Digital Payments
for a Trusted World

WORLDLINE

Merchant Subscription Management – Implementation Guide

Status	Published
Author:	Worldline
Document date:	7 March 2023
Classification:	Confidential
Version:	1.0

Version history

Version no.	Version date	Status	Edited by	Most important edit(s)
1.0	07-03-2023	Published	Joris Majoor	Initial version

Table of contents

1 Overview

The **Initiating Party and Subscription Management** provides the maintenance of all general Initiating Party, Creditor data, Account data and Service Subscription data after setup.

Additionally technical and security data are provided for further processing. The Initiating Party & Subscription Management allows the **Bank Backoffice Admin** to setup, maintain and check **Initiating Party** data.
The **Bank** offers the available services to the **Initiating Party** and can also setup, maintain and check the service subscription data.
For **Bank Backoffice Admins** all features are also maintainable via **Webservices** (e.g. exchange or delete Initiating Party data).

2 Introduction to the Implementation Guide

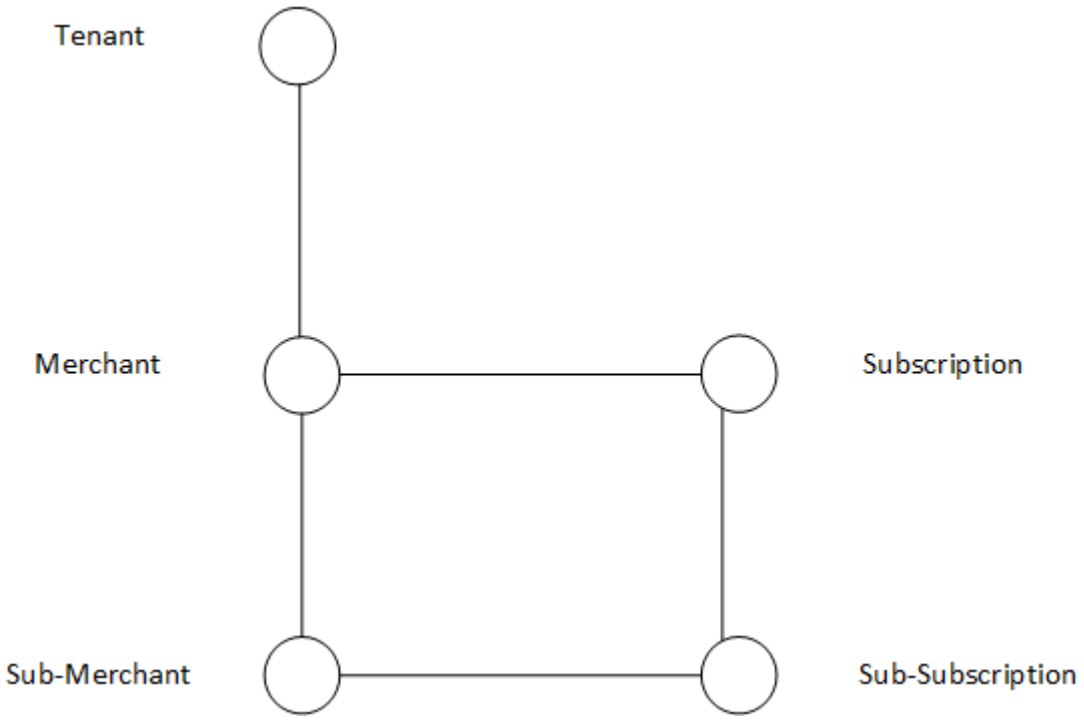
This document describes the Merchant Subscription Managements API's which can be used to setup and maintain Sub-Merchants and their Sub-Subscriptions on the Multi Service Platform.

2.1 Terminology

Term	Description
TPP Solution	The solution provided by Worldline to support Open Banking solutions for Third Party Providers
Initiating Party	The party who initiates API calls towards the TPP Solution
Tenant	Logical unit within the TPP Solution
Merchant	Organizational unit under a tenant
Sub-Merchant	Organizational unit under a merchant
Subscription	a subscription belonging to a Merchant
Sub-Subscription	a subscription belonging to a Sub-Merchant
MSP	The Multi Service Platform

2.2 Overview

The picture below gives an overview of the relations between the entities. It shows that a Sub-Subscription can only be connected to a Sub-Merchant and it can only exist if it's parent Merchant has the same Subscription.



