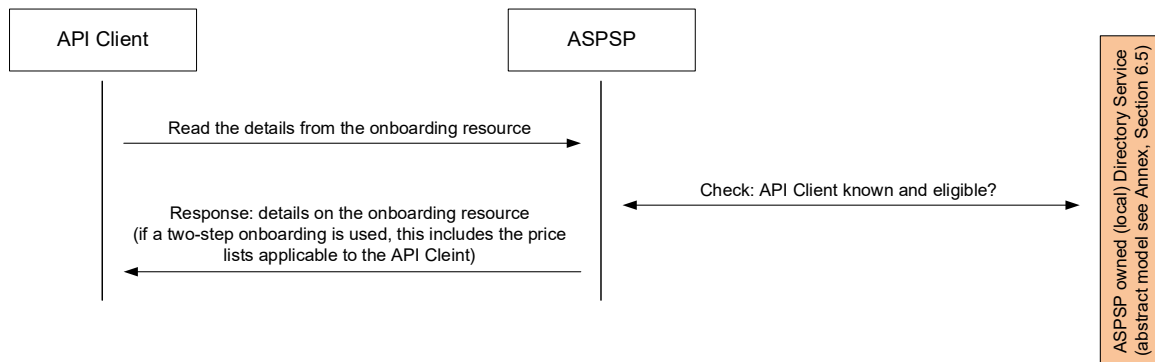


Figure 8: API Service: Read Service Contract

The response must include the onboarding resource, its status, and if the onboarding is already completed, the API-Contract-Id and the validity period must be included. If a price model was confirmed during the onboarding process, it must be included in the response as well.

The following status values might occur:

- **valid:** the onboarding process is completed and the client may initiate services using the API-Contract-Id
- **received:** used only in the two-step onboarding process, when the onboarding process was already initiated by the client and the client was verified by the ASPSP but no confirmation of the price model took place yet
- **validTimeRestricted:** the resource is still valid but a date when the onboarding resource will expire has been set. Reasons for this will be defined by the business logic.
- **expired:** the onboarding was once valid, but the expiry date has been reached, thus the resource is expired
- **rejected:** the onboarding request has been rejected during the onboarding process by the ASPSP (e.g. if an incorrect certificate was used)
- **terminatedByApiClient:** the API Client requested the deletion of the onboarding resource, and the ASPSP accepted this request, the onboarding resource is invalid, and
- **terminatedByAspsp:** the ASPSP has terminated the corresponding onboarding, the resource is invalid.

3.4 API Service: Cancellation of the API Client's onboarding (by API Client)

Using this API service the API Client can request the cancellation of its own onboarding which has been concluded with the ASPSP before. By requesting the cancellation the API Client will lose its rights to use the corresponding service of the ASPSP.

Following requests to corresponding services by the API Client will be rejected by the ASPSP when using the API-Contract-Id corresponding to the onboarding resource. All liabilities of the API Client resulting from former requests before the cancellation stay valid (i.e. the API Client still has to pay the fees resulting from former requests before the cancellation).

It is up to the ASPSP to decide when the corresponding resource will be deleted. Until that deletion all accesses to that resource are rejected with "contract cancelled".

It is up to the ASPSP to decide when the corresponding resource will be deleted. Until that deletion all accesses are rejected and the onboarding status is set to `terminatedByApiClient`.

The following figure shows only the very top level information flow:

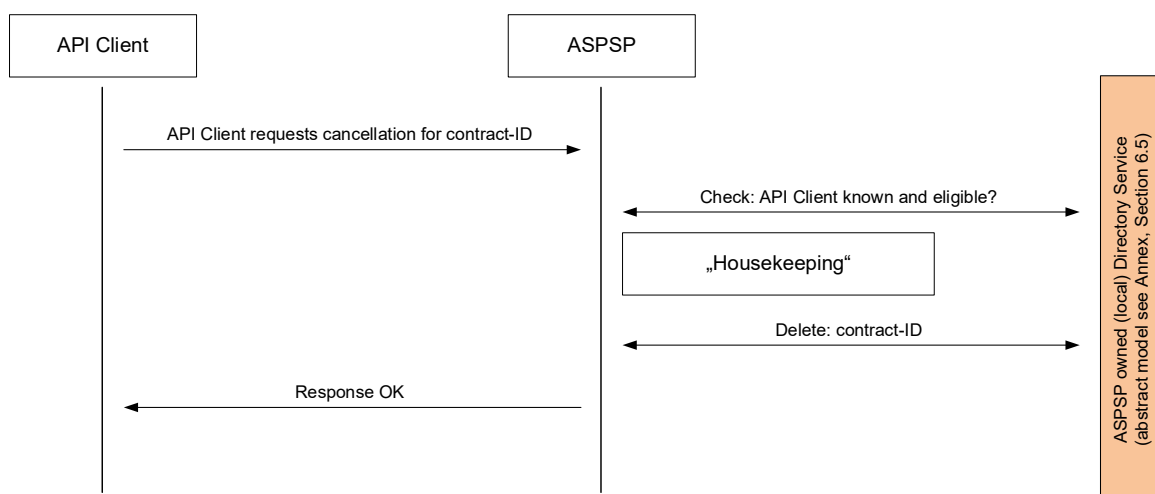


Figure 9: API Service: Cancellation of the onboarding (by API Client)

3.5 API Service: Updating the certificate of the API Client

The API Client must inform the ASPSP in case, the API Client wants to use a new certificate for future calls. Then, the API Client must update the information about the certificate corresponding to the onboarding resource.

