Federal
Public Service
FINANCE

MD 29/04/2024

REGISTERED CASH REGISTER SYSTEM – TECHNICAL SPECIFICATIONS

API-PROTOCOL BETWEEN FDM & FINCLOUD

GKS SCE

VERSIE 1.06 – 14/03/2025

GKS2 Protocol v1.06 TBP

1. Version history

Date	Version	Change
05/08/2024	1.0	Initial version
09/09/2024	1.01	Improvements based on feedback
12/09/2024	1.02	ws_init: clarifications & cleaning up the text
04/10/2024	1.03	minor corrections
12/02/2025	1.04	text improvements, minor corrections in section 12.5
22/02/2025	1.05	12.3.3. - adding solution avoiding loops (based on feedback)
14/03/2025	1.06	retryIntervalsInit added in sections 5, 11 and 12.3.3

2. Preamble

This document explains the high-level design, core technicalities and main functionalities of the protocol between the FDM and FPS Finance.

Intended audience of this document: manufacturers interested in building an FDM for the Belgian market.

3. Context

To increase tax compliance, all business transactions recorded by the use of mandatory RCRS (currently limited to establishments offering restaurant and catering services that exceed the limit of 25.000 euro sales turnover) should be sent electronically to FPS Finance for analysis/horizontal monitoring. FPS Finance can use this information source for more targeted tax audits.

4. Actors

- FPS Finance
  - main stakeholder and organizer of the project
  - provides specifications for POS & FDM (but doesn’t produce them)
  - certifies POS & FDM
  - provides infrastructure to receive business transactions electronically
  - provides infrastructure to manage authentication: creating & renewing client certificates
- Manufacturer
  - produces POS / FDM following specifications of FPS Finance
  - attributes a serial number and secret hardware key to the FDM, and dispatches this information to FPS Finance following an exchange protocol
  - gets their products certified by FPS Finance

    - provides support for their produced devices
- Businesses
    - registers its POS and FDM devices with FPS Finance
    - uses POS and FDM following legal instructions
    - buys his POS and FDM from the Manufacturer (or one of its dealers)
    - asks support for POS / FDM from the Manufacturer (or one of its dealers)
- FDM (= black box)
    - computing device permanently connected to the POS and requiring frequent internet access availability
    - electronically signs all incoming business transactions
    - sends all business transactions to Cloud Partner
- POS (= cash register)
    - manages and stores all orders and business transactions
    - manages and displays messages from FDM
    - sends all business transactions to FDM

5. Data elements in the Fiscal Data Module (FDM)

These data elements should be maintained by the FDM. Write access “ws” means that the value can only be updated with information coming from FPS Finance webservices. Write access “fdm” means that only the FDM can update the value.

Nr	Abbreviation	Explanation	Data type	Read access by user	Write access	Factory default value	Special security
1	fdmId	FDM identification number, used to uniquely identify the FDM unit	11 alphanumerical positions	yes	nobody	provided by manufacturer	Can never be modified.
2	fdmSwVersion	firmware version of FDM	text	yes	fdm		
3	fdmHkey	hardware key, used to verify the identity and to sign business transactions	private key 256 bits	no	nobody	factory generated private key using secp256r1 prime field curve	Stored in security enclave (private key). Created for lifetime of the device. Private key may never be exposed in logging.
4	fdmLock	FDM lock. If set to true, the device becomes locked and will no longer accept or	boolean	yes	ws	false	

		sign any business transactions.					
5	lockReason	lock reason for fdmLock	array of language and message for lock reason	yes	ws	FALSE	
6	posIdList	list of allowed PosId numbers by the FDM list containing data elements of 14 alphanumerical positions	array of strings	yes	ws		
7	fdmLastActions	date of last retrieval of queries task list (ws_query or ws_init)	timestamp	yes	fdm	null (only for first boot ever)	
8	fdmLastSuccessfulTrans	timestamp of last successful call to register webservice (in UTC) (`ws_reg` or `ws_nop`)	timestamp	yes	fdm		
9	fdmInit	If set to true, the device should try to perform an init during boot.	boolean	no	fdm		
10	initInterval	During unsuccessful calls to ws_query, interval in seconds at which the FDM should call ws_init.	integer	no	ws		
11	queryInterval	interval in seconds at which the FDM should call ws_query	integer	no	ws		
12	nopInterval	interval in seconds at which the FDM has to report “No Operation”	integer	no	ws		

13	retryIntervals	retry intervals expressed in seconds If the list of retry intervals is exhausted due to persistent failed calls, the last value may be repeated. Use for `ws_reg` and `ws_query`	Array of integers	no	ws	[5, 60, 300, 300, 300, 3600, 10800]	Omit from logging.
14	retryIntervalsInit	retry intervals expressed in seconds If the list of retry intervals is exhausted due to persistent failed calls, the last value may be repeated. Use for `ws_init`	Array of integers	no	ws	[5, 60, 300, 300, 300, 3600, 10800]	Omit from logging.
15	retryCount	number of retries of sent attempt	integer	yes	fdm		
16	vatFromBookingDateList	`VatRatesInput` array of `vatRates` array and `fromBookingDate` Each `VatRatesInput` element looks as follows: see 12.5.5.3 • `vatLabel` contains the VAT label A, B, C, D or X, • `outOfScope` (boolean) no VAT applied, only for `vatLabel` X	array of VatRatesInput	yes	ws		The FDM should apply the correct VAT rates according to the date of use.

		(vatRate = 0 when true) • `vatRate` contains the VAT rate in decimal value					
17	footer	receipt footer Strings may not contain new lines or carriage returns. Array item is considered as single paragraph of the footer.	array of strings	yes	ws		
18	url	url to print on receipt footer	string	yes	ws		
19	unsentEventCount	current number of unsent events transactions in the buffer	integer	no	fdm	0	Omit from logging.
20	postpone	If set to true, the FDM should postpone sending business transactions till `maxMsgSize` is reached.	boolean	no	ws		Omit from logging.
21	maxMsgSize	The maximum size in KB of register webservice message body. This value will determine how many business transactions can be sent in a single webservice call. If the size of one business transaction >= `maxMsgSize`, this transaction can be sent despite `maxMsgSize`. The message body contains only this transaction.	integer	yes	ws	32 KB	Omit from logging.

22	defaultMaxCalls	maximum number of webservice calls to register webservice per hour	integer	no	ws	0	Omit from logging.
23	maxCalls	list of regulation slots to limit the calls to the register webservice A regulation slot is an object with 3 properties: day of week, hour slot and maximum number of calls.	array of regulation slots	no	ws		Omit from logging.
24	maxBuffer	the maximum limit for unsent business transactions in the buffer	integer	no	ws	0	Omit from logging.
25	paramsVersion	Date and time of the params version currently used by the FDM. A params version is an array of `ParamsFromSystemDateInput` (`fromSystemDate` and `params`).	timestamp	yes	fdm	empty	Omit from logging.
26	ws_reg	webservice to register business transaction(s)	hostname and portnr	yes	ws	will be provided	
27	ws_nop	webservice to post NOP transaction	hostname and portnr	yes	ws	will be provided	
28	ws_init	webservice to (re)initialize an FDM	hostname and portnr	yes	user and ws	will be provided	
29	ws_query	webservice to check if there are queries to do	hostname and portnr	yes	ws	will be provided	

30	ws_ntp1, ws_ntp2, ws_ntp3	NTP time server. We use UTC time zone. The FDM should use in priority **ws_ntp1** and **ws_ntp2**. If they don't work, the FDM can use the **ws_ntp3.** **ws_ntp3** can be specified only manually via the administration console.	hostname and portnr	yes	ws & user	will be provided	
31	certstoreTrust	certificate store containing trust certificates to resolve server certificate	set of X.509 certificates (PEM)	yes	ws		Omit from logging.
32	certstoreHw	root and intermediate hardware certificate published by FDM manufacturer	set of X.509 certificates (PEM)	yes	no	preconfigured in firmware	Omit from logging.
33	certClient	client certificate used in the mTLS session This client certificate is valid for one year (this value can be modified, adapted by FPS Finance according to business or technical needs).	PKCS #12 format	yes (except the private key)	ws		Stored at least in non-user accessible memory. Ideally stored in security enclave. Private key may never be exposed in logging.
34	clientCertSn	serial number of the client certificate	string	yes	ws		
35	certClaim	claim certificate used in the initialization webservice					Omit from logging.
36	fdmDateTime	RTC time of FDM	timestamp				

		We assume here that an RTC that is not powered by main power or battery adopts a default date/time in the past.					

6. FDM certificate management

6.1 FDM device identity

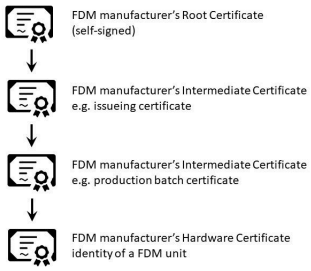
Each FDM has a unique identity which is provided by the FDM manufacturer. This identity consists of an `fdmId`, hardware key `fdmHkey` and certificate for life (use an expiration date in the very far future). The hardware key `fdmHkey` is a secp256r1 prime field curve private key and is stored in the security enclave of the FDM. The hardware certificate is a digital certificate that should have the following elements:

- SHA-256 hashing algorithm
- Elliptic Curve Digital Signature (ECDSA) signing algorithm
- subject name of the certificate contains at least the `fdmId` of the FDM
- SHA-256 fingerprint
- certificate serial number with producer prefix (All hardware certificates issued by an FDM manufacturer should have a unique certificate serial number.)
- no expiration date (use an expiration date in the very far future)

The hardware key will also be used to sign business transactions.

A secure protocol will be established so that the manufacturer can dispatch the hardware certificates to FPS Finance.

The FDM manufacturer is also responsible for establishing a root and one or more intermediate certificates for the hardware certificate conforming industry best practices. The root and intermediate certificates should never expire and should be part of the FDM firmware (a.k.a. `certstoreHw`).

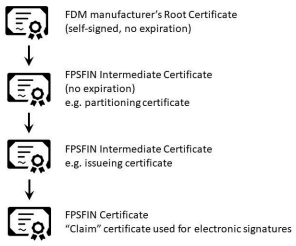


FPS Finance will establish a collaboration protocol with FDM manufacturers which includes security requirements during production of the FDM units. In case of security issues, FPS Finance reserves the right to permanently revoke access to the online services.

6.2 Claim certificate(s)

During the first boot ever of an FDM unit, or when its client certificate has expired, an mTLS session can't be established. In this case, FPS Finance adds an electronic signature to its reply. The FDM should verify this signature using the claim certificate(s). To this end, the FDM manufacturer creates a never-expiring intermediate certificate derived from its root certificate and hands over the private key of this

intermediate certificate to FPS Finance. The latter will be used by FPS Finance to establish its claim certificates. This concept is illustrated in the following diagram.



The usage of claim certificates is further explained in section 12.3.

6.3 Authentication

The main authentication and encryption mechanism of the FDM is based on mutual TLS in which both a server and client certificate are involved. The FDM contains a client certificate issued by FPS Finance. This client certificate is valid for one year (this value can be modified by FPS Finance according to business or technical needs). The private key should be stored in a security enclave of the FDM device. FPS Finance will generate the private key using secp256r1 prime field curve. The hashing and signing algorithms to use are SHA-256 and Elliptic Curve Digital Signature (ECDSA). The subject name of the certificate will amongst others contain the `fdmId` of the FDM. The procedure to issue a new client certificate is described in section 12.3. An automatic update of an operational client certificate is described in section 12.7.

If the client certificate is valid for less than 4 months, the user should receive a warning `CLIENT_CERT_NEAR_EXPIRATION` appearing on the POS at the next possible opportunity. If the client certificate has expired, the user should receive a warning `CLIENT_CERT_EXPIRED` appearing on the POS at the next possible opportunity. The user can check the details of the client certificate via the Administration Console.

In case of abuse, FPS Finance may decide to retract the client certificate. The user can consult the FPS Finance e-service to find out if his access has been retracted.

6.4 Trust certificates

FPS Finance will configure a server certificate for its webservices. This server certificate is typically updated yearly. The FDM should maintain a certificate store `certstoreTrust` to be able to resolve the server certificate. If the FDM can't resolve the server certificate, the user should receive a warning `SERVER_CERT_RESOLVE_FAILED` on the linked POS systems at the next possible opportunity. The FDM is not allowed to unconditionally trust the server certificate (except for the initialization webservice described in section 12.3).

FPS Finance will dispatch the trust certificates automatically. The FDM may delete expired certificates from the certificate store `certstoreTrust`.

7. System clock

The system clock should be synchronized at each system startup or restart, and at least every 24 hours using `ntp` service. If the synchronization fails, a warning should be shown on the POS systems at the next possible opportunity. This warning should be `RTC_NOT_INITIALIZED` when `fdmDateTime` < `fdmLastSuccessfulTrans`, or `RTC_SYNC_FAILED` otherwise.

To synchronize the system clock, the FDM first tries **ws_ntp1** and falls back to **ws_ntp2** if need be. If both fail, **ws_ntp3** can be attempted. This latter is a local configured NTP service entered manually via the administration console.

The FDM operates in the UTC time zone.

8. Booting the FDM

This section describes the minimal requirements of the FDM boot sequence. The first step during boot is to check the expiration date of the client certificate. We distinguish 2 possibilities:

- The FDM boots in **initialization mode** (see section 8.1) in the following situations. For each situation, we set `fdmInit` to `true`.
  - FDM has an expired or non-existent client certificate `certClient`.
  - FDM has an expired Trust Certificate or non-existent `certstoreTrust`.
  - If `fdmLastActions` is null.
  - Or `fdmInit` was already set to `true` during a previous boot or other.
- Otherwise, the FDM boots in **regular mode** (see section 8.2).

Before the first boot ever, the owner of the FDM should contact FPS Finance to register the FDM and the linked POS systems.

8.1 Initialization boot

The following actions should be performed in the order in which they appear. In the specific case of an expired or nonexistent client certificate, all steps are required and should be successful to get the FDM operational.

- The FDM should synchronize the system clock (see section 7).
- The FDM will call the initialization webservice `ws_init` and install the initialization package (see section 12.3). If this call fails persistently, the owner might have to verify, and update if needed, the URL of `ws_init` manually in the Administration Console. After successful execution of the initialization webservice, set `fdmInit = false` and update the `fdmLastActions` timestamp.
- If `fdmDateTime` > `fdmLastSuccessfulTrans`, the FDM verifies whether there are unsent business transactions. If so, these are sent in accordance to section 9.2 “Sending business transactions”. If there are no transactions to be sent, the FDM will send a NOP instead (section 12.4.2).

8.2 Regular boot

The following actions should be performed in the order in which they appear. None of the actions are blocking.

- The FDM should try to resynchronize the system clock with the NTP time server (see section 7).
- If `fdmDateTime` > `fdmLastSuccessfulTrans`, the FDM will:
  - Verify whether the query webservice needs to be contacted (section 12.5).
  - Verify whether there are unsent business transactions. If so, these are sent in accordance to section 9.2 “Sending business transactions”. If there are no transactions to be sent, the FDM will send a NOP instead (section 12.4.2).

9. FDM Receiving & sending business transactions

9.1 FDM buffer

All valid incoming business transactions from the POS are enriched and signed immediately. If the FDM is locked (`fdmLock = true`), all incoming business transactions will be refused and the error `lockReason` message should be shown on the linked POS systems at the next possible opportunity (in accordance with the POS language choice).