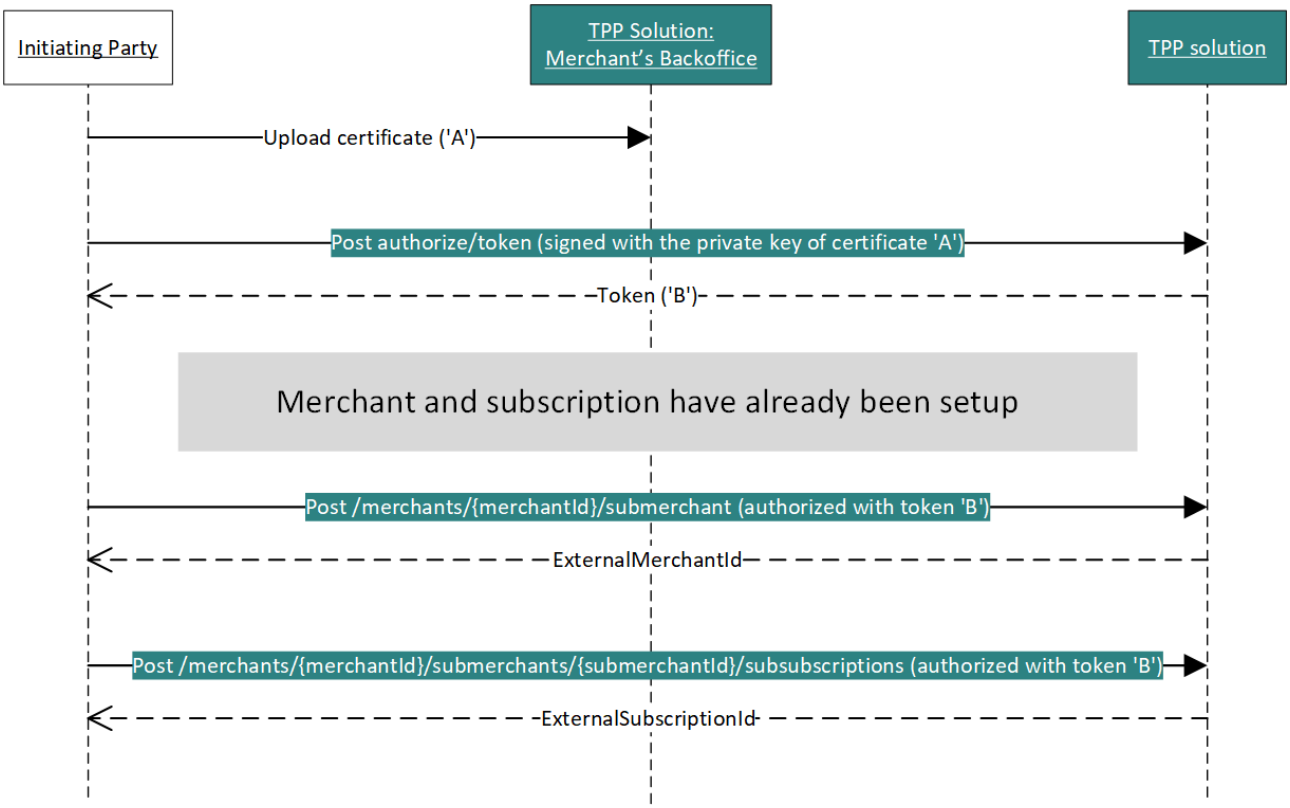
3 MSM - Sequence diagram

If you want to make use of one of the Backoffice REST APIs, e.g. to create Sub-Merchants and Sub-Subscriptions, a preceding step to authenticate and authorize the user performing these API requests is mandatory.

This preceding step is the request for an OAUTH2 access token based on a certificate ('A', in the diagram below), for which we offer a dedicated API.

More information about the security and the different API's can be found in the next chapters.

The sequence diagram provides an example of a possible flow. In this case the Merchant and Subscription have already been setup. The Initiating Party should have the 'merchantId', so it can start and create a Sub-Merchant below this merchant.



4 MSM - Security

This chapter describes how to:

- upload a Certificate to the GUI

And continues with the API security:

Authentication, authorization and message integrity of requests between the Initiating Party and the TPP solution is guaranteed through three different mechanisms.

- Transport Level Security
- Digital Signatures
- Access Tokens

4.1 Upload a Certificate to the GUI

The basic precondition for a successful use of the Authorization API is the upload of a dedicated certificate via the Backoffice GUI (Dashboard).

To do so, first log in to the Backoffice GUI (as MerchantAdmin) and navigate to main menu item "**Merchants**" and sub menu item "**Web Services**".

Here you can click the button "**New Certificate**" to start the dialog for uploading a new webservice certificate.

You see an upload button and, most important, a select box labeled with "**Owner**". Here you will see two options in the dropdown:

- Option 1: The first option in this select box is always the alias name of your own user
- Option 2: The second option is a technical virtual user, prefixed with "**TU:**", "**TU:MerchantAdmin**"

You selected Option 2 in the dropdown and clicked the button "**Confirm**".

During the use of the Authorization API, the value "**TU:MerchantAdmin**" has to be provided as the header "**Id**" (see also dedicated Chapter).

4.2 Transport Level Security

On the transport level all requests to the TPP solution, as well as all requests sent by the TPP solution MUST be encrypted using TLS and be made over HTTPS. TLS 1.3 SHOULD be used; TLS 1.2 MAY be used. Anything below TLS 1.2 MUST NOT be used and will be refused by the TPP solution. The TLS authentication method used is one-way, that means in requests to TPP the server authenticates itself with its certificate and in case of requests from TPP to the Initiating Party, the latter authenticates itself with its certificate.

Any connection without TLS encryption, such as plain http will be refused.

4.3 Digital Signatures

4.3.1 Signing the authorization request

On the application level, the Initiating Party is authenticated and authorized by sending a digitally signed request to the Authorization API endpoint (see Authorization API). The signature validation allows to check the authenticity and integrity of the request. This is achieved by applying the "Authorization" scheme as described in https://datatracker.ietf.org/doc/html/draft-cavage-http-signatures-12 and further detailed in Authorization API.

In order to generate the string that is signed with a key, the Initiating Party must use the values of each HTTP header field in the `headers` Signature Parameter, in the order they appear in the `headers` Signature Parameter. The `headers` Signature Parameter is fixed:

```
headers="app client id date"
```

The header field string is created by concatenating the lowercased header field name followed with an ASCII colon `:`, an ASCII space ` `, and the header field value. Leading and trailing optional whitespace (OWS) in the header field value MUST be omitted. If the value is not the last value, then an ASCII newline `\n` is appended.