Remark: If the API Client has got more than one onboarding resources for accessing the ASPSP's APIs, then this API Service has to be repeated for each onboarding resource.

The very top information flow is presented in the following diagram:

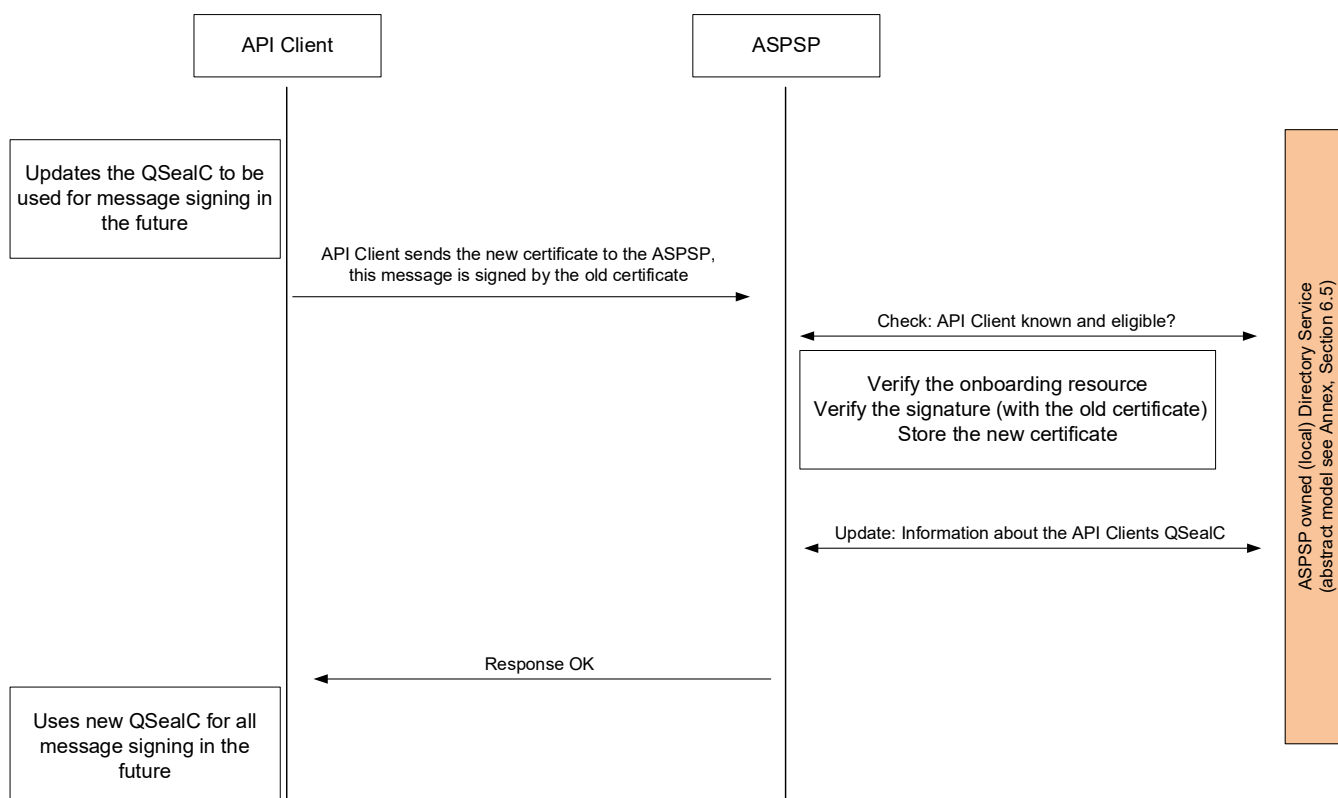


Figure 10: Updating the QSeal certificate

The API Client informs the ASPSP that it will use a new QSealC for message signing in future calls. For the signing of this request, the API Client must use the current QSealC known by the ASPSP.

After the verification of the request, the ASPSP confirms the update of the certificate. The original certificate stays valid and the API Client might still use it for message signing until it is expired. The new certificate may be used for message signing by the API Client as soon as the update has been successful.

4 Operational rules

This section summarises the operational rules specifically to be observed by each API Client accessing the openFinance API for the Administrative Services and each ASPSP providing the Administrative Services at the openFinance API. In addition, the general operational rules as specified in [oFA-OR-FW] apply.

Not all of these rules are enforced by technical means of the Administrative Services within the openFinance API.

The order of the rules does not represent an order of importance.

4.1 API Client Identification

Accessing API Clients are identified based on eIDAS certificates in analogy to the Core XS2A Interface, but without mandating PSD2 specific attributes in the certificate.

Remark:

The eIDAS certificate infrastructure is used for this to allow a homogeneous infrastructure for the issuance/usage and verification of certificates used to secure all API accesses (i.e. XS2A API and openFinance API).

