



openFinance API Framework Implementation Guidelines

Administrative Services

Version 1.2

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	Administrative Services	2
1.3	Document Structure.....	3
1.4	Document History	4
2	Application Layer: Guiding Principles.....	5
2.1	Signing Messages at Application Layer	5
2.2	Overview: Transaction flow for the onboarding process.....	5
2.3	Overview: Status changes for an onboarding resource.....	7
2.4	Overview: Structure of the price list	10
2.5	Best practises and examples for price models.....	14
2.5.1	Establish a base price of an API call	16
2.5.2	Establish a fixed price point possibly related to the value of a parameter	16
2.5.3	Establish a relative price given by a percentage and depending on the value of a parameter used in the API call	18
2.5.4	Establish minimum and maximum prices for an API call	20
2.5.5	Establish general flat fees for the API usage	21
2.5.6	Establish different price models depending on the account usage	22
2.5.7	Establish different price models depending on the API Client's usage of the API	22
2.5.8	Establish prices for the usage of a service consisting of multiple API Calls	25
2.6	Endpoints and API Access Methods	26
3	Administrative Service	28
3.1	Initiation of the onboarding process	28
3.2	Reading price lists	32
3.3	Confirmation of price lists	34
3.4	Reading the status of the onboarding	36
3.5	Reading the onboarding resource.....	37

3.6	Cancel the onboarding	39
3.7	Information about a new QSealC of the API Client	40
4	References	43
4.1	Informative References	43
4.2	Normative References	43
4.3	Detailed Change Log	44
4.3.1	Changes from Version 1.1 to Version 1.2	44



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:

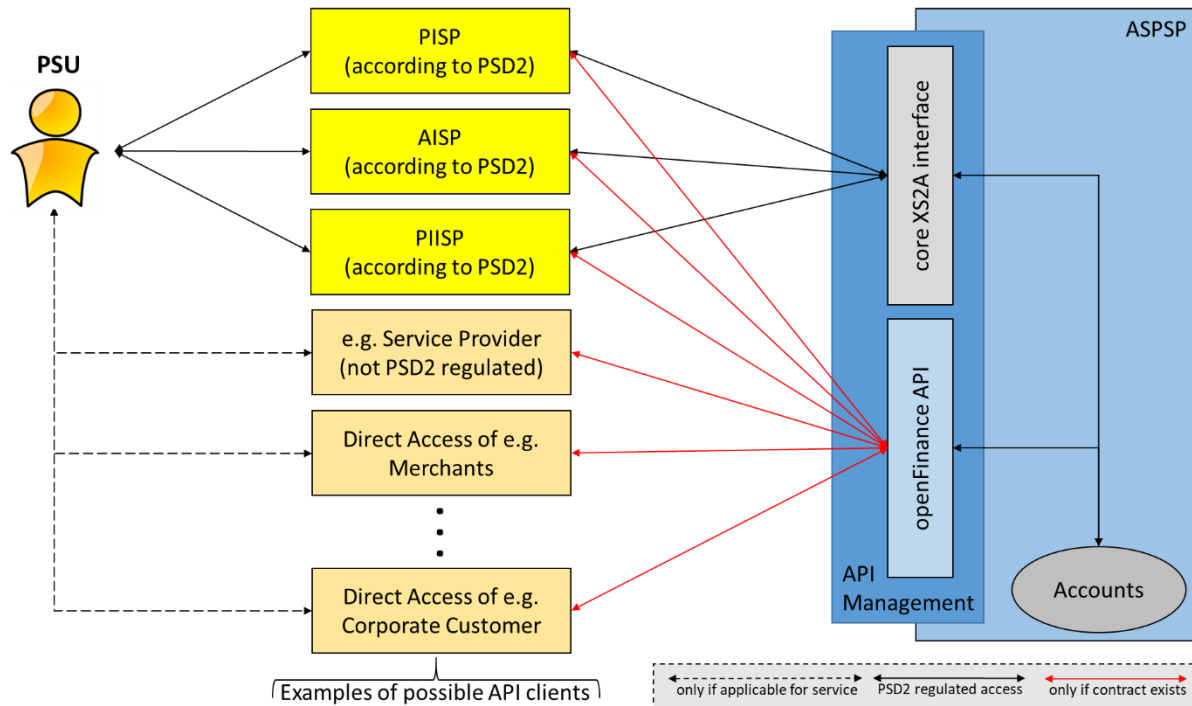


Figure 1: Core XS2A interface and openFinance API

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 out of scope of this document. These requirements are described in a dedicated operational rules document [oFA-OR-ADM].

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

1.2 Administrative Services

In contrast to the compliance API of an ASPSP the access to the openFinance API of an ASPSP with the extended services supported by the ASPSP is not open to all possible API clients. Instead, an API Client has to conclude corresponding contracts (either with a scheme or bilaterally with an ASPSP) before the API Client may access the extended services supported by an ASPSP. Which contracts are necessary will be determined either by an access scheme or the ASPSP itself (if no access scheme exists).

After conclusion of necessary contracts an API Client has to execute a technical onboarding at the openFinance API of an ASPSP. As part of the technical onboarding the ASPSP can check if all necessary contractual steps have been finished before. If everything is ok the ASPSP will assign a unique **API-Contract-Id** to the API Client and will return this to the API

Client as response of the technical onboarding. The API Client has to use this API-Contract-Id for all further accesses to the extended services at the openFinance API of the ASPSP.

Currently these implementation guidelines for Administrative Services describe only the services for the technical onboarding of an API Client at the openFinance API of an ASPSP. Please notice that [oFA-OR-ADM] contains more services which are currently not contained in these implementation guidelines.

For the technical onboarding a "one step process" and a "two step process" are distinguished.

The one step process shall be used if all conditions of the access to the extended services have been determined finally as part of the contracts concluded before. In this case nothing has to be accepted or confirmed by the API Client as part of the technical onboarding.

In contrast the two step process allows to ask the API Client for an explicit confirmation of special conditions determined by the ASPSP as part of the technical onboarding. The current specification only allows the confirmation of the special price conditions for using the extended services of an ASPSP by the API Client.

Remark:

If a scheme determines the rules and conditions for using the extended services of its participants it is up to the scheme to determine if only the one step process shall be supported for the technical onboarding or if also the two step process will be possible. If no scheme exists this has to be decided by the ASPSP.

Remark on version 1.1 of this document:

The new endpoint "onboardings/{onboardingId}/certificates" has been added.

All POST request messages used for the administrative services have to be signed http messages using the certificate (QsealC) of the API Client.

If the API Client will change this certificate for message signing after the onboarding with an ASPSP has been finished, the API Client has to inform the ASPSP about this using this new endpoint.

1.3 Document Structure

This document specifies the Administrative Services in detail.

Section 2 provides specific, but still abstract information on the application layer, including the message signing to be used by the API Client during the onboarding process in Section 2.1, the transaction flow during the onboarding process (for a one-step and a two-step approach) in Section 2.2 as well as an overview of the process of status changes for the onboarding resource in Section 2.3.

The two step-onboarding process requires the API Client to agree to the ASPSP prices for the API usage during the onboarding. For this the ASPSP must present the price lists via the API.

Details on the structure and usage of the data model of the pricelists are given in Sections 2.4 and 2.5.

Section 3 defines the API calls for the Administrative Services in detail.

1.4 Document History

Version	Change/Note	Approved
Draft (28.12.2023)	Final draft version for internal consultation	
Version 1.0 (21.02.2024)	Final Version	21.02.2024 openFinance TF
Draft 1.1 (17.07.2024)	new endpoint has been added "onboardings/{onboardingId}/certificates" Confirmation of a price list change from PUT method to POST method	
Version 1.1 (9.09.2024)	No further changes, only references adapted	9.09.2024 BWG
1.2	Integration of price list processes. Updated references. A detailed change log can be found in Section 4.3.1.	openFinance TF, 2026-07-07

2 Application Layer: Guiding Principles

2.1 Signing Messages at Application Layer

For the administrative services all POST request messages according to section 3.1, section 3.3 and section 3.7 have to be signed HTTP request messages as described in section 6 of [oFA-PFSM].

For these request messages the Qualified Seal Certificate (QSealC) of the API Client has to be included into the JWS Protected Header (element x5c), if the certificate has not already been exchanged with the ASPSP before by some other means.

If the API Client will change its certificate after the onboarding has been finished, the API Client has to inform the ASPSP by a POST request message according to section 3.7. This POST request message has to be signed by the API Client using its **current** certificate known to the ASPSP and the request body must contain the new certificate.