In order to have a repeatable hash for the electronic signature, canonical JSON should be used (see section 14 for details). The electronic signature consists of a SHA-256 hash, encrypted with the `fdmHkey` using Elliptic Curve Digital Signature Algorithm (ECDSA) in base64 output. The business transactions, their electronic signatures and localization information (GPS coordinates) are then stored temporarily in the persistent memory of the FDM. We refer to this memory as the buffer of the FDM. The FDM always tries to ventilate from the oldest business transaction first to the most recent business transactions last. The FDM should maintain an `fdmLastSuccessfulTrans` timestamp indicating the last successful call to the register webservice (`ws_reg` or `ws_nop`) and `retryCount` indicating the number of

eventual call retries after failure. After each successful sending, `fdmLastSuccessfulTrans` is updated and the `retryCount` is reset (see section 13.4).

The buffer should contain at least:

- the full business transaction
- the timestamp of successfully sending the transaction

The FDM keeps tracks of which transactions it still has to send. If a business transaction has been sent successfully, it may be deleted after 10 days. If the FDM reaches its physical memory limit, sent business transactions may be deleted exceptionally before 10 days, always starting with the oldest transactions.

The buffer with unsent business transactions is limited in size, see section 10.2 for more details and implications.

The buffer contents can be read by the user via the Administration Console (see section 11).

9.2 Sending business transactions

In the general case, all business transactions should be sent to FPS Finance without delay. Additionally, the FDM should also send a “No Operation” (NOP) message in case the buffer is empty and the current timestamp exceeds `fdmLastSuccessfulTrans` + `nopInterval`. In case the buffer remains empty, this NOP message should be repeated at each `nopInterval`. The NOP message is a regular webservice call with an empty transaction list. This is further explained in section 12.4 which explains the register webservice (`ws_reg`).

There are however some exceptions about when to call the register webservice (`ws_reg`) which are used for load balancing purposes at the client side. These are described in the following subsections 9.2.1, 9.2.2 and 9.2.3.

> The FDM may **never** ignore the exceptions described in sections 9.2.1, 9.2.2 and 9.2.3, except for NOP messages. FPS Finance will reduce the risk of a full buffer, with amongst others the following measures:
>
> - fine-tune the parameters `defaultMaxCalls`, `maxCalls`, `maxMsgSize`, and `maxBuffer`
> - require that the manufacturers install enough physical memory in the FDM devices, so that `maxBuffer` can be set sufficiently high

The FDM will only function with the POS systems it is registered for (see section 10.1).

9.2.1 Postpone transmission

If `postpone = true`, the FDM should postpone calls to the register webservice (`ws_reg`) till `maxMsgSize` is reached, so several business transactions can be bundled in 1 webservice call. If, due to a very low number of business transactions, the level of `maxMsgSize` isn’t reached within 60 minutes, a call is performed anyway if there is no regulation slot preventing it - see section 9.2.3.

9.2.2 Maximum hourly transmission speed

If `defaultMaxCalls > 0`, the FDM can call the register webservice maximally `defaultMaxCalls` times per hour. Each call to the register webservice can contain multiple business transactions, but is limited in physical size by `maxMsgSize`. The FDM should try to evenly distribute the webservice calls if possible.

9.2.3 Transmission regulation slots

This mechanism allows to further refine the maximum hourly transmission speed explained in the previous section. If the `maxCalls` array is not empty, the FDM should respect the regulation slots.

A regulation slot contains 3 elements.

- The first element refers to the day of the week: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, and Sunday.
- The second elements refers to an hourly slot based on a 24 hour clock. For example, 18 refers to the hourly timeslot between 18:00 and 19:00 o’clock.
- The last element refers to the maximum number of allowed webservice calls to the Register webservice. If this value is 0, then no calls can be performed in that hourly timeslot.

GraphQL description for regulation slots:

The enum of the first element:

```
enum WEEKDAY {
  MONDAY
  TUESDAY
  WEDNESDAY
  THURSDAY
  FRIDAY
  SATURDAY
  SUNDAY
}
```

Definition of `MaxCallsInput`:

```
input MaxCallsInput {
  dayOfWeek: WEEKDAY!
  hour: Integer!             # range 0 - 23.
  maxCallsInSlot: Integer!   # The maximum number of calls allowed in the specified time slot.
}
```

Regulation slots sample:

The following example forbids the FDM to call the register webservice on Saturdays between 18:00 and 19:00 o'clock and on Mondays between 17:00 and 18:00 o'clock. Furthermore, on Sundays, the maximum hourly transmission speed is 100 calls between 12:00 and 13:00 o'clock.

```
[
  {
    "dayOfWeek": "SATURDAY",
    "hour": 18,
    "maxCallsInSlot": 0
  },
  {
    "dayOfWeek": "SUNDAY",
    "hour": 12,
    "maxCallsInSlot": 100
  },
  {
    "dayOfWeek": "MONDAY",
    "hour": 17,
    "maxCallsInSlot": 0
  }
]
```

10. FDM Service interruption

10.1 FDM-POS mismatch

The FDM only accepts business transactions from the POS devices it is registered for. The check is done via the identification number of the POS: `posIdList`. This value is communicated by FPS Finance via the SetPosAllowlist query of the query webservice (`ws_query`). See section 12.5.5.4.

When the FDM is connected to a wrong POS system, the user should receive an error `UNKOWN_POS` via the linked POS systems at the next possible opportunity.

10.2 FDM buffer limit

The buffer that holds the unsent business transactions is limited in size. It may contain maximally `maxBuffer` unsent business transactions. The counter `unsentEventCount` holds, at each moment in time, the number of unsent event transactions in the buffer. When >70% of the buffer memory is used, the user receives a warning `BUFFER_NEAR_FULL` via the connected POS systems at the next possible opportunity. If 100% of the buffer memory is used, the FDM stops accepting incoming business transactions from the connected POS systems. The user receives an error `BUFFER_FULL` that explains the situation via the connected POS systems at the next possible opportunity.

If the buffer of the FDM is full, the user should connect its FDM to the internet so that the unsent business transactions can be ventilated.

FPS Finance can disable the buffer limit by setting the value `maxBuffer = 0`. In that case, the buffer size is limited by the physical capacity of the FDM device.

10.3 FDM lock

FPS Finance can decide to lock and unlock an FDM device (`fdmLock`). A locked FDM is still connected to the online service, but will refuse to accept and sign incoming business transactions from the POS. The user will systematically receive the error `lockReason` message (in accordance with the POS language choice).

11. Administration console

The Administration Console of the FDM should at least support the following functionalities in a stand-alone fashion, without any dependency from the manufacturer.

Functionality	User access
Read the error log.	X
Read the buffer transactions history: the percentage of used buffer (`unsentEventCount` divided by `maxBuffer` when not = 0), timestamp of last successful transmission (`fdmLastSuccessfulTrans`) and number of retries since last send attempt (`retryCount`).	X
Configure network settings.	X
Read the data elements which are accessible to the user (see section 5).	X
An electronic form to enter the URL of `ws_init` (see section 12.3). Remark : when the URL of `ws_init` is changed in the administration console, if we are in a `retryIntervalsInit` cycle in the initialization webservice, we restart the cycle at the first value of the list. The goal is to have a retry ASAP.	X
Manually adapt/fill the **ws_ntp3** URL of the NTP timeserver.	X

A button to resync the real-time clock with the NTP timeserver.	X
Consult the current version of the FDM parameters (`paramsVersion`).	X
A button to call the query webservice (download queries list and execute queries if necessary) ws_query	X

12. Webservices

12.1 General

All webservices of type “**Restful** Web Services” using **JSON** payload encoded in **UTF-8** over **HTTP 1.1**. Authentication and encryption is realized using **mTLS 1.2** using a client and server certificate. As security technology evolves, FPS Finance can require newer versions or other (security) protocols.

If for some reason the FDM received illegal byte sequence(s) from the POS, it may exceptionally represent these bytes via a JSON escape sequence `\u` .

If mTLS authorization fails, the TCP/IP connection may be closed immediately without a reply from the webservice.

12.2 Error logging

At least all failed webservice requests and responses (if any) should be logged for a minimum of 5 days, or at least the 10 last failed requests (HTTP headers, JSON payload). The FDM may deviate from this principle if it runs out of physical storage: loggings may be deleted starting from the oldest entries. Log items should be accessible for FPS Finance with a message ws_query.

Others errors, requests that fail due to no internet connection, etc. should not be logged in this way (but via the generic error log with only a message without content). These log items should be accessible to the user for inspection via the Administration Console (see section 11).

GraphQL description for error logging:

The error logging `FailedRequest` :

```
# The fields for FailedRequests
type FailedRequest {
  timestamp: String!         # "2024-04-09T04:00:00Z" for example
  requestMethod: String      # POST or GET
  requestUri: String         # address of the endpoint for the web service
  requestHeaders: String     # the headers of the request in base64
  requestBody: String        # the body of the request in base64
  replyStatusCode: Integer   # the status code obtained from the server (if reached that far)
  replyHeaders: String       # the headers obtained from the server in base64 (if reached that far)
  replyBody: String          # the body obtained from the server in base64 (if it was reached that far)
  errorMessage: String       # A error message
  data: [DataItem!]          # Generic data object (can be used for POS for example)
}

type DataItem {
    name: String!
    value: String!
}
```

12.3 Initialization webservice

The initialization webservice provides a bootstrap package for the FDM. It contains the URLs, trust/claim/client certificates, parameters, and VAT rates.

The user may manually modify the value of `ws_init` via the Administration Console. The correct value will be published on the e-service of FPS Finance.

The webservice is exposed via HTTP with TLS without validation of the server certificate, but with the use of an **authorization token**. This token is established as a JSON web token using an Elliptic Curve Digital Signature (ECDSA) and the hardware key `fdmHkey`. The payload contains the subject `init`, the "issued at time" timestamp `iat`, and the `fdmId`.

Conditionally, you must add the serial number of the client certificate `clientCertSn`:

- If the client certificate is nonexistent or expired, the `clientCertSn` was not provided in the request.
- If the client certificate is valid and not expired, the `clientCertSn` was provided in the request.

JWT header:

```
{ "alg": "ES256", "typ": "JWT" }
```

JWT payload in case of nonexistent or expired client certificate:

```
{ "sub": "init",
  "iat": 1686315756,
  "fdmId": "FDM01234567"
}
```

JWT payload in case of existing, not expired client certificate. We additionally provide the serial number of the client certificate:

```
{ "sub": "init",
  "iat": 1686315756,
  "fdmId": "FDM01234567",
  "clientCertSn": "dhjs76djs"
 }
```

> ⚠ The webservice call should take place immediately after the generation of the token, because FPS Finance enforces a very short token lifespan.

Protocol: TLS with authorization token (JWT)

Verb: POST

URL: /init (using `ws_init`)

12.3.1 Reply with retry invitation

It is possible that FPS Finance can't synchronously provide an initialization package. In this case, the reply contains a retry invitation expressed in seconds (HTTP code 200). When the waiting time has elapsed, the FDM can retry the initialization.

In the following example, the FDM is invited to wait 5 minutes before a retry.

```
{ "wait": 300 }
```

12.3.2 Successful reply

The initialization package is formatted in JSON and transcoded to base64. The reply can be with or without a new client certificate.

- If the `clientCertSn` was not provided in the request, FPS Finance will issue a new client certificate in the reply.
- If the `clientCertSn` was provided in the request (in case of an existing client certificate that is not expired), FPS Finance will issue a new client certificate in the reply if the previous client certificate is retracted. Otherwise, the `clientCert` element will be empty or not

present in the reply. This means that the FDM can continue to use its current client certificate.

We show a concrete example of a possible reply. The queries are: **SetFdmLock**, **SetParams**, **SetVatRates**, **SetPosAllowlist**, **SetTrustCert** and **SetUrls**.

Remarks:

- the `actionId` is specified only when FPS Finance needs this information in the GraphQL reply to avoid resending this query at the next `ws_query` call.

The **SetClientCert** query (in case the clientCert needs to be renewed) will be done in the next call of ws_query.