The conditions (legally, organisational, technical) for issuing these eIDAS compliant qualified certificates are regulated by the QTSP.

If an API Client has got an eIDAS compliant qualified certificate from a QTSP this certificate has to be accepted by all ASPSP for the identification of that API Client at the openFinance API, regardless which QTSP has issued the certificate.

The API Client has to inform the ASPSP/Scheme about the QTSP, who issued its qualified certificates. This can be done for example as part of the signing of a scheme contract. This has to be in time such that the ASPSP has got enough time to prepare the acceptance of the certificates within its systems. The way how this information about the QTSP is exchanged is out of scope for the operational rules of the openFinance API Framework.

Remark:

This requirement is not relevant for the access of a service provider who is already a corporate client of the ASPSP. Requirements for the identification of these corporate clients at the openFinance API of the ASPSP can be arranged bilaterally between the corporate client and the ASPSP. Of course the bilateral arrangement do not determine anything about the access to openFinance APIs of other ASPSPs.

4.2 API Client Authorisation

ASPSPs might refuse access to the openFinance API even if the related eIDAS certificate is valid, if not all related contract conditions are fulfilled. This might be controlled e.g. via a scheme directory.

4.3 Consent of the PSU

A PSU is not involved in the execution of the administrative services. Proving and verifying the PSU's consent does not apply to the administrative services.

4.4 Decision about strong customer authentication

The administrative services do not involve a PSU. Therefore strong customer authentication of a PSU is not executed.

4.5 Onboarding process of the API Client at the ASPSP's API

An API Client initiates its onboarding procedure at the ASPSP's API by sending an onboarding request to the onboarding endpoint of the ASPSP.

The onboarding of an API Client might be a one-step process or a two-step process where a retrieval and confirmation on the price model is needed to complete the onboarding. A scheme decides if a one-step approach or a two-step approach is used for the onboarding process. In case no scheme is involved, the ASPSP decides on the kind of approach.

After the onboarding process is successfully completed, the ASPSP returns a unique identifier, called the API-Contract-Id in the response message. The API Client needs to use this for all future accesses towards an API of the ASPSP.

The status of the onboarding resource created during the onboarding process is set to valid and the API Client must be informed of its validity period.

4.6 The price lists presented during the onboarding of an API Client

During the second step of the onboarding process, the potential API Client receives the ASPSP's current price lists, which must be confirmed by the API Client to complete the two-step onboarding procedure. The price list yields an organisational overview of all the prices and fees that may apply at the API for the API Client.

Each entry of the price list leads to the price model of all business services related to a service type offered by the ASPSP via its APIs. If an ASPSP does not offer an extended service, the corresponding entry for this service is not included in the price list.

If an ASPSP offers an extended service but there exists no corresponding entry yielding the price or fees that would apply for this service in the price list, the ASPSP may not demand a price for the execution of this service.

It is not allowed for (a summand of) a price to occur when calculating the price of a specific API Call if not all parameters for the calculation are used in the API Call, e.g. if a parameter may be used optionally by the API Client and is not used in the specific API call that is to be priced.

The price lists are given as a JSON data structure. The prices presented in such a price list might be confidential and the ASPSP must ensure that the prices are precisely the fees and conditions that currently apply for the client that requested them.

In particular, if a bilateral contract containing a price model differentiating from the fall-back prices was established with the client beforehand, then the price list must include the client-specific prices and the ASPSP must ensure to keep the price information confidential.

The ASPSP may return more than one price list in the response, if besides the current price list a price list for future use already exists.

The validity time of any price list must be included. Similarly, the price list might be given with or without change fees included. This must be made transparent for the API Client as well.

4.7 Fall-back price model

If no bilateral contract exists, the price lists that are offered to the potential API Client by the ASPSP reflect the fall-back price model of the ASPSP or the API access scheme.

This model is pre-defined and might also be made available to a wider group of potential API Clients. It is the ASPSP's responsibility to consider any requirements of the current legal situation when making price lists available to other parties and an ASPSP of course has to consider any requirements of national or European law.

4.8 Confirmation of a price model

The API Client sends a signed confirmation request where the price list(s) are contained in the body of the message to confirm the price model that was presented by the ASPSP beforehand.

This part of the confirmation is saved within the onboarding resource.

4.9 Information on the current price model and next invoicing date

The API Client may retrieve the onboarding resource at any time during the onboarding process as well as after the onboarding is completed.

If the onboarding of the API Client required the confirmation of price lists, the onboarding resource must include the confirmed price lists and the next date at which the invoice is issued to the API Client in addition to the other onboarding resource data.

4.10 Validity of an onboarding

After the onboarding has been completed the onboarding resource has got a fixed period of validity.

After the onboarding resource is expired, any requests of the API Client for the corresponding services will be rejected at the openFinance API of the ASPSP. All liabilities of the API Client resulting from former requests before the cancellation stay valid (i.e. the API Client still has to pay the fees resulting from former requests before the cancellation).

It is not possible to renew an expired onboarding resource. The API Client has to initiate a new onboarding process at the onboarding endpoint of the ASPSP.