2.2 Overview: Transaction flow for the onboarding process

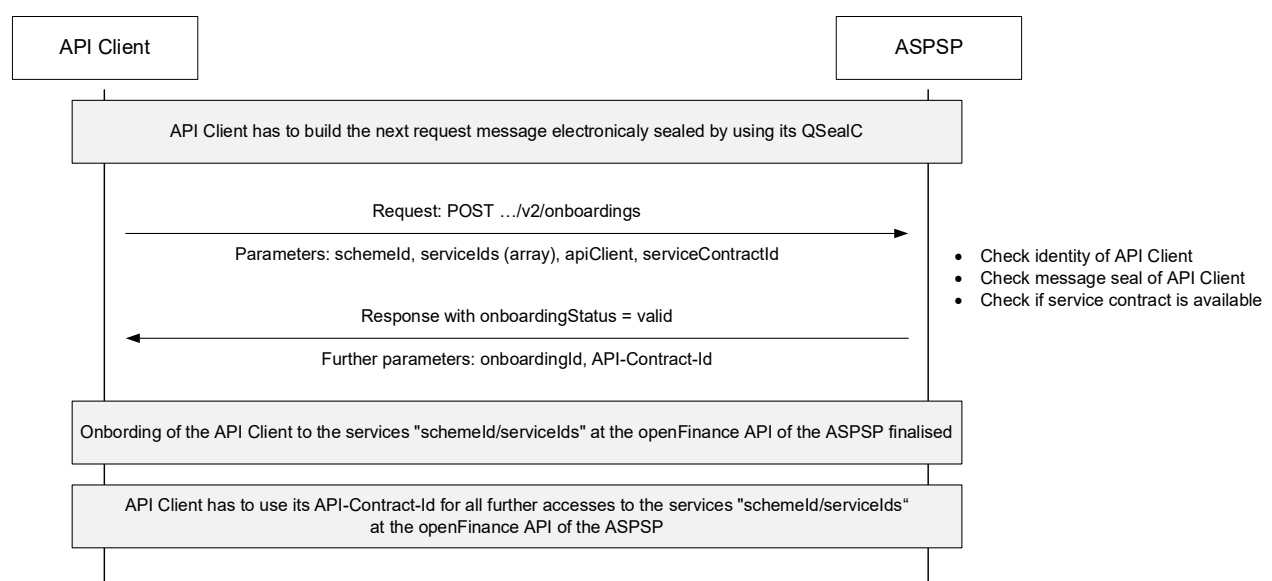
For the onboarding process of an API Client at the openFinance API of an ASPSP a "one step process" and a "two step process" can be distinguished and are described by this specification.

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.

The API Client is informed by the response body to the first POST request message if a one step process or a two step process has to be used.

The following figures will show a high-level overview of the processes for the two cases.

One step process without confirmation of (price) conditions at the API



For this use case the onboarding process consist only of one simple POST request sent by the API Client to the onboarding endpoint of the openFinance API of the ASPSP.

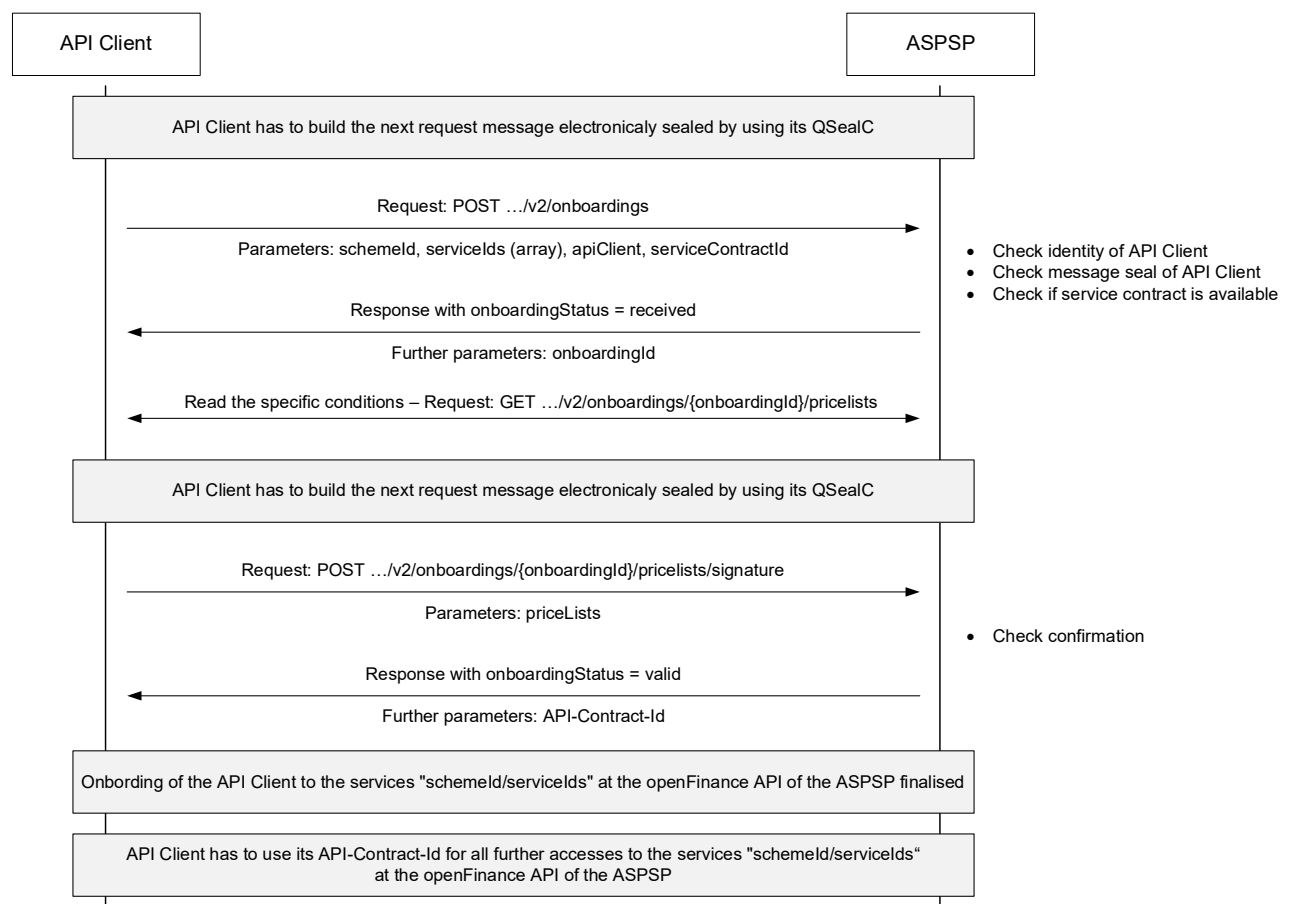
If the onboarding is successful an onboarding resource is created and the ASPSP will respond with status `valid` and the attribute `validTo` of the onboarding resource will be set depending on the validity of the service contract. In the response, the ASPSP will also provide an UUID as `API-Contract-Id`. The API Client shall use this UUID for further accesses to the corresponding services at the openFinance API of this ASPSP.

Remark:

If the service contract has got an unlimited validity the attribute `validTo` will be set to 9999-12-31.

If the onboarding is not successful no onboarding resource will be created and the ASPSP will respond with a HTTP Response Code according to [oFA-PFSM].

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



For this use case, the onboarding process consists of two POST requests sent by the API Client to the onboarding endpoint of the openFinance API of the ASPSP.

If the first POST is successful, an onboarding resource is created and the ASPSP will respond with resource status `received`. In addition, links indicating the next steps are returned by the ASPSP. The value of the attribute `validTo` of the onboarding resource is irrelevant at this time.

If the POST is not successful, no onboarding resource will be created and the ASPSP will respond with a HTTP Response Code according to [oFA-PFSM]. In this case this process is finished.

If the POST returns with status `received` the API Client can read the special price conditions (JSON data structure) from the onboarding resource created by the POST request. The API Client can check the conditions.

If the API Client agrees to the given price conditions, the API Client has to send the second POST request to confirm its acceptance to the ASPSP. This second POST request also contains a signature calculated for the HTTP request message as described by [oFA-PFSM].

If the second POST (confirmation step) is successful the ASPSP will respond with resource status `valid`. The attribute `validTo` of the onboarding resource will be set according to the validity of the price conditions confirmed by the second POST (see section 3.3). In this case the ASPSP will also provide an UUID as `API-Contract-Id`. The API Client shall use this UUID for further accesses to the corresponding services at the openFinance API of this ASPSP.

If the second POST (confirmation step) is not successful, the ASPSP will respond with the resource status `rejected`. This could be for example the case if the signature of the HTTP request message of the API Client or further content of the second POST message could not be verified successfully by the ASPSP. The API Client needs to start the onboarding process again if the API Client still wants to receive access to the ASPSP's APIs.

2.3 Overview: Status changes for an onboarding resource

If the POST command of an onboarding process is executed successfully, an onboarding resource is created.

The possible values for the status of an onboarding resource (see [oFA-DaD]) and the possible events triggering a transition of the status are different depending on the case of the onboarding process, i.e. with or without confirmation of special price conditions, and the business logic given by the contracts.

If the onboarding process follows the one-step procedure, the related onboarding resource will contain the attributes `validTo` (indicating the validity period of the corresponding onboarding) and `API-Contract-Id` (indicating the id to be used by the API Client to access the corresponding services at the openFinance API of the ASPSP).

If the onboarding process follows the two-step procedure, the first POST command will only lead to the creation of the onboarding resource. Its status is set to `received`. The resource does not contain a `validTo` attribute or the `API-Contract-Id`, yet. After the second POST command received for this onboarding resource at the openFinance API to confirm the price list (section 3.3) is successfully issued, the validity period will be included in the resource and is set by the `validTo` attribute and the `API-Contract-Id` is included.

