Technical Guide for Gateway Management

Version 2.7 – 01/02/2022

Table of content

1 Contacts

For any question, please contact the following persons:

- Business related questions:

  Contract managers – contracting_AS@elia.be

- Technical questions:

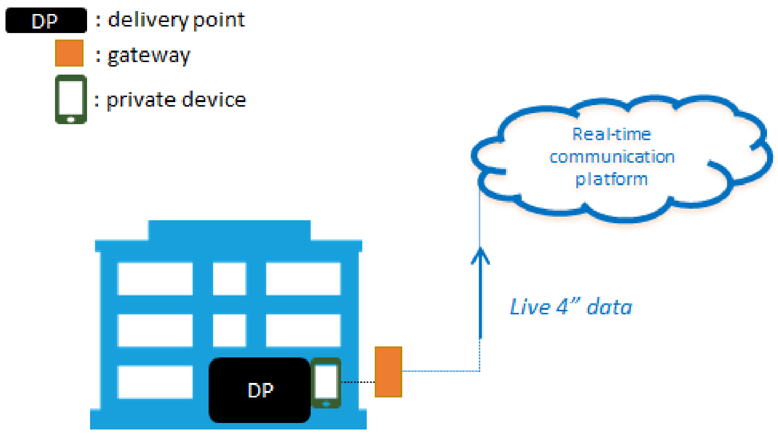
  Frate Michaël – michael.frate@elia.be – +32 472 38 10 32

2 Introduction

In the new aFRR design, a real-time data exchange of measured data and a collection of parameters, used for the aFRR-settlement process is required for delivery points DP_{PG} (i.e delivery points for which ELIA does not receive MW daily schedules) participating to the aFRR service.

For FCR, the technical units that make part of a virtual delivery point need to exchange their real-time data.

Private measurement devices must send the data, via gateways, directly or indirectly to the Communication Platform. To secure this data and the platform, we will deploy multiple mechanisms with respect to the data exchange (E2E encryption of the measured data between the gateway and the FlexHUB, certificate-based authentication) and require the upload on the Real-Time Communication Platform Web Portal of specific security-related technical documentation for each gateway model.



The following document describes technical framework related to the management of the gateways and delivery points connected to the Elia grid and their interaction with the Real-Time Communication Platform.

3 Asset configurations

The following configurations are authorised (see figure below):

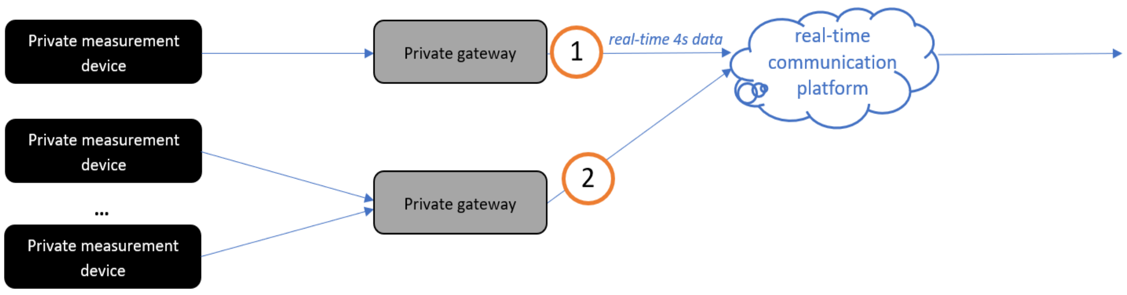
1. A single gateway transmits real-time data from one SDP measured by a measurement device.
2. A single gateway transmits real-time data from multiple SDPs measured by measurement devices.

In both configurations,

a. The private measurement device is located at the SDP. The SDP can also be defined at the level of the headpoint/access point.

b. The connection of a single gateway to SDPs located on two or more access points is not allowed.

c. A gateway must collect every 4s (exactly at second 0, 4, 8, 12, …), the instantaneous power measurement values of a measurement device and other necessary parameters required for the aFRR services, and communicate this in real-time to the real-time Communication Platform using the communication protocol determined by Elia.


Private measurement device
Private gateway
1
real-time 4s data
real-time communication platform
Private measurement device
...
Private measurement device
Private gateway
2

Centralised virtual gateways are also an allowed setup. The data will still be sent per delivery point, each delivery point being linked to a separate virtual gateway, to the Communication Platform. All specifications written in this document and corresponding business processes remain valid and must be complied to.

4 Data exchange specifications: GW to CP

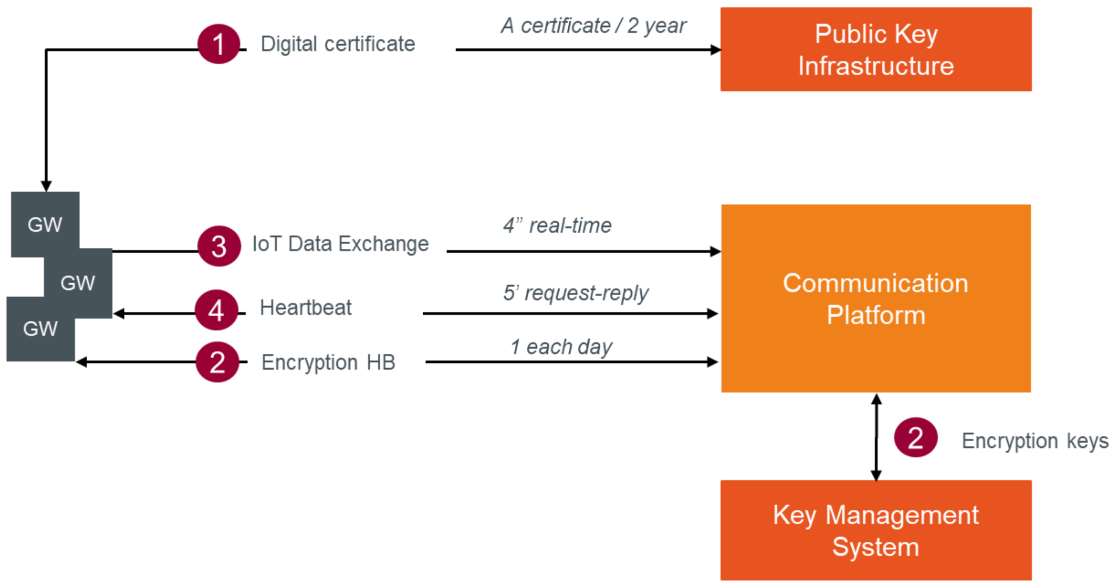
This section describes the detailed data exchange interface specifications to exchange data between the gateways, the Communication Platform and the security components. In the first version of the platform, the exchange of aFRR data is unidirectional (except for Heartbeat) from the gateways via the Communication Platform to the Flexhub. The message flow will consist of real-time 4s aFRR messages, used for the settlement of aFRR activations. One message will be sent for each delivery point connected to a gateway.

The exchange of FCR data is unidirectional (except for Heartbeat) from the gateways via the Communication Platform to the Flexhub. The message flow will consist of real-time 2s FCR messages, used for the verification of the aggregated signal sent by the BSP for FCR activations. One message will be sent for each delivery point connected to a gateway.

The security mechanisms allow a reliable and secure data exchange: the Public Key Infrastructure allows certificate-based authentication of the gateways and the Key Management System distributes encryption keys that can be used to encrypt the aFRR and FCR message body.

4.1 Data flows

Underneath you can find a visualisation of the E2E process flow of all data exchanges the gateways must be able to support.



1. Each gateway and application that will connect to the Communication Platform will need to acquire a digital certificate from the Public Key Infrastructure (valid for 2 years). This certificate is used to authenticate the gateway for all connections to the platform and Key Management System.

2. As explained in the introduction, the data (body) has to be end-to-end encrypted (from GW to Flexhub). Every day, an independent Key Management System (KMS) will generate encryption keys they need to use for message body encryption and will send these via the Communication Platform to the gateways.

3. Every 4 seconds, an aFRR message with encrypted body is send by the gateway to the Communication Platform. Every 2 seconds, a FCR message with encrypted body is send by the gateway to the Communication Platform. To be able to connect and publish the message on the queue, the gateways must have a digital certificate retrieved from the Public Key Infrastructure (PKI).

4. At regular interval (initially every 5 minutes), the Communication Platform will put a heartbeat message on the topic on which the gateway must reply. The message includes key values for specific use cases and for gateway connection status updates.

Message queues enable asynchronous communication, which means that the endpoints that are producing and consuming messages interact with the queue, not each other. In contrast to queues, in which each message is processed by a single consumer, **topics** and subscriptions provide a one-to-many form of communication, in a publish/subscribe pattern

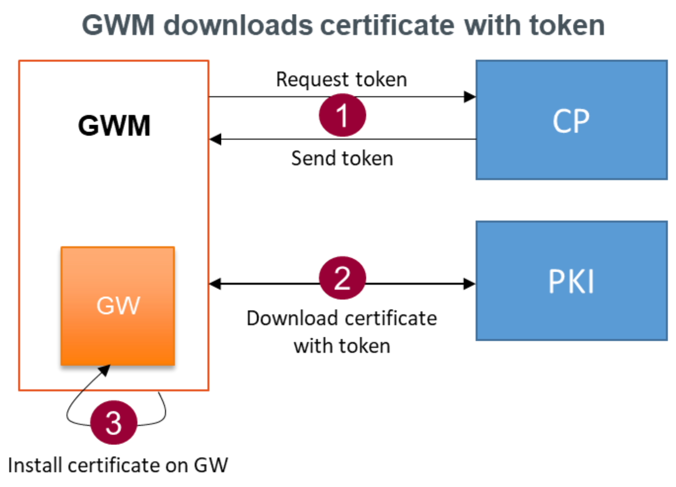
The data exchange between the gateway and the Communication Platform will be done using two different topics (1 topic for each direction see section 5.1).

4.2 Interfaces

4.2.1 Certificate base authentication

The following scenarios will be provided for acquisition of tokens and certificates:

Scenario 1: Acquisition of the Certificate through the portal


GWM downloads certificate with token
GWM
Request token
1
Send token
CP
GW
2
Download certificate with token
PKI
3
Install certificate on GW

1. The CP user requests a token via an action in the user interface of the portal for a gateway. A validation code will be generated and shown in the portal in the concerned gateway information screen, and a mail will be sent to you with a token.
2. The CP user navigates to a secure webpage via the web portal and uses the token as well as the validation code to download the certificate.
3. When the request is valid, the CP user can download a ZIP file with the PFX file and the password to extract the certificate (CERT file - X.509 Certificate). Another file is also present with an AES key. This key has to be used when the GW model is configured using AES to decrypt the received encryption key (see 4.2.3).