The ASPSP and the API Client both can terminate the service contract by issuing a delete of the onboarding resource internally resp. by a dedicated API Call. This affects the validity period of the onboarding as well, see Sections 4.11 and 4.12.

4.11 Cancellation of the onboarding resource by an API Client

An API Client may cancel an existing service contract with an ASPSP again, if the API Client wishes to do so. Periods of notice to be considered for the cancellation are ruled by the service contract.

The API Client has to use the corresponding service of the openFinance API of the ASPSP for the cancellation of a service contract. By requesting the cancellation the API Client will lose its rights to use the corresponding services of the ASPSP and pricing rules.

After the cancellation of the service contract is issued successfully, the status is set to `terminatedByApiClient` and the resource becomes invalid. Any requests of the API Client for the corresponding services using the `API-Contract-Id` corresponding to the onboarding resource will be rejected at the openFinance API of the ASPSP. Any transition period for this will be defined by the service contract. All liabilities of the API Client resulting from former requests before the cancellation stay valid (i.e. the API Client still has to pay the fees resulting from former requests before the cancellation).

4.12 Cancellation of a service contract by an ASPSP

An existing service contract may be cancelled by an ASPSP if an API Client has broken some of the requirements or rules of the service contract.

The ASPSP has to inform the API Client about this cancellation directly.

After the cancellation of the service contract gets effective, the status is set to `terminatedByAspsp` and the resource becomes invalid. Any requests of the API Client for the corresponding services using the `API-Contract-Id` corresponding to the onboarding resource will be rejected at the openFinance API of the ASPSP. Any transition period for this will be ruled by the service contract. All liabilities of the API Client resulting from former requests

before the cancellation stay valid (i.e. the API Client still has to pay the fees resulting from former requests before the cancellation).



5 Message and data model

In the following, an abstract data model is presented for the specific usage of the administrative services within the openFinance API and the directory services. The basic abstract data model for the openFinance API is defined in [oFA-OR-FW].

This data model will be further detailed in the implementation guidelines [oFA-IG-Adm].

5.1 Protocol Level

There are no specific requirements on data modelling on protocol level. The following data elements are used independently of the semantic of the related messages, building an abstract basic protocol level.

5.2 Authorisation Data Model

Not applicable since no authorisation by a PSU is part of these administrative services.

5.3 Informative: Basic data model for different entities

In this section the basic data model for the entities ASPSP, API Client and Schemes that could be used, e.g. as part of a directory service, is defined. This is not part of a API Services specified in the Implementation Guidelines [oFA-IG-Adm].

5.3.1 ASPSP data

- Name of ASPSP
- Number of bank register (registration number of NCA + name of NCA) (it is assumed that the ASPSP will be identified uniquely by this number)
- Postal address of ASPSP
- Contact persons
 - List of
 - Type of contact (business/legal/technical/etc.)
 - Name of contact (not only persons but also departments, etc. possible)
 - E-mail-address of contact
 - Telephone number of contact (optional)
 - Postal address of contact (optional)

- List of supported schemes
- URL to routing information
- URL to information about supported services
- URL to information about testing facilities

5.3.2 API Client data

- Name of API Client
- List of identification numbers (it is assumed that the ASPSP will be identified uniquely by this number) (examples:)
 - Number of public register (if any)
 - TPP register number of NCA, Name of NCA, if existing
 - LEI
- Postal Mail address
- Invoice postal address
- General account number (IBAN) for debiting fees
- Tax number
- Contact persons
 - List of
 - Type of contact (business/legal/technical)
 - Name of contact (not only persons but also departments, etc. possible)
 - E-mail-address of contact
 - Telephone number of contact (optional)
 - Postal Mail address of contact (optional)
- References to KYC process
- References to sub scheme approvals (if any)
- References to scheme/sub scheme contracts

5.3.3 Scheme data

This will be detailed later.

5.4 Initiation of the onboarding process

5.4.1 Initiation Request

- Scheme Id
- Service Ids
- apiClientContacts
- ServiceContractId

5.4.2 Initiation Response

- Onboarding Id (mandatory)

Technical identifier used during the onboarding process

- Onboarding Status (mandatory)
- API-Contract-Id (Conditional)

Included if and only if the one-step onboarding approach is used, this yields a unique identifier used by the API Client to initiate services at the ASPSP APIs in the future

- Valid to (Conditional)

Only included if the onboarding status is set to valid, indicates the end of the validity period of the onboarding resource.

- Valid From (optional)
- Links to read price conditions and confirm conditions (conditional)

Must be included for the two-step approach to initiate the second step

5.5 Reading price lists

5.5.1 Reading price lists Request

No specific parameters used.

5.5.2 Reading price lists Response

- Price Lists (Mandatory)

JSON data structure containing the fees and prices

- next invoicing date (Mandatory)

the next date when the API Client will receive an invoice if the onboarding process will be completed

- Link to confirm conditions (mandatory)

Confirmation of price lists

5.5.3 Confirmation Request

- Price Lists (Mandatory)

The price list/lists that were received before and now are confirmed via this request must be included in the request message without any changes.

5.5.4 Confirmation Response

- Onboarding Status (mandatory)
- API-Contract-ID (Mandatory)

This yields a unique identifier used by the API Client to initiate services at the ASPSP APIs in the future.