Other transitions of the status and of the value of the `validTo` attribute can be triggered for example by

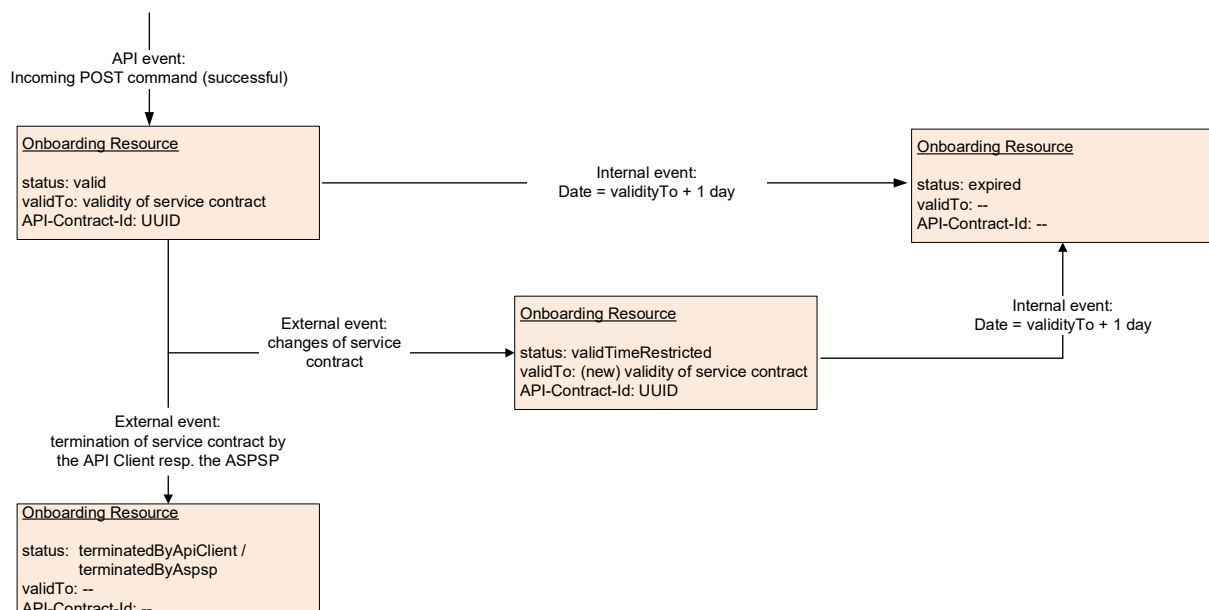
- a DELETE command received for this onboarding resource at the openFinance API (section 3.6),
- an external event to be reflected by the onboarding resource, i.e. termination of the corresponding service contract or replacement of the special price conditions by the ASPSP, or
- an internal event of the implementation, i.e. the termination of the resource due to the expiration of its validity.

Possible external events depend on the business logic which are defined by the ASPSP and API Client within the contracts. These external events are not out of scope for this specification. The same holds for the usage of the `validTo` parameter. The conditions for changing the value of this parameter depend on the business logic.

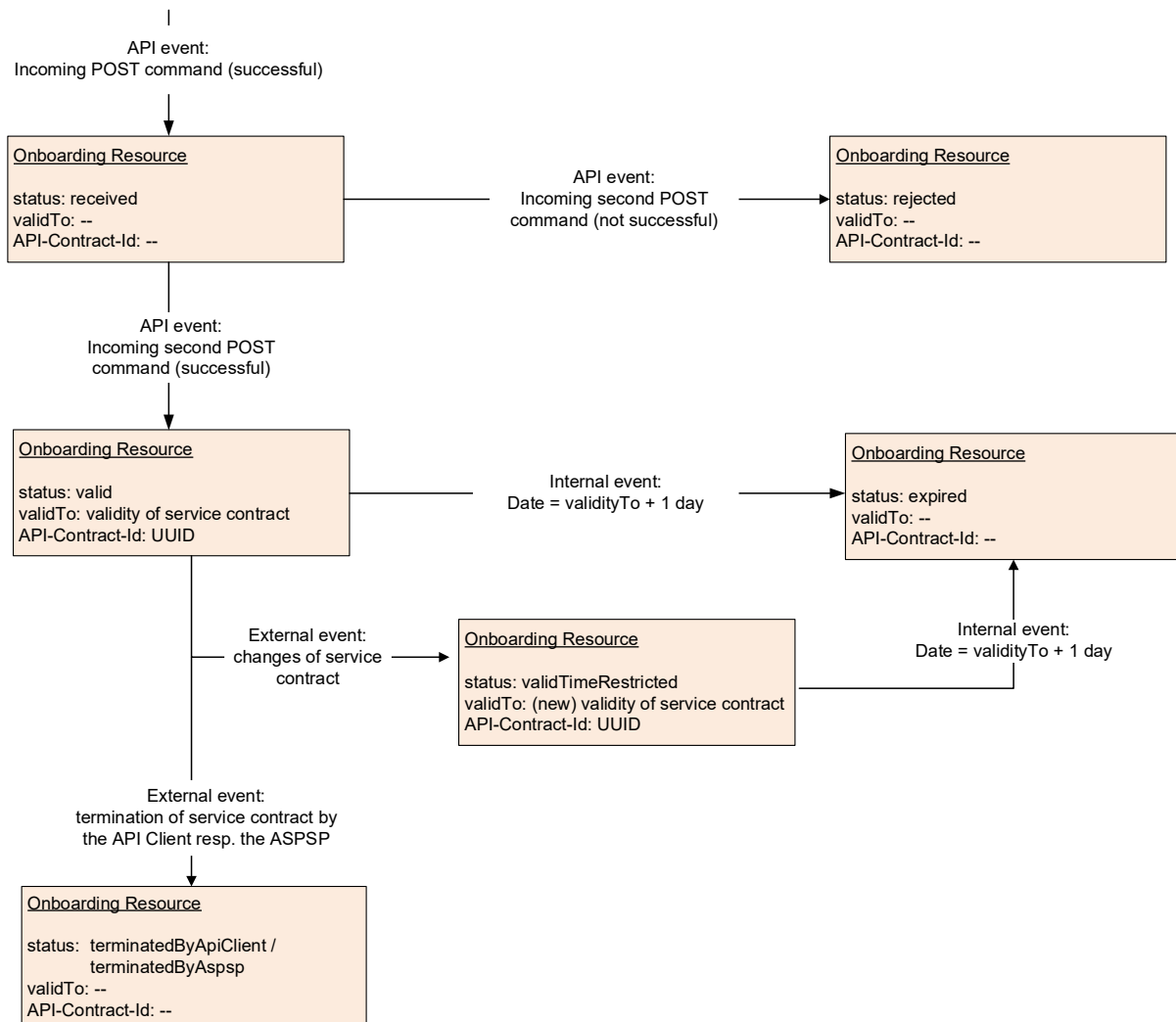
If a POST command is received (at the endpoint `onboardings/{onboardingId}/certificates`) for an onboarding resource at the openFinance API to inform about a new certificate (QSealC) of the API Client (see section 3.7), the status of the onboarding resource will not change. If this POST command is successful, the ASPSP will store the new certificate for the API Client. If this POST command is not successful (i.e. if the signature or the new certificate cannot be verified successfully), the ASPSP will reject this message. In both cases the status of the onboarding resource will not be changed.

The following figures give an overview of possible transitions of an onboarding resource:

One step process without confirmation of (price) conditions at the API



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



The onboarding resource in this two-step process with confirmation of price conditions obtains the (to be) confirmed price conditions in addition to all other onboarding resource data. This is part of the service contract.

Remark:

The validity of an onboarding resource starts always on the day its status is set to valid. It is not possible to execute the onboarding with a validity starting in the future.

Nevertheless, the response body to a POST request may contain an optional validFrom parameter for implementation reasons. This parameter indicates the date from which the API Client may access the corresponding services at the openFinance API of the ASPSP.

2.4 Overview: Structure of the price list

In this section, the attributes needed for the implementation of the pricing model are presented. These attributes must correspond to the price model that would apply to the potential API Client requesting the price list. If no bilateral contract between the API Client and the ASPSP exists, the price list contains the fallback price model of the ASPSP.

Modelling the API prices with price lists

Price lists allow to convey the prices for different services (given by their service type) in form of the different entries. The ASPSP might implement different prices depending on whether the related account is a privately used account or a corporate account. Each entry of the price list must always convey the information of the dedicated service type or list of 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.

For each service type, the price model can be defined by one or a combination of the following three approaches:

- Using a flat fee for the whole service type: any service call related to this service type is covered by this flat fee,
- Using a flat fee per service endpoint or parent endpoint: any service call addressed to a service endpoint or to an endpoint that obtains the given endpoint as parent endpoint (including the addressed parent endpoint itself) is covered by this flat fee,
- Having variable prices for each API call possibly depending on the entries of the issued requests.

The ASPSP may combine these approaches when structuring the price model, e.g. by having a fixed flat fee for the general API usage and having a variable partial price depending on the values of the parameters of the API Call.

The prices that apply for an API Client using the ASPSP's API are calculated at the end of a pre-determined operating period (e.g. at the end of each month) such that the API usage of the API Client during that period might affect the prices that apply for the API Client for this period. This allows the ASPSP to offer the API Client adapted conditions corresponding to the usage.

If the price of an API call depends on the parameters and/or values used in the specific call (i.e. following the last approach listed above), the price of the call can be given by the sum of one or several partial prices:

- a fixed price, or
- a percentual price depending on a specified parameter that is used as unit of measurement, or
- a price depending on both a fixed and a percentual part, and/or
- a base price for the API Call that is not dependent on any parameters/values used in the call.

The partial prices yield the total price of that specific API call for the API Client at the ASPSP's APIs. The sum of all these partial price points of the API call can be restricted by

- a minimum, and/or
- a maximum price.

These parameters can be included to yield an upper or lower limit for the pricing of a singular API Call.

Both fixed and percentual prices can be given by different amounts or percentages for different intervals. These intervals may not overlap and must relate to a base parameter that is defined or given by the call. This could for example be the initiated amount or the number of previous API calls relating to the same resource (e.g. for multiple payment initiation services). The unit of this base parameter might be used to define these intervals. If intervals are used to define different pricing rules for different ranges, then for any range not covered by the intervals, no price is defined.