Scenario 2: Acquisition of the Certificate by the Gateway using a token

This second scenario will be available in a subsequent release. The detailed specification will follow in a next update of this document.

4.2.1.1 Request

Documentation will come.

4.2.1.2 Reply

Documentation will come.

4.2.1.3 Technical information

Information will come.

4.2.2 Product Messages

The messages in the data exchange will be composed of a functional header and a message body dependent on the product.

All required (and optional) fields are described in the following sections. In the element column, we use abbreviations to make the message tags smaller to reduce the message size.

With respect to datetimes, we use the ticks datetime format, which are the milliseconds, counted from the reference date: **01-01-2019 00:00:00 UTC**.

4.2.2.1 Header

Element	Data Type	Origin	Description
MT - Message Type	String	Data source originated	Represents the message type & frequency. This makes sure that every message type is unique no matter what frequency is requested.
SID – Sender Id	String	Data source originated	The Endpoint Id as generated by the Communication Platform
GID – Gateway Id	String	Date source originated	The Gateway Id of the gateway as generated by the Communication Platform.
EKV – Encrypted key version	String (optional)	Data source originated	The version of the encryption key used (changes at certain periods). If not sent then the message body is to be considered: not encrypted.
HV – Header version	Integer	Data source originated	The header version allows communication on the same message type but with different versions in case the message header structure is updated. This way, senders have time to adapt and a receiver knows how to interpret the message.
BV – Body version	Integer	Data source originated	The body version allows communication on the same message type but with different versions in case the message body structure is updated. This way, senders have time to adapt and a receiver knows how to interpret the message.
CTS - Creation timestamp	Ticks (UTC)	Date source originated	The timestamp when the message has been sent by the sender

4.2.2.2 Body (to be encrypted – see next sections)

For each product, a different body is defined according to the product requirements.
These bodies can be found in Product messages 8.1.
These bodies have to be encrypted using the encryption algorithm mentioned below.

4.2.2.3 Protocol

MQTTS protocol has to be used between the GW and the Communication Platform.

4.2.2.4 Encryption Algorithm

In order to encrypt the message bodies, the Advanced Encryption Standard (AES) / Rijndael algorithm (128 bits) using symmetric keys is used. A lot of implementation libraries are available in Python, JAVA, C#, …

The algorithm is described in the ISO/IEC 18033-3 standard. A simple description of this algorithm can be found here:
https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

This algorithm is used with, as default, the following parameters:
- Block size: 128 bits
- Key size: 128 bits
- Cypher: CBC
- Padding: PKCS7

4.2.3 Encryption keys

As described in the process flows, a Key Management System will generate encryption keys and put them available through to each separate GW through the Communication Platform.
Therefore, a specific message type will be exchanged.

4.2.3.1 Header

Parameter	Value	Description
MT – Message Type	String (ENCRYPTIONKEY)	Represents the message type & frequency. This makes sure that every message type is unique no matter what frequency is requested.

4.2.3.2 Body

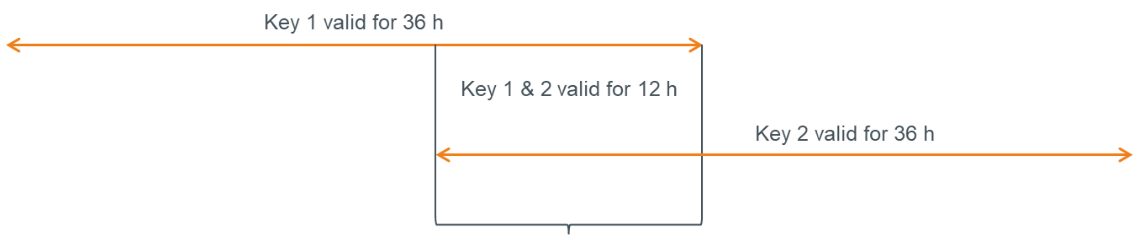
Pay attention that the body is a collection of keys. At the moment, only two MessageTypes are supported (aFRR and FCR) and thus only one element will be present in this collection.

Parameter	Value	Description
MT – Message Type		The message type for which the key is requested
KEY	string	The encryption key itself. This key is encrypted from the secure KMS using the GW certificate.

KV - Key version	String	The key version of the requested key
KT – Key Type	string	The algorithm supported for encryption
VF - Valid From (Start Validity)	Ticks	Validity start datetime of the encryption key
VT - Valid To (Stop Validity)	Ticks	Validity end datetime of the encryption key

<u>Gateways</u>

An encryption key is valid for **36 hours** and a new key will be retrieved daily. This means we will have an encryption key overlap of 12 hours within which period the new key must be received and used :



4.2.3.3 Technical information

The Communication Platform will exchange this message type with the same principles as the aFRR and FCR messages but in the other direction. Therefore a second topic is used (see later).

Please note that currently, only the AES / Rijndael algorithm is supported by the platform to encrypt the AFFR messages. Others can be added later on.

To guarantee the confidentiality on the key, the key present in the message will be encrypted also. The way is encrypt (and thus decrypt) the message and receive the key is configured on GW Model level. There are 2 possibilities: RSA or AES. We advise to use RSA.

Using RSA: the GW will need to use its own certificate private key to decrypt the key.

Using AES: the GW will need to use the AES key given with the certificate to decrypt the key. The parameters are the same than the one described in 4.2.2.4.

Message example:

Body (this will always be encrypted):

[{"MT":"aFRR","KV":"0jV0ly","KEY":"sapS9WSIpkSqG/TLEUY5tQ==","VF":47735117728,"VT":47864717728}]

Complete message:

```
{
   "MT": "ENCRYPTIONKEY",
    "Body":
"hj7EFc+S5giTCk41loj21ILGOT4aZkafhXzSbmt/gy4ANB4as1MZsnyAwixU76vm4AEmniUw2
9+8gNLEg9Yq0LeR8Hc3zEqGXFaplqNv+6TrSQy+VvZG2NR4xaK1EvAUF8GeP6U9FMVz4eB8
MWB94RW44n3QOYfCQz7CTEJXvbwbwclGHJN4wsfGPMMxdZUeUiLAuhHvGG7KeLPefTl2
DoHS4N8B2mol7lXFZcSD1vnCy4kcF3Jyd6KPEzKfhkFJc2FZaidIjSWuo/Z5HQb74hAmg2m/R
EQnw7yXfaHjJ3E8ZzoFZhw+sR7TsBnZvDInni74zuv0R7UFTg2eHmKHnA=="  }
```

4.2.4 Encryption key Request

As described in the process flows, a Key Management System will generate encryption keys and put them available through to each separate GW through the Communication Platform. When the GW has to be replaced or restarted with an empty configuration, the latest encryption key(s) has(ve) to be requested to be able to send new messages again. Therefore, a specific message type will be exchanged.
Note that you will receive one message (as described in section 4.2.3) for each message type and version managed by your gateway with an active aFRR service (normally only one because there is currently only one message type with only one version).
You will only receive a key when the EP managed by your Gateway has an active service. If it is not the case, you can ask for a dummy key. This key has exactly the same format as a normal key, but is recognized by the platform as a dummy key and your message is logged with a specific error code and not transferred to the Flexhub.

4.2.4.1 Header

Parameter	Value	Description
MT - Message Type	String (ENCRYPTION KEYREQUEST)	Represents the message type & frequency. This makes sure that every message type is unique no matter what frequency is requested.
GID – Gateway Id	String	The Gateway Id of the gateway as generated by the Communication Platform.
CTS - Creation timestamp	Ticks (UTC)	The timestamp when the message has been sent by the sender.

4.2.4.2 Body

Body is empty.

4.2.4.3 Technical information

The Communication Platform will exchange this message type with the same principles as the aFRR and FCR messages but in the other direction. A specific topic for this message exchange will be foreseen.

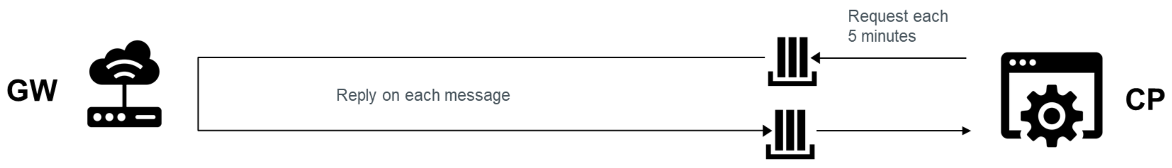
Message example:

```
{
  "MT": "ENCRYPTIONKEYREQUEST","GID":"ABCDE","CTS":46184713978

}
```

4.2.5 Heartbeat

The heartbeat mechanism allows to exchange key values between the gateways and the Communication Platform that are not related to the exchange of market data from endpoints.



The Communication Platform indicates the pace of the heartbeat messages and will be initially set to every five minutes.

The heartbeat message has two functioning methods:

- Ad hoc: an action button in the management portal will be provided in order to initiate a one-time heartbeat message sent to the gateway (see platform user manual). If this message is successfully replied to by the gateway, its communication status will be set to ‘Connected’. This allows the user to test the connection and authentication of a gateway.

- Recurrent: once a service is activated on this endpoint, the CP will initiate a heartbeat at the interval it choses (5 minutes initially). Also here, the communication status of the gateway will be updated in the portal in case a single heartbeat is not replied to. The time to live of the heartbeat message will equal the heartbeat frequency (5 minutes initially).

4.2.5.1 CP to GW

Header

Parameter	Value	Description

MID - MessageId	Integer	A counter that can be reinitialized
MT - Message Type	String (HEARTBEAT)	Represents the message type & frequency. This makes sure that every message type is unique no matter which message heartbeat is posted.

Body

Parameter	Value	Description
TS - Time Sync	1	Only present when a gateway must synchronize its internal clock with an NTP server
GWV - GW Version	1	Only present when a gateway must send its firmware and software version. This will be requested daily.

TimeSync et GW version parameters are 2 keys that can be added as list of parameters in the message. Other parameter(s) can be added later on in body.