Example: For the request

```
POST /authorize/token HTTP/1.1
App: IDEAL
Date: Fri, 25 Mar 2022 20:51:35 GMT
Client : idealClient
Id : TU:MerchantAdmin
```

The concatenated String to sign would be:

```
app: IDEAL
client: idealClient
id: TU:MerchantAdmin
date: Fri, 25 Mar 2022 20:51:35 GMT
```

The signature algorithm is fixed:

```
algorithm="SHA256withRSA"
```

The ‘keyId’ Signature Parameter is the thumbprint of the used certificate, viewed with the SHA1 algorithm. The private key associated with `keyId` is used to generate a digital signature on the concatenated signature string applying the SHA256withRSA algorithm. The complete Authorization header looks then like this:

```
Authorization: Signature keyId=”DCAC7209573D506FC56095B8B23E8555A8F38B29”, algorithm=”SHA256withRSA”,
headers=”app client id date”, signature=”
guoLSHgl/zGRujqkDnmaWCL8kgCVnDazqkKu7nWU/uAHrS+M9eQsI8ueB4uWgxyPOnZps3vpNgkW1f4aBsdFYLS0jYeup4yhCMN6vis2z
fMKxUhZFkjELslQkit9Gwc9pqvcyH0IxUnDLbCQwkiYjf6nGbP1YNfoxVXQpfq6i6CbIXCotLfwH2kbkrnSWwAS5skZY77+znmLDjtP3e
t2K94C36yPo0EEGqGkQ5xkD7owA7YxzA30xzsvkDvU3hzDzTK5wZmsgVsoyjRvMrokG0HrszUpNTwUtxflukcgs0pH7GuT+JrIpQ55f1d
pzULqxeBggnCvD9DRSuKeTakqlw==”
```

The Initiating Party must upload the used public certificate in the Backoffice GUI so that TPP is able to validate the signature. If the signature could be validated and the sender has a valid subscription a response containing an access token is returned. This access token can be used in all follow-up requests until it is expired. After expiration a new access token must be requested from the Authorization API endpoint.

4.3.2 Signing requests and responses

Signing requests and responses could be enforced depending on the TPP configuration. The following options exist:

- Requests to the Payment Initiation, Payment Status and Debtor Preference APIs, sent from Initiating Party to the TPP Solution
- Requests to the Notification API, sent from the TPP solution to the Initiating Party
- Responses sent back from the TPP Solution to the Initiating Party

The digital signing is done by applying the “Signature” scheme as described in https://datatracker.ietf.org/doc/html/draft-cavage-http-signatures-12. This is equivalent to the “Authorization” scheme and the same procedure is followed to generate the signature and header parts but it uses the Signature header instead of the Authorization header.

A notification request or response from TPP Solution to the Initiating Party may contain for example a header like:

```
Signature: keyId="2DOXXL7lNBNKJSMHKO2IBQC1", algorithm="SHA256withRSA", headers="digest x-request-id
messagecreatedatetime",
signature="lulVhOhRwFs5G8+uGCh3BjJHBG540AyTCyaKhr9QE71YTtljxFt1b7i55C9QROw/zGt+iac3cBfGGRiTAfW4ta9Hn8+u7L
vyfYHFcdxBhNj7T6dgrv+MLG6aI6hw/3Cwmkz/OwESBrQJzISWf8/0bgYNuXnuPf5r7BGMhHdhIr+RNxocW6wkSOEVQfOGYazy7YsoJhV
EwNEt9gN++sw9HVfaxmwjl8MqxGNcLoVAgoGMcUOvNhjATA1MSzOj2cmw6kdi2yY2w7XiMuU9Tma0jQ3CGKhxIkD8Na2Vq1K2bs/n2DOx
xL7lNbnKjsmhkO2ibQc1+RV3pXQJ1SDSI3EW/w=="
```

The Initiating Party may then validate that the content of the sent message is correct and has not been altered during transmission or storage. For this he will use the public certificate of the TPP Solution as can be downloaded from the Backoffice GUI.

4.4 Access Token

The access token as has been returned from the Authorization request endpoint can be used for subsequent requests. The default validity time of the access token is one hour after which a new one must be requested from the Authorization endpoint. A refresh token does not exist for this type of request.

The access token must be put into the request Authorization header like for example:

```
Authorization:Bearer 4944daeae6c9115a10dafecbfad4a9c
```

With the access token the TPP Solution can validate and authorize the request.

5 Retrieve token

Endpoint: POST /authorize/token

This API retrieves the token which is used in the communication between the Initiating Party and the TPP solution. The Initiating Party is using his private key to sign the request. In the response he will receive an access token from the TPP solution. This token is used in all subsequent API calls towards the TPP solution.

Request

Location	Name	Comments	Type
Query Param	**grant_type** *required*	To be set to 'client_credentials'	String

Header	**Authorization** *required*	The signature. It contains the header attributes 'app', 'client', 'id' and 'date' signed with the private key of the client.
		The signature will be used to sign the authorization request with the private key which corresponds to the certificate provided for the onboarding.
		Structure
		`Signature keyId="<thumbprint of certificate>", algorithm="SHA256withRSA", headers="app client id date", signature="<signature>"`
		Example
		`Signature keyId="58AF4EC5ADD4C4A3F28D3AEFF60656B2F2xxxxxx", algorithm="SHA256withRSA", headers="app client id date", signature="Abczym2rZF…r5qcvgmA=="`
		Generating rules
		The signature must be created over a String where app, client, id and date are concatenated with the following rules:
		• The keyId is the thumbprint of the certificate, viewed with the SHA1 algorithm.
		• Create the header field string by concatenating the lowercased header field name followed with an ASCII colon `:`, an ASCII space ` `, and the header field value. Leading and trailing optional whitespace (OWS) in the header field value MUST be omitted (as specified in RFC7230 [RFC7230], Section 3.2.4 [7]). If value is not the last value then append an ASCII newline `\n`.
		More details can be found here: https://datatracker.ietf.org/doc/draft-ietf-httpbis-message-signatures/
			String

