FlexMeasures Documentation

Release 0.3.2

Seita B.V.

Mar 02, 2021

CONTENTS

In a world with renewable energy, flexibility is crucial and valuable. Planning ahead allows flexible assets to serve the whole system with their flexibility, e.g. by shifting or curtailing energy use. This can also be profitable for their owners.

The FlexMeasures Platform is a tool for scheduling flexible actions for energy assets. For this purpose, it performs three services:

- Monitoring of incoming measurements
- Forecasting of expected measurements
- Scheduling flexible actions with custom optimization

For more on FlexMeasures services, read services.

Using FlexMeasures benefits operators as well as asset owners, by allowing or automation, insight, autonomy and profit sharing. For more on benefits, consult benefits.

FlexMeasures is compliant with the Universal Smart Energy Framework (USEF). Therefore, this documentation uses USEF terminology, e.g. for role definitions. The intended users of FlexMeasures are a Supplier (energy company) and its Prosumers (asset owners who have energy contracts with that Supplier). The platform operator of FlexMeasures can be an Aggregator.

CHAPTER

ONE

FLEXMEASURES CHANGELOG

1.1 v0.2.3 | February 27, 2021

1.1.1 New features

- Power charts available via the API [see PR #39]
- User management via the API [see PR #25]
- Better visibility of asset icons on maps [see PR #30]

Note: Read more on these features on the FlexMeasures blog.

1.1.2 Bugfixes

- Fix maps on new asset page (update MapBox lib) [see PR #27]
- Some asset links were broken [see PR #20]
- Password reset link on account page was broken [see PR #23]

1.1.3 Infrastructure/Support

- CI via Github Actions [see PR #1]
- Integration with timely beliefs lib: Sensors [see PR #13]
- Apache 2.0 license [see PR #16]
- Load js & css from CDN [see PR #21]
- Start using marshmallow for input validation, also introducing `HTTP status 422` in the API [see PR #25]
- Replace `solarpy` with `pvlib` (due to license conflict) [see PR #16]
- Stop supporting the creation of new users on asset creation (to reduce complexity) [see PR #36]

CHAPTER

TWO

DASHBOARD

The dashboard shows where the user's assets are located and how many different asset types are connected to the platform. The view serves to quickly identify the status of assets, such as whether they are taking part in any flexibility actions or whether there are upcoming opportunities to valorise on flexibility actions. In particular, the page contains:

2.1 Interactive map of assets

The map shows all of the user's assets, using color codes to indicate whether the platform has identified upcoming flexibility opportunities. Clicking on an asset allows the user to navigate to the *Flexibility actions* page to verify the offered flexibility action and to possibly order the action.

2.2 Summary of asset types

The summary below the map lists all asset types that the user has hooked up to the platform.

img/screenshot_dashboard.png

CHAPTER

THREE

PORTFOLIO OVERVIEW

The portfolio overview shows results and opportunities regarding the user's asset portfolio. The view serves to quickly identify upcoming opportunities to valorise on flexibility actions. In particular, the page contains:

3.1 Financial statements about energy and flexibility actions

The financial statements separate the effects of energy consumption/production and flexibility actions over two tables.

3.1.1 Statements about energy

The top table lists the effects of energy trading for each asset type in the user's portfolio. Production and consumption values are total volumes within the selected time window.[1]

Costs and revenues are calculated based on the relevant market prices for the user within the selected time window. A consumer will only have costs, while a prosumer may have both costs and revenues. A supplier always has both costs and revenues, since it trades energy both with its customers and with external markets. Finally, the financial statements show the total profit or loss per asset.

3.1.2 Statements about flexibility actions

The bottom table lists the effects of flexibility actions for each asset type in the user's portfolio. Separate columns are stated for each type of action, e.g. curtailment and shifting (see *Types of flexibility actions*), with relevant total volumes within the selected time window.[2]

Costs and revenues are calculated based on the internal method for profit sharing. Asset owners that provide flexibility actions via the platform will generate revenues. Suppliers that order flexibility action via the platform will generate both costs and revenues, where the revenues come from interacting with external markets. Finally, the financial statements show the total profit or loss per asset.

[1] For time windows that include future time slots, future values are based on forecasts.