Message example without time synchronization and GW version needed:

```
{
   "MID": 36,
   "MT": "HEARTBEAT",
 },
```

Message example with time synchronization and without GW version needed:

```
{
   "MID": 36,
   "MT": "HEARTBEAT",
   "Body": {"TS":1}
 },
```

Message example without time synchronization and with GW version needed:

```
{
   "MID": 36,
   "MT": "HEARTBEAT",
   "Body": {"GWV":1}
 },
```

Message example with time synchronization and GW version needed:

```
{
   "MID": 36,
   "MT": "HEARTBEAT",
   "Body": {"TS":1, "GWV”:1}
},
```

4.2.5.2 GW to CP

Header

Parameter	Value	Description
MID - MessageId	Integer	The message ID of the Heartbeat request message.
MT - Message Type	String (HEARTBEAT)	Represents the message type & frequency. This makes sure that every message type is unique no matter what frequency is requested.
GID – Gateway Id	String	The serial number of the gateway as registered in the Communication Platform.
CTS - Creation timestamp	Ticks (UTC)	The timestamp when the message has been sent by the sender

Body

Parameter	Value	Description
SV - Software version	String	The model software version on which the gateway is running. Only to be sent when the GW Version field in the request is sent.
FWV - Firmware version	String	The model firmware version on which the gateway is running. Only to be sent when the GW Version field in the request is sent.

Message example without software and firmware version needed:

```
{
   "MID": 36,
   "MT": "HEARTBEAT ",
   "GID": "123-ABCD",
   "CTS": 29666589696
 },
```

Message example with software and firmware version needed:

```
{
   "MID": 36,
   "MT": "HEARTBEAT ",
   "GID": "123-ABCD",
   "CTS": 29666589696,
   "Body": {"SV":"1.2", "FWV":"1.74"}
 },
```

4.2.5.3 Technical information

The Heartbeat will be pushed regularly on the GW receiver topic. The response is sent to the same topic as the aFRR messages.

4.3 Exception handling

4.3.1 Buffering

A local buffering of at least 5 days has to be done locally. This will be used when the communication between the GW and the Communication Platform is interrupted. The data has to be timestamped at the moment they are produced.
Once the communication is back up, the messages not sent during the interruption have to be sent.

4.3.2 Throttling

To avoid congestion, a maximum of **1** message can be sent per second per gateway.

4.3.3 Message grouping

- Message grouping can be done for a period of **1** minute (15 data of 4’’ or 30 data of 2’’). Pay attention that it is only valid during exception handling (communication failure, …)
- When grouping, the header is sent only once and the bodies of the specific time series will be grouped in one body.
- The body will be encrypted only once

4.3.4 Fallback files

In the event that Elia does not receive the data through real time communication for bigger gaps, the following is put in place:

- The FSP must, on the request of Elia, be able to provide a fallback file with time series containing the same parameters requested in the aFRR or FCR message.
- Elia can only request fallback files in a period covering maximum 90 days before the day of request.
- The delivery of the fallback file must be fulfilled within five working days.

4.4 Service level agreements

To assure correct, complete and real-time data exchange, there will be a monitoring foreseen on predefined KPIs.

5 Data exchange specifications: App to CP

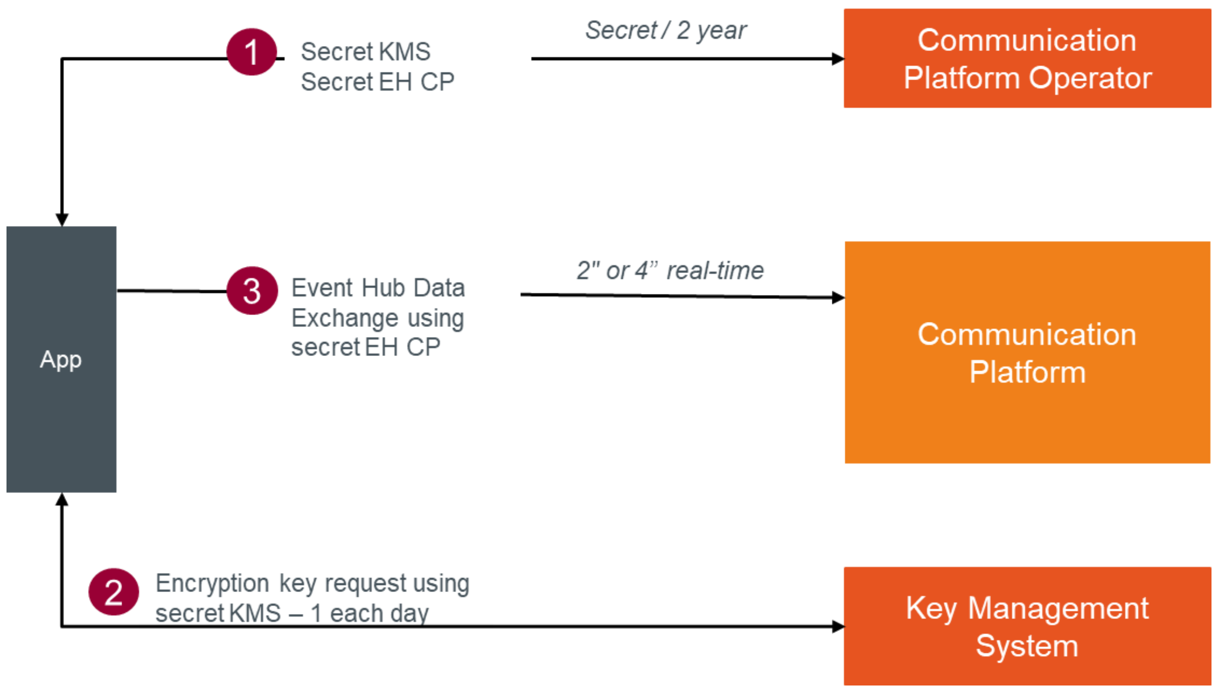
This section describes the detailed data exchange interface specifications to exchange data between applications, the Communication Platform and the security components. This data exchange is suited for exchanging data for a large amount of delivery points, like in low voltage connected delivery points.

The exchange of data has the same granularity as in the GW to CP configuration.

The security mechanisms allow a reliable and secure data exchange: the Public Key Infrastructure allows secure authentication of the application and the Key Management System distributes encryption keys that can be used to encrypt the aFRR and FCR message body.

To connect an APP to the platform, you will need to contact the Communication Platform Operator to receive the needed credentials for EventHub and KMS system.

5.1 Data flows


1
Secret KMS
Secret EH CP
Secret / 2 year
Communication Platform Operator
App
3
Event Hub Data Exchange using secret EH CP
2" or 4" real-time
Communication Platform
2
Encryption key request using secret KMS – 1 each day
Key Management System

1. Each application that will receive two secrets from the Communication Platform Operator (valid for 2 years). One secret is to connect to the Key Management System, the other to connect to the Event Hub of the communication platform.

2. As explained in the introduction, the data (body) has to be end-to-end encrypted (from App to Flexhub). Every day, the app can connect to the independent Key Management

System (KMS) using the first secret to retrieve the encryption keys they need to use for message body encryption.

3. Every 4 seconds, an aFRR message with encrypted body is send by the app to the Communication Platform Event Hub. Every 2 seconds, a FCR message with encrypted body is send by the app to the Communication Platform Event Hub. To be able to connect and publish the message on the queue, the app must the second secret.

5.2 Interfaces

5.2.1 Product Messages

The messages in the data exchange will be composed of a functional header and a message body dependent on the product.

All required (and optional) fields are described in the following sections. In the element column, we use abbreviations to make the message tags smaller to reduce the message size.

With respect to datetimes, we use the ticks datetime format, which are the milliseconds, counted from the reference date: **01-01-2019 00:00:00 UTC**.

5.2.1.1 Header

Element	Data Type	Origin	Description
MT - Message Type	String	Data source originated	Represents the message type & frequency. This makes sure that every message type is unique no matter what frequency is requested.
AID – Application ID	String	Data source originated	The Application Id of the send application as generated by the Communication Platform
SID – Sender Id	String (optional)	Date source originated	The Endpoint Id as generated by the Communication Platform. Mandatory for FCR/AFRR use cases.
ERIDS – Received Id's	Array of Strings (Optional)	Data source originated	The list of endpoint receivers of the messages (when the message is specific for some endpoints). Null for FCR/AFRR use cases.
ARIDS – Received Id's	Array of String (optional)	Data source originated	The list of application receivers of the message (when the message is specific for some applications).

			Null for FCR/AFRR use cases.
EKV – Encrypted key version	String (optional)	Data source originated	The version of the encryption key used (changes at certain periods). If not sent then the message body is to be considered: not encrypted
HV – Header Version	Integer	Date source originated	The header version allows communication on the same message type but with different versions in case the message header structure is updated. This way, senders have time to adapt and a receiver knows how to interpret the message.
BV – Body Version	Integer	Data source originated	The body versions allows communication on the the same message type but with different versions in case the message body structure is updated. This way, senders have time to adapt and a receiver knows how to interpret the message
CTS – Creation timestamp	Ticks (UTC)	Data source originated	The timestamp when the message has been sent by the sender.

5.2.1.2 Body (to be encrypted – see next sections)

For each product, a different body is defined according to the product requirements.
These bodies can be found in Product messages 8.1.
These bodies have to be encrypted using the encryption algorithm mentioned below.
These bodies have the same content as in the GW to Communication Platform data exchange.

5.2.1.3 Protocol

Connection to EventHub (see 5.3 Connection to Eventhub).

5.2.1.4 Encryption Algorithm

In order to encrypt the message bodies, the Advanced Encryption Standard (AES) / Rijndael algorithm (128 bits) using symmetric keys is used. A lot of implementation libraries are available in Python, JAVA, C#, …

The algorithm is described in the ISO/IEC 18033-3 standard. A simple description of this algorithm can be found here:
https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

This algorithm is used with, as default, the following parameters:

- Block size: 128 bits
- Key size: 128 bits
- Cypher: CBC
- Padding: PKCS7

5.3 Connection to EventHub

When an application is registered on the Communication Platform, connection details will be provided to connect to a specific queue.
All the messages approved to be sent to your application will be sent to this queue (whatever the message type).

Please note that the header of the messages received on the Eventhub will be different from the messages sent by an application or the Gateway. Indeed, some additional fields will be added (like a unique message id, the timestamp when the platform receives the message, the timestamp when the platform process the message, …) and some fields will be removed (Gateway ID for example). The body of the message will never be altered by the platform.

2 different topics will be provided: 1 to send data (used) and 1 to receive the data (currently unused). Depending on the role played on the platform, one will be more relevant than the other but both are always provided.

Documentation over Event Hubs can be found here:

Azure Event Hubs documentation | Microsoft Docs

Quickstart section gives you information how to connect in different programming languages: .NET Core, Java, Python, JavaScript, Apache Kafka, Apache Storm, Go.