JSON data structure of the price model

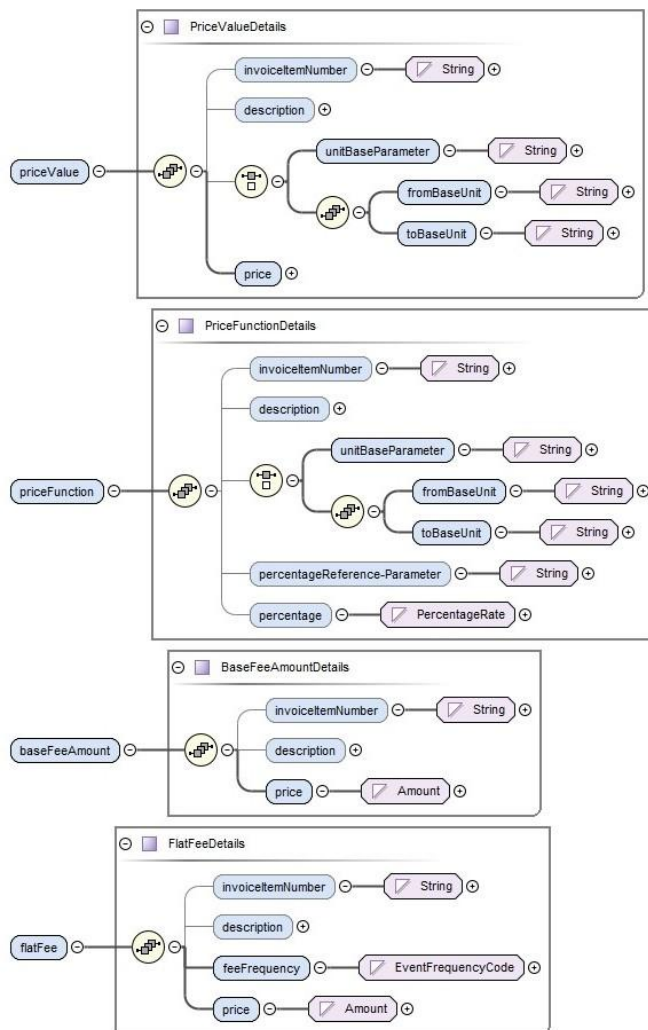
The following diagram gives a structural overview of the multilayered structure of the price API. Please note that the priceListDetails3 and the flat fee attribute on the Service Price List level are spread out and continued on the next page.

The following notation is used in the representation:

- Whether a parameter must be used or may be used optionally is indicated by the thickness of the connecting lines: If the line is bold, this implies a mandatory field, if a line is thin this field that may be included optionally.
- The cardinality of a field/layered structure respectively to the connected parent node is given by a small notion on top of a connecting line:
 - $0..∞$ implies the parameter may be used as many times (including zero) as needed,
 - $1..∞$ implies the parameter may be used as many times as needed and must be used at least once, or
 - if this is not given explicitly, it is implied that the parameter may/must be used precisely one time (depending on whether the line is bold or thin).
- the symbol  specifies that the following entry of the price list schema is a component possibly consisting of multiple attributes.
- the symbol  implies that at most/exactly one of the following connected attributes may be used (depending on whether the line going into this is bold or thin)

The overview shows the data structure of a service price list. All service price lists are bundled together in an overall price list that the API Client can retrieve during the onboarding process.





Clarification: There are two different categories of prices and price points that are covered in this price model:

- **Flat fee prices** that apply generally for a given time period and do not depend on any API calls. These prices are always listed in a "flatFee" parameter yielding a fixed price for the usage of the API/APIs. They are set for a fixed 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.
- **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 total price of an API Call might be given as the sum of partial price points. All related partial price points that might be considered for the total price of a specific API Call must be listed in the "invoiceItems" parameter. Additionally, the sum of all

partial price points that might apply to the API Call can be restricted by a `minimumFeeAmount` and a `maximumFeeAmount`. If used, these restrictions are applied after all partial price points given in the "invoiceltems" parameter that are applicable for a respective API Call are summed up.

Moreover, the price model that applies for a certain API Client might also depend on factors that are independent of the (singular) API call 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).

2.5 Best practises and examples for price models

The prices are given by a JSON following the structure outlined below:

```
"pricelists": [
  {
    "serviceType": "SERVICE_TYPE_01",
    "validFrom": "2025-01-01",
    "validTo": "2025-12-31",
    "accountUsage": "...",
    "flatFee": {
      "feeFrequency": "...",
      "price": {
        "amount": "n.mm",
        "currency": "EUR"
      }
    }
  },
  "subServiceEndpoints": [
    {
      "serviceEndpoint": "SERVICE_ENDPOINT_01_a",
      "flatFee": {
        "feeFrequency": "...",
        "price": {
          "amount": "n.mm",
          "currency": "EUR"
        }
      }
    }
  ],
  "accessMethods": [
    {
      "method": "POST",
      "responseCode": "xxx",
      "transactionAuthorised": true,
      "perUsage": false,
      "minimumFeeAmount": {
        "feeAmount": {
          ...
        }
      },
      "maximumFeeAmount": {
        "feeAmount": {
          ...
        }
      },
      "initiatedVolumePerPeriod": {
        ...
      },
      "initiatedAccessPerPeriod": {
        ...
      }
    }
  ]
}
```

```

    },
    "initiatedCallsPerPeriod": {
      ...
    },
    "deferredPricing": "...",
    "referencePeriod": "...",
    "invoiceItems": [
      {
        "baseParameter": "...",
        "baseUnit": "...",
        "priceValue": [
          ...
        ]
      },
      {
        "baseParameter": "...",
        "baseUnit": "...",
        "priceFunction": [
          ...
        ]
      },
      {
        "baseParameter": "...",
        "baseFeeAmount": {
          ...
        }
      }
    ],
    "method": "GET",
    ...
  }
],
{
  "serviceEndpoint": "SERVICE_ENDPOINT_01_b",
  ...
}
],
{
  "serviceType": "SERVICE_TYPE_02",
  ...
}
]

```

Within this structure the ASPSP may define partial price points that depend on multiple factors. The parameters conveying information about the prices are **highlighted** to allow easier orientation within the structure. The structure around these data elements describes the circumstances (e.g. the kind of API call, general circumstances of the API Client's API usage etc.) apply such that the given price point factors into the total sum of the price for the API usage.

In the following subsections, several examples on how certain (partial) price points should be integrated in the structure are described. The focus there lies on the intended usage of the highlighted parameters.

2.5.1 Establish a base price of an API call

Part of the price of an API call can be a base fee. This base fee applies to each call at the respective endpoint using the specified access method and is independent of any parameters or values used in the call.

In this case, the *baseFeeAmount* parameter within *invoiceItems* should be used to represent this price point and the *baseParameter* attribute should be set to „null“.

```
{
  "baseParameter": null,
  "baseFeeAmount": {
    "invoiceItemNumber": "Number_Z",
    "description": "base price for the API call",
    "price": {
      "currency": "EUR",
      "amount": "0.01"
    }
  }
}
```

2.5.2 Establish a fixed price point possibly related to the value of a parameter

A (partial) price of an API call can consist of a "price for the usage of a specific parameter". If that parameter is used in an API call, then the ASPSP may charge the client for its usage.

This is depicted by following diagram, the price point yields the same price for the parameter, the range of the price point does not change, thus is independent of the value of the parameter.



This price point should be conveyed by defining an object in the *invoiceItems* using the *priceValue*. As this is a fixed price point not depending on the value of the parameter, just on the usage, no intervals for certain values of the parameter need to be specified.

The following example uses the "remittanceInformationUnstructured" as parameter:

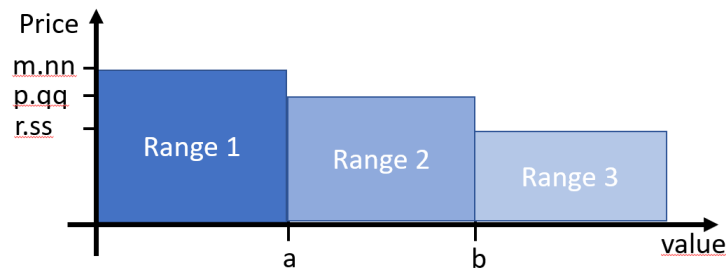
```
{
  "baseParameter": "remittanceInformationUnstructured",
  "description": "cost for the usage of unstructured remittance information in the API call",
  "priceValue": [
    {
      "invoiceItemNumber": "NUMBER_A",
      "price": {
        "currency": "EUR",
        "amount": "0.05"
      }
    }
  ]
}
```

```

    }
  }
  ...
]
}

```

The price point for the usage of the parameter might also depend on the value of the used parameter, also. This leads to different fixed prices for different values or value ranges of the *baseParameter*. The different prices are defined for fixed ranges of the *baseParameter* is shown in the following diagram:



Such a price point should also be depicted by an object in the *invoiceItems* using the *priceValue* structure. In contrast to the above, the *priceValue* obtains more than one object to present the different ranges for which the different prices apply. The *baseParameter* might be defined by discrete data (e.g. frequency codes) or by continues data (e.g. amount of a payment). For continues data the different ranges shall be defined using intervals given by the *fromBaseUnit* and *toBaseUnit* attributes. Discrete data shall rely on the *unitBaseParameter* which then includes all discrete values as a colon separated string for which the pricing applies.