3.2 Power profile measurements and forecasts

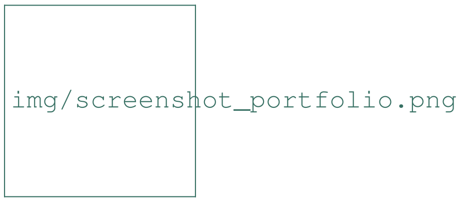
The power profile shows total production and consumption over the selected time window. A switch allows the user to view the contribution of each asset type to either total as a stacked plot. Past time slots show measurement data, whereas future time slots show forecasts. When flexibility opportunities exist in the selected time window, the plot is overlaid with highlights (see *Opportunities to valorise on flexibility actions*).

3.3 Changes to the power profile due to flexibility actions

Just below the power profile, the net effect of flexibility actions that have previously been ordered is plotted. The profile indicates the change in power resulting from actions that are planned in the future, and from actions that had been planned in the past. Positive values indicate an increase in production or a decrease in consumption, both of which result in an increased load on the network. For short-term flexibility actions, this is sometimes called up-regulation. Negative values indicate a decrease in production or an increase in consumption, which result in a decreased load on the network (down-regulation). When flexibility opportunities exist in the selected time window, the plot is overlaid with highlights (see *Opportunities to valorise on flexibility actions*).

3.4 Opportunities to valorise on flexibility actions

When flexibility opportunities exist in the selected time window, plots are overlaid with highlights indicating time slots in which flexibility actions can be taken in the future or were missed in the past. The default time window (the next 24 hours) shows immediately upcoming opportunities to valorise on flexibility actions. The user can follow up on identified opportunities by taking a flexibility action on the *Flexibility actions* page.



CHAPTER

FOUR

FLEXIBILITY ACTIONS

Flexibility actions have commercial value that users can valorise on. In the Flexibility actions page, FlexMeasures shows all flexibility actions that the user can take for a selected time window. When FlexMeasures has identified commercial value for flexibility actions, the user is suggested to order them. The user can opt to automate this otherwise manual process. Listed flexibility actions include previously ordered actions and currently offered actions. Currently offered actions are presented as an order book, where they are sorted according to their commercial value. The user can place orders and check the expected value of offers.

4.1 Types of flexibility actions

The platform distinguishes between different types of flexibility actions that an asset can take.

4.1.1 Curtailment

Curtailment happens when an asset temporarily lowers or stops its production or consumption. A defining feature of curtailment is that total production or consumption at the end of the flexibility action has decreased.

- A typical example of curtailing production is when a wind turbine adjusts the pitch angle of its blades to decrease the generator torque.
- An example of curtailing consumption is load shedding of energy intensive industries.

Curtailment offers may specify some freedom in terms of how much energy can be curtailed. In these cases, the user can select the energy volume (in MWh) to be ordered, within constraints set by the relevant Prosumer. The net effect of a curtailment action is also measured in terms of an energy volume (see the flexibility metrics in the *Portfolio overview* page). Note that the volume ordered is not necessarily equal to the volume curtailed: the ordered volume relates only to the selected time window, while the curtailed volume may include volumes outside of the selected time window. For example, an asset that runs an all-or-nothing consumption process of 2 hours can be ordered to curtail consumption for 1 hour, but will in effect stop the entire process. In this case, the curtailed volume will be higher than the ordered volume, and the platform will take into account the total expected curtailment in its calculations.

4.1.2 Shifting

Shifting happens when an asset delays or advances its energy production or consumption. A defining feature of shifting is that total production or consumption at the end of the flexibility action remains the same.

- An example of delaying consumption is when a charging station postpones the charging process of an electric vehicle.
- An example of advancing consumption is when a cooling unit starts to cool before the upper temperature bound was reached (pre-cooling).

Shifting offers may specify some freedom in terms of how much energy can be shifted. In these cases, the user can select the energy volume (in MWh) to be ordered, within constraints set by the relevant Prosumer. This energy volume represents how much energy is shifting into or out of the selected time window. The net effect of a shifting action is measured in terms of an energy-time volume (see the flexibility metrics in the *Portfolio overview* page). This volume is a multiplication of the energy volume being shifted and the duration of that shift.

4.2 Visualisation of actions

Flexibility actions cause changes to the power profile of an asset. Depending on the time window selection and constraints set by the asset owner, the effects of an action may partially take place outside of the selected time window. Such effects are taken into account by FlexMeasures and shown to the user, e.g. as a part of expected value calculations and power profile forecasts.

img/screenshot_control.png

CHAPTER

FIVE

CLIENT ANALYTICS

The client analytics page shows relevant data to an asset's operation: production and consumption, market prices and weather data. The view serves to browse through available data and to assess how the app is monitoring and forecasting data streams from various sources. In particular, the page contains:

5.1 Data browsing

…

5.2 Data visualisation

…

5.3 Statistics

…

img/screenshot_analytics.png

CHAPTER SIX

ADMINISTRATION

The administrator can edit assets and user accounts here.