5.4 Connection to KMS

ACC:

Swagger URL of the KMS: https://cp-app-acc-sec-we-01.synergrid.be/swagger/index.html

URL of the service: https://synergridafrracc.onmicrosoft.com/security-api

TenantID (B2C) for identification: 7e362a04-a71b-491d-83c3-75d2b17b5d9b

PRE-PROD

Swagger URL of the KMS: https://cp-app-pre-sec-we-01.synergrid.be/swagger/index.html

URL of the service: https://synergridafrrpre.onmicrosoft.com/security-api

TenantID (B2C) for identification: 6b9b2089-a838-4cbe-963e-58bdc3aaeeaa

PROD

Swagger URL of the KMS: https://cp-app-prod-sec-we-01.synergrid.be/swagger/index.html

URL of the service: https://synergridrtcp.onmicrosoft.com/security-api

TenantID (B2C) for identification: ed729655-fe49-4600-9b1a-f4b64fe4a5ee

Client: xxx (specific for each application)

Secret: yyy (specific for each application)

To obtain a token, the library 'Microsoft.IdentityModel.Client' can be used. The request is composed as follows:
https://login.microsoftonline.com/{tenantId}/oauth2/token

In the ACC case, we obtain:
https://login.microsoftonline.com/7e362a04-a71b-491d-83c3-75d2b17b5d9b/oauth2/token

5.2.1 Get a key to decrypt a received message

Each received message on the EventHub will contain:

- A Key Version
- A Message Type

These 2 information will be needed to fetch the key from the Key Management System.
Pay attention: your (active) service can only receive the keys for message types used by this service. Requests for other message types will be denied.
When a service is registered on the Communication Platform, connection details will be provided to connect to the KMS.

API (GET): /api/v1/Key/decrypt/{messageType}/{encryptionKeyVersion}

Request:

messageType: the message type for which a key is requested

encryptionKeyVersion: the key version received in the message header

Response:

messageType: the message type for which a key is requested

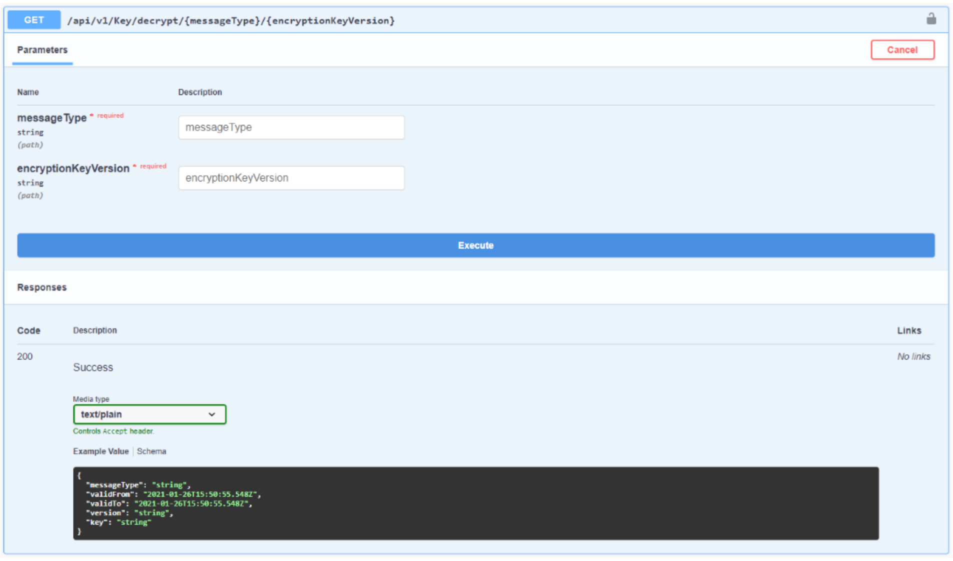
validFrom: the date from which the key is valid

validTo: the date to which the key is valid

version: the key version

key: the key itself

```
{
  "messageType": "string",
  "validFrom": "2021-07-05T12:00:46.525Z",
  "validTo": "2021-07-05T12:00:46.525Z",
  "version": "string",
  "key": "string"
}
```


GET /api/v1/Key/decrypt/{messageType}/{encryptionKeyVersion}

Parameters

Name	Description
messageType * required string (path)	messageType
encryptionKeyVersion * required string (path)	encryptionKeyVersion

Responses

Code	Description	Links
200	Success Media type: text/plain Controls Accept header. Example Value | Schema	No links

```
{
  "messageType": "string",
  "validFrom": "2021-01-26T15:50:55.548Z",
  "validTo": "2021-01-26T15:50:55.548Z",
  "version": "string",
  "key": "string"
}
```

5.2.2 Get the last valid key to encrypt a message

When a message needs to be sent to the Communication Platform, a valid key has to be used. The key is specific per message type.
When a service is registered on the Communication Platform, connection details will be provided to connect to the KMS.
Pay attention: your (active) service can only receive the keys for message types used by this service. Requests for other message types will be denied.

API (GET): /api/v1/Key/encrypt/{messageType}

Request:

messageType: the message type for which a key is requested

Response:

messageType: the message type for which a key is requested

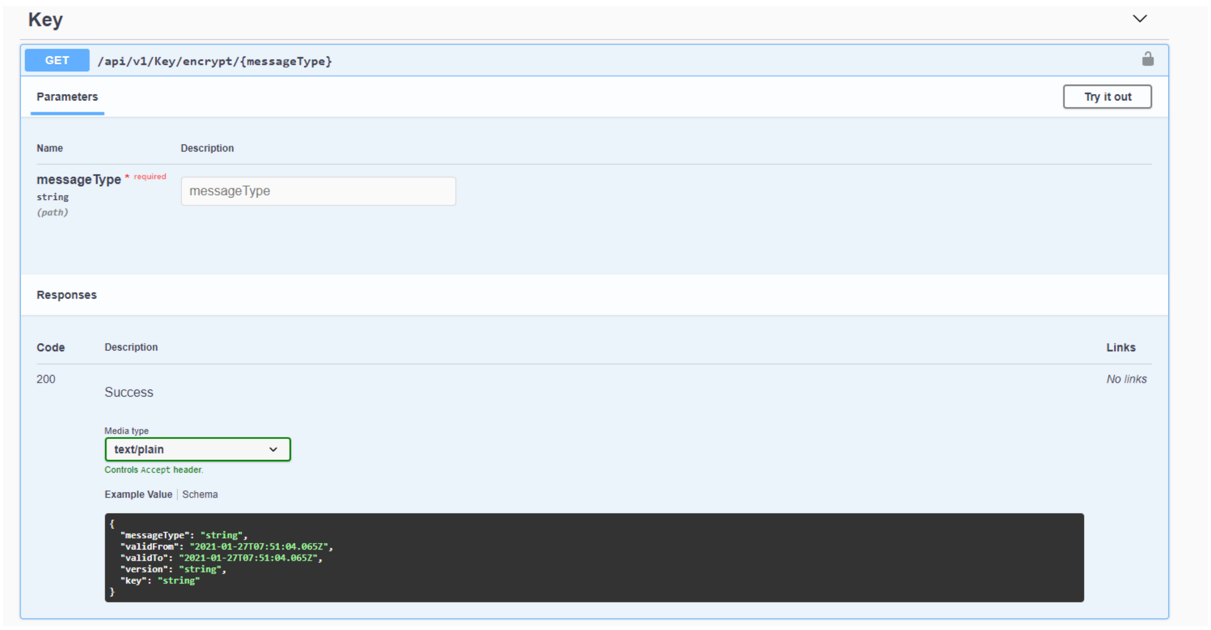
validFrom: the date from which the key is valid

validTo: the date to which the key is valid

version: the key version

key: the key itself

```
{
  "messageType": "string",
  "validFrom": "2021-07-05T11:58:17.062Z",
  "validTo": "2021-07-05T11:58:17.062Z",
  "version": "string",
  "key": "string"
}
```


Key

GET /api/v1/Key/encrypt/{messageType}

Parameters

Try it out

Name	Description
messageType * required string *(path)*	messageType

Responses

Code	Description	Links
200	Success	*No links*

Media type

text/plain

Controls Accept header.

Example Value | Schema

```
{
  "messageType": "string",
  "validFrom": "2021-01-27T07:51:04.065Z",
  "validTo": "2021-01-27T07:51:04.065Z",
  "version": "string",
  "key": "string"
}
```

5.5 Exception handling

5.5.1 Buffering

A local buffering of at least 5 days has to be done locally. This will be used when the communication between the App and the Communication Platform is interrupted. The data has to be timestamped at the moment they are produced.
Once the communication is back up, the messages not sent during the interruption have to be sent.

5.5.2 Message grouping

- Message grouping can be done for a period of **1** minute (15 data of 4'' or 30 data of 2''). Pay attention that it is only valid during exception handling (communication failure, …)
- When grouping, the header is sent only once and the bodies of the specific time series will be grouped in one body.
- The body will be encrypted only once

5.5.3 Fallback files

In the event that Elia does not receive the data through real time communication for bigger gaps, the following is put in place:

- The FSP must, on the request of Elia, be able to provide a fallback file with time series containing the same parameters requested in the aFRR or FCR message.
- Elia can only request fallback files in a period covering maximum 90 days before the day of request.
- The delivery of the fallback file must be fulfilled within five working days.

5.6 Service level agreements

To assure correct, complete and real-time data exchange, there will be a monitoring foreseen on predefined KPIs.

6 Technical features

6.1 URL's and config

The platform will be available at the following URL's:

ACC: https://rtcp-acc.synergrid.be/

DEMO: https://rtcp-pre.synergrid.be/

PROD: https://rtcp.synergrid.be/

Please note that the first tests starting from May 18th have to be done with the Pre-Prod environment. The acceptance environment will be used when updates of the platform will be released. The production environment (to use for the pre-qualifications tests) is released in and will be available after contractual agreement with your Key Account Manager.

To connect to the platform, 2 steps are needed:

1) Connect to the Device Provisioning System (DPS) to receive the URL of the MQTT broker. This URL can changed in time due to load spread for example or during a DRP. Therefore, each time a new connection has to be established, a call to the DPS has to be made to receive the broker URL.
2) Connection to the MQTT broker thanks to the URL given by the DPS

Note that you can use the Microsoft Azure IOTHub SDK available on Azure platform. This SDK is available in different programming language. There is no obligation to use it. An example of Gateway programming without the use of the SDK can be provided on demand (in C#). Device Twins functionalities are not used.

<u>DPS</u>

- The Device Provisioning System URL is the following without using the Microsoft SDK:

  https://global.azure-devices-provisioning.net/{connectionScope}/registrations/{GatewayBusinessId}/register?api-version=2019-03-31

- The GatewayBusinessId is generated by the platform when a new Gateway is created.
- The GW certificate has to be used to connect to the DPS
- Connection scope :

  ACC: 0ne000F2E25

  DEMO: 0ne000F7DB8

  PROD: 0ne000FEA0A

  Info: With the Microsoft SDK, the connection string is the following:

  global.azure-devices-provisioning.net

Note that these URL's & configurations will not change in case of DRP.