Sample: (for more information about mutations used in the sample, see 12.5.5.1, 12.5.5.2, 12.5.5.3, 12.5.5.4, 12.5.5.5 & 12.5.5.6)

```
[
    {
        "dataResultUrl": "https://www.fod/data_endpoint",
        "errorResultUrl": "https://www.fod/error_endpoint",
        "actionId": "2023-01-05T18:03:23.000Z",
        "messageDateTimeEcho": "2024-04-17T04:00:00Z",
        "operationType": "SET_FDM_LOCK",
        "executable": {
            "query": "mutation SetFdmLock($fdmLockFromDateList: [FdmLockInput!]!)
                        { setFdmLock(fdmLockFromDateList: $fdmLockFromDateList)
                            {
                                device { fdmId fdmDateTime bufferCapacityUsed }
                                configuration { paramsVersionObject { paramsVersion }
                                fdmLockFromDateList
                                    {fromSystemDate fdmLock lockReason { language message } } } } }",
            "operationName": "SetFdmLock",
            "variables": {
                "fdmLockFromDateList": [
                    {
                        "fromSystemDate": "2024-06-01",
                        "fdmLock": false,
                        "lockReason": null
                    },
                    {
                        "fromSystemDate": "2024-07-01",
                        "fdmLock": true,
                        "lockReason": [{"language": "EN", "message": "FDM_LOCKED-EN" },
                                       {"language": "NL", "message": "FDM_LOCKED-NL" },
                                       {"language": "FR", "message": "FDM_LOCKED-FR" },
                                       {"language": "DE", "message": "FDM_LOCKED-DE" }]
                    }
                ]
            }
        }
    },
    {
        "dataResultUrl": "https://www.fod/data_endpoint",
        "errorResultUrl": "https://www.fod/error_endpoint",
        "actionId": "2023-01-05T18:03:23.000Z",
        "messageDateTimeEcho": "2024-04-17T04:00:00Z",
        "operationType": "SET_PARAMS",
        "executable": {
            "query": "mutation SetParams($paramsVersionObject: ParamsVersionObjectInput!)
                        { setParams(paramsVersionObject: $paramsVersionObject)
                        {
                            device { fdmId fdmDateTime bufferCapacityUsed }
```

```
                            configuration { paramsVersionObject { paramsVersion }
                            fdmLockFromDateList
                                {fromSystemDate fdmLock lockReason { language message } } } } }",
            "operationName": "SetParams",
            "variables": {
                "paramsVersionObject": {
                    "paramsVersion": "2023-01-05T18:03:23.000Z",
                    "paramsArray": [
                        {
                            "fromSystemDate": "2024-07-01",
                            "params": {
                                "url": "https://www.gs2.be",
                                "initInterval": 345600,
                                "nopInterval": 21600,
                                "maxBuffer": 20000,
                                "queryInterval": 21600,
                                "retryIntervals": [5, 60, 300, 300, 300, 3600, 10800],
                                "retryIntervalsInit": [5, 60, 300, 300, 300, 3600, 10800],
                                "maxMsgSize": 32,
                                "postpone": true,
                                "defaultMaxCalls": 30,
                                "maxCalls": [{"dayOfWeek": "SATURDAY", "hour": 18, "maxCallsInSlot": 0},
                                    {"dayOfWeek": "SUNDAY", "hour": 12, "maxCallsInSlot": 100},
                                    {"dayOfWeek": "MONDAY", "hour": 17, "maxCallsInSlot": 0}],
                                "footer": ["myfootertext1", "myfootertext1"]
                            }
                        },
                        {
                            "fromSystemDate": "2024-08-01",
                            "params": {
                                "url": "https://www.gs2.be",
                                "initInterval": 345600,
                                "nopInterval": 21600,
                                "maxBuffer": 20000,
                                "queryInterval": 21600,
                                "retryIntervals": [5, 60, 300, 300, 300, 3600, 10800],
                                "retryIntervalsInit": [5, 60, 300, 300, 300, 3600, 10800],
                                "maxMsgSize": 32,
                                "postpone": true,
                                "defaultMaxCalls": 30,
                                "maxCalls": [{"dayOfWeek": "SATURDAY", "hour": 18, "maxCallsInSlot": 0},
                                    {"dayOfWeek": "SUNDAY", "hour": 12, "maxCallsInSlot": 100},
                                    {"dayOfWeek": "MONDAY", "hour": 17, "maxCallsInSlot": 0}],
                                "footer": ["myfootertext2", "myfootertext2"]
                            }
                        }
                    ]
                }
            }
        }
    },
    {
        "dataResultUrl": "https://www.fod/data_endpoint",
        "errorResultUrl": "https://www.fod/error_endpoint",
        "actionId": "2023-01-05T18:03:23.000Z",
        "messageDateTimeEcho": "2024-04-17T04:00:00Z",
        "operationType": "SET_VAT_RATES",
        "executable": {
```

```
            "query": "mutation SetVatRates($vatFromBookingDateList: [VatRatesInput!]!)
                        { setVatRates(vatFromBookingDateList: $vatFromBookingDateList)
                        {
                            device { fdmId fdmDateTime bufferCapacityUsed }
                            configuration { paramsVersionObject { paramsVersion }
                            fdmLockFromDateList
                                {fromSystemDate fdmLock lockReason { language message } } } } }",
            "operationName": "SetVatRates",
            "variables": {
                "vatFromBookingDateList": [
                    {
                    "fromBookingDate": "2020-01-01",
                    "vatRates": [
                        {
                            "vatLabel": "A",
                            "vatRate": 21,
                            "outOfScope": false
                        },
                        {
                            "vatLabel": "B",
                            "vatRate": 12,
                            "outOfScope": false
                        },
                        {
                            "vatLabel": "C",
                            "vatRate": 6,
                            "outOfScope": false
                        },
                        {
                            "vatLabel": "D",
                            "vatRate": 0,
                            "outOfScope": false
                        },
                        {
                            "vatLabel": "X",
                            "vatRate": 0,
                            "outOfScope": true
                        }
                    ]
                    },
                    {
                    "fromBookingDate": "2025-04-01",
                    "vatRates": [
                        {
                            "vatLabel": "A",
                            "vatRate": 21,
                            "outOfScope": false
                        },
                        {
                            "vatLabel": "B",
                            "vatRate": 17,
                            "outOfScope": false
                        },
                        {
                            "vatLabel": "C",
                            "vatRate": 6,
                            "outOfScope": false
                        },
```

```
                        {
                            "vatLabel": "D",
                            "vatRate": 0,
                            "outOfScope": false
                        },
                        {
                            "vatLabel": "X",
                            "vatRate": 0,
                            "outOfScope": true
                        }
                    ]
                    }
                ]
            }
        }
    },
    {
        "dataResultUrl": "https://www.fod/data_endpoint",
        "errorResultUrl": "https://www.fod/error_endpoint",
        "actionId": "2023-01-05T18:03:23.000Z",
        "messageDateTimeEcho": "2024-04-17T04:00:00Z",
        "operationType": "SET_POS_ALLOWLIST",
        "executable": {
            "query": "mutation SetPosAllowlist($posIdFromDateList: [PosAllowlistInput!]!)
                        { setPosAllowlist (posIdFromDateList: $posIdFromDateList)
                        {
                            device { fdmId fdmDateTime bufferCapacityUsed }
                            configuration { paramsVersionObject { paramsVersion }
                            fdmLockFromDateList {fromSystemDate fdmLock lockReason { language message } }
                            posIdFromDateList { fromSystemDate posIdList } } } }",
            "operationName": "SetPosAllowlist",
            "variables": {
                "posIdFromDateList": [
                    {
                        "fromSystemDate": "2024-07-01",
                        "posIdList": ["Csnpos00000001", "Csnpos00000003"]
                    },
                    {
                        "fromSystemDate": "2024-08-01",
                        "posIdList": ["Csnpos00000001", "Csnpos00000003", "Csnpos00000004"]
                    }
                ]
            }
        }
    },
    {
        "dataResultUrl": "https://www.fod/data_endpoint",
        "errorResultUrl": "https://www.fod/error_endpoint",
        "actionId": "2023-01-05T18:03:23.000Z",
        "messageDateTimeEcho": "2024-04-17T04:00:00Z",
        "operationType": "SET_TRUST_CERT",
        "executable": {
            "query": "mutation SetTrustCert($trustCertFromDateList: [TrustCertInput!]!)
                        { setTrustCert(trustCertFromDateList: $trustCertFromDateList)
                        {
                            device { fdmId fdmDateTime bufferCapacityUsed }
                            configuration { paramsVersionObject { paramsVersion }
                            fdmLockFromDateList
```

```
                                {fromSystemDate fdmLock lockReason { language message } } } } }",
            "operationName": "SetTrustCert",
            "variables": {
                "trustCertFromDateList": [
                    {
                        "fromSystemDate": "2024-07-01",
                        "trustCert": [ "cert1x509pem", "cert2x509pem", "cert3x509pem" ]
                    },
                    {
                        "fromSystemDate": "2024-08-01",
                        "trustCert": [ "cert1x509pem2", "cert2x509pem2", "cert3x509pem2" ]
                    }
                ]
            }
        }
    },
    {
        "dataResultUrl": "https://www.fod/data_endpoint",
        "errorResultUrl": "https://www.fod/error_endpoint",
        "actionId": "2023-01-05T18:03:23.000Z",
        "messageDateTimeEcho": "2024-04-17T04:00:00Z",
        "operationType": "SET_URLS",
        "executable": {
            "query": "mutation SetUrls($urlsFromDateList: [UrlInput!]!)
                        { setUrls(urlsFromDateList: $urlsFromDateList)
                        {
                            device { fdmId fdmDateTime bufferCapacityUsed }
                            configuration { paramsVersionObject { paramsVersion }
                            fdmLockFromDateList
                                {fromSystemDate fdmLock lockReason { language message } } } } }",
            "operationName": "SetUrls",
            "variables": {
                "urlsFromDateList": [
                    {
                        "fromSystemDate": "2024-07-01",
                        "urls": [
                            { "key": "ws_reg", "value": "https://myurl:8080" },
                            { "key": "ws_nop", "value": "https://myurl:8080" },
                            { "key": "ws_query", "value": "https://myurl:8080" },
                            { "key": "ws_init", "value": "https://myurl:8080" },
                            { "key": "ws_ntp1", "value": "https://myurl:8080" },
                            { "key": "ws_ntp2", "value": "https://myurl:8080" }
                        ]
                    },
                    {
                        "fromSystemDate": "2024-08-01",
                        "urls": [
                            { "key": "ws_reg", "value": "https://myurl2:8080" },
                            { "key": "ws_nop", "value": "https://myurl2:8080" },
                            { "key": "ws_query", "value": "https://myurl2:8080" },
                            { "key": "ws_init", "value": "https://myurl2:8080" },
                            { "key": "ws_ntp1", "value": "https://myurl2:8080" },
                            { "key": "ws_ntp2", "value": "https://myurl2:8080" }
                        ]
                    }
                ]
            }
        }
```

```
    }
]
```

The full reply contains the current `clientCert` (empty value when `clientCertSn` is specified in the JWT), the initialization package in base64 encoding, the digital signature [of the client certificate as `"pem"` + `"init-package-in-base64"`] and the claim certificates. The electronic signature is established with SHA-256 and Elliptic Curve Digital Signature (ECDSA) in base64 output using the private key of the claim certificate. The claim certificate and its intermediates are specific to an FDM vendor (see section 6.2). The `PKCS12` contains the client certificate and the private key (encryption password = secret, details can be requested from the relevant department). The `pem`, used for the digital signature, contains only the certificate.

When the `clientCertSn` is not specified:

```
{ "clientCert": "pkcs12pem-with-password", "init": "init-package-in-base64", "digitalSignature": "hjd76gd5",
"claim": [ "cert1x509pem", "cert2x509pem" ] }
```

When the `clientCertSn` is specified:

```
{ "clientCert": "", "init": "init-package-in-base64", "digitalSignature": "hjd76gd5",
"claim": [ "cert1x509pem", "cert2x509pem" ] }
```

The FDM should perform the following steps:

- Validate the chain of claim certificates using the root certificate that is built into the firmware.
- Validate the digital signature using the claim certificate. To this end, SHA-256 and ECDSA with base64 output should be used.
- If the validation of the claim certificates and digital signature is successful, the FDM is allowed to install the initialization package.
- First, the client certificate from the initialization package can be imported using the shared secret.
- Then the queries are executed (see 12.5 for more information).
- Then the FDM calls `ws_query` to receive pending queries and execute them (if there are any).

12.3.3 Exceptions

- HTTP code 400/404: Bad Request/General error
- HTTP code 401/403: Authorization token is invalid
- HTTP code 404: Page not found (bad ws_init URL)
- HTTP code 406: FPS Finance refuses to issue a new client certificate (normally will never happen, we use the `fdmLock` parameter)

In case of an unsuccessful execution of the initialization webservice, the warning `INITIALIZATION_FAILED` should be shown on the linked POS systems at the next possible opportunity.

If a retry of the initialization webservice is needed (bad client certificate, bad claim certificates,…), to avoid a loop in the initialization webservice calls, the `retryIntervalsInit` must be applied by the FDM before next call. The parameter `retryIntervalsInit` specifies a series of time intervals expressed in seconds at which a retry should take place after a failed call to the initialization webservice. If the list of retry intervals is exhausted due to persistent failed calls, the last value may be repeated. Without availability of `retryIntervalsInit`, the FDM must use a interval of 600 seconds (10 minutes) before next call.

Remark : when the URL of `ws_init` is changed in the administration console, if we are in a `retryIntervalsInit` cycle in the initialization webservice, we restart the cycle at the first value of the list. The goal is to have a retry ASAP (see section 11).

12.4 Register webservice

The business transactions and their electronic signatures, which are stored in the FDM buffer (see section 9.1), should be dispatched to FPS Finance using the register webservice.

12.4.1 Register business transactions

The payload of the register webservice contains the following elements:

- The size of the FDM buffer `unsentEventCount` at the moment of the webservice call.
- The fdmId.
- An array of business transaction `events`. Each array element contains a single business transaction. By default, only one business transaction per webservice call is preferred, although multiple business transactions in one call is permitted. If `postpone = true`, multiple business transactions must be sent (`maxMsgSize`, see 9.2.1 for more details) in a single webservice call. The maximum size of the body payload is determined by the `maxMsgSize` parameter.
- a single business transaction consists of the detailed transaction description (`enrichedEventData`), the electronic signature in base64 (`digitalSignature`), a short signature for printed ticket (`shortSignature`) and the localisation information (`fdmLocalisation`).
- When a read transaction is corrupted (due to bad or defective sectors in the memory) or becomes unreadable in the memory, the FDM replaces the standard business transaction with `rawCounter` (required), `rawBytes` (optional) and `rawInfo` (required). `rawCounter` is a counter incremented every time the FDM sends a corrupt transaction to the FPS Cloud. `rawBytes` is sent when the corrupt transaction is readable. `rawInfo` is a comment/info of the FDM. The warning `CORRUPT_RECORD_ENCOUNTERED` should be shown on the linked POS systems at the next possible opportunity. If the issue (`CORRUPT_RECORD_ENCOUNTERED`) persists, FPS Finance can decide to set `fdmLock` to true with a `lockReason` message. All incoming business transactions will then be refused.

You need a valid client certificate `certClient` for calling this webservice (see section 6.3).

The conditions under which the FDM is allowed to call this webservice are described in section 9.2.

Protocol: mTLS using `certClient`

Verb: POST

URL: /reg (using `ws_reg`)

Example request looks as follows:

```
{
    "unsentEventCount": 999,
    "fdmId": "FOD01000001",
    "events": [
        {
            "enrichedEventData": { ... },
            "digitalSignature": "base64h2f4sh1",
            "shortSignature": "sha-1",
            "fdmLocalisation": {
                "geoLocation": {
                    "latitude": 50.083207,
                    "longitude": 5.417244
                }
            }
        },
        {
            "enrichedEventData": { ... },
            "digitalSignature": "base64h2f4sh2",
            "shortSignature": "sha-1",
            "fdmLocalisation": {
            }
        },
        {
            "enrichedEventData": { ... },
            "digitalSignature": "base64h2f4sh3",
            "shortSignature": "sha-1",
            "fdmLocalisation": {
            }
        }
    ]
}
```

Example request with corrupt transaction in memory:

```
{
    "unsentEventCount": 999,
    "fdmId": "FOD01000001",
    "events": [
        {
            "enrichedEventData": { ... },
            "digitalSignature": "base64h2f4sh1",
            "shortSignature": "sha-1",
            "fdmLocalisation": {
                "geoLocation": {
                    "latitude": 50.083207,
                    "longitude": 5.417244
                }
            }
        },
        {
            "rawCounter": 1,
            "rawBytes": "base64h2f4sh1",
            "rawInfo": "eventueel opmerking door fdm"
        },
        {
            "rawCounter": 2,
            "rawInfo": "eventueel opmerking door fdm"
        },
        {
            "enrichedEventData": { ... },
            "digitalSignature": "base64h2f4sh2",
            "shortSignature": "sha-1",
            "fdmLocalisation": {
            }
        },
        {
            "enrichedEventData": { ... },
            "digitalSignature": "base64h2f4sh3",
            "shortSignature": "sha-1",
            "fdmLocalisation": {
            }
        }
    ]
}
```

If the business transactions call fails, the warning `TRANSACTION_UPLOAD_FAILED` should be shown on the linked POS systems at next possible opportunity.

12.4.2 No Operation message

The payload can potentially be empty in case of a “No Operation” (NOP) message which is sent at frequent intervals (see section 9.2). Example of a NOP:

Protocol: mTLS using `certClient`

Verb: POST

URL: /reg (using `ws_nop`)

```
{ "unsentEventCount": 0,
  "fdmId": "FOD01000001",
  "events": []
}
```

If the NOP call fails, the warning `NOP_FAILED` should be shown on the linked POS systems at next possible opportunity.

12.4.3 Replies

HTTP code 200 indicates a successful call. The value of `fdmLastSuccessfulTrans` should be updated and the counter should be reset.

All other replies different than HTTP code 200 indicate an error. In that case, the counter `retryCount` should be incremented. If `retryCount > 10`, the user should receive warning `TRANSACTION_UPLOAD_FAILED` on the connected POS systems at the next possible opportunity.

The parameter `retryIntervals` specifies a series of time intervals expressed in seconds at which a retry should take place after a failed call to the register webservice. If the list of retry intervals is exhausted due to persistent failed calls, the last value may be repeated. Retries are not necessary for failed NOP messages.