6.1 Assets

Edit meta data

6.2 Users

None yet.

CHAPTER

SEVEN

INTRODUCTION

This document details the Application Programming Interface (API) of the FlexMeasures web service. The API supports user automation for flexibility valorisation in the energy sector, both in a live setting and for the purpose of simulating scenarios. The web service adheres to the concepts and terminology used in the Universal Smart Energy Framework (USEF). We assume in this document that the FlexMeasures instance you want to connect to is hosted at https://company.flexmeasures.io.

New versions of the API are released on:

```
https://company.flexmeasures.io/api
```

A list of services offered by (a version of) the FlexMeasures web service can be obtained by sending a *getService* request. An optional parameter "access" can be used to specify a user role for which to obtain only the relevant services.

Example request

```
{
    "type": "GetServiceRequest",
    "version": "1.0"
}
```

Example response

```
{
    "type": "GetServiceResponse",
    "version": "1.0",
    "services": [
        {
            "name": "getMeterData",
            "access": ["Aggregator", "Supplier", "MDC", "DSO", "Prosumer", "ESCo"],
            "description": "Request meter reading"
        },
        {
            "name": "postMeterData",
            "access": ["MDC"],
            "description": "Send meter reading"
        }
    ]
}
```

7.1 Authentication

Service usage is only possible with a user access token specified in the request header, for example:

```
{
    "Authorization": "<token>"
}
```

A fresh "<token>" can be generated on the user's profile after logging in:

```
https://company.flexmeasures.io/account
```

or through a POST request to the following endpoint:

```
https://company.flexmeasures.io/api/requestAuthToken
```

using the following JSON message for the POST request data:

```
{
    "email": "<user email>",
    "password": "<user password>"
}
```

Note that each access token has a limited lifetime, see auth.

7.2 Roles

We distinguish the following roles with different access rights to the individual services. Capitalised roles are defined by USEF:

- public
- user
- admin
- Aggregator
- Supplier: an energy retailer (see supplier)
- Prosumer: an asset owner (see prosumer)
- ESCo: an energy service company (see esco)
- MDC: a meter data company (see mdc)
- DSO: a distribution system operator (see dso)

7.3 Sources

Requests for data may limit the data selection by specifying a source, for example, a specific user. USEF roles are also valid source selectors. For example, to obtain data originating from either a meter data company or user 42, include the following:

```
{
    "sources": ["MDC", "42"],
}
```

7.4 Notation

All requests and responses to and from the web service should be valid JSON messages.

7.4.1 Singular vs plural keys

Throughout this document, keys are written in singular if a single value is listed, and written in plural if multiple values are listed, for example:

```
{
    "keyToValue": "this is a single value",
    "keyToValues": ["this is a value", "and this is a second value"]
}
```

The API, however, does not distinguish between singular and plural key notation.

7.4.2 Connections

Connections are end points of the grid at which an asset is located. Connections should be identified with an entity address following the EA1 addressing scheme prescribed by USEF[1], which is mostly taken from IETF RFC 3720 [2]:

This is the complete structure of an EA1 address:

```
{
    "connection": "ea1.{date code}.{reversed domain name}:{locally unique string}"
}
```

Here is a full example for a FlexMeasures connection address:

```
{
    "connection": "ea1.2021-02.io.flexmeasures.company:30:73"
}
```

where FlexMeasures runs at *company.flexmeasures.io* and the owner ID is 30 and the asset ID is 73. The owner ID is optional. Both the owner ID and the asset ID, as well as the full entity address can be obtained on the asset's listing after logging in:

```
https://company.flexmeasures.io/assets
```

Some deeper explanations about an entity address:

- "ea1" is a constant, indicating this is a type 1 USEF entity address

- The date code "must be a date during which the naming authority owned the domain name used in this format, and should be the first month in which the domain name was owned by this naming authority at 00:01 GMT of the first day of the month.
- The reversed domain name is taken from the naming authority (person or organization) creating this entity address
- The locally unique string can be used for local purposes, and FlexMeasures uses it to identify the resource (more information in parse_entity_address).

TODO: This needs to be in the FlexMeasures documentation.