- Valid to (Mandatory)

Only included if the onboarding status is set to valid, indicates the end of the validity period.

- Valid From (optional)

5.6 Reading the status of the onboarding process

5.6.1 Status Request

No specific parameters.

5.6.2 Status Response

- Onboarding status (Mandatory)

5.7 Reading the onboarding resource

5.7.1 Reading the onboarding resource Request

No specific parameters.

5.7.2 Reading the onboarding resource Response

- Onboarding status (Mandatory)
- Scheme Id (mandatory)
- Service Ids (Mandatory)
- API-Contract-ID (Conditional)

Unique identifier of the successful onboarding of an API Client. This must be included if it is contained in the onboarding resource (i.e. after the API Client's onboarding is completed successfully)

- Valid to (Conditional)

Only included if the onboarding status is set to valid, indicates the end of the validity period.

- Valid from (Optional)
- Price Lists (Conditional)

JSON data structure containing the fees and prices that apply for the API Client, only included if the two-step onboarding approach is used

- next invoicing date (Conditional)

the next date when the API Client will receive an invoice based on the price conditions given in the price lists, only included if price lists are included in the onboarding resource.

5.8 Cancellation of the onboarding by the API Client

5.8.1 Cancel Request

No specific parameters.

5.8.2 Cancel Response

No specific parameters.

5.9 Updating the QSealC certificate used by the API Client

5.9.1 Updating Request

- new certificate (Conditional)

The new QSealC must be provided in the request body.

5.9.2 Updating Response

No specific parameters.

6 Annexe

6.1 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].

API Client

Service provider accessing the openFinanceAPI of an ASPSP. A TPP is a possible example of an API Client.

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].

LEI

Legal Entity Identifier

openFinance API

Premium access to account interface – interface provided by an ASPSP to API Clients for accessing extended services offered by the ASPSP.

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 for PSD2 regulated services.

6.2 References

- [oFA-IG-Adm] openFinance API Framework, Implementation Guidelines, Administrative Services, Version 1.2, 31 July 2026
- [oFA-OR-FW] openFinance API Framework, Operational Rules, Basic Operational Rules for the Framework, Version 2.1, 31 March 2026
- [oFA-OR-Com] openFinance API Framework, Operational Rules, Compliance Services, Version 2.1.1, 31 July 2026
- [oFA-PFSM] openFinance API Framework, Implementation Guidelines, Protocol Functions and Security Measures, Version 2.4.1, 31 July 2026
- [oFA-IG-Com] openFinance API Framework, Implementation Guidelines, Compliance Services, Version 2.4.2, 31 July 2026
- [oFA-DaD] openFinance API Framework, Implementation Guidelines, Data Dictionary, Version Draft 2.4.1, 31 July 2026
- [RTS] Commission Delegated Regulation (EU) 2018/§() 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
- [EBA-OP] Opinion of the European Banking Authority on the implementation of the RTS on SCA and CSC, EBA-Op-2018-04, 13.06.2018
- [EBA-OP2] Opinion of the European Banking Authority on obstacles under Article 32(3) of the RTS on SCA and CSC, EBA-Op-2020-10, 4 June 2020
- [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
- [TS 119 495] Draft ETSI TS 119 495, Electronic Signatures and Infrastructures (ESI); Sector Specific Requirements; Qualified Certificates Profiles and TSP Policy Requirements under the Payment Service Directive 2015/2366/EU, V1.1.2 (2018-07)
- [PSD2] Directive (EU) 2015/2366 of the European Parliament and of the Council on Payment Services in the Internal Market, published 25 November 2016
- [OBE-JWS] JSON Web Signature Profile for Open Banking, proposed final draft for approval, 4.09.2020



6.3 List of figures

Figure 1: Core XS2A interface and openFinance API	2
Figure 2: Scenario "services organised by a scheme"	7
Figure 3: Scenario "services organised by ASPSP – client is new partner"	8
Figure 4: Scenario "services organised by ASPSP – client is a known partner"	9
Figure 5: layers of response presenting the price list	17
Figure 6: API Service: Start of the onboarding	28
Figure 7: API Service: Accept the price lists – first time	31
Figure 8: API Service: Read Service Contract	32
Figure 9: API Service: Cancellation of the onboarding (by API Client)	33
Figure 10: Updating the QSeal certificate	34
Figure 14: Overview directory services	51
Figure 15: Directory Service Add entity to directory	53
Figure 16: Directory Service Update entity within directory	54
Figure 17: Directory Service Read entity from directory	54
Figure 18: Directory Service List of entities	55
Figure 19: Directory Service Delete entity from directory	56

6.4 List of tables

Table 1: Administrative services	25
Table 2: API Services of the Administrative Services	26
Table 3: API Services of the Directory Services	52

6.5 Informative: Abstract model of directory services

6.5.1 Introduction

For the access to an openFinance API by an API Client as well as for the execution of requests at an openFinance API by the ASPSP information about the status of an API Client or of an ASPSP are needed. This status information about an API Client or an ASPSP may change by time or by the execution of different administrative steps.

This status information about API Clients will **not** be contained in the certificates API Clients will use for their identification at the openFinance API of an ASPSP. It is assumed/suggested by the openFinance API Framework that this information is managed and distributed using directory services.