> The retry policy may slow down the transmission frequency (see section 9.2). FPS Finance will configure the `retryIntervals` with caution.

12.5 Query webservice

FPS Finance can issue queries to be executed by the FDM. The FDM should call the query webservice (`ws_query`) at each `queryInterval` to check if there are queries to be done (when `fdmDateTime` > `fdmLastActions` + `queryInterval`).

FPS Finance always sends all the queries to be executed at that time (array of queries). If FPS Finance doesn't receive the response to a query, the query will be resent at the next call to `ws_query` (after `queryInterval`). So, after `queryInterval`, the FDM can drop the not already done reply of the previous `ws_query` call.

Each query should be executed by the FDM as soon as possible.

The queries are written in JSON GraphQL compatible format for compatibility with GraphQL on the FDM.

Next, once a query is executed (with or without success), the FDM should reply to `dataResultUrl` (success) or `errorResultUrl` (errors). This is explained in sections 12.5.2. & 12.5.3.

If needed, in the future, FPS Finance can add new queries (mutation or query). In that case, a firmware update is needed.

The FDM accepts only these described queries from the FPS Finance query webservice (`ws_query`). No other queries can be used to modify the parameters described in these queries (mutations). More info about an update of the parameters can be found in section 5.

12.5.1 Get the queries to-do list

Protocol: mTLS with `certClient`

Verb: GET

URL: /query (using `ws_query`)

The possible queries are:

- **Mutation:** `SetClientCert`: the FDM is requested to renew its client certificate `certClient` as soon as possible. See section 12.5.5.7 of this procedure.
- **Mutation:** `SetTrustCert`: the FDM is requested to fetch the chain of trust certificates related to the server certificate as soon as possible. See section 6.4 and 12.5.5.5 for more info.

- **Mutation:** `SetParams` : the FDM is requested to renew its parameters as soon as possible. See section 12.5.5.2 for the details of this procedure.
- **Mutation:** `SetUrls` : the FDM is requested to renew the URLs of the webservices as soon as possible. See section 12.5.5.6 for the details of this procedure.
- **Mutation:** `SetVatRates` : the FDM is requested to renew the VAT rates as soon as possible. See section 12.5.5.3 for the details of this procedure.
- **Mutation:** `SetFdmLock` : the FDM is requested to set the `fdmLock` as soon as possible. See section 12.5.5.1 for the details of this procedure.
- **Mutation:** `SetPosAllowlist` : the FDM is requested to set the `posIdList` as soon as possible. See section 12.5.5.4 for the details of this procedure.
- **Query:** `FailedRequests` : the FDM is requested to send failed request as soon as possible. See section 12.5.5.8 for the details of this procedure.
- **Query:** `FdmConfiguration` : the FDM is requested to send all FDM configurations as soon as possible. See section 12.5.5.9 for the details of this procedure.

An query is always announced with a `actionId` timestamp, so that we can refer to a particular action in the FPS system. There is no fixed order in which the queries appear in the list.

A success call to the ws_query implies an update of `fdmLastActions` timestamp.

When an error occurs and the FDM can't receive the queries to-do list, Warning `UPDATE_TASK_LIST_FAILED` should be shown on the linked POS systems at the next possible opportunity.

If the error concerns a problem with the client certificate or the trust certificate (http code **401**, **403**) of this webservice, or a bad URL (http codes **400**, **404**), set `fdmInit = true` , call the initialization webservice (ws_init) and install the initialization package (see section 12.3). If this call fails persistently, the owner might have to verify, and update if needed, the URL of `ws_init` manually in the Administration Console. After successful execution of the initialization webservice, set `fdmInit = false` and update the `fdmLastActions` timestamp.

For other errors the FDM applies the `retryIntervals` to call the query webservice. The parameter `retryIntervals` specifies a series of time intervals expressed in seconds at which a retry should take place after a failed call to the webservice. If the list of retry intervals is exhausted due to persistent failed calls, the FDM evaluates the condition to execute a call to the initialization webservice (ws_init): If `fdmDateTime` > `fdmLastActions` + `initInterval` , the FDM sets `fdmInit = true` , calls the initialization webservice (ws_init) and installs the initialization package (see section 12.3). If this call fails persistently, the owner might have to verify, and update if needed, the URL of `ws_init` manually in the Administration Console. After successful execution of the initialization webservice, set `fdmInit = false` and update the `fdmLastActions` timestamp.

Otherwise, the FDM continues the try again and the last value of the `retryIntervals` may be repeated until a successful call to query webservice or the condition to call the initialization webservice is reached.

12.5.2 Queries root structure

- Query samples
  - One query:

```
[
    {
        "dataResultUrl": "https://www.fod/data_endpoint",
        "errorResultUrl": "https://www.fod/error_endpoint",
        "actionId": "2023-01-05T18:03:23.000Z",
        "messageDateTimeEcho": "2024-04-17T04:00:00Z",
        "operationType": "SET_FDM_LOCK",
        "executable": {
            "query": "mutation SetFdmLock($fdmLockFromDateList: [FdmLockInput!]!)
                        { setFdmLock(fdmLockFromDateList: $fdmLockFromDateList)
```

```
                        {
                            device { fdmId fdmDateTime bufferCapacityUsed }
                            configuration { paramsVersionObject { paramsVersion }
                            fdmLockFromDateList
                                {fromSystemDate fdmLock lockReason { language message } } } } }",
            "operationName": "SetFdmLock",
            "variables": {
                "fdmLockFromDateList": [
                    {
                        "fromSystemDate": "2024-06-01",
                        "fdmLock": false,
                        "lockReason": null
                    },
                    {
                        "fromSystemDate": "2024-07-01",
                        "fdmLock": true,
                        "lockReason": [{"language": "EN", "message": "FDM_LOCKED-EN" },
                        {"language": "NL", "message": "FDM_LOCKED-NL" },
                        {"language": "FR", "message": "FDM_LOCKED-FR" },
                        {"language": "DE", "message": "FDM_LOCKED-DE" }]
                    }
                ]
            }
        }
    }
]
```

- two queries:

```
[
    {
        "dataResultUrl": "https://www.fod/data_endpoint",
        "errorResultUrl": "https://www.fod/error_endpoint",
        "actionId": "2023-01-05T18:03:23.000Z",
        "messageDateTimeEcho": "2024-04-17T04:00:00Z",
        "operationType": "SET_FDM_LOCK",
        "executable": {
            "query": "mutation SetFdmLock($fdmLockFromDateList: [FdmLockInput!]!)
                        { setFdmLock(fdmLockFromDateList: $fdmLockFromDateList)
                        {
                            device { fdmId fdmDateTime bufferCapacityUsed }
                            configuration { paramsVersionObject { paramsVersion }
                            fdmLockFromDateList
                                {fromSystemDate fdmLock lockReason { language message } } } } }",
            "operationName": "SetFdmLock",
            "variables": {
                "fdmLockFromDateList": [
                    {
                        "fromSystemDate": "2024-06-01",
                        "fdmLock": false,
                        "lockReason": null
                    },
                    {
                        "fromSystemDate": "2024-07-01",
                        "fdmLock": true,
                        "lockReason": [{"language": "EN", "message": "FDM_LOCKED-EN" },
                        {"language": "NL", "message": "FDM_LOCKED-NL" },
                        {"language": "FR", "message": "FDM_LOCKED-FR" },
                        {"language": "DE", "message": "FDM_LOCKED-DE" }]
```

```
                    }
                ]
            }
        }
    },
    {
        "dataResultUrl": "https://www.fod/data_endpoint",
        "errorResultUrl": "https://www.fod/error_endpoint",
        "actionId": "2023-01-05T18:03:23.000Z",
        "messageDateTimeEcho": "2024-04-17T04:00:00Z",
        "operationType": "SET_PARAMS",
        "executable": {
            "query": "mutation SetParams($paramsVersionObject: ParamsVersionObjectInput!)
                        { setParams(paramsVersionObject: $paramsVersionObject)
                        {
                            device { fdmId fdmDateTime bufferCapacityUsed }
                            configuration { paramsVersionObject { paramsVersion }
                            fdmLockFromDateList
                                {fromSystemDate fdmLock lockReason { language message } } } } }",
            "operationName": "SetParams",
            "variables": {
                "paramsVersionObject": {
                    "paramsVersion": "2023-01-05T18:03:23.000Z",
                    "paramsArray": [
                        {
                            "fromSystemDate": "2024-07-01",
                            "params": {
                                "url": "https://www.gs2.be",
                                "initInterval": 345600,
                                "nopInterval": 21600,
                                "maxBuffer": 20000,
                                "queryInterval": 21600,
                                "retryIntervals": [5, 60, 300, 300, 300, 3600, 10800],
                                "retryIntervalsInit": [5, 60, 300, 300, 300, 3600, 10800],
                                "maxMsgSize": 32,
                                "postpone": true,
                                "defaultMaxCalls": 30,
                                "maxCalls": [{"dayOfWeek": "SATURDAY", "hour": 18, "maxCallsInSlot": 0},
                                {"dayOfWeek": "SUNDAY", "hour": 12, "maxCallsInSlot": 100},
                                {"dayOfWeek": "MONDAY", "hour": 17, "maxCallsInSlot": 0}],
                                "footer": ["myfootertext1", "myfootertext1"]
                            }
                        },
                        {
                            "fromSystemDate": "2024-08-01",
                            "params": {
                                "url": "https://www.gs2.be",
                                "initInterval": 345600,
                                "nopInterval": 21600,
                                "maxBuffer": 20000,
                                "queryInterval": 21600,
                                "retryIntervals": [5, 60, 300, 300, 300, 3600, 10800],
                                "retryIntervalsInit": [5, 60, 300, 300, 300, 3600, 10800],
                                "maxMsgSize": 32,
                                "postpone": true,
                                "defaultMaxCalls": 30,
                                "maxCalls": [{"dayOfWeek": "SATURDAY", "hour": 18, "maxCallsInSlot": 0},
                                {"dayOfWeek": "SUNDAY", "hour": 12, "maxCallsInSlot": 100},
```

```
                                {"dayOfWeek": "MONDAY", "hour": 17, "maxCallsInSlot": 0}],
                                "footer": ["myfootertext2", "myfootertext2"]
                            }
                        }
                    ]
                }
            }
        }
    }
]
```

- Query root description
  - **dataResultUrl**: reply endpoint for success - **mandatory**
  - **errorResultUrl**: reply endpoint for errors - **mandatory**
  - **executable**: the core executable query in GraphQL mutation format (for compatibility with GraphQL on the FDM) - **mandatory**
    - **query**: query in JSON GraphQL mutation format or query format - **mandatory**
    - **operationName**: Name of the query - **mandatory**
    - **variables**: to put data for the query & mutation, if needed - **mandatory**
  - All other fields in the root are considered as “echo” and will be copied as is in the reply (in the sample: “actionId” & “messageDateTimeEcho”) - **optional fields in the queries**

12.5.3 FDM reply structure

- Reply sample success:

```
{
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "SET_FDM_LOCK",
    "result": {
        "data": {
            "setFdmLock": {
                "device": {
                    "fdmId": "FOD01000001",
                    "fdmDateTime": "2024-04-18T04:00:05Z",
                    "bufferCapacityUsed": 35.12,
                },
                "configuration": {
                    "paramsVersionObject": {
                        "paramsVersion": "2023-01-05T18:03:23.000Z"
                    },
                    "fdmLockFromDateList": [
                        {
                            "fromSystemDate": "2024-06-01",
                            "fdmLock": false,
                            "lockReason": null
                        },
                        {
                            "fromSystemDate": "2024-07-01",
                            "fdmLock": true,
                            "lockReason": [{"language": "EN", "message": "FDM_LOCKED-EN" },
                            {"language": "NL", "message": "FDM_LOCKED-NL" },
                            {"language": "FR", "message": "FDM_LOCKED-FR" },
```

```
                            {"language": "DE", "message": "FDM_LOCKED-DE" }]
                        }
                    ]
                }
            }
        }
    }
}
```

- Reply sample error:

```
{
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "SET_FDM_LOCK",
    "result": {
        "errors": [
            {
                "message": "A message",
                "extensions": {
                    "fdmId": "FOD01000001",
                    "timestamp": "2024-04-17T04:00:00Z"
                }
            }
        ]
    }
}
```

- Reply description:
  - **actionId**: echo field - **optional (depending on the query)**
  - **messageDateTimeEcho**: echo field - **optional (depending on the query)**
  - **result**: field for data or errors- **mandatory**
    - **data**: field for mutation replies - **mandatory**
      - fields for mutation replies depending on the query - **optional**
    - **errors**: replace data when something wrong occurs during the execution of the query by the FDM.
      - **message**: message sent by FDM - **mandatory**
      - **extensions**: specific fields - **mandatory**
        - **fdmId**: fdmId - **mandatory**
        - **timestamp**: error timestamp - **mandatory**

- GraphQL Type system

```
##### Mutation #####
type Mutation {
  setVatRates(vatFromBookingDateList: [VatRateInput!]!): ConfigurationResult
  setParams(paramsVersionObject: ParamsVersionObjectInput!): ConfigurationResult
  setFdmLock(fdmLockFromDateList: [FdmLockInput!]!): ConfigurationResult
  setPosAllowlist(posIdFromDateList: [PosAllowlistInput!]!): ConfigurationResult
  setTrustCert(trustCertFromDateList: [TrustCertInput!]!): ConfigurationResult
  setUrls(urlsFromDateList: [UrlInput!]!): ConfigurationResult
  setClientCert(clientCertFromDateList: [ClientCertInput!]!): ConfigurationResult
}

##### Query #####
type Query {
```

```
  failedRequests(selectRange: DateTimeRangeInput!): [FailedRequest]
  allFdmInfos: AllFdmInfos!
}

################################################ OUTPUT Fields ##############################################
# The fields for AllFdmInfos
type AllFdmInfos {
  device: Device!
  transactionState: TransactionState!
  taskState: TaskState!
  configuration: Configuration!
}

# The fields for ConfigurationResult
type ConfigurationResult {
    device: Device!
    configuration: Configuration!
}


# The fields for device
type Device {
    fdmId: String!
    fdmSwVersion: String!
    fdmDateTime: String!
    bufferCapacityUsed: Float!
}

# The fields for transactionState
type TransactionState {
    unsentEventCount: Int!          # number of new transactions waiting for upload
    fdmLastSuccessfulTrans: String  # date and time of the last successful upload
    retryCount: Int!                # number of attempts taken to upload the next batch of new transactions
}

# The fields for taskState
type TaskState {
    lastDownload: String # date and time of the last successful download [=fdmLastActions].
}

# The fields for Configuration
type Configuration {
  fdmLockFromDateList: [FdmLock!]
  paramsVersionObject: ParamsVersionObject
  vatFromBookingDateList: [VatRates!]
  posIdFromDateList: [PosAllowlist!]
  trustCertFromDateList: [TrustCert!]!
  urlsFromDateList: [Urls!]!
  clientCertFromDateList: [ClientCert!]!
  localConfig: LocalConfig!
}

# The fields for local config
type LocalConfig {
    ws_ntp3: String
}
```

```
# The fields for paramsVersionObject
type ParamsVersionObject {
    paramsVersion: String!    # date time
    paramsArray: [ParamsFromSystemDate!]
}

# The fields for paramsVersionObject
type ParamsFromSystemDate {
    fromSystemDate: String! # YYYY-MM-DD
    params: Params!
}

type Params {
  url: String!
  initInterval: Integer!
  nopInterval: Integer!
  maxBuffer: Integer!
  queryInterval: Integer!
  retryIntervals: [Integer!]!
  retryIntervalsInit: [Integer!]!
  maxMsgSize: Integer!
  postpone: Boolean!
  defaultMaxCalls: Integer!
  maxCalls: [MaxCalls!]!
  footer: [String!]!
}

type MaxCalls {
  dayOfWeek: WEEKDAY!
  hour: Integer!           # range 0 - 23.
  maxCallsInSlot: Integer! # The maximum number of calls allowed in the specified time slot.
}

enum WEEKDAY {
    MONDAY
    TUESDAY
    WEDNESDAY
    THURSDAY
    FRIDAY
    SATURDAY
    SUNDAY
}

# The fields for posIdFromDateList
type  PosAllowlist
{
    fromSystemDate: String! # YYYY-MM-DD
    posIdList: [String!]!
}

# The fields for urlsFromDateList
type Urls {
    fromSystemDate: String! # YYYY-MM-DD
    urls: [KeyValue!]!
}
type KeyValue {
   key: String!
   value: String!
```

```
}

# The fields for vatFromBookingDateList
type VatRates {
   fromBookingDate: String!  # YYYY-MM-DD
   vatRates: [VatRate!]!
}
type VatRate {
  vatLabel: VatLabel!
  vatRate: float!
  outOfScope: Boolean!
}
enum VatLabel {
  A
  B
  C
  D
  X
}

# The fields for fdmLockFromDateList
type FdmLock {
  fromSystemDate: String # YYYY-MM-DD
  fdmLock: Boolean!
  lockReason: [Message!]
}

type Message {
  language: Language!
  message: String!
}

# The fields for trustCertFromDateList
type TrustCert {
  fromSystemDate: String! # YYYY-MM-DD
  trustCert: [String!]!
}

# The fields for clientCertFromDateList
type ClientCert {
    fromSystemDate: String! # YYYY-MM-DD
    clientCertSn: String!
}


# The fields for failedRequests
see 12.2



############################## All the queries get an enum name so they can be echoed back in the result. #####
enum OperationTypes {
  SET_POS_ALLOWLIST
  SET_VAT_RATES
  SET_URLS
  SET_TRUST_CERT
  SET_PARAMS
  SET_FDM_LOCK
```

```
  SET_CLIENT_CERT
  FAILED_REQUESTS
  ALL_FDM_INFOS
}
```