Header	**App** *required*	The name of the service. For Merchant subscription management the value "**MSM**" should be used.	String
Header	**Client** *required*	The name of the tenant	String
Header	**Id** *required*	**TU:MerchantAdmin**	String
Header	**Date** *required*	Should be filled with the current date. The following date formats are supported: 1. EEE MMM dd HH:mm:ss zzz yyyy 2. ISO DATE: for example 2011-12-03T10:15:30+01:00 3. RFC 1123: for example Tue, 3 Jun 2008 11:05:30 GMT	Date

Response

Location	Name	Comments	Type
[1..1]	access_token	Token to be used in further API calls	String
[1..1]	token_type	Type of the token: Bearer	String
[1..1]	expires_in	Expiration time in seconds	Integer

5.1 Example: Authorization

Request

```
    Address:
https://localhost:8443/xs2a/routingservice/services/authorize/token?grant_type=client_credentials
    HttpMethod: POST

    Headers: {App=MSM, Accept=application/json, Date=2022-03-25T09:41:31.256Z, Authorization=Signature
KeyID="8D0F688AD3E6C2D4D5FB99FE129F2A2E3B496AF7", algorithm="SHA256withRSA", headers="app client id
date", signature=
"kAIepMoo6CRTWz9CLUFcpZj8eNQTdjXq6V8+kdk/9M1GmVud2CVrP1NMNTEiXgKzBlFQQlhv1iaFhMVOLVq7u8aEV4eeoNxjTLDK+lk4
zkjCBjeOyXtr32dtfjsvytlzhXw7KJizgOGd+m4Gh9xtSjY0I5QM/p+znKZsJCVKNSUUBZndAxIudsxy2Srp/yzexmvWpsoAvWIZzwtDS
03h4PjGTGKloXz6KyC+/I+GSBjw9M3GATUMMVrrgTKoR8oI0Xcr9v7ZTr3KpT1d1/LrcxQ82o2kq0+4ECVoJdVRezr2oZRmZ5hTHTIHeh
MNkASnuDqzDaQxQvMInUTg8tFKGA==", Id=TU:MerchantAdmin, Client=Worldline, }
```

Response

```
Content-Type: application/json;charset=UTF-8
    ResponseCode: 200

    Headers: {X-Request-ID=23eeed2d-f163-43c5-94b1-eeadcbb393e3, MessageCreateDateTime=2022-03-
25T09:41:31.819Z, Date=Fri, 25 Mar 2022 09:41:31 GMT, Content-Type=application/json;charset=UTF-8}
    Payload: {
    "access_token": "abb5468b4845dffff9cccd7c950e529",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

6 MSM - Sub-Merchant API

6.1 Merchant status

Merchant and sub-merchant status:

- Onboarding does not exist for sub-merchants)
- Applied (does not exist for sub-merchants)
- Active (available while creation)
- Cancelled (available while creation)
- Blocked (available while creation)

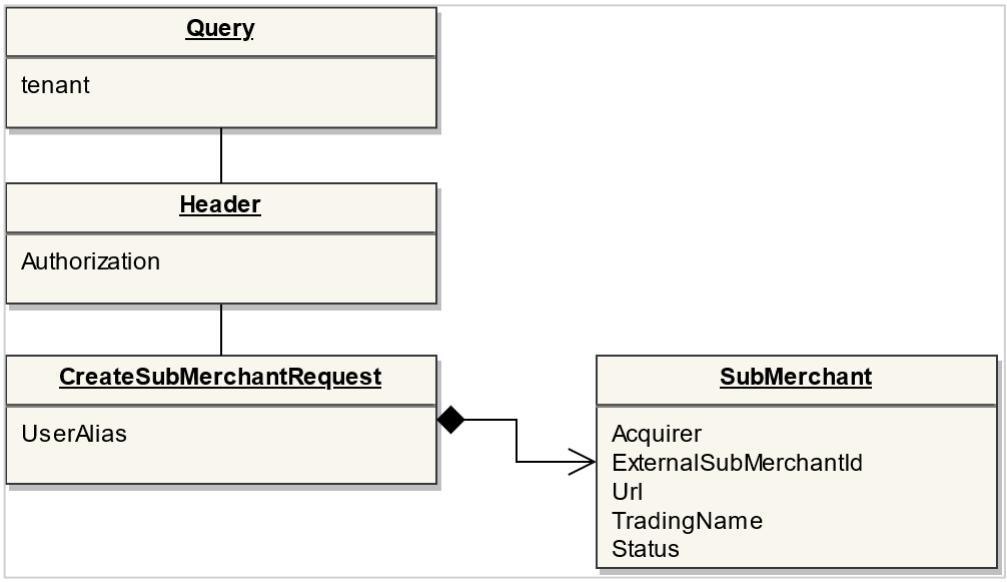
When the Sub-Merchant is created it will inherit the status of the Merchant.

6.2 Sub-Merchant Endpoints

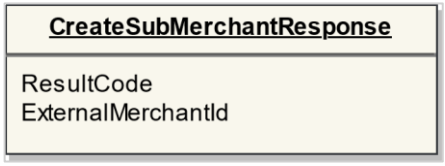
6.2.1 POST Sub-merchant

Endpoint: Post /merchants/{merchantId}/submerchants

6.2.1.1 request



response



6.2.1.2 Example: post sub merchant