A related directory service standardisation is not (yet) part of the openFinance API Framework, since it is not directly impacting the API standard as such. Still, it was seen helpful to add an abstract model of directory services, to give API access schemes and ASPSPs a guidance for developing directory services. This abstract model defines functional requirements for the directory services to be used in the context of the openFinance API.

It is up to API access schemes and/or ASPSPs to specify the directory services in detail that are needed in their environment. The directory services can be provided by an external Directory Service Provider or by an ASPSP itself.

ASPSP information needed by an API Client is (b.o.) the following:

- Is the ASPSP a participant of a given scheme?
- Which extended services are offered by an ASPSP at its openFinance API?
- How can the openFinance API of an ASPSP be accessed? (routing information)
- Persons to be contacted?

API Client information needed by an ASPSP is (b.o.) the following:

- Has the API Client already signed a business contract to access an extended service?
- Has the API Client already finished successfully the certification process of a certification body for an extended service?
- Has the API Client got an approval certificate for the extended service?
- Has the API Client finished the technical onboarding for the extended service?
- Persons to be contacted?

API Clients can only get information from these directory services and update some contact data, while ASPSP can get and add information from/to these directory services. In addition it

might be necessary that also API scheme owner and certification bodies can get and add information from/to these directory services. The following figure shows an overview:

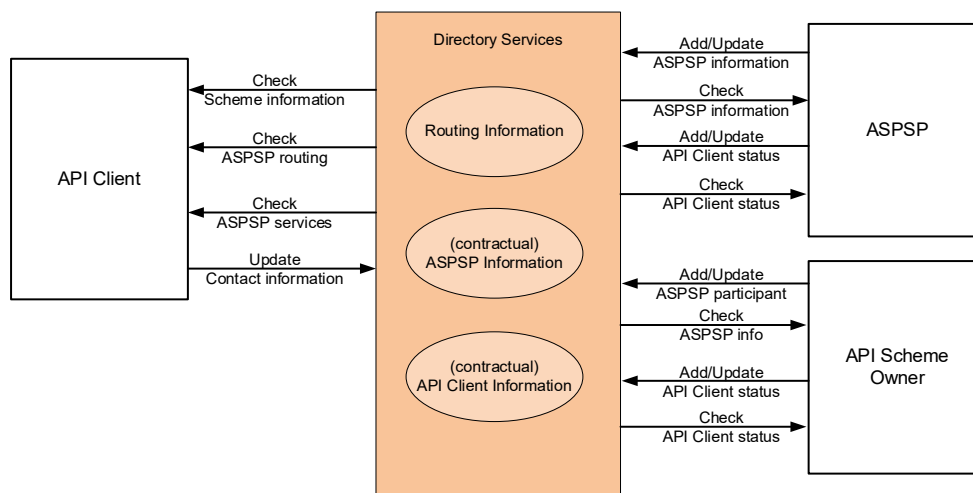


Figure 11: Overview directory services

Remarks:

- This section defines only an abstract model for directory services. Nothing is determined if the necessary information will be contained in one directory or will be distributed over different directories.
- Directories can be for example managed on a local level (by an ASPSP) or centralised on a scheme level (by a scheme owner).
- These (contractual) directory services indicating the participation of ASPSP and API Clients to schemes or contractual arrangements should not be confused with the (functional) Discovery Service specified by the openFinance API Framework. Information about the special arrangements of an extended service supported by an ASPSP will be part of the Discovery Service, not part of the directory service. The same holds for information about testing support and facilities offered by an ASPSP.
- These directory services should also not be confused with services supported by a QTSP to validate certificates of an API Client. These directory services will not contain any information about certificates.
- Detailed specification and requirements for the implementation of these directory services is out of the scope of the openFinance API Framework.
- The Berlin Group will not define anything about the organisation and operation of these directory services.

- In any case the owner of a directory (API access scheme or ASPSP) is responsible that the content is reliable and up to date. Requirements for update cycles have to be defined by the owner of the directory service. These requirements are not determined by the openFinance API framework.
- If a scheme owned directory exists at the scheme level and a local directory owned by an ASPSP it will be necessary that the relevant content of these directories has to be synchronised time by time. Requirements and processes for the updates of a directory owned by an ASPSP by a (partial) download from a directory owned by a scheme will be defined by the framework.

For this abstract model the access to the directory services will also be defined as API services. This is only done to show the necessary functionality.

6.5.2 Overview: Functionality of a directory service

The following table gives an overview of the API services:

API Services		Technical Functionalities	
Add entity to directory		Add new entity to directory	
Update entity within directory		Update content of an existing entity of the directory	
Read entity from directory		Read content of an existing entity of the directory	
List entities		Read a list of entities with given attributes	
Delete entity from directory		Delete existing entity from directory	

Table 3: API Services of the Directory Services

For modelling the API services to support the (contractual) Directory Services the following is assumed:

- It is assumed that the information contained in the directory service is organised in entities. Each entity is defined by a set of parameters.
- Each entity has got a type (API Client information, ASPSP information, scheme information, etc.) and is represented by a unique key.