The following example uses the instructed amount (e.g. of a payment initiation):

```

{
  "baseParameter": "instructedAmount",
  "baseUnit": "amount",
  "description": "the price of the API call depends on the instructed Amount of the payment",
  "priceValue": [
    {
      "invoiceItemNumber": "NUMBER_A",
      "fromBaseUnit": "0.00",
      "toBaseUnit": "999.99",
      "price": {
        "currency": "EUR",
        "amount": "0.05"
      }
    },
    {
      "invoiceItemNumber": "NUMBER_B",
      "fromBaseUnit": "1000.00",
      "toBaseUnit": "99999.99",
      "price": {
        "currency": "EUR",
        "amount": "0.025"
      }
    },
    {
      "invoiceItemNumber": "NUMBER_C",
      "fromBaseUnit": "100000.00",
      "price": {

```

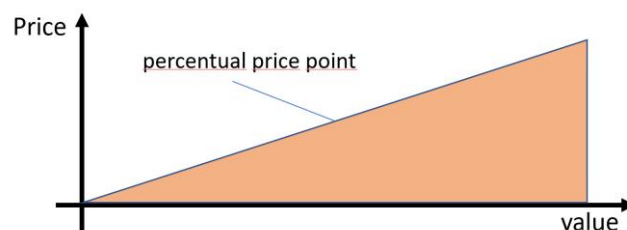
```

    "currency": "EUR",
    "amount": "0.01"
  }
]
}

```

2.5.3 Establish a relative price given by a percentage and depending on the value of a parameter used in the API call

A (partial) price of an API call can consist of a percentual price, defined as the fixed percentage of the value of an attribute which is included in the API call. In the price structure, this attribute is defined as *percentageReferenceParameter*. Such a percentual price is depicted by following diagram. The price point is defined as the partial amount (given by a fixed percentage value) of the attribute set to be the *percentageReferenceParameter*. In the diagram, only a single value is used for the percentage over the whole value range of the *percentageReferenceParameter*.



Such a percentual price point should be conveyed by defining an object in the *invoiceItems* using the *priceFunction*. As this is a fixed percentage value, it does not depend on the value of any parameter used in the call.

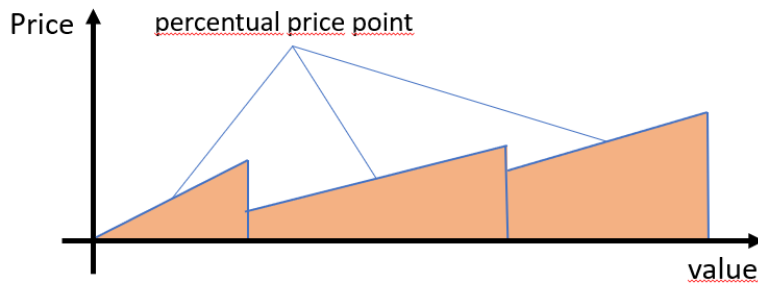
The following example uses the "instructedAmount" as *baseParameter* and *percentageReferenceParameter*. The value of the percentage is fixed in this example:

```

{
  "baseParameter": "instructedAmount",
  "baseUnit": "Amount",
  "description": "the price of an API call requesting the initiation is partially
defined relatively on the instructed amount of the API call",
  "priceFunction": [
    {
      "invoiceItemNumber": "XMRPIS_INITIATION_01_a",
      "percentageReferenceParameter": "instructedAmount",
      "percentage": "1.0"
    }
  ]
}

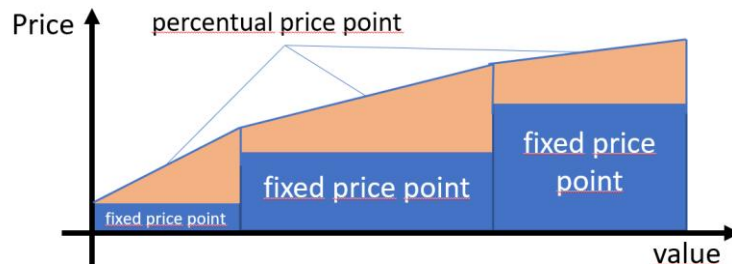
```

Similarly to the fixed price (see Section 2.5.2), the percentual price point might depend on the value of the *baseParameter*, leading to the usage of different percentual values for the calculation of the price point. The following diagram shows such scenario, where percentual values are used for different ranges of the *baseParameter*:



If only the percentual price is studied, the price point is not a continuous function but obtains "steps" whenever the percentage value changes.

It is possible to define the total price of an API call as a **combined price** via a percentual price point and a fixed price point using the same *baseParameter* and the same ranges, and defined in such a way that the fixed price point levels the percentual price so that the sum of both price points is a continuous function. The following diagram and the example below show how this could be defined in the price model:



The relative price point should also be depicted by an object in the *invoiceItems* using the *priceFunction* structure. In contrast to the above, the *priceFunction* obtains more than one object to present the different ranges for which the different prices apply. The *baseParameter* and the *percentageReferenceParameter* might be the same or different parameters given in the call. The following gives an example for pricing the creation of a multiple recurring payment, using the recurring frequency as *baseParameter* and the instructedAmount as *percentageReferenceParameter*. The price model for the fixed price points follows the same rules as given in the above section.

```
"invoiceItems": [
  {
    "baseParameter": "frequency",
    "baseUnit": "Frequency Code",
    "description": "relative price point for establishing a multiple recurring
payment depends on the frequency of the recurring period",
    "priceFunction": [
      {
        "invoiceItemNumber": "XMRPIS_CREATION_01_a",
        "unitBaseParameter": "Daily; Weekly",
        "percentageReferenceParameter": "instructedAmount",
        "percentage": "2.0"
      },
      {
        "invoiceItemNumber": "XMRPIS_CREATION_01_b",
        "unitBaseParameter": "EveryTwoWeeks; Monthly; EveryTwoMonths;
MonthlyVariable",
        "percentageReferenceParameter": "instructedAmount",
        "percentage": "1.0"
      }
    ]
  }
]
```

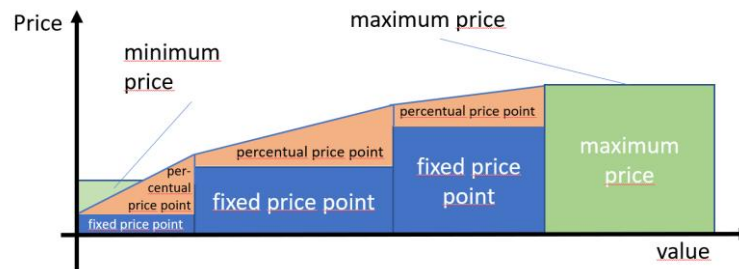
```

        {
            "invoiceItemNumber": "XMRPIS_CREATION_01_c",
            "unitBaseParameter": "Quarterly; SemiAnnual; Annual",
            "percentageReferenceParameter": "instructedAmount",
            "percentage": "0.5"
        }
    ],
},
{
    "baseParameter": "frequency",
    "baseUnit": "Frequency Code",
    "description": "fixed price point for establishing a multiple recurring payment
depends on the frequency of the recurring period",
    "priceValue": [
        {
            "invoiceItemNumber": "XMRPIS_CREATION_02_a",
            "unitBaseParameter": "Daily; Weekly",
            "price": {
                "currency": "EUR",
                "amount": "0.00"
            }
        },
        {
            "invoiceItemNumber": "XMRPIS_CREATION_02_b",
            "unitBaseParameter": "EveryTwoWeeks; Monthly; EveryTwoMonths;
MonthlyVariable",
            "price": {
                "currency": "EUR",
                "amount": "0.10"
            }
        },
        {
            "invoiceItemNumber": "XMRPIS_CREATION_02_c",
            "unitBaseParameter": "Quarterly; SemiAnnual; Annual",
            "price": {
                "currency": "EUR",
                "amount": "1.00"
            }
        }
    ]
}
]

```

2.5.4 Establish minimum and maximum prices for an API call

The ASPSP defines objects in the *invoiceItems* to yield a detailed price model for an API call that possibly relies on the values and parameters used in the call. The sum of all these partial price points given this way, yields the total price of the API call. As some partial prices are defined relatively, the ASPSP cannot foresee the range of the overall price of an API call. Therefore, it is possible for the ASPSP to define a minimum and/or maximum amount for the total price of the addressed API call by setting a *minimumFeeAmount* and *maximumFeeAmount* for the API service. These are not part of the objects listed in the *invoiceItems* but are given separately.



In case the sum of all partial price points given in the *invoiceItems* yields a total price below (resp. above) the price given in the *minimumFeeAmount* (resp. the *maximumFeeAmount*), this price specified in as the minimum (resp. maximum) fee applies for the addressed API call instead.

The following gives an Example how such a minimum and maximum price point might be defined. Additional to the fee amount, a description and invoice item number might be added.

```
"minimumFeeAmount": {
  "invoiceItemNumber": "SERVICE_PRICE_MIN",
  "description": "If the sum of all applicable price points in the invoiceItems is
lower than the price given in this parameter, the price of the API call shall be
set to this price",
  "feeAmount": {
    "currency": "EUR",
    "amount": "0.10"
  }
},
"maximumFeeAmount": {
  "invoiceItemNumber": "SERVICE_PRICE_MAX",
  "description": "If the sum of all applicable price points in the invoiceItems is
higher than the price given in this parameter, the price of the API call shall be
set to this price",
  "feeAmount": {
    "currency": "EUR",
    "amount": "5.00"
  }
}
```

2.5.5 Establish general flat fees for the API usage

Flat fees may be used to define a recurring price for the usage of the API. A flat fee may be defined for the overall service (defined by its service type) or in relation to a service or parent endpoint, such that all API calls addressed to the endpoint(s) are included in this flat fee. Flat fees are independent of any API call. They are applied when the API is used by the API Client in any way and flat fees are to be paid once during the period they are specified for. Due to their nature, flat fees are not included as part of the price points for an API call and may not be included in the sum of all partial price points given in the *invoiceItems*. Thus, flat fees do not affect the minimum and maximum fee amount described in Section 2.5.4.

It is the ASPSP's decision if its pricing model uses flat fees only, in addition to the price points given in the *invoiceItems* or not at all.