[1] https://www.usef.energy/app/uploads/2020/01/USEF-Flex-Trading-Protocol-Specifications-1.01.pdf [2] https://tools.ietf.org/html/rfc3720

Notation for simulation

For version 1 of the API, the following simplified addressing scheme may be used:

```
{
    "connection": "<owner-id>:<asset-id>"
}
```

or even simpler:

```
{
    "connection": "<asset-id>"
}
```

7.4.3 Groups

Data such as measurements, load prognoses and tariffs are usually stated per group of connections. When the attributes "start", "duration" and "unit" are stated outside of "groups" they are inherited by each of the individual groups. For example:

```
{
    "groups": [
        {
            "connections": [
                "CS 1",
                "CS 2"
            ],
            "values": [
                306.66,
                306.66,
                0,
                0,
                306.66,
                306.66
            ]
        },
        {
            "connection": "CS 3",
            "values": [
                306.66,
                0,
```

(continues on next page)

(continued from previous page)

```
                0,
                0,
                306.66,
                306.66
            ]
        }
    ],
    "start": "2016-05-01T12:45:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

In case of a single group of connections, the message may be flattened to:

```
{
    "connections": [
        "CS 1",
        "CS 2"
    ],
    "values": [
        306.66,
        306.66,
        0,
        0,
        306.66,
        306.66
    ],
    "start": "2016-05-01T12:45:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

7.4.4 Timeseries

Timestamps and durations are consistent with the ISO 8601 standard. All timestamps in requests to the API must be timezone aware. The timezone indication "Z" indicates a zero offset from UTC. Additionally, we use the following shorthand for sequential values within a time interval:

```
{
    "values": [
        10,
        5,
        8
    ],
    "start": "2016-05-01T13:00:00Z",
    "duration": "PT45M"
}
```

is equal to:

```
{
    "timeseries": [
        {
            "value": 10,
            "start": "2016-05-01T13:00:00Z",
```

(continues on next page)

(continued from previous page)

```
                "duration": "PT15M"
            },
            {
                "value": 5,
                "start": "2016-05-01T13:15:00Z",
                "duration": "PT15M"
            },
            {
                "value": 8,
                "start": "2016-05-01T13:30:00Z",
                "duration": "PT15M"
            }
        ]
}
```

This intuitive convention allows us to reduce communication by sending univariate timeseries as arrays.

Notation for v1

For version 1 of the API, only univariate timeseries data is expected to be communicated. Therefore:

- only the array notation should be used,
- "start" should be a timestamp on the hour or a multiple of 15 minutes thereafter, and
- "duration" should be a multiple of 15 minutes.

7.4.5 Prognoses

When POSTing a prognosis, the message should state a time horizon, i.e. the duration between the time at which the prognosis was made and the time of realisation (commonly at the end of the prognosed time interval). The horizon can be stated explicitly by including a "horizon", consistent with the ISO 8601 standard, as follows:

```
{
    "values": [
        10,
        5,
        8
    ],
    "start": "2016-05-01T13:00:00Z",
    "duration": "PT45M",
    "horizon": "PT6H"
}
```

This message implies that the entire prognosis was made at 7:45 AM UTC, i.e. 6 hours before the end of the time interval. Alternatively, a rolling horizon can be stated as an ISO 8601 repeating time interval:

```
{
    "values": [
        10,
        5,
        8
    ],
    "start": "2016-05-01T13:00:00Z",
    "duration": "PT45M",
```

(continues on next page)

(continued from previous page)

```
    "horizon": "R/PT6H"
}
```

Here, the number of repetitions and the repeat rule is omitted as it is implied by our notation for univariate timeseries (a complete representation of the "horizon" would have been "R3/PT6H/FREQ=MI;INTR=15"). This message implies that the value for 1:00-1:15 PM was made at 7:15 AM, the value for 1:15-1:30 PM was made at 7:30 AM, and the value for 1:30-1:45 PM was made at 7:45 AM.

A "horizon" may be omitted, in which case the web service will infer the horizon from the arrival time of the message. Negative horizons may also be stated (breaking with the ISO 8601 standard) to indicate a prognosis about something that has already happened (i.e. after the fact, or simply *ex post*). For example, the following message implies that the entire prognosis was made at 1:55 PM UTC, 10 minutes after the fact:

```
{
    "values": [
        10,
        5,
        8
    ],
    "start": "2016-05-01T13:00:00Z",
    "duration": "PT45M",
    "horizon": "-PT10M"
}
```

For a rolling horizon indicating a prognosis 10 minutes after the start of each 15-minute interval, the "horizon" would have been "R/PT5M" since in fact only the last 5 minutes of each interval occurs before the fact (*ex ante*). That is, for ex-ante prognoses, the timeseries resolution (here 15 minutes) is included in the horizon, because the horizon is relative to the end of the timeseries.