- An entity can be added to the directory and deleted from the directory.
- The content of the parameters of an entity can be read from the directory.
- The content of some of the parameters of an entity can be changed within the directory.
- It is possible to get a list of all entities of a given type.

For controlling the access to a (contractual) Directory Service the following is assumed:

- As a best practise approach it is assumed that all external accesses to the directory service will be secured by client certificates for web site authentication (i.e. with authentication of the client). For an API Client this can be the same certificate as used for accessing the openFinance API.
- For internal accesses the provider of the directory service has to determine the requirements for securing these accesses.
- The access to the Directory Service is secured by a sufficient Access Control System, by which for each request it can be decided if the requestor is eligible to access the Directory Service or not.

6.5.3 Directory Service: Add entity to directory

With this service a new entity can be added to the directory. The type, the unique key and the content of the new entity have to be provided with the request.

The following figure shows only the very top level information flow:

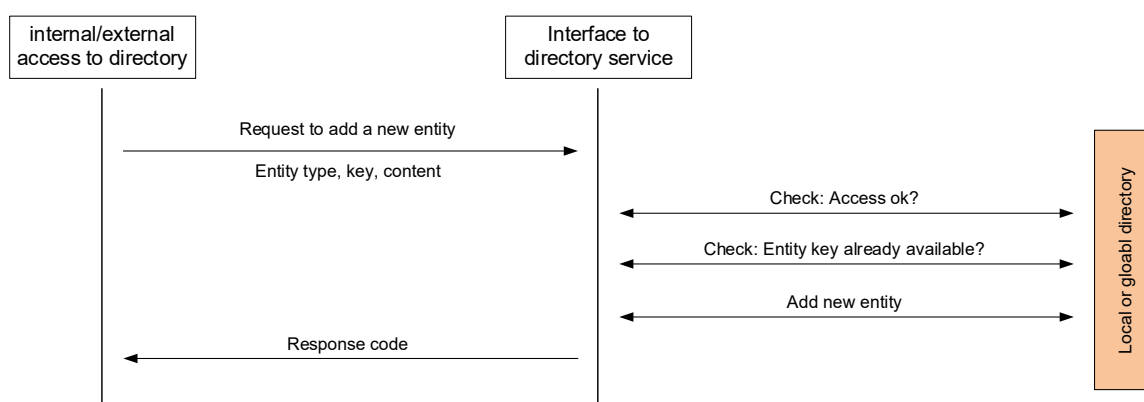


Figure 12: Directory Service Add entity to directory

If an entity with the key given by the request exists already, the request is rejected. Otherwise the new entity with the content of the request is added to the directory.

6.5.4 Directory Service: Update entity within directory

With this service the content of an entity contained in the directory can be changed. The type, the unique key and the new content of the entity have to be provided with the request.

Of course not all parameters of an entity may be changed. Any change request for these parameters have to be rejected.

The following figure shows only the very top level information flow:

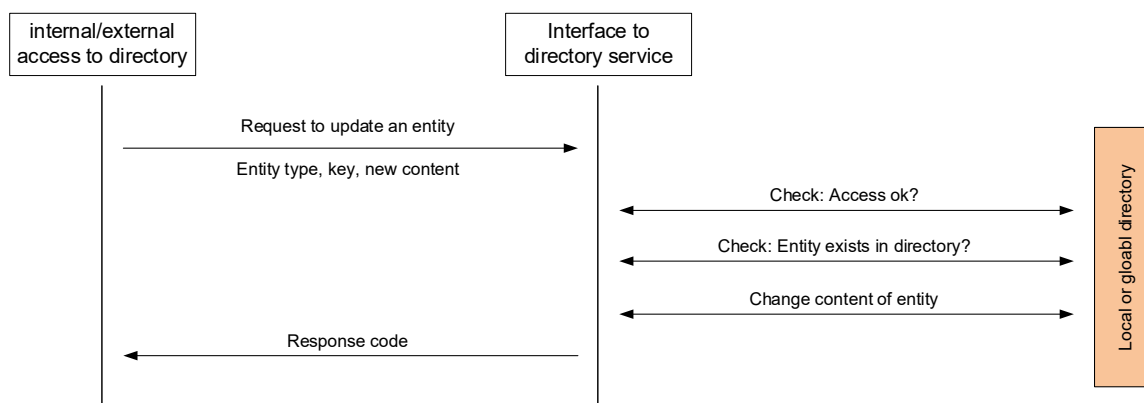


Figure 13: Directory Service Update entity within directory

If an entity with the key given by the request does not exist in the directory, the request is rejected. Otherwise the content of the entity within the directory is changed to the new content of the request.

6.5.5 Directory Service: Read entity from directory

With this service the content of an entity contained in the directory can be read. The type and the unique key of the entity have to be provided with the request.