```
curl --location --request POST 'https://digitalroutingservice-
integration.awltest.de/msp/services/merchant-service/v2/merchants/000522/submerchants' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer 46de4ec2442b8ebc6f0429e8e269d386add9071a481000de7b8806ec2d8ebea4' \
--header 'Cookie:
TS01fa099a=016a988a239ea2bc1b0bcf3be3cd20f1e80044855a7cec66ff63743afc4d8312f6d0d8306220cb6f8e441a7696a78b
17922da0ea09;
msp_persistence_cookie=!01XKCNjBAwqAvSOXUbhpI2X8Q+dbBLaZMKvyenoQu9ZjmhJVPRLTmAjN3XN0X3Q3oIijSlWcEn2MrzuKe
FPwdWLB8mozlc5S69tqvbpUVSA=' \
--data-raw '{
    "SubMerchant": {
        "ExternalSubMerchantId": "2",
        "Url": "https://sub1.worldline.com",
        "TradingName": "Trading-name-6123-2",
        "Status": "ACTIVE"
    }
}'
```

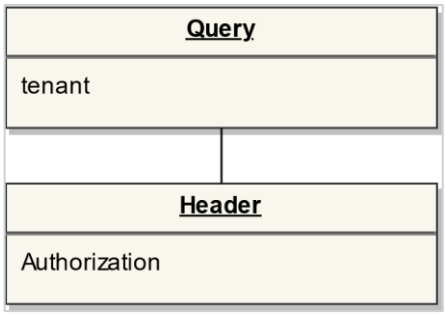
Code Block 1 post submerchant

6.2.2 Get Sub-merchant List

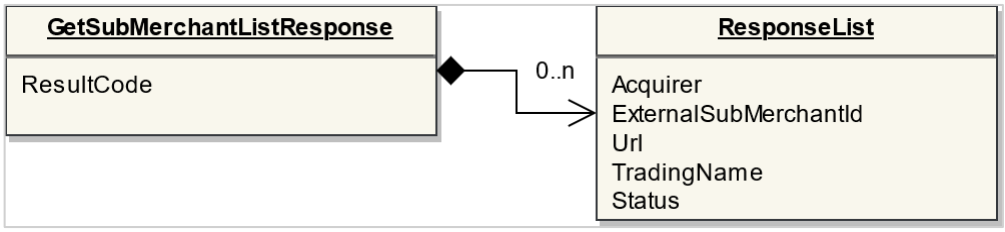
Endpoint: Get /merchants/{merchantId}/submerchants

Get a list of sub-merchants for a specific parent merchant.

request



6.2.2.1 response

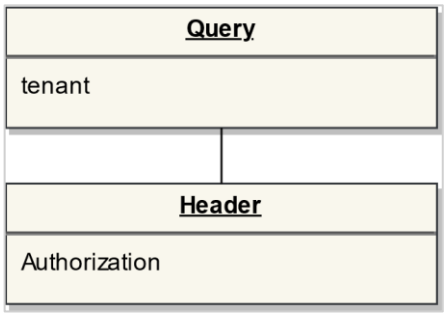


6.2.3 Get Sub-merchant

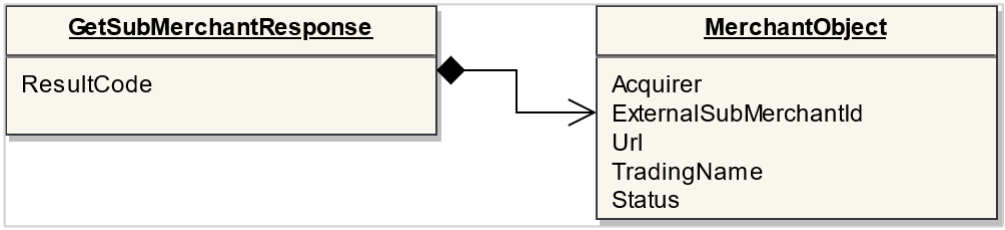
Endpoint: Get /merchants/{merchantId}/submerchants/{submerchantId}

Get a specific sub merchant

6.2.3.1 request



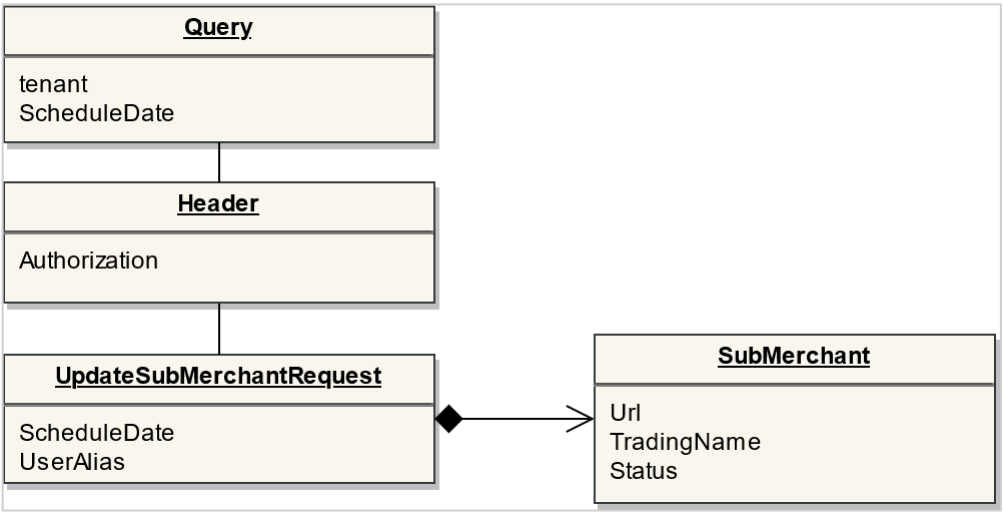
6.2.3.2 response



6.2.4 Put Sub-merchant

Endpoint: Put /merchants/{merchantId}/submerchants/{submerchantId}

6.2.4.1 request



6.2.4.2 response

7 MSM - Sub-Subscription API

7.1 Sub-Subscription status

- Active
- Blocked
- Onboarding
- Wait_for_start
- Validated