<u>MQTT broker</u>

- The connection has to be made thanks to the URL received by the DPS. During testing phase, this URL will remain fix (cp-iothub-pre-we-01.azure-devices.net).
- Both root certificate et GW certificate are needed to connect to the MQTT broker
- TLS 1.0, 1.1 and 1.2 are still supported but both 1.0 and 1.1 are deprecated and will not be supported in the future

So, it is needed to create an MQTTClient with these settings:
Hostname: cp-iothub-pre-we-01.azure-devices.net
Port: 8883
Secure: True
CA cert: rootCertificate (AzureBaltimoreRoot.cer)
Client Cert: Gateway certificate
MqttSslProtocols: TLSv1_2

Then we do a Connect on this client object with these settings:
ClientId: Gateway business Id
User name: cp-iothub-pre-we-01.azure-devices.net/{clientId}/?api-version=2018-06-30
Password: null
CleanSession: false
Keep alive: 10

- The name of the 2 topics are :

Cloud to Device: "devices/{GatewayBusinessId}/messages/devicebound/#"
Device to Cloud: "devices/{GatewayBusinessId}/messages/events/"

Heartbeat request and encryption keys comes on the same topic (Cloud to Device). The received message type is different.
Hearbeat response and AFFR messages needs to be sent to the same topic also (Device to Cloud).

6.2 Message format testing

You can test the validity of JSON messages in the communication portal interface. See the user manual of the platform for further explanations.

6.3 Time synchronization

Gateways have to be synchronized with an NTP server or an equivalent system at all times. The precision of the timestamp should be at least 20ms. In case of consistent time difference, the CPO will request, via a heartbeat message, to synchronise to an NTP server.

6.4 Examples

Here below, some examples of messages are given. It will also be possible to test your message format (JSON Validation) in our test platform.

To receive more detail how to connect to the platform and a detailed example (in C#) of the code to connect to our platform, please use the technical reference as defined in paragraph 1 of this document.

Other examples (in different programming languages) can be found here: https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-sdks.

The section to use is '**IoT Hub Device SDKs**'.

6.4.1 Data exchange

Messages have to be sent with encrypted body. In this section, we will give you the overview of unencrypted and encrypted data to allow you to generate the correct JSON before encryption. As previously described, the body can contain multiple 4 seconds data to cover some exception flows. Both cases are detailed here under.

- aFRR data – Unencrypted JSON with one 4'' data :

```
 {
  "MT": "AFRR",
  "HV": 1,
  "BV": 1,
  "GID": "SN4589674",
  "CTS": 33496996088,
  "EKV": 1,
  "SID": "84V-UOU-40P",
  "Body":
[{"DPM":0.123,"DPB":0.987,"AS":1,"PS":0.0,"MTS":0,"SDP":"541122334455667788"}]
 }
```

- aFRR data – Encrypted JSON with one 4'' data :

The encryption key to use for this message has the following properties:

Encryption type: RijndaelManaged -> KeySize: 128, Padding: PKCS7, Mode: CBC
Encryption key: 9xu0DqrgaFYgrPhudq9s6A==

Encryption IV: 9xu0DqrgaFYgrPhudq9s6A==

```
{
   "MT": "AFRR",
   "HV": 1,
   "BV": 1,
   "GID": "SN4589674",
   "CTS": 33496996088,
   "EKV": 1,
"SID": "84V-UOU-40P",
   "Body":
"9pMzn4mX5b/+y5SSPVzi6vgebzyLDQJ5bog4c3mg+8cIXS1eVw5ELNlbBUqllhYznMt872Nu7dwUyBTb
Ykl7IPcC9NK8XFy9wnFtVLLmFjM="
  }
```

- FCR data – Unencrypted JSON with one 2’’ data :

```
{
     "MT": "FCR",
     "HV": 1,
     "BV": 1,
     "GID": "SN4589674",
     "CTS": 33496996088,
     "EKV": 1,
     "SID": "84V-UOU-40P",
     "Body": “[{"MTS": 33496996088,"DPM": 0.15,"SDP": "11987"}]”
}
```

7 Master Data APIs

7.1 Introduction

This section describes the Communication Platform API’s for master data management. These APIs enable the onboarding of your master data as well as querying information.

In order to be able to connect to the API’s, you will need to contact the RTCP Operator (thanks to the Contact us form or specific CPO email address) to receive the needed credentials.

Authentication will be done based on a client secret mechanism linked to your RTCP account.

7.2 Master data setup

In case of setup the technical units under a virtual delivery point. An endpoint as explained below needs to be created for each technical unit. The virtual delivery point will be created in the Flexhub.

In that case the headpointEAN is the EAN of the virtual delivery point and the FriendlyName is the EAN of access point where the technical unit belongs to (for example, the EAN code of the house)

7.3 GatewayModel Management

7.3.1 GetGatewayModelList

Get: /api/interface/v1/GatewayModel

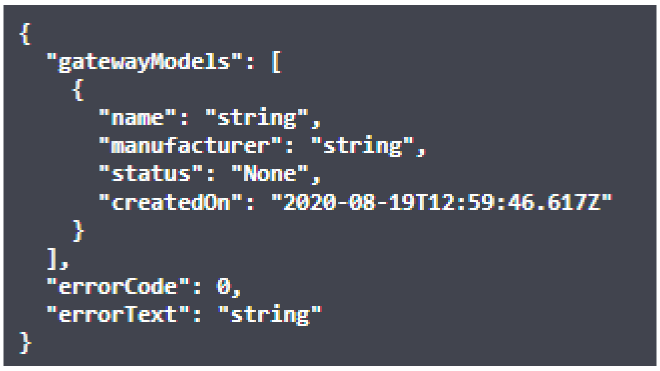
7.3.1.1 Domain

Request

No Parameter. The account is deduced from the authentication.

Response

Fields	Type	Description
id	String	The GW model id
Name	String	The model name
Status	Enum	Status (Active, Inactive)
Manufacturer	String	The manufacturer name
CreatedOn	DateTime	Creation date and time



```
{
  "gatewayModels": [
    {
      "name": "string",
      "manufacturer": "string",
      "status": "None",
      "createdOn": "2020-08-19T12:59:46.617Z"
    }
  ],
  "errorCode": 0,
  "errorText": "string"
}
```

7.3.1.2 Business Rules

No specific business rules

7.3.2 CreateGatewayModel

Put: /api/interface/v1/GatewayModel

7.3.2.1 Domain

Request

Fields	Type	Description
Name	String	The model name
Manufacturer	String	The manufacturer name
Encryption Key Type	Enum	Encryption Key Type (AES or RSA)

```
{
  "name": "string",
  "manufacturer": "string",
  "encryptionKeyType": "None"
}
```

Response

Fields	Type	Description
gatewayModelId	String	The model id
gatewayModelStatus	Enum	Status (Active, Inactive, DocumentationRequired)

```
{
  "gatewayModelId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "gatewayModelStatus": "None",
  "errorCode": 0,
  "errorText": "string"
}
```

7.3.2.2 Business Rules

ErrorCode	Error Text
0	
1	GW Model Name must be unique
2	GW Model Name is empty
3	GW Model Manufacturer is empty
4	EncryptioKeyAlgorithm unknown
100	Unexpected error

7.3.3 UpdateGatewayModel

Post: /api/interface/v1/GatewayModel/{gatewayModelId}

7.3.3.1 Domain

Request

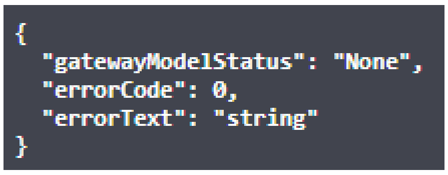
The function will edit the GW model.

Fields	Type	Description
Name	String	The model name
Manufacturer	String	The manufacturer name

```
{
  "name": "string",
  "manufacturer": "string"
}
```

Response

Fields	Type	Description
gatewayModelStatus	Enum	Status (Active, Inactive, DocumentationRequired)



```
{
  "gatewayModelStatus": "None",
  "errorCode": 0,
  "errorText": "string"
}
```

7.3.3.2 Business Rules

ErrorCode	Error Text
0	
1	GW Model not existsing for account
2	GW Model name is already existing or empty
3	GW Manufacturer name is empty
4	GW Model name already exists for account
100	Unexpected error

7.3.4 ActivateGatewayModel

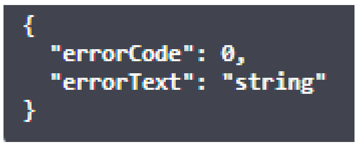
Post: /api/interface/v1/GatewayModel/{gatewayModelId}/activate

7.3.4.1 Domain

Request

No Parameter. The function will put the GW model to active.

Response



```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.3.4.2 Business Rules

ErrorCode	Error Text
0	
1	GW Model ID not existing for current account
2	GW Model status is already active
100	Unexpected error

7.3.5 DesactivateGatewayModel

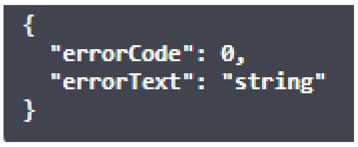
Post: /api/interface/v1/GatewayModel/{gatewayModelId}/deactivate

7.3.5.1 Domain

Request

No Parameter. The function will put the GW model to inactive.

Response



```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.3.5.2 Business Rules

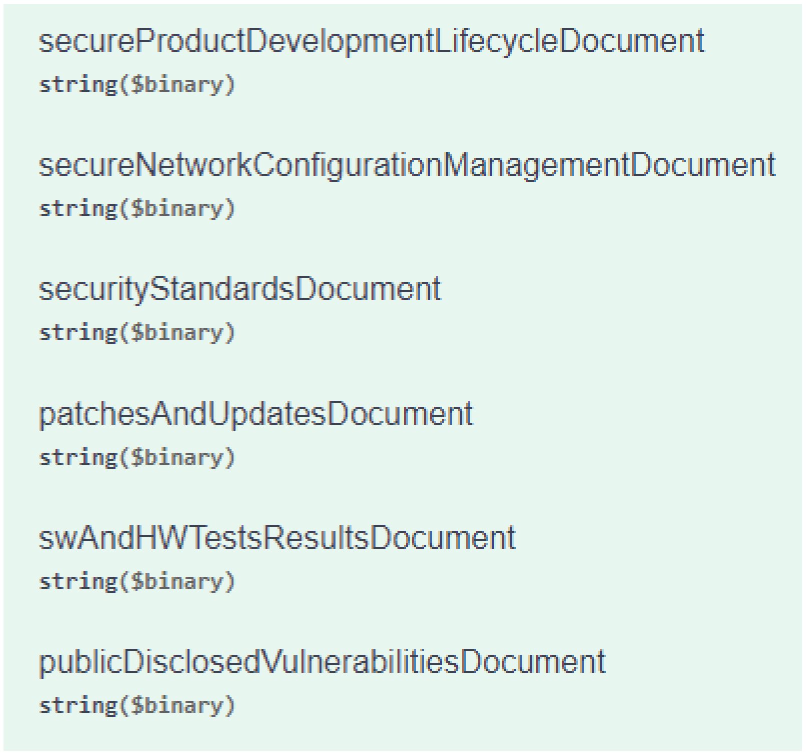
ErrorCode	Error Text
0	
1	GW Model ID not existing for current account
2	GW Model status is already inactive
100	Unexpected error