12.5.4 Replies executed queries

For each query execution, successful or not, the FDM should send a reply to FPS Finance. See section 12.5.3 for the root structure of the reply and in the queries description.

Protocol: mTLS with `certClient`

Verb: POST

URL: `dataResultUrl` or `errorResultUrl`

Possible replies are:

- HTTP code 200: the replies have been processed successfully.
- All other replies indicate an error. Warning `TASK_FEEDBACK_FAILED` should be shown on the linked POS systems at the next possible opportunity. At the next queries cycle, determined by `queryInterval`, the FDM should retry to send the reply of successful queries on this new cycle.

12.5.5 Queries description

12.5.5.1 Mutation: SetFdmLock

This query sets the `fdmLock` boolean, the `fromSystemDate` and the `lockReason` array of `MessageInput`. If `fdmLock` = true, the device is locked and will not accept and sign business transactions anymore and the error `lockReason` message should be shown on the linked POS systems at the next possible opportunity (in accordance with the POS language choice).

The `fromSystemDate` value(s) indicate when the other values in the object are applied. There can be many `fromSystemDate` because it's an array. When the FDM receives this array, it applies the values that are in the object with the `fromSystemDate` that corresponds to the current date. After that, when the FDM reaches another `fromSystemDate`, it applies the values that are in this object.

For example:

"fromSystemDate": "2024-06-01" → object1
"fromSystemDate": "2024-07-01" → object2
The FDM receives this array from the ws_query on “2024-06-10”, so the FDM applies the values in object1 directly after installation.
On "2024-07-01", the FDM applies object2.

Remark: When the FDM receives the mutation, the array **must** contain a value for the current date.

In the previous example, FPS Finance must send an object where the `fromSystemDate` covers the current date.
At the writing date of the query, FPS Finance enters a value for the writing date and values for other dates in the future if necessary.
It is not possible to have this example if the date is 2024-05-19.

When the FDM receives this mutation, it installs the new values in his memory and in case of successful installation (for all `fromSystemDate` values) drops all the old values. The new values replace the old ones.

FPS Finance doesn't require a history of installed values, but the manufacturers are free to create one.

- ~~Json query from ws_query~~
  - ~~GraphQL Type system:~~

```
# +++++ Data in the mutation ++++

# The fields for FDM lock
input FdmLockInput {
  fromSystemDate: String! # YYYY-MM-DD
  fdmLock: Boolean!
  lockReason: [MessageInput!]
}

input MessageInput {
  language: Language!
  message: String!
}

enum Language  {
   EN
   NL
   FR
   DE
}
```

- Query sample:

```
{
   "dataResultUrl": "https://www.fod/data_endpoint",
   "errorResultUrl": "https://www.fod/error_endpoint",
   "actionId": "2023-01-05T18:03:23.000Z",
   "messageDateTimeEcho": "2024-04-17T04:00:00Z",
   "operationType": "SET_FDM_LOCK",
   "executable": {
      "query": "mutation SetFdmLock($fdmLockFromDateList: [FdmLockInput!]!)
                    { setFdmLock(fdmLockFromDateList: $fdmLockFromDateList)
                        {
                            device { fdmId fdmDateTime bufferCapacityUsed }
                            configuration { paramsVersionObject { paramsVersion }
                            fdmLockFromDateList
                                { fromSystemDate fdmLock lockReason { language message } } } } }",
      "operationName": "SetFdmLock",
      "variables": {
        "fdmLockFromDateList": [
            {
                "fromSystemDate": "2024-06-01",
                "fdmLock": false,
                "lockReason": null
            },
            {
                "fromSystemDate": "2024-07-01",
                "fdmLock": true,
                "lockReason": [{"language": "EN", "message": "FDM_LOCKED-EN" },
                    {"language": "NL", "message": "FDM_LOCKED-NL" },
                    {"language": "FR", "message": "FDM_LOCKED-FR" },
                    {"language": "DE", "message": "FDM_LOCKED-DE" }]
            }
        ]
      }
   }
}
```

- Query description:

- **query**: `"mutation SetFdmLock($fdmLockFromDateList: [FdmLockInput!]!) { setFdmLock(fdmLockFromDateList: $fdmLockFromDateList) { device { fdmId fdmDateTime bufferCapacityUsed } configuration { paramsVersionObject { paramsVersion } fdmLockFromDateList {fromSystemDate fdmLock lockReason { language message } } } } }"`, graphQL query (**$fdmLockFromDateList** for input and **ConfigurationResult** for reply)
  - **operationName**: mutation name
  - **variables**:
    - **fdmLockFromDate**: `FdmLockInput` array.

- Json reply from FDM: (to `dataResultUrl` endpoint)
  - Reply sample:

```
{
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "SET_FDM_LOCK",
    "result": {
        "data": {
            "setFdmLock": {
                "device": {
                    "fdmId": "FOD01000001",
                    "fdmDateTime": "2024-04-18T04:00:05Z",
                    "bufferCapacityUsed": 35.12,
                },
                "configuration": {
                    "paramsVersionObject": {
                        "paramsVersion": "2023-01-05T18:03:23.000Z"
                    },
                    "fdmLockFromDateList": [
                        {
                            "fromSystemDate": "2024-06-01",
                            "fdmLock": false,
                            "lockReason": null
                        },
                        {
                            "fromSystemDate": "2024-07-01",
                            "fdmLock": true,
                            "lockReason": [{"language": "EN", "message": "FDM_LOCKED-EN" },
                                {"language": "NL", "message": "FDM_LOCKED-NL" },
                                {"language": "FR", "message": "FDM_LOCKED-FR" },
                                {"language": "DE", "message": "FDM_LOCKED-DE" }]
                        }
                    ]
                }
            }
        }
    }
}
```

  - Reply description:
    - **result**:
      - **data**:
        - **ConfigurationResult fields**: generic fields for mutations replies - **mandatory (see 12.5.3 Type system)**

    If the query fails, the eventual previous values remain valid.

12.5.5.2 Mutation: SetParams

The SetParams query is sent to the FDM to install the latest configuration parameters. The data of the query contains the latest timestamp version of the parameters (`paramsVersion`) and all current parameters.

The `fromSystemDate` value(s) indicate when the others values in the object are applied. There can be many `fromSystemDate` because it's an array. When the FDM receives this array, it applies the values that are in the object with the current `fromSystemDate` . When the FDM reaches a `fromSystemDate` , it applies the values that are in this object.

For example:

"fromSystemDate": "2024-06-01" → object1

"fromSystemDate": "2024-07-01" → object2

The FDM receives this array from the ws_query on “2024-06-10”, so the FDM applies the values in object1.

On "2024-07-01", the FDM applies object2.

Remark: When the FDM receives the mutation, the array must contain a value for the current date.

When the FDM receives this mutation, it installs the new values and drops all the old values. The new values replace the old ones.

- ~~Json query from ws_query~~
  - ~~GraphQL Type system:~~

```
# The fields for paramsVersionObject
input ParamsVersionObjectInput {
    paramsVersion: String!    # date time
    paramsArray: [ParamsFromSystemDateInput!]
}

input ParamsFromSystemDateInput {
    fromSystemDate: String! # YYYY-MM-DD
    params: ParamsInput!
}

input ParamsInput {
  url: String!
  initInterval: Integer!
  nopInterval: Integer!
  maxBuffer: Integer!
  queryInterval: Integer!
  retryIntervals: [Integer!]!
  retryIntervalsInit: [Integer!]!
  maxMsgSize: Integer!
  postpone: Boolean!
  defaultMaxCalls: Integer!
  maxCalls: [MaxCallsInput!]!
  footer: [String!]!
}

enum WEEKDAY {
    MONDAY
    TUESDAY
    WEDNESDAY
    THURSDAY
    FRIDAY
    SATURDAY
    SUNDAY
}

input MaxCallsInput {
  dayOfWeek: WEEKDAY!
```

```
  hour: Integer! # range 0 - 23.
  maxCallsInSlot: Integer! # The maximum number of calls allowed in the specified time slot.
}
```

- ~~Query sample:~~

```
{
   "dataResultUrl": "https://www.fod/data_endpoint",
   "errorResultUrl": "https://www.fod/error_endpoint",
   "actionId": "2023-01-05T18:03:23.000Z",
   "messageDateTimeEcho": "2024-04-17T04:00:00Z",
   "operationType": "SET_PARAMS",
   "executable": {
      "query": "mutation SetParams($paramsVersionObject: ParamsVersionObjectInput!)
                    { setParams(paramsVersionObject: $paramsVersionObject)
                        {
                            device { fdmId fdmDateTime bufferCapacityUsed }
                            configuration { paramsVersionObject { paramsVersion }
                            fdmLockFromDateList
                                {fromSystemDate fdmLock lockReason { language message } } } } }",
      "operationName": "SetParams",
      "variables": {
        "paramsVersionObject": {
            "paramsVersion": "2023-01-05T18:03:23.000Z",
            "paramsArray": [
                {
                    "fromSystemDate": "2024-07-01",
                    "params": {
                        "url": "https://www.gs2.be",
                        "initInterval": 345600,
                        "nopInterval": 21600,
                        "maxBuffer": 20000,
                        "queryInterval": 21600,
                        "retryIntervals": [5, 60, 300, 300, 300, 3600, 10800],
                        "retryIntervalsInit": [5, 60, 300, 300, 300, 3600, 10800],
                        "maxMsgSize": 32,
                        "postpone": true,
                        "defaultMaxCalls": 30,
                        "maxCalls": [{"dayOfWeek": "SATURDAY", "hour": 18, "maxCallsInSlot": 0},
                            {"dayOfWeek": "SUNDAY", "hour": 12, "maxCallsInSlot": 100},
                            {"dayOfWeek": "MONDAY", "hour": 17, "maxCallsInSlot": 0}],
                        "footer": ["myfootertext1", "myfootertext1"]
                    }
                },
                {
                    "fromSystemDate": "2024-08-01",
                    "params": {
                        "url": "https://www.gs2.be",
                        "initInterval": 345600,
                        "nopInterval": 21600,
                        "maxBuffer": 20000,
                        "queryInterval": 21600,
                        "retryIntervals": [5, 60, 300, 300, 300, 3600, 10800],
                        "retryIntervalsInit": [5, 60, 300, 300, 300, 3600, 10800],
                        "maxMsgSize": 32,
                        "postpone": true,
                        "defaultMaxCalls": 30,
```

```
                        "maxCalls": [{"dayOfWeek": "SATURDAY", "hour": 18, "maxCallsInSlot": 0},
                            {"dayOfWeek": "SUNDAY", "hour": 12, "maxCallsInSlot": 100},
                            {"dayOfWeek": "MONDAY", "hour": 17, "maxCallsInSlot": 0}],
                        "footer": ["myfootertext2", "myfootertext2"]
                    }
                }
            ]
        }
      }
   }
}
```

- ~~Query description:~~
  - **query**: `"mutation SetParams($paramsVersionObject: ParamsVersionObjectInput!) { setParams(paramsVersionObject: $paramsVersionObject) { device { fdmId fdmDateTime bufferCapacityUsed } configuration { paramsVersionObject { paramsVersion } fdmLockFromDateList {fromSystemDate fdmLock lockReason { language message } } } } }"`, graphQL query (**$paramsVersionObject** for input and **ConfigurationResult** for reply)
    - **operationName**: mutation name
    - **variables**:
      - **paramsVersionObject**: `ParamsVersionObjectInput` If this object is empty or absent, the FDM must ignore this mutation.

- ~~Json reply from FDM:~~ (to `dataResultUrl` endpoint)
  - ~~Reply sample:~~

```
{
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "SET_PARAMS",
    "result": {
        "data": {
            "setParams": {
                "device": {
                    "fdmId": "FOD01000001",
                    "fdmDateTime": "2024-04-18T04:00:05Z",
                    "bufferCapacityUsed": 35.12,
                },
                "configuration": {
                    "paramsVersionObject": {
                        "paramsVersion": "2023-01-05T18:03:23.000Z"
                    },
                    "fdmLockFromDateList": [
                        {
                            "fromSystemDate": "2024-06-01",
                            "fdmLock": false,
                            "lockReason": null
                        },
                        {
                            "fromSystemDate": "2024-07-01",
                            "fdmLock": true,
                            "lockReason": [{"language": "EN", "message": "FDM_LOCKED-EN" },
                                {"language": "NL", "message": "FDM_LOCKED-NL" },
                                {"language": "FR", "message": "FDM_LOCKED-FR" },
                                {"language": "DE", "message": "FDM_LOCKED-DE" }]
                        }
                    ]
```

```
                }
            }
        }
    }
}
```

- Reply description:
  - **result**:
    - **data**:
      - **ConfigurationResult fields**: generic fields for mutation replies - **mandatory (see 12.5.3 Type system)**

    The query was successful and the parameters are updated.
    If the execution of the query was unsuccessful, the eventual previous parameter values stored in the FDM remain valid. Warning `UPDATE_PARAMS_FAILED` should be shown on the linked POS systems at the next possible opportunity.