7.4.6 Beliefs

By regarding all time series data as beliefs that have been recorded at a certain time, data can be filtered accordingly. Some GET endpoints have two optional timing parameters to allow such filtering. The "prior" parameter (a timestamp) can be used to select beliefs recorded before some moment in time. It can be used to "time-travel" to see the state of information at some moment in the past. In addition, the "horizon" parameter (a duration) can be used to select beliefs recorded before some moment in time, relative to each event. For example, to filter out meter readings communicated within a day (denoted by a negative horizon) or forecasts created at least a day beforehand (denoted by a positive horizon). In addition to these two timing filters, beliefs can be filtered by their source (see *Sources*).

The two timing parameters follow the ISO 8601 standard and are interpreted as follows:

- "horizon": recorded at least <duration> before the fact (indicated by a positive horizon), or at most <duration> after the fact (indicated by a negative horizon).
- "prior": recorded prior to <timestamp>.

For example:

```
{
    "horizon": "PT6H",
    "prior": "2020-08-01T17:00:00Z"
}
```

These parameters denote that the data should have been recorded at least 6 hours before the fact (i.e. forecasts) and prior to 5 PM on August 1st 2020 (UTC).

7.4.7 Resolutions

Specifying a resolution is redundant for POST requests that contain both “values” and a “duration”. Also, posted data is checked against the required resolution of the assets which are posted to.

GET requests (such as *getMeterData*) return data in the resolution which the sensor is configured for. A “resolution” may be specified explicitly to obtain the data in downsampled form, which can be very beneficial for download speed. The specified resolution needs to be a multiple of the asset’s resolution, e.g. hourly or daily values if the asset’s resolution is 15 minutes.

7.4.8 Units

Valid units for timeseries data in version 1 of the API are “MW” only.

7.4.9 Signs

USEF recommends to use positive power values to indicate consumption and negative values to indicate production, i.e. to take the perspective of the Prosumer. If an asset has been configured as a pure producer or pure consumer, the web service will help avoid mistakes by checking the sign of posted power values.

CHAPTER

EIGHT

SIMULATION

This document details examples for using a FlexMeasures server for simulation. The API on a server that is set up for simulation is extended with several features that make it possible to run simulations of energy flows and control actions. Please read the *Introduction* for explanations of the message fields, specifically regarding:

- The sign of values (*Signs*)
- Valid durations (*Resolutions*)
- Valid horizons (*Prognoses*)
- Valid units (*Simulation*)

Table of contents

8.1 Setting up

Researchers require an admin account to set up a new simulation with a number of assets.

8.1.1 Creating assets and owners

New assets can be created through the UI on:

```
https://company.flexmeasures.io/assets/new
```

We recommend that researchers choose their own admin account as the asset's owner. This way, the simulation will only require refreshing of the access token for the admin account. Alternatively, researchers can set up unique accounts for each agent in a multi-agent simulation by creating new owners. In this case, access tokens need to be refreshed by each agent separately.

8.1.2 Authentication

Service usage is only possible with a user authentication token specified in the request header, for example:

```
{
    "Authorization": "<token>"
}
```

The "<token>" can be obtained on your profile after logging in:

```
https://company.flexmeasures.io/account
```

For security reasons, tokens expire after a certain amount of time (see _auth). To automate token renewal, use the following POST endpoint:

```
https://company.flexmeasures.io/api/requestAuthToken
```

Providing applicable user credentials:

```
{
    "email": "<email>",
    "password": "<password>"
}
```

8.2 Posting weather data

Weather data (both observations and forecasts) can be posted to the following POST endpoint:

```
https://company.flexmeasures.io/api/<version>/postWeatherData
```

Weather data can be posted for the following three types of weather sensors:

- "radiation" (with kW/m^2 as unit)
- "temperature" (with °C as unit)
- "wind_speed" (with m/s as unit)

The sensor type is part of the unique entity address for each sensor, together with the sensor's latitude and longitude.