In contrast to other price points given at the API, a flat fee allows the API Client to address all the related API endpoint(s) in general. The *feeFrequency* yields the recurring time period for

which the fee applies and the *price* yields the total price for the fee. Additionally, a description and invoice item number can be added.

```
"flatFee": {
  "feeFrequency": "Monthly",
  "invoiceItemNumber": "SERVICE_FLAT",
  "description": "monthly flat fee for the overall service",
  "price": {
    "amount": "n.mm",
    "currency": "EUR"
  }
}
```

2.5.6 Establish different price models depending on the account usage

The overall pricelist entry specifying the service type and validity period for which the embedded price model applies may also specify for which kind of account usage the prices are defined. This way the ASPSP may establish different price models whether the addressed account is a personal private or a professional account. For this the attribute *accountUsage* is used.

If the ASPSP specifies the account usage in its price model, then a service type possibly has two price list models (for the same point in time), one for each account usage. If the account usage is not explicitly specified, the price model for the addressed service type applies to both types of accounts.

2.5.7 Establish different price models depending on the API Client's usage of the API

The ASPSP may set different fees and conditions for the same API Client depending on the usage of the API. This allows the ASPSP to adapt the prices to the API Client's usage for a certain pre-set timespan.

The attributes: *initiatedVolumePerPeriod*, *initiatedAccessPerPeriod*, *initiatedCallsPerPeriod* can be used to differentiate between different price models for the same API call depending on the amount (volume) that was initiated at the API, on the number of PSUs for which the API Client accessed the API or/and the number of Calls the API Client send to the API, all within a certain preset period. This period must be specified in the *referencePeriod* parameter if any of the attributes are used.

If the price model uses more than one of these attributes, then they must be related to the same time period, defined by the *referencePeriod* attribute. Moreover, if more than one of these attributes is used in the same entry, then the price entry only applies if all these conditions on the API Client's API usage hold true.

If the ASPSP imposes different price models depending on the API Client's API usage, then there are two options how this can apply. This is indicated by the Boolean parameter *deferredPricing*. If the API Client's API usage does imply the usage of a different price list entry (i.e. due to a certain initiated volume, number of access or/and API calls) at some point during the reference period, then:

- Option (A): all API calls that are issued within the reference period are priced after the billing period is over. The price model used for billing relies on the total API

usage of the API Client during the related period. If during the period, the API usage of the API Client implies a change of the price model to be used (i.e. because the number of API Calls exceeds a certain limit), then all API Calls are priced using the (new) related model, including API Calls that were already initiated before the API Client exceeded the limit of the previous price model.

Thus, the actual price of an API Call can only be specified after the related billing period is over. This must be indicated in the price list by setting the deferredPricing attribute to "true".

- Option (B): all the API Calls are priced directly using the price model that currently applies to the API Client within the billing period up to the point of issuing this API Call. This implies a different price model applies to the API Client in the future (within the same billing period), only the API Calls that are issued afterwards are priced using this (new) price model as it became the price model applicable to the current API usage of the API Client. All API Calls that were issued before the API Client's API usage exceeded the former price model, are still priced using the former price model.

Therefore, the price of an API Call can be calculated directly and the deferredPricing attribute must be set to "false" to indicate this option.

The ASPSP prices are given by the following price model for payment initiations with reservation of funds depending on the already initiated volume of the API Client during that period:

```
{
  "serviceType": "XDFIS",
  "serviceDescription": "payment with reservation of funds",
  "validFrom": "2025-01-01",
  "validTo": "2099-12-31",
  "subServiceEndpoints": [
    {
      "serviceEndpoint": ".../v2/deferred-payments/{payment-product}",
      "accessMethod": [
        {
          "method": "POST",
          "initiatedVolumePerPeriod": {
            "fromInitiatedVolume": "0.00",
            "toInitiatedVolume": "99999.99"
          },
          "deferredPricing": "... */ depending on the ASPSP's implementation",
          "referencePeriod": "Monthly",
          "invoiceItems": [
            {
              "baseParameter": null,
              "baseFeeAmount": {
                "description": "price per API call",
                "price": {
                  "amount": "0.05",
                  "currency": "EUR"
                }
              }
            }
          ],
          {
            "baseParameter": "instructedAmount",
            "baseUnit": "amount",
            "description": "price depending on the amount of this payment",
            "priceFunction": [
```

```

        {
            "percentageReferenceParameter": "instructedAmount",
            "percentage": "1.000"
        }
    ]
}
],
{
    "method": "POST",
    "initiatedVolumePerPeriod": {
        "fromInitiatedVolume": "100000.00",
        "toInitiatedVolume": null
    },
    "referencePeriod": "Monthly",
    "invoiceItems": [
        {
            "baseParameter": null,
            "baseFeeAmount": {
                "description": "price per API call",
                "price": {
                    "amount": "0.05",
                    "currency": "EUR"
                }
            }
        },
        {
            "baseParameter": "instructedAmount",
            "baseUnit": "amount",
            "description": "price depending on the amount of this payment",
            "priceFunction": [
                {
                    "percentageReferenceParameter": "instructedAmount",
                    "percentage": "0.500"
                }
            ]
        }
    ]
}
]
}
]
}
]
}
}

```

If the ASPSP prices the API Client using the option (A), then an API Client's request for initiating a payment with reservation of funds for 100.00€ via the API will cost

- 1.05€ if the total initiated amount of the API Client at the end of the month does not exceed 99999.99€, and
- 0.55€ if the API Client's total initiated amount at the API does exceed 99999.99€ at the end of the month.

If the ASPSP uses option (B) for the price model, then the API Client's request for initiating a payment with reservation of funds for 100.00€ via the API will cost

- 1.05€ if the amount that the API Client has already initiated during that month up until the moment of the API Call is less or equal than 99999.99€, and
- 0.55€ if the API Client has already initiated more than 100000.00€ during that month at the moment that the API Call was issued.

2.5.8 Establish prices for the usage of a service consisting of multiple API Calls

The ASPSP might set the price for a business service that consists of (possibly) multiple API calls issued by the API Client. Then the prices given in the price entry do yield the price of the overall service and not for the single API call.

The following yields an example for a price model specifying the price for the retrieval of cards and the respective card details of a related PSU. As this service could consist of multiple API calls – one receiving an overview of all cards and then one for each card to retrieve the details – the price model sums up the price for the service per service usage.

```
{
  "serviceType": "XAIS-CRD",
  "serviceDescription": "extended account information for cards",
  "validFrom": "2025-01-01",
  "validTo": "2099-12-31",
  "subServiceEndpoints": [
    {
      "serviceEndpoint": ".../v2/cards",
      "accessMethods": [
        {
          "method": "GET",
          "preUsage": true,
          "initiatedAccessPerPeriod": {
            "fromInitiatedAccess": "0",
            "toInitiatedAccess": "9999"
          },
          "invoiceItems": [
            {
              "baseParameter": null,
              "baseFeeAmount": {
                "description": "price per API service reading the list of cards
related to the PSU and the card details",
                "price": {
                  "amount": "0.80",
                  "currency": "EUR"
                }
              }
            }
          ]
        }
      ]
    },
    {
      "serviceEndpoint": ".../v2/cards/{cardId}",
      "accessMethods": [
        {
          "method": "GET",
          "preUsage": true,
          "invoiceItems": [
            {
              "baseParameter": null,
              "baseFeeAmount": {
                "description": "price per API service reading the list of cards
related to the PSU and the card details",
                "price": {
                  "amount": "0.00",
                  "currency": "EUR"
                }
              }
            }
          ]
        }
      ]
    }
  ]
}
```

```

    }
  ]
}

```

2.6 Endpoints and API Access Methods

The following table gives an overview on the HTTP access methods supported by the API endpoints for the Administrative Services. The conditions are relative and are provided for the case that the related parent endpoint exists.

Endpoints/Resources	Method	Condition	Description
onboardings	POST	Mandatory	Create an onboarding resource addressable under {onboardingId}. Note: All POST request messages sent to this endpoint shall be sealed by the API Client using its QSealC. See section 3.1.
onboardings/{onboardingId}/pricelists	GET	conditional	Read the data structure containing one or more price lists valid for using the corresponding extended services by the given API Client. Shall be supported if the two step approach is required by the business definitions. See section 3.2.
onboardings/{onboardingId}/pricelists/signature	POST	conditional	POST a signature to confirm the price lists contained in the data structure read before. Shall be supported if the two step approach is required by the business definitions. See section 3.3.
onboardings/{onboardingId}/certificates	POST	Optional	POST a signature (using the old certificate (QSealC)) to inform about a new certificate used by the API Client for http-message signing. See section 3.7.

Endpoints/Resources	Method	Condition	Description
onboardings/{onboardingId}/status	GET	Mandatory	Read the status of the onboarding resource. See section 3.4.
onboardings/{onboardingId}	GET	Mandatory	Read the onboarding resource See section 3.5.
onboardings/{onboardingId}	DELETE	Optional	Cancel the onboarding. See section 3.6.

Remark:

The support of the endpoint `onboardings/{onboardingId}/certificates` is optional for an ASPSP. If an ASPSP supports this endpoint the following requirements hold:

- The ASPSP shall also support to receive signed http messages where the JWS Protected Header does not contain the certificate in the element `x5c` but contains only a hash value of the certificate in the element `x5t#S256`. See section 6 of [oFA-PFSM].
- The ASPSP informs about the support of this endpoint using its Discovery API.
- The ASPSP can require that an API Client has to include the hash value of the certificate (using the element `x5t#S256`) instead of the certificate itself (using the element `x5c`) into the JWS Protected Header if the API Client sends a signed http message. In this case a signed http request message containing the certificate in the JWS protected header will be rejected by the ASPSP.
- The ASPSP will inform using its Discovery API if the ASPSP accepts for signed http messages, that the certificate is included in the JWS Protected Header (using the element `x5c`) or if the API Client has to include the hash value of the certificate using the element `x5t#S256` instead.
- The last two points do not hold for the POST request messages defined as part of the Administrative Services defined in this document. For these signed http request messages, the requirements as defined by section 2.1 hold.