The following figure shows only the very top level information flow:

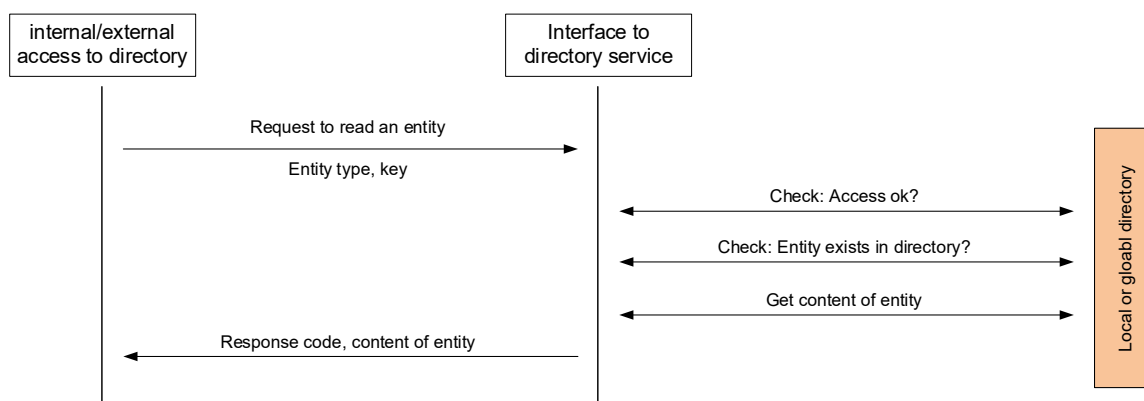


Figure 14: Directory Service Read entity from directory

If an entity with the key given by the request does not exist in the directory, the request is rejected. Otherwise the content of the entity within the directory is given back to the requestor as part of the response.

6.5.6 Directory Service: List of entities

With this API Service a list of all entities with a given type can be read from the directory. The type of the entities has to be provided with the request.

The following figure shows only the very top level information flow:

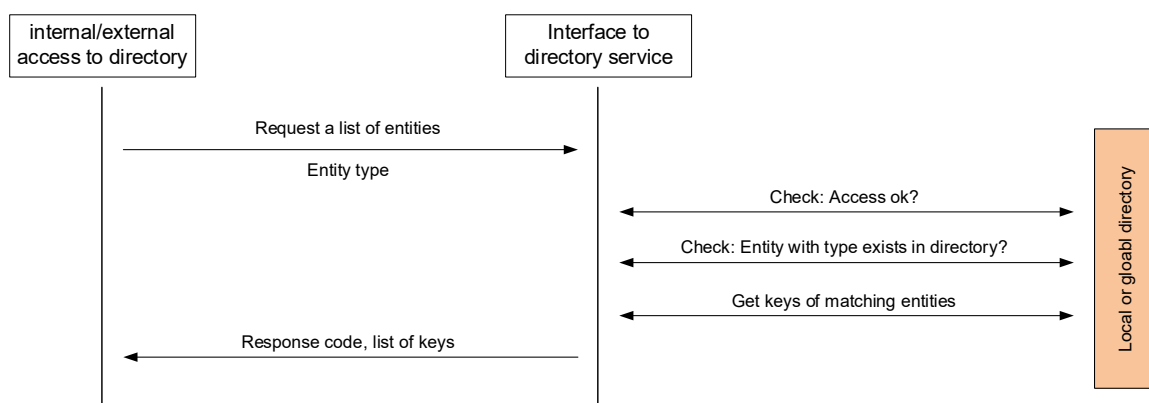


Figure 15: Directory Service List of entities

If no entity with the requested type exists in the directory, the request is rejected. Otherwise a list containing the keys of all matching entities within the directory is given back to the requestor as part of the response.

6.5.7 Directory Service: Delete entity from directory

With this service an entity contained in the directory can be deleted from the directory. The type and the unique key of the entity have to be provided with the request.

The following figure shows only the very top level information flow:

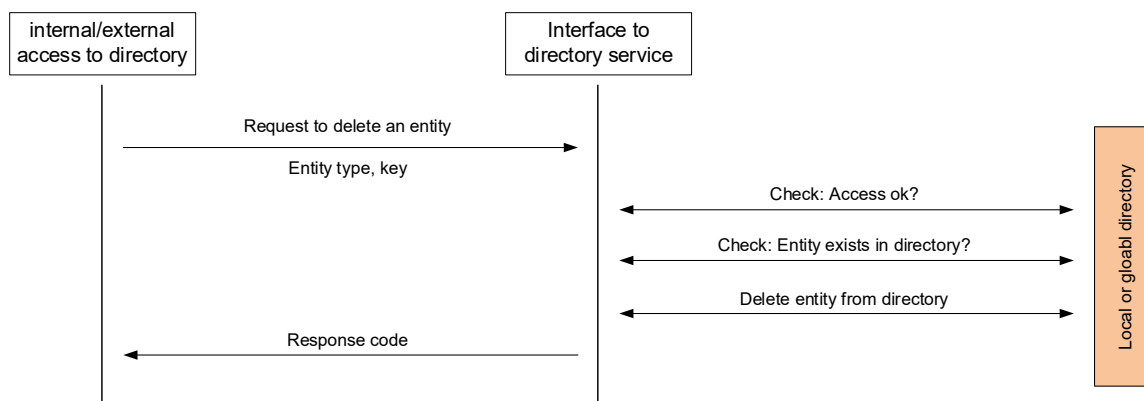


Figure 16: Directory Service Delete entity from directory

If an entity with the key given by the request does not exist in the directory, the request is rejected. Otherwise the entity is deleted from the directory.