12.5.5.3 Mutation: SetVatRates

This query sends the latest VAT rates to the FDM.

The `fromBookingDate` value(s) indicate when the other values in the object are applied. There can be many `fromBookingDate` because it's an array. When the FDM receives this array, it applies the values that are in the object with the current `fromBookingDate`. When the FDM reaches a `fromBookingDate`, it applies the values that are in this object.

For example:

"fromBookingDate": "2024-06-01" → object1
"fromBookingDate": "2024-07-01" → object2
The FDM receives this array from the ws_query on "2024-06-10", so the FDM applies the values in object1.
On "2024-07-01", the FDM applies object2.

Remark: When the FDM receives the mutation, the array **must** contain a value for the current date.

When the FDM receives this mutation, it installs the new values and drops all the old values. The new values replace the old ones.

- Json query from ws_query
  - GraphQL Type system:

```
# +++++ Data in the mutation ++++

# The fields for a VAT rate
input VatRatesInput {
   fromBookingDate: String!  # YYYY-MM-DD
   vatRates: [VatRateInput!]!
}

input VatRateInput {
  vatLabel: VatLabel!
  vatRate: float!
  outOfScope: Boolean!
}

enum VatLabel {
  A
  B
  C
  D
  X
}

```

- ~~Query sample:~~

```
{
   "dataResultUrl": "https://www.fod/data_endpoint",
   "errorResultUrl": "https://www.fod/error_endpoint",
   "actionId": "2023-01-05T18:03:23.000Z",
   "messageDateTimeEcho": "2024-04-17T04:00:00Z",
   "operationType": "SET_VAT_RATES",
   "executable": {
      "query": "mutation SetVatRates($vatFromBookingDateList: [VatRatesInput!]!)
                { setVatRates(vatFromBookingDateList: $vatFromBookingDateList)
                    {
                        device { fdmId fdmDateTime bufferCapacityUsed }
                        configuration { paramsVersionObject { paramsVersion }
                        fdmLockFromDateList
                            {fromSystemDate fdmLock lockReason { language message } } } } }",
      "operationName": "SetVatRates",
      "variables": {
         "vatFromBookingDateList": [
            {
               "fromBookingDate": "2020-01-01",
               "vatRates": [
                  {
                     "vatLabel": "A",
                     "vatRate": 21,
                     "outOfScope": false
                  },
                  {
                     "vatLabel": "B",
                     "vatRate": 12,
                     "outOfScope": false
                  },
                  {
                     "vatLabel": "C",
                     "vatRate": 6,
                     "outOfScope": false
                  },
                  {
                     "vatLabel": "D",
                     "vatRate": 0,
                     "outOfScope": false
                  },
                  {
                     "vatLabel": "X",
                     "vatRate": 0,
                     "outOfScope": true
                  }
               ]
            },
            {
               "fromBookingDate": "2025-04-01",
               "vatRates": [
                  {
                     "vatLabel": "A",
                     "vatRate": 21,
                     "outOfScope": false
                  },
```

```
                {
                   "vatLabel": "B",
                   "vatRate": 17,
                   "outOfScope": false
                },
                {
                   "vatLabel": "C",
                   "vatRate": 6,
                   "outOfScope": false
                },
                {
                   "vatLabel": "D",
                   "vatRate": 0,
                   "outOfScope": false
                },
                {
                   "vatLabel": "X",
                   "vatRate": 0,
                   "outOfScope": true
                }
             ]
          }
       ]
    }
 }
}
```


- ~~Query description:~~
  - **query**: `"mutation SetVatRates($vatFromBookingDateList: [VatRatesInput!]!) { setVatRates(vatFromBookingDateList: $vatFromBookingDateList) { device { fdmId fdmDateTime bufferCapacityUsed } configuration { paramsVersionObject { paramsVersion } fdmLockFromDateList {fromSystemDate fdmLock lockReason { language message } } } } }"` graphQL query (**$vatFromBookingDateList** for input and **ConfigurationResult** for reply)
    - **operationName**: mutation name
    - **variables**:
      - **vatFromBookingDateList**: VatRatesInput array. If this array is empty or absent, the FDM must ignore this mutation,

    The FDM should store the new or modified VAT rates.

- ~~Json reply from FDM:~~ (to `dataResultUrl` endpoint)
  - ~~Reply sample:~~


```
{
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "SET_VAT_RATES",
    "result": {
        "data": {
            "setVatRates": {
                "device": {
                    "fdmId": "FOD01000001",
                    "fdmDateTime": "2024-04-18T04:00:05Z",
                    "bufferCapacityUsed": 35.12,
                },
                "configuration": {
                    "paramsVersionObject": {
                        "paramsVersion": "2023-01-05T18:03:23.000Z"
```

```
                    },
                    "fdmLockFromDateList": [
                        {
                            "fromSystemDate": "2024-06-01",
                            "fdmLock": false,
                            "lockReason": null
                        },
                        {
                            "fromSystemDate": "2024-07-01",
                            "fdmLock": true,
                            "lockReason": [{"language": "EN", "message": "FDM_LOCKED-EN" },
                                {"language": "NL", "message": "FDM_LOCKED-NL" },
                                {"language": "FR", "message": "FDM_LOCKED-FR" },
                                {"language": "DE", "message": "FDM_LOCKED-DE" }]
                        }
                    ]
                }
            }
        }
    }
}
```

- ~~Reply description:~~
  - **result**:
    - **data**:
      - **ConfigurationResult fields**: generic fields for mutation replies - **mandatory (see 12.5.3 Type system)**

    If the execution of the query fails, the eventual previous values in `vatList` remain valid. Warning `UPDATE_VAT_RATES_FAILED` should be shown on the linked POS systems at the next possible opportunity.

12.5.5.4 Mutation: SetPosAllowlist

This query sets the `posIdList`.

The `fromSystemDate` value(s) indicate when the other values in the object are applied. There can be many `fromSystemDate` because it's an array. When the FDM receives this array, it applies the values that are in the object with the current `fromSystemDate`. When the FDM reaches a `fromSystemDate`, it applies the values that are in this object.

For example:

"fromSystemDate": "2024-06-01" → object1

"fromSystemDate": "2024-07-01" → object2

The FDM receives this array from the ws_query on “2024-06-10”, so the FDM applies the values in object1.

On "2024-07-01", the FDM applies object2.

Remark: When the FDM receives the mutation, the array **must** contain a value for the current date.

When the FDM receives this mutation, it installs the new values and drops all the old values. The new values replace the old ones.

- ~~Json query from ws_query~~
  - ~~GraphQL Type system:~~

```
# The fields for posIdFromDateList
Input PosAllowlistInput
{
    fromSystemDate: String! # YYYY-MM-DD
    posIdList: [String!]!
}
```

- ~~Query sample:~~

```
{
  "dataResultUrl": "https://www.fod/data_endpoint",
  "errorResultUrl": "https://www.fod/error_endpoint",
  "actionId": "2023-01-05T18:03:23.000Z",
  "messageDateTimeEcho": "2024-04-17T04:00:00Z",
  "operationType": "SET_POS_ALLOWLIST",
  "executable": {
     "query": "mutation SetPosAllowlist($posIdFromDateList: [PosAllowlistInput!]!)
                { setPosAllowlist (posIdFromDateList: $posIdFromDateList)
                    {
                        device { fdmId fdmDateTime bufferCapacityUsed }
                        configuration { paramsVersionObject { paramsVersion }
                        fdmLockFromDateList {fromSystemDate fdmLock lockReason { language message } }
                        posIdFromDateList { fromSystemDate posIdList } } } }",
     "operationName": "SetPosAllowlist",
     "variables": {
        "posIdFromDateList": [
            {
                "fromSystemDate": "2024-07-01",
                "posIdList": ["Csnpos00000001", "Csnpos00000003"]
            },
            {
                "fromSystemDate": "2024-08-01",
                "posIdList": ["Csnpos00000001", "Csnpos00000003", "Csnpos00000004"]
            }
        ]
     }
  }
}
```

- ~~Query description:~~
  - **query**: `"mutation SetPosAllowlist($posIdFromDateList: [PosAllowlistInput!]!) { setPosAllowlist (posIdFromDateList: $posIdFromDateList) {device { fdmId fdmDateTime bufferCapacityUsed } configuration { paramsVersionObject { paramsVersion } fdmLockFromDateList {fromSystemDate fdmLock lockReason { language message } } posIdFromDateList { fromSystemDate posIdList } } } }"` graphQL query (**$posIdFromDateList** for input and **ConfigurationResult** for reply)
    - **operationName**: mutation name
    - **variables**:
      - **posIdFromDateList**: array of PosAllowlistInput.

- ~~Json reply from FDM:~~ (to `dataResultUrl` endpoint)
  - ~~Reply sample:~~

```
{
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "SET_POS_ALLOWLIST",
    "result": {
        "data": {
            "setPosAllowlist": {
                "device": {
                    "fdmId": "FOD01000001",
                    "fdmDateTime": "2024-04-18T04:00:05Z",
```

```
                    "bufferCapacityUsed": 35.12,
                },
                "configuration": {
                    "paramsVersionObject": {
                        "paramsVersion": "2023-01-05T18:03:23.000Z"
                    },
                    "fdmLockFromDateList": [
                        {
                            "fromSystemDate": "2024-06-01",
                            "fdmLock": false,
                            "lockReason": null
                        },
                        {
                            "fromSystemDate": "2024-07-01",
                            "fdmLock": true,
                            "lockReason": [{"language": "EN", "message": "FDM_LOCKED-EN" },
                                {"language": "NL", "message": "FDM_LOCKED-NL" },
                                {"language": "FR", "message": "FDM_LOCKED-FR" },
                                {"language": "DE", "message": "FDM_LOCKED-DE" }]
                        }
                    ],
                    "posIdFromDateList": [
                        {
                            "fromSystemDate": "2024-07-01",
                            "posIdList": ["Csnpos00000001", "Csnpos00000003"]
                        },
                        {
                            "fromSystemDate": "2024-08-01",
                            "posIdList": ["Csnpos00000001", "Csnpos00000003", "Csnpos00000004"]
                        }
                    ]
                }
            }
        }
    }
}
```

- ~~Reply description:~~
  - **result**:
    - **data**:
      - **ConfigurationResult fields**: generic fields for mutation replies - **mandatory (see 12.5.3 Type system)**

    If the execution of the query fails, the eventual previous values in `posIdFromDateList` remain valid. Warning `UPDATE_POS_ALLOWLIST_FAILED` should be shown on the linked POS systems at next possible opportunity.

12.5.5.5 Mutation: SetTrustCert

This query provides the chain of trust certificates which allow the FDM to resolve the server certificate from FPS Finance.

The `fromSystemDate` value(s) indicate when the other values in the object are applied. There can be many `fromSystemDate` because it's an array. When the FDM receives this array, it applies the values that are in the object with the current `fromSystemDate`. When the FDM reaches a `fromSystemDate`, it applies the values that are in this object.

For example:

"fromSystemDate": "2024-06-01" → object1

"fromSystemDate": "2024-07-01" → object2

The FDM receive this array from the ws_query on “2024-06-10”, so the FDM applies the values in object1.

On "2024-07-01", the FDM applies object2.

Remark: When the FDM receives the mutation, the array **must** contain a value for the current date.

When the FDM receives this mutation, it installs the new values and drops all the old values. The new values replace the old ones.

- ~~Json query from ws_query~~
  - ~~GraphQL Type system:~~

```
# +++++ Data in the mutation ++++

# The fields for TrustCertInput
input TrustCertInput {
    fromSystemDate: String! # YYYY-MM-DD
    trustCert: [String!]!
}
```

  - ~~Query sample:~~

```
{
   "dataResultUrl": "https://www.fod/data_endpoint",
   "errorResultUrl": "https://www.fod/error_endpoint",
   "actionId": "2023-01-05T18:03:23.000Z",
   "messageDateTimeEcho": "2024-04-17T04:00:00Z",
   "operationType": "SET_TRUST_CERT",
   "executable": {
      "query": "mutation SetTrustCert($trustCertFromDateList: [TrustCertInput!]!)
                    { setTrustCert(trustCertFromDateList: $trustCertFromDateList)
                        {
                            device { fdmId fdmDateTime bufferCapacityUsed }
                            configuration { paramsVersionObject { paramsVersion }
                            fdmLockFromDateList
                                {fromSystemDate fdmLock lockReason { language message } } } } }",
      "operationName": "SetTrustCert",
      "variables": {
        "trustCertFromDateList": [
            {
                "fromSystemDate": "2024-07-01",
                "trustCert": [ "cert1x509pem", "cert2x509pem", "cert3x509pem" ]
            },
            {
                "fromSystemDate": "2024-08-01",
                "trustCert": [ "cert1x509pem2", "cert2x509pem2", "cert3x509pem2" ]
            }
        ]
      }
   }
}
```

  - ~~Query description:~~
    - **query**: `"mutation SetTrustCert($trustCertFromDateList: [TrustCertInput!]!) { setTrustCert(trustCertFromDateList: $trustCertFromDateList) { device { fdmId fdmDateTime bufferCapacityUsed } configuration { paramsVersionObject { paramsVersion } fdmLockFromDateList {fromSystemDate fdmLock lockReason { language message } } } } }"` graphQL query (**$trustCertFromDateList** for input and **ConfigurationResult** for reply)
      - **operationName**: mutation name
      - **variables**:
        - **trustCertFromDateList**: array of `TrustCertInput`. If this array is empty or absent, the FDM must ignore this mutation,

The query `trustCert` contains a list of X.509 certificates in PEM format. The certificates should be stored in the FDM's certificate store `certstoreTrust`. The FDM may delete all other trust certificates from the certificate store `certstoreTrust`.

- ~~Json reply from FDM:~~ (to `dataResultUrl` endpoint)
  - ~~Reply sample:~~

```
{
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "SET_TRUST_CERT",
    "result": {
        "data": {
            "setTrustCert": {
                "device": {
                    "fdmId": "FOD01000001",
                    "fdmDateTime": "2024-04-18T04:00:05Z",
                    "bufferCapacityUsed": 35.12,
                },
                "configuration": {
                    "paramsVersionObject": {
                        "paramsVersion": "2023-01-05T18:03:23.000Z"
                    },
                    "fdmLockFromDateList": [
                        {
                            "fromSystemDate": "2024-06-01",
                            "fdmLock": false,
                            "lockReason": null
                        },
                        {
                            "fromSystemDate": "2024-07-01",
                            "fdmLock": true,
                            "lockReason": [{"language": "EN", "message": "FDM_LOCKED-EN" },
                                {"language": "NL", "message": "FDM_LOCKED-NL" },
                                {"language": "FR", "message": "FDM_LOCKED-FR" },
                                {"language": "DE", "message": "FDM_LOCKED-DE" }]
                        }
                    ]
                }
            }
        }
    }
}
```

  - ~~Reply description:~~
    - **result**:
      - **data**:
        - **ConfigurationResult fields**: generic fields for mutation replies - **mandatory (see 12.5.3 Type system)**

    If the execution of the query fails, the eventual previous `certstoreTrust` remains unchanged. Warning `UPDATE_TRUST_CERT_FAILED` should be shown on the linked POS systems at next possible opportunity.

12.5.5.6 Mutation: SetUrls

The SetUrls query is used to install the latest URLs of the different webservices.