- Cancelled

When creating a Sub-Subscription it will inherit the status of the parent Subscription.

If the parent Subscription is set to the status Blocked or Cancelled it's Sub-Subscription will also be Blocked or Cancelled.

7.2 Sub-Subscription Endpoints

7.2.1 POST Sub-Subscriptions

Endpoint: Post /merchants/{merchantId}/submerchants/{submerchantId}/subsubscriptions

Creates a sub-subscription for a sub merchant. A sub-subscription can only be created when the merchant has the corresponding subscription.

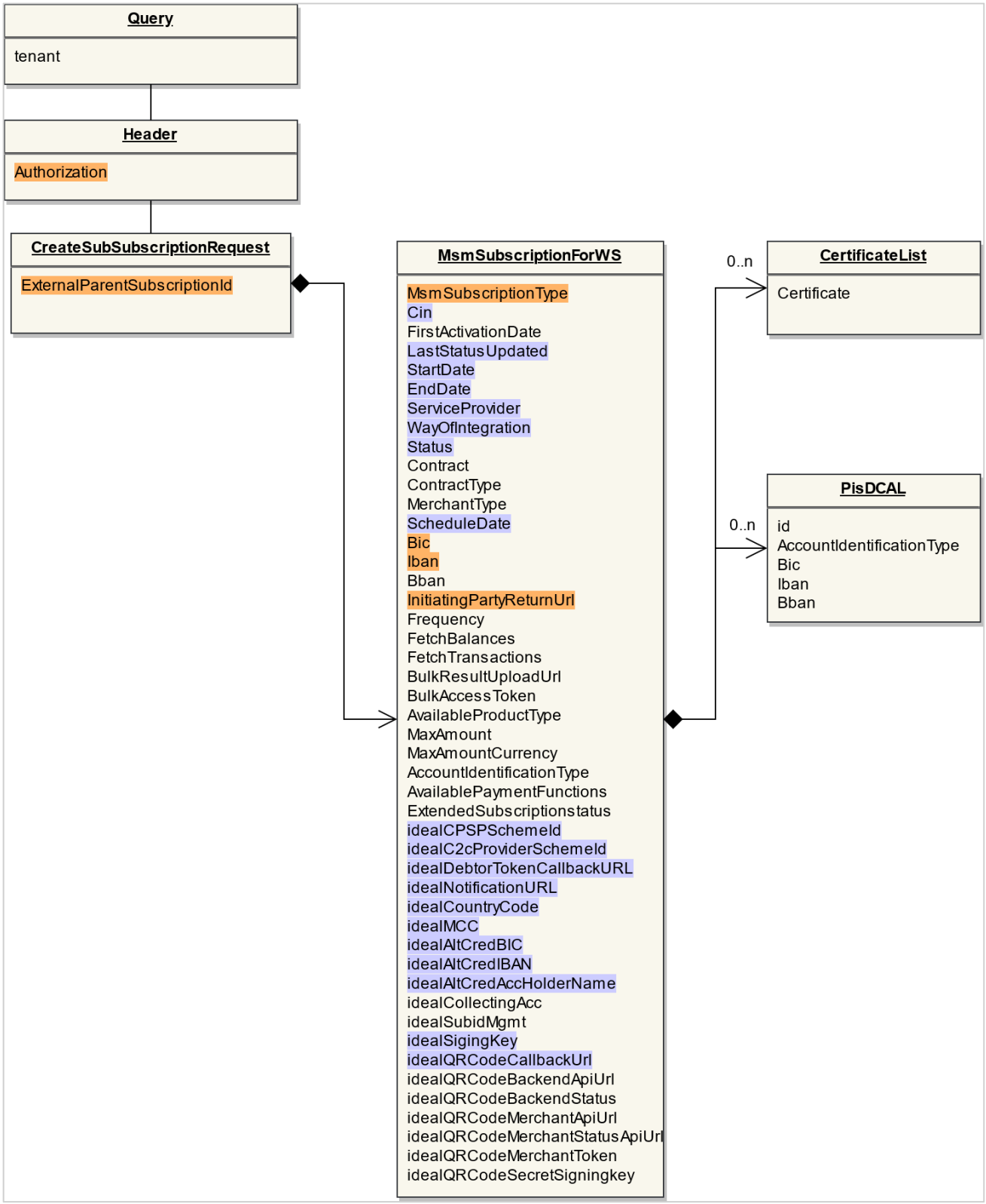
The following table displays which fields are relevant for the MsmSubscriptionType = IDEAL.

- M = a mandatory field (orange color in the diagram below)
- O = an optional field (purple in the diagram below)

The fields not mentioned in the table should not be used while creating an iDEAL Sub-Subscription.

Fieldname	iDEAL	Comment
ExternalParentSubscriptionId	M	
MsmSubscriptionType	M	
Cin	O	
LastStatusUpdated	O	
StartDate	O	
EndDate	O	
ServiceProvider	O	
WayOfIntegration	O	
Status	O	
ScheduleDate	O	
Bic	M	
Iban	M	
InitiatingPartyReturnUrl	M	
idealCPSPSchemeId	O	
idealC2cProviderSchemeId	O	
idealDebtorTokenCallbackURL	O	
idealNotificationURL	O	
idealCountryCode	O	
idealMCC	O	
idealAltCredBIC	O	currently only there in iDEAL 1.0 context.
idealAltCredIBAN	O	currently only there in iDEAL 1.0 context.
idealAltCredAccHolderName	O	currently only there in iDEAL 1.0 context.
idealSigingKey	O	
idealQRCodeCallbackUrl	O	

7.2.1.1 Request



7.2.1.2 response

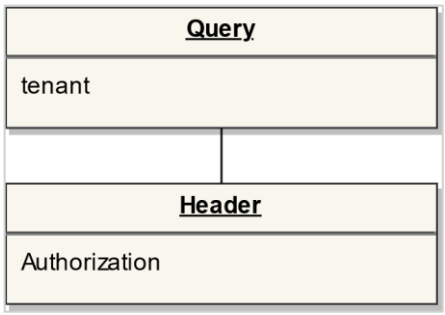
7.2.1.3 Example:

```
POST 'https://digitalroutingservice-integration.awltest.de/msp/services/subscription-
service/v2/merchants/000522/submerchants/1/subsubscriptions' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer 46de4ec2442b8ebc6f0429e8e269d386add9071a481000de7b8806ec2d8ebea4' \
--header 'Cookie:
TS01fa099a=016a988a239ea2bc1b0bcf3be3cd20f1e80044855a7cec66ff63743afc4d8312f6d0d8306220cb6f8e441a7696a78b
17922da0ea09;
msp_persistence_cookie=!01XKCNjBAwqAvSOXUbhpI2X8Q+dbBLaZMKvyenoQu9ZjmhJVPRLTmAjN3XN0X3Q3oIijSlWcEn2MrzuKe
FPwdWLB8mozlc5S69tqvbpUVSA=' \
--data-raw '{
    "MsmSubscriptionForWS": {
        "MsmSubscriptionType": "IDEAL",
        "Bic": "VOLKNL2U",
        "Iban": "NL19VOLK5460924692",
        "InitiatingPartyReturnUrl": "https://www.return.url.com"
    },
    "ExternalParentSubscriptionId": "002000462"
}'
```

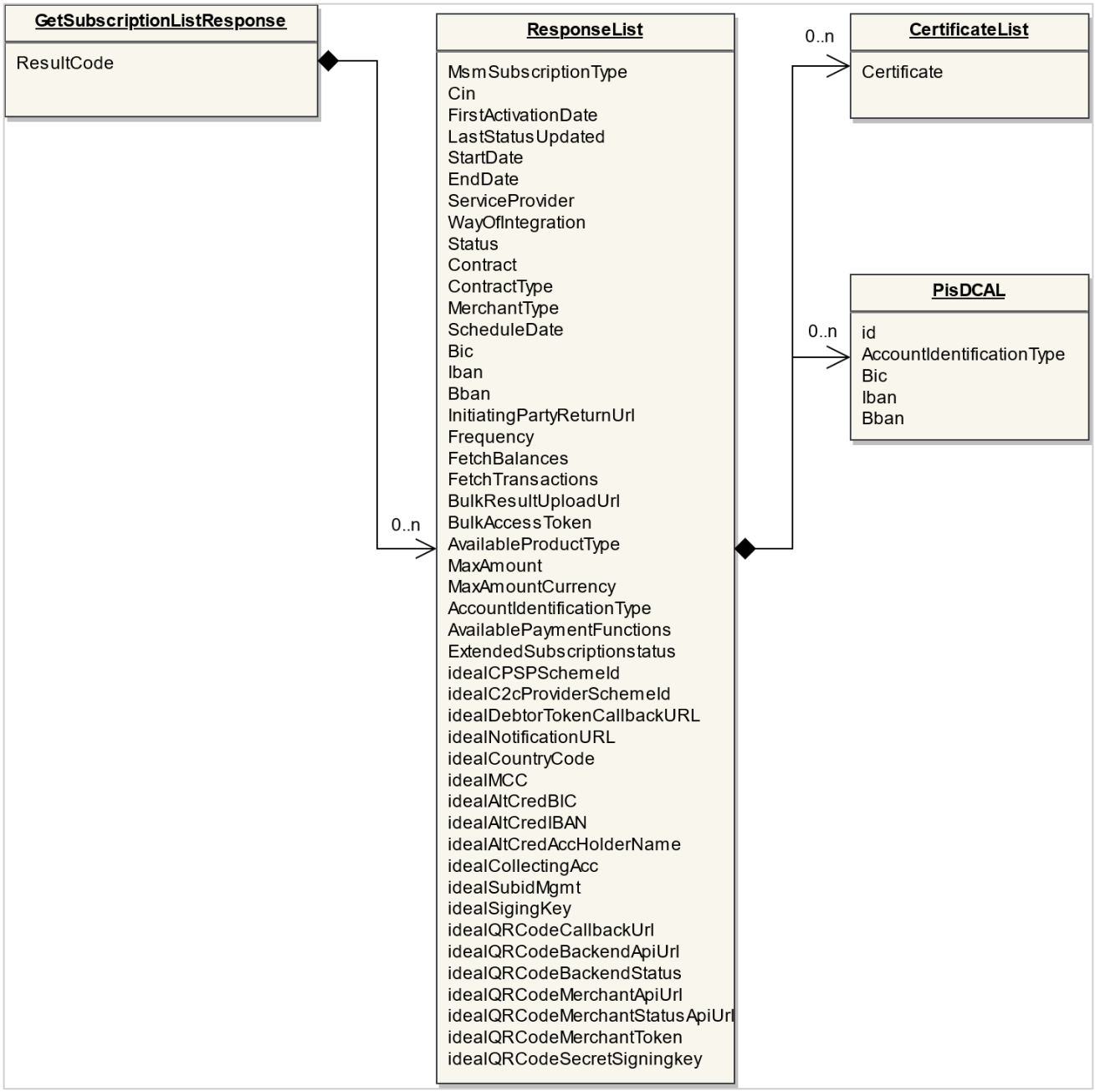
7.2.2 GET Sub-Subscriptions List

Endpoint: Get /merchants/{merchantId}/submerchants/{submerchantId}/subsubscriptions

7.2.2.1 request



7.2.2.2 response



7.2.3 PUT Sub-Subscriptions