7.3.6 UpdateGatewayModelDocuments

Post: /api/interface/v1/GatewayModel/{gatewayModelId}/documents

7.3.6.1 Domain

Request



secureProductDevelopmentLifecycleDocument
`string($binary)`

secureNetworkConfigurationManagementDocument
`string($binary)`

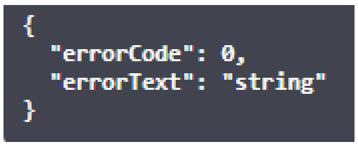
securityStandardsDocument
`string($binary)`

patchesAndUpdatesDocument
`string($binary)`

swAndHWTestsResultsDocument
`string($binary)`

publicDisclosedVulnerabilitiesDocument
`string($binary)`

Response



```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.3.6.2 Business Rules

No specific business rules

7.4 Gateway Management

7.4.1 GetGatewayList

Get: /api/interface/v1/Gateway

7.4.1.1 Domain

Request

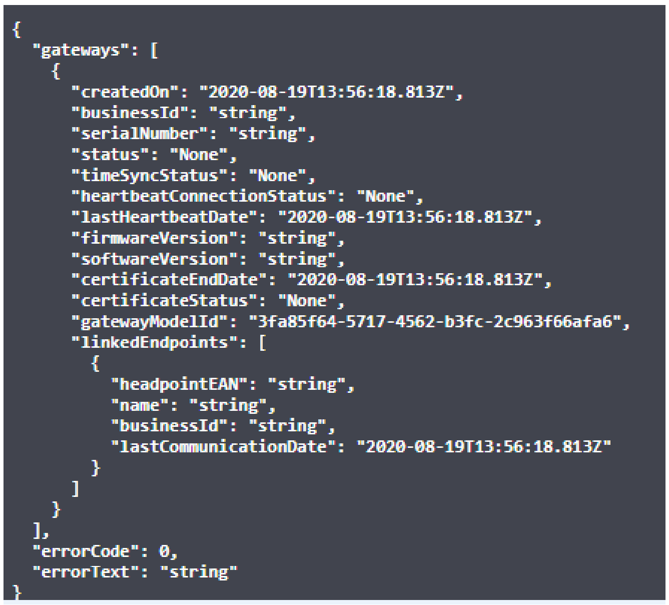
No Parameter. The account is deduced from the authentication.

Response

Fields	Type	Description
BusinessID	String	Gateway Business ID
SerialNumber	String	Gateway SN
Status	Enum	Status (Active, Inactive)
CreatedOn	DateTime	Creation date and time
LastHeartbeatDate	DateTime	Last Heartbeat Received date and time
SWVersion	String	Software version
FWVersion	String	Firmware version
HeartbeatConnectionStatus	Enum	Connection Status (NotConnected, ConnectionFailed, Connected, ConnectionRequested)
LinkedEndpoints	Array of Endpoint	
CertificateStatus	Enum	Certificate status enum (TokenRequested, TokenReceived, TokenRejected, CertificateReceived, CertificateRegistered, CertificatedRevoked)
CertificateEndDate	DateTime	Certificate Validity until date and time
GatewayModelID	String	The Gateway Model Id
TimeSyncStatus	Enum	Time sync status (InsufficientResponse, UnacceptableLatency, TimeSyncSuccessful)

Endpoint

Fields	Type	Description
HeadpointEAN	String	Headpoint EAN
Name	String	Endpoint name
BusinessID	String	Endpoint business ID
LastCommunicationDate	DateTime	Last message received



```
{
  "gateways": [
    {
      "createdOn": "2020-08-19T13:56:18.813Z",
      "businessId": "string",
      "serialNumber": "string",
      "status": "None",
      "timeSyncStatus": "None",
      "heartbeatConnectionStatus": "None",
      "lastHeartbeatDate": "2020-08-19T13:56:18.813Z",
      "firmwareVersion": "string",
      "softwareVersion": "string",
      "certificateEndDate": "2020-08-19T13:56:18.813Z",
      "certificateStatus": "None",
      "gatewayModelId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
      "linkedEndpoints": [
        {
          "headpointEAN": "string",
          "name": "string",
          "businessId": "string",
          "lastCommunicationDate": "2020-08-19T13:56:18.813Z"
        }
      ]
    }
  ],
  "errorCode": 0,
  "errorText": "string"
}
```

7.4.1.2 Business Rules

No specific business rules

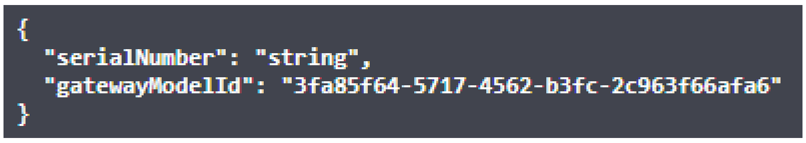
7.4.2 CreateGateway

Put: /api/interface/v1/Gateway

7.4.2.1 Domain

Request

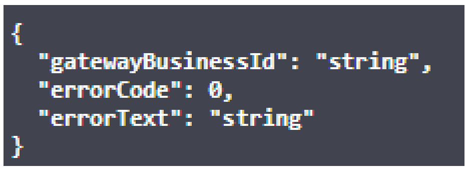
Fields	Type	Description
Serial Number	String	Gateway Serial Number
GatewayModelID	String	Gateway Model ID



```
{
  "serialNumber": "string",
  "gatewayModelId": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
}
```

Response

Fields	Type	Description
GWBusinessID	String	GW Bus ID



```
{
  "gatewayBusinessId": "string",
  "errorCode": 0,
  "errorText": "string"
}
```

7.4.2.2 Business Rules

ErrorCode	Error Text
0	
1	GW Model ID unknow for this account
2	Serial Number already existing for this account
100	Unexpected error

7.4.3 EditGateway

Post: /api/interface/v1/Gateway/{gatewayBusinessId}

7.4.3.1 Domain

Request

Fields	Type	Description
Serial Number	String	Gateway Serial Number
GatewayModelID	String	Gateway Model ID

```
{
  "gatewayModelId": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
  "serialNumber": "string"
}
```

Response

```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.4.3.2 Business Rules

ErrorCode	Error Text
0	
1	GW Business ID not existing for current account
2	GW Serial Number is already existing for another GW
3	GW Model ID not existing for current account
100	Unexpected error

7.4.4 ActivateGateway

Post: /api/interface/v1/Gateway/{gatewayBusinessId}/activate

7.4.4.1 Domain

Request

No Parameter. The function will put the Gateway status to active.

Response

```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.4.4.2 Business Rules

ErrorCode	Error Text
0	
1	GW Business ID not existing for current account
2	GW status is already active
100	Unexpected error

7.4.5 DeactivateGateway

Post: /api/interface/v1/Gateway/{gatewayBusinessId}/deactivate

7.4.5.1 Domain

Request

No Parameter. The function will put the GW status to inactive.

Response

```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.4.5.2 Business Rules

ErrorCode	Error Text
0	
1	GW Business ID not existing for current account
2	GW status is already inactive
3	GW is currently linked to an EP
100	Unexpected error

7.4.6 RequestCertificateToken

Post: /api/interface/v1/Gateway/{gatewayBusinessId}/certificate

7.4.6.1 Domain

Request

Fields	Type	Description
GatewayBusinessId	String	Gateway Business Id

Response

Fields	Type	Description
VerificationCode	String	The verification code

7.4.6.2 Business Rules

7.5 System Operator Management

7.5.1 GetSystemOperatorList

Get: /api/interface/v1/SystemOperator

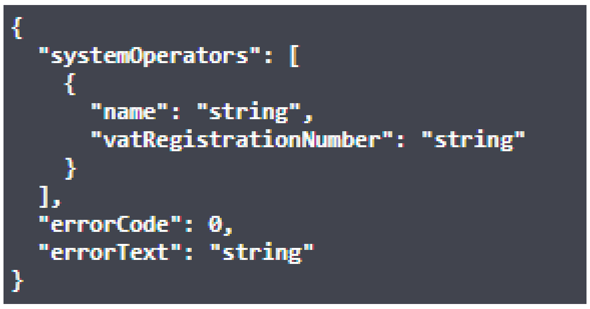
7.5.1.1 Domain

Request

No Parameter.

Response

Fields	**Type**	**Description**
SystemOperators	Array of SystemOperator	List of System Operators
SystemOperator		
Fields	Type	Description
ID	string	Id of the System Operator
Name	string	Name of System Operator
VATRegistrationNumber	DateTime	VAT of System Operator



```
{
  "systemOperators": [
    {
      "name": "string",
      "vatRegistrationNumber": "string"
    }
  ],
  "errorCode": 0,
  "errorText": "string"
}
```

7.5.1.2 Business Rules

No specific business rules

7.6 EndPoint Management

7.6.1 GetEndpointList

Get: /api/interface/v1/Endpoint

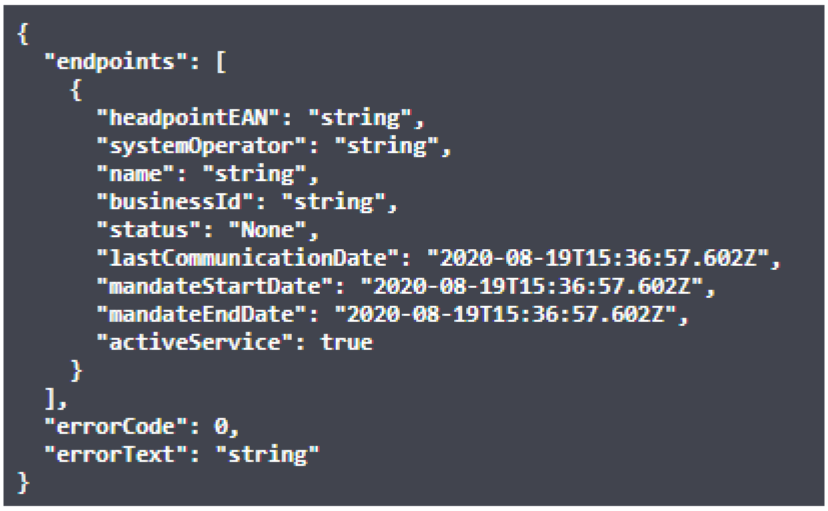
7.6.1.1 Domain

Request

No Parameter. The account is deduced from the authentication.