The `fromSystemDate` value(s) indicate when the others values in the object are applied. There can be many `fromSystemDate` because it's an array. When the FDM receives this array, it applies the values that are in the object with the current `fromSystemDate`. When the FDM reaches a `fromSystemDate`, it applies the values that are in this object.

For example:

"fromSystemDate": "2024-06-01" → object1

"fromSystemDate": "2024-07-01" → object2

The FDM receives this array from the ws_query on “2024-06-10”, so the FDM applies the values in object1.

On "2024-07-01", the FDM applies object2.

Remark: When the FDM receives the mutation, the array **must** contain a value for the current date.

When the FDM receives this mutation, it installs the new values and drops all the old values. The new values replace the old ones.

- ~~Json query from ws_query~~
  - ~~GraphQL Type system:~~

```
# The fields for a URLs values
input UrlInput {
    fromSystemDate: String! # YYYY-MM-DD
    urls: [KeyValueInput!]!
}

input KeyValueInput {
   key: String!
   value: String!
}
```

  - ~~Query sample:~~

```
{
   "dataResultUrl": "https://www.fod/data_endpoint",
   "errorResultUrl": "https://www.fod/error_endpoint",
   "actionId": "2023-01-05T18:03:23.000Z",
   "messageDateTimeEcho": "2024-04-17T04:00:00Z",
   "operationType": "SET_URLS",
   "executable": {
      "query": "mutation SetUrls($urlsFromDateList: [UrlInput!]!)
                    { setUrls(urlsFromDateList: $urlsFromDateList)
                        {
                            device { fdmId fdmDateTime bufferCapacityUsed }
                            configuration { paramsVersionObject { paramsVersion }
                            fdmLockFromDateList
                                {fromSystemDate fdmLock lockReason { language message } } } } }",
      "operationName": "SetUrls",
      "variables": {
        "urlsFromDateList": [
            {
                "fromSystemDate": "2024-07-01",
                "urls": [
                    { "key": "ws_reg", "value": "https://myurl:8080" },
                    { "key": "ws_nop", "value": "https://myurl:8080" },
                    { "key": "ws_query", "value": "https://myurl:8080" },
                    { "key": "ws_init", "value": "https://myurl:8080" },
                    { "key": "ws_ntp1", "value": "https://myurl:8080" },
                    { "key": "ws_ntp2", "value": "https://myurl:8080" }
                ]
            },
```

```
            {
                "fromSystemDate": "2024-08-01",
                "urls": [
                    { "key": "ws_reg", "value": "https://myurl2:8080" },
                    { "key": "ws_nop", "value": "https://myurl2:8080" },
                    { "key": "ws_query", "value": "https://myurl2:8080" },
                    { "key": "ws_init", "value": "https://myurl2:8080" },
                    { "key": "ws_ntp1", "value": "https://myurl2:8080" },
                    { "key": "ws_ntp2", "value": "https://myurl2:8080" }
                ]
            }
        ]
      }
   }
}
```

- ~~Query description:~~
  - **query**: `"mutation SetUrls($urlsFromDateList: [UrlInput!]!) { setUrls(urlsFromDateList: $urlsFromDateList) { device { fdmId fdmDateTime bufferCapacityUsed } configuration { paramsVersionObject { paramsVersion } fdmLockFromDateList {fromSystemDate fdmLock lockReason { language message } } } } }"` graphQL query (**$urlsFromDateList** for input and **ConfigurationResult** for reply)
    - **operationName**: mutation name
    - **variables**:
      - **urlsFromDateList**: UrlInput array. If this array is empty or absent, the FDM must ignore this mutation,

- ~~Json reply from FDM:~~ (to `dataResultUrl` endpoint)
  - ~~Reply sample:~~

```
{
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "SET_URLS",
    "result": {
        "data": {
            "setUrls": {
                "device": {
                    "fdmId": "FOD01000001",
                    "fdmDateTime": "2024-04-18T04:00:05Z",
                    "bufferCapacityUsed": 35.12,
                },
                "configuration": {
                    "paramsVersionObject": {
                        "paramsVersion": "2023-01-05T18:03:23.000Z"
                    },
                    "fdmLockFromDateList": [
                        {
                            "fromSystemDate": "2024-06-01",
                            "fdmLock": false,
                            "lockReason": null
                        },
                        {
                            "fromSystemDate": "2024-07-01",
                            "fdmLock": true,
                            "lockReason": [{"language": "EN", "message": "FDM_LOCKED-EN" },
                                {"language": "NL", "message": "FDM_LOCKED-NL" },
                                {"language": "FR", "message": "FDM_LOCKED-FR" },
```

```
                                {"language": "DE", "message": "FDM_LOCKED-DE" }]
                        }
                    ]
                }
            }
        }
    }
}
```

- ~~Reply description:~~
  - **result**:
    - **data:**
      - **ConfigurationResult fields**: generic fields for mutations replies - **mandatory (see 12.5.3 Type system)**

    All other replies: the update failed. The eventual previous values remain valid. Warning `UPDATE_URLS_FAILED` should be shown on the linked POS systems at the next possible opportunity.
    If the execution of the query fails, the eventual previous values remain unchanged. Warning `UPDATE_URLS_FAILED` should be shown on the linked POS systems at the next possible opportunity.

12.5.5.7 Mutation: SetClientCert

The `SetClientCert` query is used to renew an existing operational client certificate. The renewal involves 2 steps: recollection (in the query) and validation of the new client certificate (with the reply).

The `fromSystemDate` value(s) indicate when the other values in the object are applied. There can be many `fromSystemDate` because it's an array. When the FDM receives this array, it applies the values that are in the object with the current `fromSystemDate`. When the FDM reaches a `fromSystemDate`, it applies the values that are in this object.

For example:
"fromSystemDate": "2024-06-01" → object1
"fromSystemDate": "2024-07-01" → object2
The FDM receives this array from the ws_query on "2024-06-10", so the FDM applies the values in object1.
On "2024-07-01", the FDM applies object2.

Remark: When the FDM receives the mutation, the array **must** contain a value for the current date.

When the FDM receives this mutation, it installs the new values (after step 2, with positive reply recieved) and drops all the old values. The new values replace the old ones.

- ~~Json query from ws_query~~
  - ~~GraphQL Type system:~~

```graphql
# +++++ Data in the mutation ++++

# The fields for ClientCertInput
input ClientCertInput {
    fromSystemDate: String! # YYYY-MM-DD
    clientCert: String!
}
```

  - ~~Query sample:~~

```
{
    "dataResultUrl": "https://www.fod/data_endpoint",
    "errorResultUrl": "https://www.fod/error_endpoint",
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "SET_CLIENT_CERT",
```

```
  "executable": {
     "query": "mutation SetClientCert($clientCertFromDateList: [ClientCertInput!]!)
                  { setClientCert(clientCertFromDateList: $clientCertFromDateList)
                      {
                          device { fdmId fdmDateTime bufferCapacityUsed }
                          configuration { paramsVersionObject { paramsVersion }
                          fdmLockFromDateList
                              {fromSystemDate fdmLock lockReason { language message } }
                          clientCertFromDateList { fromSystemDate clientCertSn } } } }",
     "operationName": "SetClientCert",
     "variables": {
       "clientCertFromDateList": [
           {
               "fromSystemDate": "2024-07-01",
               "clientCert": "pkcs12pem-with-password"
           },
           {
               "fromSystemDate": "2024-08-01",
               "clientCert": "pkcs12pem2-with-password"
           }
       ]
     }
  }
}
```

- ~~Query description:~~
  - **query**: `"mutation SetClientCert($clientCertFromDateList: [ClientCertInput!]!) { setClientCert(clientCertFromDateList: $clientCertFromDateList) { device { fdmId fdmDateTime bufferCapacityUsed } configuration { paramsVersionObject { paramsVersion } fdmLockFromDateList {fromSystemDate fdmLock lockReason { language message } } clientCertFromDateList { fromSystemDate clientCertSn } } } }"` graphQL query (**$clientCertFromDateList** for input and **ConfigurationResult** for reply)
    - **operationName**: mutation name
    - **variables**:
      - **clientCertFromDateList**: `clientCertFromDateList` array with object `clientCert` containing the client certificate and the private key (encryption password = hash of the `fdmHkey` private key). If this array is empty or absent, the FDM must ignore this mutation.

- ~~Json reply from FDM:~~ (to `dataResultUrl` endpoint)
  - **Step 1**: The FDM can import the candidate client certificate(s) and private key(s) using the shared secret. The FDM should however not remove its operational client certificate.
  - **Step 2**: In this step, the FDM performs a confirmation call with the candidate (depending on the right `fromSystemDate` with the current date) client certificate from the previous step.
    - Protocol: mTLS using candidate client certificate from the previous step
    - Verb: GET
    - URL: `"dataResultUrl"`
    - Possible outcomes are:
      - The FDM receives a positive reply (HTTP code 200) → The FDM should promote the candidate client certificate to the new operational client certificate.
      - The FDM receives a negative reply (HTTP code 4XX or 5XX) → The confirmation failed. The FDM can't promote the client certificate. The renewal procedure is a failure.
  - ~~Reply sample:~~

```
{
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "SET_CLIENT_CERT",
    "result": {
        "data": {
            "setClientCert": {
                "device": {
                    "fdmId": "FOD01000001",
                    "fdmDateTime": "2024-04-18T04:00:05Z",
                    "bufferCapacityUsed": 35.12,
                },
                "configuration": {
                    "paramsVersionObject": {
                        "paramsVersion": "2023-01-05T18:03:23.000Z"
                    },
                    "fdmLockFromDateList": [
                        {
                            "fromSystemDate": "2024-06-01",
                            "fdmLock": false,
                            "lockReason": null
                        },
                        {
                            "fromSystemDate": "2024-07-01",
                            "fdmLock": true,
                            "lockReason": [{"language": "EN", "message": "FDM_LOCKED-EN" },
                                {"language": "NL", "message": "FDM_LOCKED-NL" },
                                {"language": "FR", "message": "FDM_LOCKED-FR" },
                                {"language": "DE", "message": "FDM_LOCKED-DE" }]
                        }
                    ],
                    "clientCertFromDateList": [
                        {
                            "fromSystemDate": "2024-07-01",
                            "clientCertSn": "dhjs76djs"
                        },
                        {
                            "fromSystemDate": "2024-08-01",
                            "clientCertSn": "dhjs76djs2"
                        }
                    ]
                }
            }
        }
    }
}
```


- ~~Reply description:~~
  - **result**:
    - **data**:
      - **ConfigurationResult fields**: generic fields for mutation replies - **mandatory (see 12.5.3 Type system)**
        - **clientCertSn**: serial number of the candidate client certificate

    If the query was unsuccessful, warning `UPDATE_CLIENT_CERT_FAILED` should be shown on the linked POS systems at next possible opportunity.

12.5.5.8 Query: FailedRequests

This query requests the failed requests error log (see 12.2 Error logging).

- ~~Json query from ws_query~~
  - ~~GraphQL Type system:~~

```
#### Input fields for Failed Requests

input DateTimeRangeInput {
  from: String!            # "2024-04-09T04:00:00Z" for example
  to: String!              # "2024-04-09T04:00:00Z" for example
  maxItemCount: Integer!   # maximum items that will be send
                           #   (last maxItemCount for the range)
}

```

  - ~~Query sample:~~

```
{
    "dataResultUrl": "https://www.fod/data_endpoint",
    "errorResultUrl": "https://www.fod/error_endpoint",
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "FAILED_REQUESTS",
    "executable": {
        "query": "query FailedRequests($selectRange: DateTimeRangeInput!)
                    { failedRequests(selectRange: $selectRange)
                        { timeStamp requestMethod requestUri requestHeaders
                        requestBody replyStatusCode replyHeaders replyBody errorMessage } }",
        "operationName": "FailedRequests",
        "variables": {
            "selectRange": {
                "from": "2024-04-17T00:00:00Z",
                "to": "2024-04-18T00:00:00Z",
                "maxItemCount": 30
            }
        }
    }
}
```

- ~~Query description:~~
  - **query**: `"query FailedRequests($selectRange: DateTimeRangeInput!) { failedRequests(selectRange: $selectRange) { {timeStamp requestMethod requestUri requestHeaders requestBody replyStatusCode replyHeaders replyBody errorMessage } }"` graphQL query (**$selectRange** for input and [**FailedRequest]** for reply)
    - **operationName**: query name
    - **variables**:
      - **selectRange**: selectRange object
        - Remark: the result is limited to the last `maxItemCount` items in the range
- ~~Json reply from FDM:~~ (to `dataResultUrl` endpoint)
  - ~~Reply samples:~~

```
{
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "FAILED_REQUESTS",
    "result": {
```

```
        "data": {
            "failedRequests": [
                {
                    "timeStamp": "2024-04-18T04:00:05Z",
                    "requestMethod": "request Method",
                    "requestUri": "request Uri",
                    "requestHeaders": "request Headers",
                    "requestBody": "request Body",
                    "replyStatusCode": 500,
                    "replyHeaders": "reply Headers",
                    "replyBody": "reply Body",
                    "errorMessage": "error"
                },
                {
                    "timeStamp": "2024-04-20T04:00:05Z",
                    "requestMethod": "request Method",
                    "requestUri": "request Uri",
                    "requestHeaders": "request Headers",
                    "requestBody": "request Body",
                    "replyStatusCode": 500,
                    "replyHeaders": "reply Headers",
                    "replyBody": "reply Body",
                    "errorMessage": "error"
                }
            ]
        }
    }
}
```


No error logging result:


```
{
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "FAILED_REQUESTS",
    "result": {
        "data": {
            "failedRequests": []
        }
    }
}
```


- ~~Reply description~~
  - **result**:
    - **data:**
      - **[FailedRequests] array**: - **mandatory**

12.5.5.9 Query: AllFdmInfos

This query requests all the FDM configurations/info.