3 Administrative Service

For all access methods defined in this document further applicable header parameters and error handling are defined in [oFA-PFSM]. In addition, also the authentication of the API Client is done based on a QWAC as described in the above document. Data structures are used as defined in [oFA-DaD].

3.1 Initiation of the onboarding process

Call

POST .../v2/onboardings

Creates the onboarding resource at the ASPSP.

Path Parameters

No Path Parameters

Query Parameters

No Query Parameter

Request Header

No special Request Header parameter

Request Body

Attribute	Type	Condition	Description
schemelds	Max70Text	Mandatory	Identification of the corresponding scheme, in case of no scheme involved use constant BILATERAL Example: giroAPI-payment The schemelds supported by the ASPSP should be provided in the Discovery API [oFA-IG-Disc].
servicelds	Array of Service Type	Mandatory	Identification of the extended services for which the onboarding will be done. Possible values to identify a service have to be defined by the service owner (scheme or ASPSP). The service types supported by the ASPSP should be provided in the Discovery API [oFA-IG-Disc].

Attribute	Type	Condition	Description
apiClientContacts	Array of Contact	Optional	Contacts (of different categories) of the API Client. Only to be given if not already available by e.g. a directory service.
serviceContractId	Max70Text	Mandatory	Identification of the service contract concluded between the service owner (scheme or ASPSP) and the API Client to use these services.

Response Code

The HTTP response code equals 201.

Response Header

No special Response Header parameter

Response Body

Attribute	Type	Condition	Description
onboardingId	Max70Text	Mandatory	Unique identification of the onboarding resource created by this initiation.
onboardingStatus	Onboarding Status	Mandatory	Status of the onboarding process.
API-Contract-Id	UUID	Conditional	Unique identification of the successful onboarding of the API Client for the list of services. Shall be used by the API Client if accessing the services after the onboarding. The attribute API-Contract-Id shall only be provided if the onboarding resource has been created and the resource status is set to <code>valid</code>. Otherwise this attribute shall be missing.
validTo	ISODate	Conditional	Date until when the successful onboarding is valid. The attribute validTo shall only be provided if the onboarding resource has been created and the resource status is set to <code>valid</code>. Otherwise this attribute shall be missing.

Attribute	Type	Condition	Description
			If the validity of the created resource is unlimited, the value 9999-12-31 shall be provided.
validFrom	ISODate	Optional	Indicates the date after which the API Client can access the services at the openFinance API of the ASPSP using the API-Contract-Id returned with this response.
_links	Links	Conditional	A list of hyperlinks to be recognised by the API Client. The actual hyperlinks used in the response depend on dynamic decisions of the ASPSP when processing the request. The attribute _links shall only be provided if the technical onboarding requires the API Client to read and confirm the price list, i.e. the onboarding follows the two step process according to section 2.2. Otherwise, this attribute shall be missing. Remark: All links can be relative or full links, to be decided by the ASPSP. Type of links admitted in this response, (further links might be added for extensions defined by the API Server): "readPricelists": the link to retrieve the price lists that need to be confirmed by the API Client before the onboarding of the API Client may be completed "confirmPricelists": the link to confirm the price list that was previously read by the API Client

Remark:

The parameters onboardingId and API-Contract-Id are two identifiers. The onboardingId is a technical resource identifier, which will always be provided for the first step if successful (and which is recommended to be a UUID). The API-Contract-Id is a UUID provided only after successful completion of the whole onboarding process, which might need more than one API interaction in the future.

Example**Request**

POST <https://api.testbank.com/openfinance/v2/onboardings>

```
Content-Type:          application/json
X-Request-ID:          99391c7e-ad88-49ec-a2ad-99ddcb1f7721
Date:                  Thu, 28 Dec 2023 15:02:37 GMT
Digest:                see section 6.2.2.1 of [oFA-PFSM]
x-jws-signature:       see section 6.2.2.2 of [oFA-PFSM]

{
  "schemeId": "giroAPI-Payment",
  "serviceIds": ["XFPIS", "XDFPIS", "XMDFPIS"],

  "serviceContractId": "giroAPI-Contract-ab10cd19ef58-20231229"
}
```

Response for a one step onboarding process

```
HTTP/1.x 201 Created
X-Request-ID:          99391c7e-ad88-49ec-a2ad-99ddcb1f7721
Date:                  Thu, 28 Dec 2023 15:02:42 GMT
Location:
    https://www.testbank.com/openfinance/v2/onboardings/3d9a81b3-a47d-
    4130-8765-a9c0ff861100
Content-Type:          application/json

{
  "onboardingId": "3d9a81b3-a47d-4130-8765-a9c0ff861100",
  "onboardingStatus": "valid",
  "API-Contract-Id": "24a990c2-e990-42f0-8cc4-56c822d4e2ec",
  "validTo": "2025-12-31",
  "validFrom": "2024-01-01"
}
```

Response for a two step onboarding process

```
HTTP/1.x 201 Created
X-Request-ID:          99391c7e-ad88-49ec-a2ad-99ddcb1f7721
Date:                  Thu, 28 Aug 2023 15:02:42 GMT
Location:
    https://www.testbank.com/openfinance/v2/onboardings/3d9a81b3-a47d-
    4130-8765-a9c0ff861100
Content-Type:          application/json

{
  "onboardingId": "3d9a81b3-a47d-4130-8765-a9c0ff861100",
  "onboardingStatus": "received",
  "_links": {
    "readPricelists":
      {"href": "/openfinance/v2/onboardings/3d9a81b3-a47d-4130-8765-
a9c0ff861100/pricelists"},
  }
```



```
"confirmPricelist":
  {"href": "/openfinance/v2/onboardings/3d9a81b3-a47d-4130-8765-
a9c0ff861100/pricelists/signature"}
}
```

3.2 Reading price lists

During the initiation of the onboarding process, the API Client is authenticated securely via its QWAC, thus the ASPSP may present the (potentially confidential) price list applicable to the API Client.

The API Client's confirmation of the presented price list is part of the two-step process of the technical onboarding of the API Client to the ASPSP's API and necessary for the onboarding process of the API Client. The onboarding shall only be successful if the terms and conditions of the price list are confirmed by the API Client.

Call

GET .../v2/onboardings/{onboardingId}/pricelists

Path Parameters

Attribute	Type	Description
onboardingId	UUID	Onboarding ID as returned with the response message to the initiation of the onboarding process according to section 3.1.

Query Parameters

No Query Parameter

Request Header

No special Request Header parameter

Request Body

No Request Body

Response Code

The HTTP response code equals 200.

Response Header

No special Response Header parameter

Response Body

Attribute	Type	Condition	Description
priceLists	Array of Service Price List	Mandatory	The API Client is presented the (potentially confidential) price list applicable exactly to the API Client for the API services. Price lists are given via a JSON data structure. There might be different price lists, specifying the account usage of accounts for that the price list applies to and the validity period of the given price list must be included as well. For each service type and account usage (if specified), if there exists more than one price list within this attribute their validity periods may not overlap. The ASPSP has to ensure that at most one service price list is presented in this attribute as currently valid to the API Client (per account usage, if specified).
nextInvoicing Date	ISODate Time	Mandatory	This attribute yields the next date when the invoice is issued to the API Client.
_links	Links	Mandatory	"confirmPricelists": the link to confirm the price list

Example

Request

GET <https://api.testbank.com/openfinance/v2/onboardings/3d9a81b3-a47d-4130-8765-a9c0ff861100/pricelists>

Content-Type: application/json
 X-Request-ID: 99391c7e-ad88-49ec-a2ad-99ddcb1f7725
 Date: Thu, 28 Dec 2023 15:03:15 GMT

3.3 Confirmation of price lists

Call

POST .../v2/onboardings/{onboardingId}/pricelists/signature

When the API Client accepts the price conditions that were presented in the previous API Call, the API Client must confirm the presented conditions to finish its onboarding process to the ASPSP's APIs.

If this is successful, the status of the onboarding resource changes and the onboarding has been completed, and the related onboarding resource becomes valid. The API Client receives the API-Contract-Id in the response of this API Call which must be used for all future accesses of the API Client to the API.

This POST command has to be secured by HTTP message signing. In case of signature failure an http Response Code 401 with a message code SIGNATURE_INVALID will be returned according to error handling defined in [oFA-PFSM]. See also [oFA-DaD] for related message codes.

Path Parameters

Attribute	Type	Description
onboardingId	UUID	Onboarding ID as returned with the response message to the initiation of the onboarding process according to section 3.1.

Query Parameters

No Query Parameter

Request Header

No special Request Header parameter

Request Body

To confirm the price list read according to section 3.2 the price list has to be return in the Request Body without any changes.

Attribute	Type	Condition	Description
priceLists	Array of Service Price List	Mandatory	Copy of the price lists read before without any changes.

Response Code

The HTTP response code equals 200.

Response Header

No special Response Header parameter

Response Body

Attribute	Type	Condition	Description
onboardingStatus	Onboarding Status	Mandatory	Status of the onboarding process.
API-Contract-Id	UUID	Mandatory	Unique identification of the API Client. It is provided after the onboarding of the API Client has been completed successfully. It shall be used by the API Client if accessing the services after the onboarding.
validTo	ISODate	Mandatory	Date until when the successful onboarding is valid.
validFrom	ISODate	Optional	Indicates the date after which the API Client can access the services at the openFinance API of the ASPSP using the API-Contract-Id returned with this response.