Response

Fields	Type	Description
HeadpointEAN	String	Headpoint EAN
System operator	String	System operator name
Name	String	Endpoint name
BusinessID	String	Endpoint business ID
Status	enum	The endpoint status
LastCommunicationDate	DateTime	Last message received
Madate Start Date	DateTime	Mandate start date
Mandate End Date	DateTime	Mandate end date
Active service	Boolean	Is an active service present on EP



```
{
  "endpoints": [
    {
      "headpointEAN": "string",
      "systemOperator": "string",
      "name": "string",
      "businessId": "string",
      "status": "None",
      "lastCommunicationDate": "2020-08-19T15:36:57.602Z",
      "mandateStartDate": "2020-08-19T15:36:57.602Z",
      "mandateEndDate": "2020-08-19T15:36:57.602Z",
      "activeService": true
    }
  ],
  "errorCode": 0,
  "errorText": "string"
}
```

7.6.1.2 Business Rules

No specific business rules

7.6.2 CreateEndpoint

Post: /api/interface/v1/Endpoint/{headpointEAN}

7.6.2.1 Domain

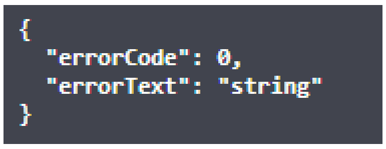
Request

Fields	Type	Description
headpointEAN	String	HeadPoint EAN
FriendlyName	String	EP Friendly Name
SystemOperatorId	String	System Operator Id
AccessHolder	boolean	Are you the contract access Holder (true/ false)
CPDesignationDocument	PDF (optional)	CP Designation document

Response

The response contains the endpoint business id and error code & text.

Fields	Type	Description
BusinessID	String	Endpoint business ID



```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.6.2.2 Business Rules

ErrorCode	Error Text
0	
1	EAN format invalid
2	EP Friendly name is already in use or empty
3	System operator is unknown
4	AccessHolder or CP Designation document needed
5	Document extension not allowed
100	Unexpected error

7.6.3 EditEndpoint

Post: /api/interface/v1/Endpoint/{endpointBusinessId}

7.6.3.1 Domain

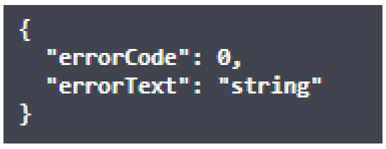
Request

Fields	Type	Description
endpointBusinessId	String	EP Business ID
friendlyName	String	EP friendly name

cpDesignationDocument	String (optional)	Communication Platform DesignationDocument

Response

The response contains only error code and error text.



```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.6.3.2 Business Rules

ErrorCode	**Error Text**
0	
1	EP Business ID not existing for current account
2	EP status is inactive
3	No new information provided (FriendlyName and CPUser Doc empty)
4	Incorrect format of the CP User Doc
5	Friendly name is already in use for headpoint
100	Unexpected error

7.6.4 ActivateEndpoint

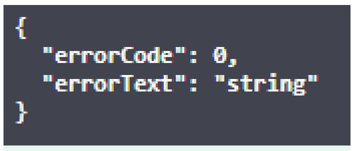
Post: /api/interface/v1/Endpoint/{endpointBusinessId}/activate

7.6.4.1 Domain

Request

No Parameter. The function will put the Endpoint status to active.

Response



```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.6.4.2 Business Rules

ErrorCode	**Error Text**
0	
1	EP Business ID not existing for current account
2	EP status is already active
100	Unexpected error

7.6.5 DeactivateEndpoint

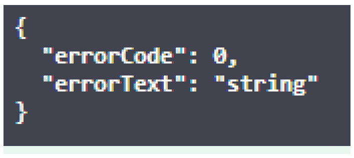
Post: /api/interface/v1/Endpoint/{endpointBusinessId}/deactivate

7.6.5.1 Domain

Request

No Parameter. The function will put the Endpoint status to inactive.

Response



```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.6.5.2 Business Rules

ErrorCode	Error Text
0	
1	EP Business ID not existing for current account
2	EP status is already inactive
3	EP is currently linked to a GW
4	EP is currently linked to a endpoint service
100	Unexpected error

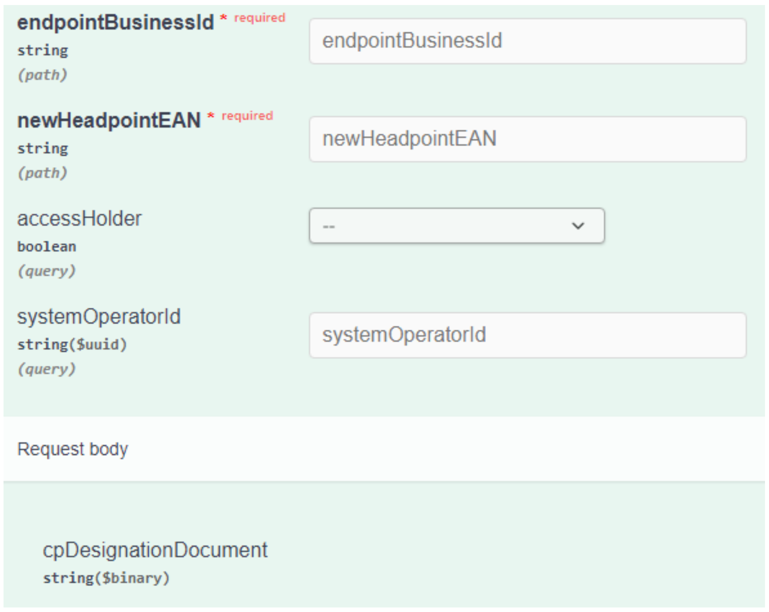
7.6.6 MoveEndpoint

Post: /api/interface/v1/Endpoint/{endpointBusinessId}/move/{newHeadpointEAN}

7.6.6.1 Domain

Request

Fields	Type	Description
endpointBusinessId	String	EP Business ID
newHeadpointEAN	EAN	New Headpoint EAN
accessHolder	Boolean	Are you the contract access holder (true/false)
systemOperatorId	string	The system operator id
cpDesignationDocument	String (optional)	Communication Platform Designation Document



endpointBusinessId * required
string
(path)
endpointBusinessId

newHeadpointEAN * required
string
(path)
newHeadpointEAN

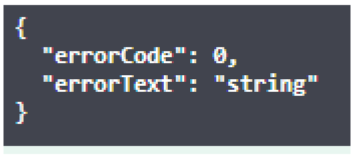
accessHolder
boolean
(query)
--

systemOperatorId
string($uuid)
(query)
systemOperatorId

Request body

cpDesignationDocument
string($binary)

Response



```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.6.6.2 Business Rules

ErrorCode	Error Text
0	
1	EAN format invalid
2	EP not existsing for current account
3	System operator is unknown
4	AccessHolder or CP Designation document needed
100	Unexpected error

7.6.7 LinkEndpointToGateway

Post: /api/interface/v1/Endpoint/{endpointBusinessId}/link/{gatewayBusinessId}

7.6.7.1 Domain

Request

Fields	Type	Description
endpointBusinessId	String	EP Business ID
gatewayBusinessId	Date	GW Business ID

Response

```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.6.7.2 Business Rules

ErrorCode	Error Text
0	
1	GW Business ID not existing for current account
2	GW status is inactive
3	EP Business ID not existing for current account
4	EP status is inactive
5	EP is already coupled to another GW
6	GW is already coupled to another EP in another HP
100	Unexpected error

7.6.8 DecoupleEndpointFromGateway

Post: /api/interface/v1/Endpoint/{endpointBusinessId}/decouple/{endDate}

7.6.8.1 Domain

Request

Fields	Type	Description
endpointBusinessId	String	EP Business ID
endDate	Date	Date of decoupling

Response

```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.6.8.2 Business Rules

ErrorCode	Error Text
0	
1	EP Business ID not existing for current account
2	EP is not coupled to any GW
3	Date of decoupling has to be in the future
100	Unexpected error

7.6.9 ReplaceGateway

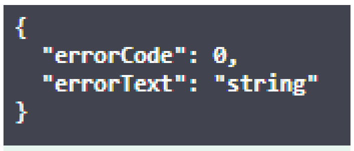
Post: /api/interface/v1/Endpoint/{endpointBusinessId}/replace/{gatewayBusinessId}/{replacementDate}

7.6.9.1 Domain

Request

Fields	Type	Description
endpointBusinessId	String	EP Business ID
gatewayBusinessId	String	GW Business ID
replacementDate	Date	Date of replacement

Response



```
{
  "errorCode": 0,
  "errorText": "string"
}
```

7.6.9.2 Business Rules

ErrorCode	Error Text
0	
1	EP Business ID not existing for current account
2	GW Business ID not existing for current account
3	EP is inactive
4	EP is not linked to any GW at the moment
5	GW is already linked to another HP
100	Unexpected error

7.7 Data Source / Statistics

7.7.1 GetDashboardStatistics

Get: /api/interface/v1/DataSource/active/metrics

7.7.1.1 Domain

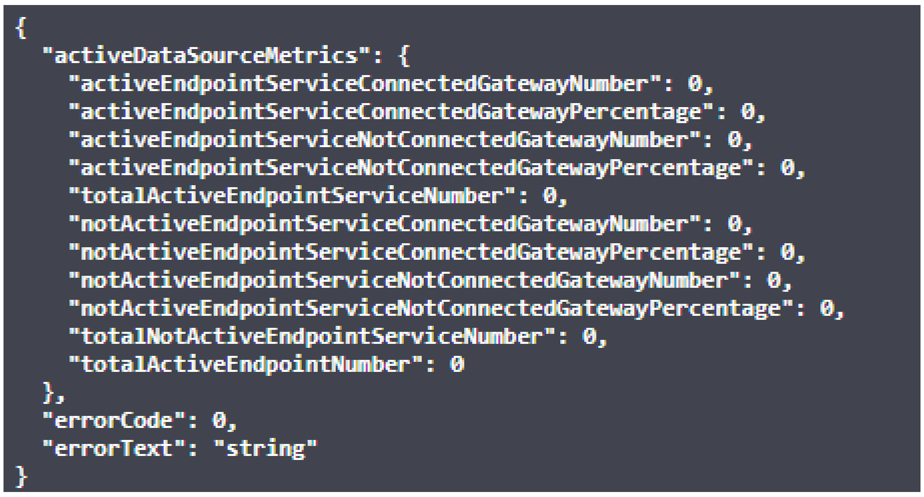
Request

No Parameter.

Response

Fields	Type	Description
ActiveEndpointServiceConnectedNumber	int	Number of active data source with Active Service that are Connected
ActiveEndpointServiceConnectedPercentage	double	Percentage of active data source with Active Service that are Connected (value between 0 and 1)
ActiveEndpointServiceNotConnectedNumber	int	Number of active data source with Active Service that are Not Connected
ActiveEndpointServiceNotConnectedPercentage	double	Percentage of active data source with Active Service that are Not Connected (value between 0 and 1)
TotalActiveEndpointServiceNumber	int	Total number of active data source with active service
NotActiveEndpointServiceConnectedGatewayNumber	int	Number of active data source without Active Service that are Connected
NotActiveEndpointServiceConnectedGatewayPercentage	double	Percentage of active data source without Active Service that are Connected (value between 0 and 1)
NotActiveEndpointServiceNotConnectedGatewayNumber	int	Number of active data source without Active Service that are Not Connected

NotActiveEndpointServiceNotConnectedGatewayPercentage	double	Percentage of active data source without Active Service that are Not Connected (value between 0 and 1)
TotalNotActiveEndpointServiceNumber	int	Total number of active data source without active service
TotalActiveEndpointNumber	int	Total number of active data source