Endpoint: Put /merchants/{merchantId}/submerchants/{submerchantId}/subsubscriptions/{subscriptionId}

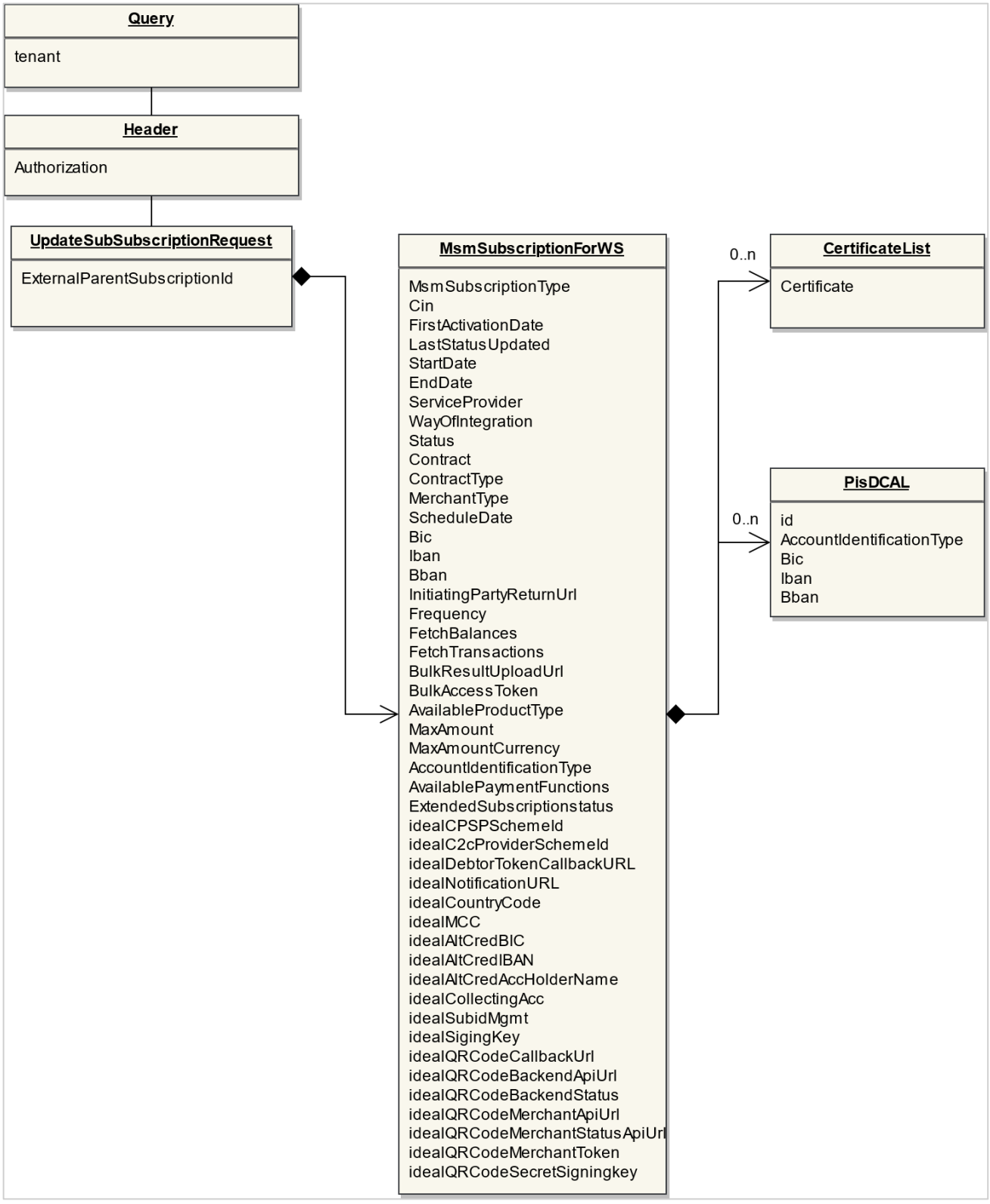
Updating a Sub-Subscription from iDEAL 1 to iDEAL 2

In the table below you can see which fields where added for iDEAL 2 compared to iDEAL 1.

Check with your acquirer if any of these fields are mandatory when you update a iDEAL 1 sub-subscription to iDEAL 2.

Fields newly added for iDEAL 2.0		
Fieldname	iDEAL	Comment
InitiatingPartyReturnUrl	M	name in the GUI: Return URL a default return URL. The Initiating Party can also sent it in each iDEAL request.
idealCPSPSchemeId	O	name in the GUI: CPSP Scheme ID
idealC2cProviderSchemeId	O	name in the GUI: C2C Provider SCheme ID
idealDebtorTokenCallbackURL	O	name in the GUI: Debtortoken callback URL
idealNotificationURL	O	name in the GUI: Status Notification URL
idealCountryCode	O	name in the GUI: Country code
idealMCC	O	name in the GUI: Merchant Category Code (MCC)
idealQRCodeCallbackUrl	O	name in the GUI: QR Code callback URL

7.2.3.1 request



7.2.3.2 response