Example

Request

POST <https://api.testbank.com/openfinance/v2/onboardings/3d9a81b3-a47d-4130-8765-a9c0ff861100/pricelists/signature>

Content-Type: application/json
 X-Request-ID: 99391c7e-ad88-49ec-a2ad-99ddcb1f7727
 Date: Thu, 28 Dec 2023 15:33:05 GMT
 Digest: see section 6 of [oFA-PFSM]
 x-jws-signature: see section 6 of [oFA-PFSM]

```
{
  "priceLists": /* please see section 2.5 for multiple examples of the
price list,
}
```

Response

HTTP/1.x 200 ok

X-Request-ID: 99391c7e-ad88-49ec-a2ad-99ddcb1f7727

Date: Thu, 28 Dec 2023 15:33:35 GMT

Content-Type: application/json

```
{
  "onboardingStatus": "valid",
  "API-Contract-Id": "24a990c2-e990-42f0-8cc4-56c822d4e2ec",
  "validTo": "2025-12-31",
  "validFrom": "2024-01-01"
}
```

3.4 Reading the status of the onboarding

Call

GET .../v2/onboardings/{onboardingId}/status

Path Parameters

Attribute	Type	Description
onboardingId	UUID	Onboarding ID as returned with the response message to the initiation of the onboarding process according to section 3.1.

Query Parameters

No Query Parameter

Request Header

No special Request Header parameter

Request Body

No Request Body

Response Code

The HTTP response code equals 200.

Response Header

No special Response Header parameter

Response Body

Attribute	Type	Condition	Description
onboardingStatus	Onboarding Status	Mandatory	Current status of this onboarding resource.

Example

Request

GET <https://api.testbank.com/openfinance/v2/onboardings/3d9a81b3-a47d-4130-8765-a9c0ff861100/status>

Content-Type: application/json

X-Request-ID: 99391c7e-ad88-49ec-a2ad-99ddcb1f7729

Date: Fri, 29 Dec 2023 15:03:15 GMT

3.5 Reading the onboarding resource

Call

GET .../v2/onboardings/{onboardingId}

Path Parameters

Attribute	Type	Description
onboardingId	UUID	Onboarding ID as returned with the response message to the initiation of the onboarding process according to section 3.1.

Query Parameters

No Query Parameter

Request Header

No special Request Header parameter

Request Body

No Request Body

Response Code

The HTTP response code equals 200.

Response Header

No special Response Header parameter

Response Body

Attribute	Type	Condition	Description
onboardingStatus	Onboarding Status	Mandatory	Current status of this onboarding resource.
schemeld	Max70Text	Mandatory	Identification of the corresponding scheme, in case of no scheme involved use constant <code>BILATERAL</code> Example: giroAPI-payment
servicelds	Array of Service Type	Mandatory	Identification of the extended services for which the onboarding will be done. Possible values to identify a service have to be defined by the service owner (scheme or ASPSP).
serviceContractId	Max70Text	Mandatory	Identification of the service contract concluded between the service owner (scheme or ASPSP) and the API Client to use these services.
API-Contract-Id	UUID	Conditional	Unique identification of the successful onboarding of the API Client for the list of services contained in servicelds of the request body. Shall be used by the API Client if accessing the services after the onboarding. Has to be returned if the API-Contract-Id is set within the onboarding resource.
validTo	ISODate	Conditional	Date until when the successful onboarding is valid.

Attribute	Type	Condition	Description
			Has to be returned if the attribute validTo is set within the onboarding resource.
validFrom	ISODate	Conditional	Indicates the date after which the API Client can access the services at the openFinance API of the ASPSP using the API-Contract-Id. Has to be returned if the attribute validFrom is set within the onboarding resource.
priceLists	Array of Service Price List	Conditional	If the ASPSP requires a two-step onboarding process, this attribute must be included. The (potentially confidential) price list applicable exactly to the API Client for the API services. For all service types given in the serviceIds attribute above, the corresponding price list must be included or the ASPSP offers the related services to a certain service type for free.
nextInvoicing Date	ISODate Time	Conditional	Must be included if a price list is given. This attribute yields the next date when the invoice is issued to the API Client.

Example

Request

GET <https://api.testbank.com/openfinance/v2/onboardings/3d9a81b3-a47d-4130-8765-a9c0ff861100>

Content-Type: application/json
 X-Request-ID: 99391c7e-ad88-49ec-a2ad-99ddcb1f7731
 Date: Fri, 29 Dec 2023 15:03:15 GMT

3.6 Cancel the onboarding

Call

DELETE .../v2/onboardings/{onboardingId}

Path Parameters

Attribute	Type	Description
onboardingId	UUID	Onboarding ID as returned with the response message to the initiation of the onboarding process according to section 3.1.

Query Parameters

No Query Parameter

Request Header

No special Request Header parameter

Request Body

No Request Body

Response Code

The HTTP response code equals 204.

Response Header

No special Response Header parameter

Response Body

No Response Body.

Example

Request

DELETE <https://api.testbank.com/openfinance/v2/onboardings/3d9a81b3-a47d-4130-8765-a9c0ff861100>

Content-Type: application/json

X-Request-ID: 99391c7e-ad88-49ec-a2ad-99ddcb1f7732

Date: Fri, 29 Dec 2023 15:03:15 GMT

3.7 Information about a new QSealC of the API Client

Call

POST .../v2/onboardings/{onboardingId}/certificates

With this POST command the API Client informs the ASPSP that for future calls a new certificate (QSealC) will be used for message signing.

This POST command has to be secured by HTTP message signing using the current QSealC of the API Client known to ASPSP, see section 2.1. The new QSealC has to be included in the body of this request.

In case of signature failure an http Response Code 401 with a message code SIGNATURE_INVALID will be returned according to error handling defined in [oFA-PFSM]. See also [oFA-DaD] for related message codes.

The validity of the old certificate is not affected by this call. The old certificate stays valid as stated in the old certificate.

Path Parameters

Attribute	Type	Description
onboardingId	UUID	Onboarding ID as returned with the response message to the initiation of the onboarding process according to section 3.1.

Query Parameters

No Query Parameter

Request Header

No special Request Header parameter

Request Body

Attribute	Type	Condition	Description
newCertificate	String	Mandatory	New QSealC used by the API Client to sign the HTTP request message in the future. The certificate has to be included base64url encoded.

Response Code

The HTTP response code equals 200.

Response Header

No special Response Header parameter

Response Body

No Response Body

Example

Request

```
POST https://api.testbank.com/openfinance/v2/onboardings/3d9a81b3-a47d-4130-8765-a9c0ff861100/certificates
Content-Type:          application/json
X-Request-ID:          99391c7e-ad88-49ec-a2ad-99ddcb1f7727
Date:                  Thu, 28 Dec 2023 15:33:05 GMT
Digest:                of empty body, see section 6.2.2.1 of [oFA-PFSM]
x-jws-signature:       see section 6.2.2.2 of [oFA-PFSM]
```

Response

```
HTTP/1.x 200 ok
X-Request-ID:          99391c7e-ad88-49ec-a2ad-99ddcb1f7727
Date:                  Thu, 28 Dec 2023 15:33:35 GMT
Content-Type:          application/json
```

4 References

4.1 Informative References

[oFA-OR-Com] openFinance API Framework, Operational Rules, Compliance Services, Version 2.1.1, 31 July 2026

[oFA-IG-Com] openFinance API Framework, Implementation Guidelines, Compliance Services, Version 2.4.2, 31 July 2026

4.2 Normative References

[oFA-OR-XPIS] openFinance API Framework, Operational Rules, Extended Payment Initiation Services, Version 2.3, 31 July 2026

[oFA-PFSM] openFinance API Framework, Implementation Guidelines, Protocol Functions and Security Measures, Version 2.4.1, 31 July 2026

[oFA-DaD] openFinance API Framework, Data Dictionary, Version 2.4.1, 31 July 2026

[oFA-IG-Disc] openFinance API Framework, Implementation Guidelines, Discovery Services, Version 1.3, 31 July 2026

[oFA-OR-ADM] openFinance API Framework, Operational Rules, Administrative Services, Version 1.0, 31 July 2026

[EBA-RTS] Commission Delegated Regulation (EU) 2018/389 of 27 November 2017 supplementing Directive 2015/2366 of the European Parliament and of the Council with regard to Regulatory Technical Standards for Strong Customer Authentication and Common and Secure Open Standards of Communication, C(2017) 7782 final, published 13 March 2018

[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

[PSD2] Directive (EU) 2015/2366 of the European Parliament and of the Council on payment services in the internal market, published 23 December 2015



4.3 Detailed Change Log

4.3.1 Changes from Version 1.1 to Version 1.2

Section	Changes	Reason
whole document	renamed "apiContractId" to "API-Contract-ID" to adapt to framework changes	editorial
2.1	Updated the process of signing messages at the API Layer	Errata
2.3	Updated the section editorially for a better understanding and clarification purposes	Clarification
2.4 and 2.5	Added new Sections on the price list structure and possibilities, to explain the enhancement of the onboarding processes in detail	Inclusion of new service functions in the specification
3.1	renamed the "apiContractId" attribute to "API-Contract-ID" to be compliant with all other openFinance Specifications	Erratum
3.1	updated the description of the "_links" attribute	Inclusion of new service functions in the specification
3.2, 3.3	Update of the service to include the new service function	Inclusion of new service functions in the specification
3.5	Price lists and next invoicing date shall be included in the response, if the onboarding process required the confirmation of price lists	Inclusion of new service functions in the specification
3.6	HTTP response code changed to "204" instead of "200" to be compliant with all other openFinance Specifications	Erratum
3.7	Updated the process of signing messages at the API Layer	Errata
4.1, 4.2	Updated the references	Editorial