```
{
  "activeDataSourceMetrics": {
    "activeEndpointServiceConnectedGatewayNumber": 0,
    "activeEndpointServiceConnectedGatewayPercentage": 0,
    "activeEndpointServiceNotConnectedGatewayNumber": 0,
    "activeEndpointServiceNotConnectedGatewayPercentage": 0,
    "totalActiveEndpointServiceNumber": 0,
    "notActiveEndpointServiceConnectedGatewayNumber": 0,
    "notActiveEndpointServiceConnectedGatewayPercentage": 0,
    "notActiveEndpointServiceNotConnectedGatewayNumber": 0,
    "notActiveEndpointServiceNotConnectedGatewayPercentage": 0,
    "totalNotActiveEndpointServiceNumber": 0,
    "totalActiveEndpointNumber": 0
  },
  "errorCode": 0,
  "errorText": "string"
}
```

7.7.1.2 Business Rules

No specific business rules

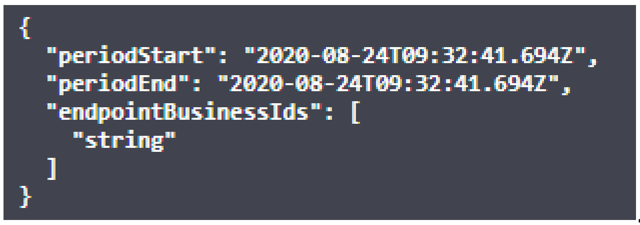
7.7.2 GetEndpointStatistics

Put: /api/interface/v1/Endpoint/Statistics

7.7.2.1 Domain

Request

Fields	**Type**	**Description**
periodStart	DateTime	Start of the statistic period
periodEnd	DateTime	End of the statistic period
endpointBusinessIds	Array of string	List of EP Business ID's



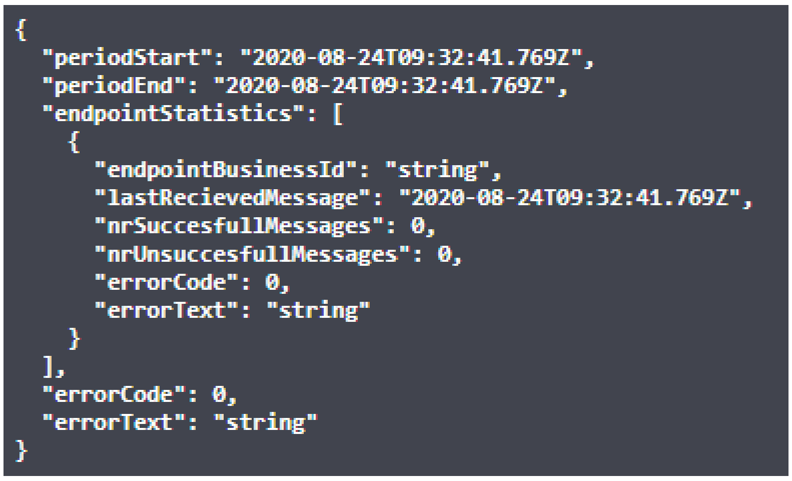
```
{
  "periodStart": "2020-08-24T09:32:41.694Z",
  "periodEnd": "2020-08-24T09:32:41.694Z",
  "endpointBusinessIds": [
    "string"
  ]
}
```

.

Response

Receive a collection of Gateway statistics.

Response:		
Fields	Type	Description
periodStart	DateTime	Start of the statistic period
periodEnd	DateTime	End of the statistic period
endpointStatisticArray	EPStatistic	EP Statistics
EPStatistic		
Fields	Type	Description
EP Busines ID	string	EP Business ID
LastReceivedMessage	DateTime	Last Received Message
NrSuccesfullMessages	int	Messages with successful status
NrUnsuccesfullMessages	int	Messages with other status



```
{
  "periodStart": "2020-08-24T09:32:41.769Z",
  "periodEnd": "2020-08-24T09:32:41.769Z",
  "endpointStatistics": [
    {
      "endpointBusinessId": "string",
      "lastRecievedMessage": "2020-08-24T09:32:41.769Z",
      "nrSuccesfullMessages": 0,
      "nrUnsuccesfullMessages": 0,
      "errorCode": 0,
      "errorText": "string"
    }
  ],
  "errorCode": 0,
  "errorText": "string"
}
```

7.7.2.2 Business Rules

The EP array size is currently limited to 50

The maximum length of the statistic period is 24h located in the last 3 monthes.

ErrorCode	**Error Text**
0	
1	EP Business ID not existing for current account
2	EP Business ID not existing for current account for current interval

3	EP Business ID status is inactive
100	Unexpected error

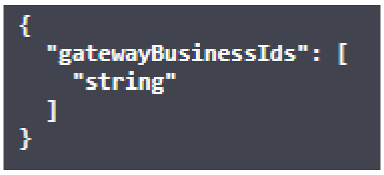
7.7.3 GetGatewayStatistics

Put: /api/interface/v1/Gateway/statistics

7.7.3.1 Domain

Request

Fields	**Type**	**Description**
GatewayBusinessIds	Array of string	List of GW Business ID's

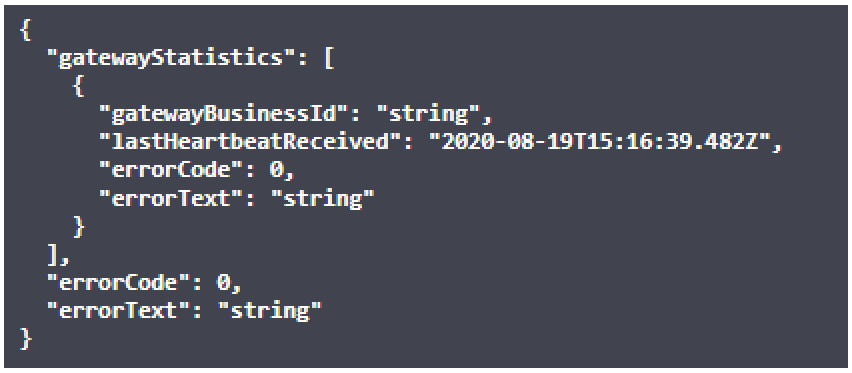


```
{
  "gatewayBusinessIds": [
    "string"
  ]
}
```

Response

Receive a collection of Gateway statistics.

Response:		
Fields	Type	Description
GWStatisticArray	GWStatistic	GW Statistics
GWStatistic		
Fields	Type	Description
GW Busines ID	string	GW Business ID
LastHearbeatReceived	DateTime	Last Heartbeat Received



```
{
  "gatewayStatistics": [
    {
      "gatewayBusinessId": "string",
      "lastHeartbeatReceived": "2020-08-19T15:16:39.482Z",
      "errorCode": 0,
      "errorText": "string"
    }
  ],
  "errorCode": 0,
  "errorText": "string"
}
```

7.7.3.2 Business Rules

The GW array size is currently limited to 50

ErrorCode	Error Text
0	
1	GW Business ID not existing for current account
2	GW Business ID status is inactive
100	Unexpected error

7.8 Business Rules

The following business rules will be implemented according to the used method.

7.8.1 All indicated time is in UTC

All time is in UTC. Period start are included, period end is excluded.

8 Annexes

8.1 Product messages

For each product, separate message are defined. These all use the common header and encryption as defined in 4.2.2

8.1.1 aFRR body

Element	Data Type	Origin	Description
SDP – SDP EAN	String	SCADA / FSP BE	The aFRR service delivery point EAN number.
DPM – DPmeasured	Decimal (JSON)	Metering device	The instantaneous net (gross if the net value cannot be measured) power measurement (in MW) per delivery point.
DPB – DPbaseline	Decimal (JSON)	SCADA / FSP BE	The power (in MW) that the delivery point would have injected/consumed without the activation of aFRR service. The baseline is sent 60 seconds in advance.
AS – DPaFRR	Integer (JSON)	SCADA / FSP BE	This is a logical (0 or 1) signal that indicates whether the delivery point is delivering the service for the concerned timeframe.
PS – DPaFRR, supplied	Decimal (JSON)	SCADA / FSP BE	The number of MW of ΔPsec_tot4 that is attributed by the BSP to the delivery point in question.
MTS – Measure timestamp	Ticks (UTC)	Metering device / gateway	The datetime on which the snapshot of the Pmeasured is taken. The Pbaseline in this message represents its value for this timestamp + 1 minute in the future. As described in paragraph 3, this timestamp has to be an exact multiple of 4 seconds (without some ms delay).

8.1.2 FCR body

Messages from technical units under a virtual delivery point can be sent at the timestamps of the aggregated signal in an event driven manner, so that only deviations are communicated towards the communication platform. In case of no deviations, a refresh value should be sent each 5 minutes.

Element	Data Type	Origin	Description
SDP – SDP EAN	String	SCADA / FSP BE	The deviceId of the participating unit
DPM – DPmeasured	Decimal (JSON)	Metering device	The instantaneous net (gross if the net value cannot be measured) power

			measurement (in MW) per delivery point.
MTS – Measure timestamp	Ticks (UTC)	Metering device / gateway	The datetime on which the snapshot of the measured object is taken