- ~~Json query from ws_query~~
  - ~~Query sample:~~


```
{
```

```
    "dataResultUrl": "https://www.fod/data_endpoint",
    "errorResultUrl": "https://www.fod/error_endpoint",
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "ALL_FDM_INFOS",
    "executable": {
        "query": "query AllFdmInfos { allFdmInfos
            {
                device { fdmId fdmSwVersion fdmDateTime bufferCapacityUsed }
                transactionState { unsentEventCount fdmLastSuccessfulTrans retryCount }
                taskState { lastDownload }
                configuration {
                    fdmLockFromDateList { fromSystemDate fdmLock lockReason { language message } }
                    paramsVersionObject { paramsVersion paramsArray { fromSystemDate params
                        { url initInterval nopInterval maxBuffer queryInterval retryIntervals retryIntervalsInit
                    maxMsgSize postpone defaultMaxCalls maxCalls { dayOfWeek hour maxCallsInSlot} footer } } }
                    vatFromBookingDateList { fromBookingDate vatRates { vatLabel vatRate outOfScope } }
                    posIdFromDateList { fromSystemDate  posIdList }
                    trustCertFromDateList { fromSystemDate trustCert }
                    urlsFromDateList { fromSystemDate urls { key value } }
                    clientCertFromDateList { fromSystemDate clientCertSn }
                    localConfig { ws_ntp3 }
                } } }",
        "operationName": "AllFdmInfos"
    }
}
```

- ~~Query description:~~
  - **query**: `{ device { fdmId fdmSwVersion fdmDateTime bufferCapacityUsed } transactionState { unsentEventCount fdmLastSuccessfulTrans retryCount } taskState { lastDownload } configuration { fdmLockFromDateList { fromSystemDate fdmLock lockReason { language message } } paramsVersionObject { fromSystemDate params { paramsVersion url initInterval nopInterval maxBuffer queryInterval retryIntervals retryIntervalsInit maxMsgSize postpone defaultMaxCalls maxCalls { dayOfWeek hour maxCallsInSlot} footer } } vatFromBookingDateList { fromBookingDate vatRates { vatLabel vatRate outOfScope } } posIdFromDateList { fromSystemDate posIdList } trustCertFromDateList { fromSystemDate trustCert } urlsFromDateList { fromSystemDate urls { key value } } clientCertFromDateList { fromSystemDate clientCertSn } localConfig { ws_ntp3 } } }` graphQL query (**$AllFdmInfos** for reply)
    - **operationName**: query name
    - **variables**:

- ~~Json reply from FDM:~~ (to `dataResultUrl` endpoint)
  - ~~Reply samples:~~

```
{
    "actionId": "2023-01-05T18:03:23.000Z",
    "messageDateTimeEcho": "2024-04-17T04:00:00Z",
    "operationType": "ALL_FDM_INFOS",
    "result": {
        "data": {
            "allFdmInfos": {
                "device": {
                    "fdmId": "FOD01000001",
                    "fdmSwVersion": "1.2.0",
                    "fdmDateTime": "2022-10-10T10:00:39",
                    "bufferCapacityUsed": 10.02
```

```
                },
                "transactionState": {
                    "unsentEventCount": 0,
                    "fdmLastSuccessfulTrans": "2022-10-09T10:00:39",
                    "retryCount": 0
                },
                "taskState": {
                    "lastDownload": "2022-10-08T10:00:39"
                },
                "configuration": {
                    "fdmLockFromDateList": [
                        {
                            "fromSystemDate": "2024-06-01",
                            "fdmLock": false,
                            "lockReason": null
                        }
                    ],
                    "paramsVersionObject": {
                        "paramsVersion": "2023-01-05T18:03:23.000Z",
                        "paramsArray": [
                            {
                                "fromSystemDate": "2024-07-01",
                                "params": {
                                    "url": "https://www.gs2.be",
                                    "initInterval": 345600,
                                    "nopInterval": 21600,
                                    "maxBuffer": 20000,
                                    "queryInterval": 21600,
                                    "retryIntervals": [5, 60, 300, 300, 300, 3600, 10800],
                                    "retryIntervalsInit": [5, 60, 300, 300, 300, 3600, 10800],
                                    "maxMsgSize": 32,
                                    "postpone": true,
                                    "defaultMaxCalls": 30,
                                    "maxCalls": [{"dayOfWeek": "SATURDAY", "hour": 18, "maxCallsInSlot": 0},
                                        {"dayOfWeek": "SUNDAY", "hour": 12, "maxCallsInSlot": 100},
                                        {"dayOfWeek": "MONDAY", "hour": 17, "maxCallsInSlot": 0}],
                                    "footer": ["myfootertext1", "myfootertext1"]
                                }
                            },
                            {
                                "fromSystemDate": "2024-08-01",
                                "params": {
                                    "url": "https://www.gs2.be",
                                    "initInterval": 345600,
                                    "nopInterval": 21600,
                                    "maxBuffer": 20000,
                                    "queryInterval": 21600,
                                    "retryIntervals": [5, 60, 300, 300, 300, 3600, 10800],
                                    "retryIntervalsInit": [5, 60, 300, 300, 300, 3600, 10800],
                                    "maxMsgSize": 32,
                                    "postpone": true,
                                    "defaultMaxCalls": 30,
                                    "maxCalls": [{"dayOfWeek": "SATURDAY", "hour": 18, "maxCallsInSlot": 0},
                                        {"dayOfWeek": "SUNDAY", "hour": 12, "maxCallsInSlot": 100},
                                        {"dayOfWeek": "MONDAY", "hour": 17, "maxCallsInSlot": 0}],
                                    "footer": ["myfootertext2", "myfootertext2"]
                                }
                            }
```

```
                    ]
                }
                "vatFromBookingDateList":
                [
                    {
                        "fromBookingDate": "2020-01-01",
                        "vatRates": [
                            {
                                "vatLabel": "A",
                                "vatRate": 21,
                                "outOfScope": false
                            },
                            {
                                "vatLabel": "B",
                                "vatRate": 12,
                                "outOfScope": false
                            },
                            {
                                "vatLabel": "C",
                                "vatRate": 6,
                                "outOfScope": false
                            },
                            {
                                "vatLabel": "D",
                                "vatRate": 0,
                                "outOfScope": false
                            },
                            {
                                "vatLabel": "X",
                                "vatRate": 0,
                                "outOfScope": true
                            }
                        ]
                    },
                    {
                        "fromBookingDate": "2025-04-01",
                        "vatRates": [
                            {
                                "vatLabel": "A",
                                "vatRate": 21,
                                "outOfScope": false
                            },
                            {
                                "vatLabel": "B",
                                "vatRate": 17,
                                "outOfScope": false
                            },
                            {
                                "vatLabel": "C",
                                "vatRate": 6,
                                "outOfScope": false
                            },
                            {
                                "vatLabel": "D",
                                "vatRate": 0,
                                "outOfScope": false
                            },
                            {
```

```
                                "vatLabel": "X",
                                "vatRate": 0,
                                "outOfScope": true
                            }
                        ]
                    }
                ],
                "posIdFromDateList": [
                    {
                        "fromSystemDate": "2024-07-01",
                        "posIdList": ["Csnpos00000001", "Csnpos00000003"]
                    },
                    {
                        "fromSystemDate": "2024-08-01",
                        "posIdList": ["Csnpos00000001", "Csnpos00000003", "Csnpos00000004"]
                    }
                ],
                "trustCertFromDateList": [
                    {
                        "fromSystemDate": "2024-07-01",
                        "trustCert": [ "cert1x509pem", "cert2x509pem", "cert3x509pem" ]
                    },
                    {
                        "fromSystemDate": "2024-08-01",
                        "trustCert": [ "cert1x509pem2", "cert2x509pem2", "cert3x509pem2" ]
                    }
                ],
                "urlsFromDateList": [
                    {
                        "fromSystemDate": "2024-07-01",
                        "urls": [
                            { "key": "ws_reg", "value": "https://myurl:8080" },
                            { "key": "ws_nop", "value": "https://myurl:8080" },
                            { "key": "ws_query", "value": "https://myurl:8080" },
                            { "key": "ws_init", "value": "https://myurl:8080" },
                            { "key": "ws_ntp1", "value": "https://myurl:8080" },
                            { "key": "ws_ntp2", "value": "https://myurl:8080" }
                        ]
                    },
                    {
                        "fromSystemDate": "2024-08-01",
                        "urls": [
                            { "key": "ws_reg", "value": "https://myurl2:8080" },
                            { "key": "ws_nop", "value": "https://myurl2:8080" },
                            { "key": "ws_query", "value": "https://myurl2:8080" },
                            { "key": "ws_init", "value": "https://myurl2:8080" },
                            { "key": "ws_ntp1", "value": "https://myurl2:8080" },
                            { "key": "ws_ntp2", "value": "https://myurl2:8080" }
                        ]
                    }
                ],
                "clientCertFromDateList": [
                    {
                        "fromSystemDate": "2024-07-01",
                        "clientCertSn": "dhjs76djs"
                    },
                    {
                        "fromSystemDate": "2024-08-01",
```

```
                            "clientCertSn": "dhjs76djs2"
                        }
                    ],
                    "localConfig": {
                        "ws_ntp3": "https://myurl8:8080"
                    }
                }
            }
        }
    }
}
```

- Reply description:
  - **result**:
    - **data**:
      - AllFdmInfos

13. Warnings and errors summary

Warnings and errors are sent to the connected POS systems at the next possible opportunity for each incident. The POS system is responsible for handling the warnings and errors and translating them into useful output for the user(s).

Warnings are incidents that can escalate over time into an error. An error means that the FDM will refuse to accept and sign the business transaction.

Code (old)	Code	Type	Message	Remedy
W001	CLIENT_CERT_NEAR_EXPIRATION	warning	Validity of client certificate is less than 4 months.	The FDM should call the query webservice to receive the query SetClientCert.
W002	CLIENT_CERT_EXPIRED	warning	Client certificate has expired.	Check the e-service of FPS Finance if your FDM unit is eligible for a client certificate. FDM should automatically reinitialize itself. In case of issues, contact your dealer.
W003	RTC_SYNC_FAILED	warning	FDM could not synchronize real-time clock.	Possible causes: • issue with internet connection; • url of time server is outdated; • unavailability of time server. The user can manually try to resync the real-time clock with the NTP timeserver. The user can manually try to call the query webservice (ws_query). The user can manually try to update the URLs or adapt the URL of the time server manually via the Administration Console. If the issue, related to an outdated URL, persists, contact your dealer.

W004	UPDATE_URLS_FAILED	warning	FDM could not update URLs.	Possible causes: • Internal problem Contact your dealer.
W005	UPDATE_TRUST_CERT_FAILED	warning	The fiscal data module could not update the trust certificates for the webservices.	Possible causes: • Internal problem Contact your dealer.
W006	UPDATE_TASK_LIST_FAILED	warning	The fiscal data module could not obtain its task list from the webservice.	Possible causes: • issue with internet connection, client certificate or trust store; • unavailability of online service. The user can manually try to call query webservice (ws_query). The FDM should try to reinitialize itself automatically if the issue, related to client certificate, trust store or outdated URLs, persists. You might need to manually update the init URL via the Administration Console if it is outdated. If auto-reinitialization does not work, contact your dealer.
W007	TASK_FEEDBACK_FAILED	warning	The fiscal data module could not report feedback on a task to the webservice.	Possible causes: • issue with internet connection, client certificate or trust store; • unavailability of online service. The user can manually try to call query webservice (ws_query). The FDM should try to reinitialize itself automatically if the issue, related to client certificate, trust store or outdated URLs, persists. You might need to manually update the init URL via the Administration Console if it is outdated. If auto-reinitialization does not work, contact your dealer.
W008	NOP_FAILED	warning	The fiscal data module could not post a no operation message to the webservice.	Possible causes: • issue with internet connection, client certificate or trust store; • unavailability of online service. The user can manually try to call query webservice (ws_query). FDM should try to reinitialize itself automatically if the issue, related to client certificate, trust store or outdated URLs, persists. You might need to manually update the init URL via the Administration Console if it is outdated. If auto-reinitialization does not work, contact your dealer.

W009	BUFFER_NEAR_FULL	warning	The buffer usage exceeds 70%.	Possible causes: • issue with internet connection, client certificate or trust store; • unavailability of online service. The user can consult the last date and time of successful transmission, and the error log. FDM should try to reinitialize itself automatically if the issue, related to client certificate, trust store or outdated URLs, persists. You might need to manually update the init URL via the Administration Console if it is outdated. If auto-reinitialization does not work, contact your dealer.
W010	SERVER_CERT_RESOLVE_FAILED	warning	The fiscal data module could not resolve a server certificate.	The user can manually try to call query webservice (ws_query). If the issue (`SERVER_CERT_RESOLVE_FAILED`) persists, the FDM should set `fdmInit` = true to perform the initialization webservice (ws_init) at the next boot.
W011	TRANSACTION_UPLOAD_FAILED	warning	The fiscal data module could not post a business transaction to the webservice. The FDM puts the `retryCount` in the message.	The user should troubleshoot the internet connection or the webservices if they are not not available. Furthermore, via de e-service of FPS Finance, you can check if your FDM is properly registered and activated. Possible causes: • issue with internet connection, client certificate or trust store; • URLs are outdated; • unavailability of online service. The user can manually try to call query webservice (ws_query). FDM should try to reinitialize itself automatically if the issue, related to client certificate, trust store or outdated URLs, persists. You might need to manually update the init URL via the Administration Console if it is outdated. If auto-reinitialization does not work, contact your dealer.
W012	INITIALIZATION_FAILED	warning	Initialization failed.	Check internet connection. Check URL of initialization webservice. Check date/time settings. Check e-service of FPS Finance if your device is eligible for online services.
W013	RTC_NOT_INITIALIZED	warning	Check FDM battery.	Check FDM battery. If the problem persist, possible causes: • issue with internet connection; • URLs are outdated; • unavailability of online service ws_ntp1, ws_ntp2 and ws_ntp3 if specified.

W014	CORRUPT_RECORD_ENCOUNTERED	warning	Corrupt transaction in memory	Check FDM memory. If the problem persist, contact FDM Dealer/Manufacturer
W015	UPDATE_PARAMS_FAILED	warning	FDM could not update params.	Possible causes: • Internal problem Contact your dealer.
W016	UPDATE_CLIENT_CERT_FAILED	warning	FDM could not update client certificate.	Possible causes: • Internal problem Contact your dealer.
W017	UPDATE_VAT_RATES_FAILED	warning	FDM could not update VAT rates.	Possible causes: • Internal problem Contact your dealer.
W018	UDPATE_POS_ALLOWLIST_FAILED	warning	The fiscal data module could not update the POS allowlist.	Possible causes: • Internal problem Contact your dealer.
E001	BUFFER_FULL	error	FDM buffer is full.	User should troubleshoot internet connection, client certificate and trust store. Furthermore, via the e-service of FPS Finance, you can check if your FDM is properly registered and activated.
E002	UNKOWN_POS	error	POS is not registered for the FDM.	User should connect a registered POS to the FDM. You can change the relation between a POS and FDM via the e-service of FPS Finance.
E003	FDM_LOCKED	error	FDM is locked.	Contact FPS Finance.
E004	TOO_MANY_MEMORY_ERRORS	error	Too many corrupt transactions because of defect memory.	Contact FDM Dealer/Manufacturer (FDM memory problem)

14. Canonical JSON for repeatable hashes

Before creating the hash of the JSON, the following JSON re-encoding should be applied for the hash to be repeatable.

- Numbers are only allowed to contain the characters 0 through 9, the minus sign (-), and a decimal point. This means that values in scientific notation need to be re-encoded in their most basic form. Trailing zeroes after the decimal point are not allowed.
- All whitespace as defined in the JSON specifications are removed, no whitespace is added.
- Characters with a code point > 0x1F (31) and < 0x7F (127) are never escaped with the exception of code points 0x22 (34 - quotation mark) and 0x5C (92 - reverse solidus) which are always escaped as per JSON specification. Those two have to be escaped in their shortest possible notation (\" and \\ respectively).
- Characters with a code point <0x20 (32) or > 0x7E (126) are always escaped, and are escaped in their shortest possible notation. Backspace, form feed, linefeed, carriage return, and horizontal tab have such short notation (\b, \f, \n, \r, and \t respectively).
- When escaping a code point as one or more sets of four hex digits, the hex digits are always expressed in uppercase (i.e. \uAAFF).
- The JSON encoder needs to adhere to a strict implementation of the specifications. For example: the literals true, false, null need to be in lower case and without quotation marks, the names of the name/value pairs in the objects need to be wrapped in quotation marks, etc.

- The name/value pairs of each object in the JSON document are sorted, in ascending order, on the numerical value of the code points in the name. Assumed is that no duplicate names are used since this would be a bad practice as no standard exists in how various JSON decoders would process them.