This "PostWeatherDataRequest" message posts temperature forecasts for 15-minute intervals between 3.00pm and 4.30pm for a weather sensor located at latitude 33.4843866 and longitude 126.477859. The forecasts were made at noon.

```json
{
    "type": "PostWeatherDataRequest",
    "sensor": "ea1.2018-06.io.flexmeasures.company:temperature:33.4843866:126.477859",
    "values": [
        20.04,
        20.23,
        20.41,
        20.51,
        20.55,
        20.57
    ],
    "start": "2015-01-01T15:00:00+09:00",
    "duration": "PT1H30M",
    "horizon": "PT4H30M",
    "unit": "°C"
}
```

8.2.1 Observations vs forecasts

To post an observation rather than a forecast, simply set the horizon to "PT0H". This denotes that the observation was made exactly after realisation of this list of temperature readings, i.e. at 4.30pm.

Alternatively, to indicate that each individual observation was made directly after the end of its 15-minute interval (i.e. at 3.15pm, 3.30pm and so on), set the horizon to "R/PT0H".

Finally, delays in reading out sensor data can be simulated by setting the horizon field to a negative value. For example, a horizon of "-PT1H" would denote that this list of temperature readings was observed one hour after the fact (i.e. at 5.30pm).

8.3 Posting price data

Price data (both observations and forecasts) can be posted to the following POST endpoint:

```
https://company.flexmeasures.io/api/<version>/postPriceData
```

This example "PostPriceDataRequest" message posts prices for hourly intervals between midnight and midnight the next day for the Korean Power Exchange (KPX) day-ahead auction. The horizon indicates that the prices were published at 3pm on December 31st 2014 (i.e. 33 hours ahead of midnight the next day).

```json
{
    "type": "PostPriceDataRequest",
    "market": "ea1.2018-06.io.flexmeasures.company:kpx_da",
    "values": [
        52.37,
        51.14,
        49.09,
        48.35,
        48.47,
```

(continues on next page)

(continued from previous page)

```
        49.98,
        58.7,
        67.76,
        69.21,
        70.26,
        70.46,
        70,
        70.7,
        70.41,
        70,
        64.53,
        65.92,
        69.72,
        70.51,
        75.49,
        70.35,
        70.01,
        66.98,
        58.61
    ],
    "start": "2015-01-01T15:00:00+09:00",
    "duration": "PT24H",
    "horizon": "PT33H",
    "unit": "KRW/kWh"
}
```

8.3.1 Observations vs forecasts

For markets, the time at which the market is cleared (i.e. when contracts are signed) determines the difference between an ex-post observation and an ex-ante forecast. For example, at the KPX day-ahead auction this is every day at 3pm. To post a forecast rather than an observation, simply increase the horizon. For example, a horizon of "PT57H" would denote a forecast of 24 hours ahead of clearing.

8.4 Posting power data

For power data, USEF specifies separate message types for observations and forecasts. Correspondingly, FlexMeasures uses separate endpoints to communicate these messages. Observations of power data can be posted to the following POST endpoint:

```
https://company.flexmeasures.io/api/<version>/postMeterData
```

while forecasts of power data can be posted to the following POST endpoint:

```
https://company.flexmeasures.io/api/<version>/postPrognosis
```

For both endpoints, power data can be posted in various ways. The following examples assume that the endpoint for power data observations (i.e. meter data) is used.

8.4.1 Single value, single connection

A single average power value for a 15-minute time interval for a single connection, posted 5 minutes after realisation.

```
{
    "type": "PostMeterDataRequest",
    "connection": "ea1.2018-06.io.flexmeasures.company:1:1",
    "value": 220,
    "start": "2015-01-01T00:00:00+00:00",
    "duration": "PT0H15M",
    "horizon": "-PT5M",
    "unit": "MW"
}
```

8.4.2 Multiple values, single connection

Multiple values (indicating a univariate timeseries) for 15-minute time intervals for a single connection, posted 5 minutes after realisation.

```
{
    "type": "PostMeterDataRequest",
    "connection": "ea1.2018-06.io.flexmeasures.company:1:1",
    "values": [
        220,
        210,
        200
    ],
    "start": "2015-01-01T00:00:00+00:00",
    "duration": "PT0H45M",
    "horizon": "-PT5M",
    "unit": "MW"
}
```

8.4.3 Single identical value, multiple connections

Single identical value for a 15-minute time interval for two connections, posted 5 minutes after realisation. Please note that both connections consumed at 10 MW, i.e. the value does not represent the total of the two connections. We recommend to use this notation for zero values only.

```
{
    "type": "PostMeterDataRequest",
    "connections": [
        "ea1.2018-06.io.flexmeasures.company:1:1",
        "ea1.2018-06.io.flexmeasures.company:1:2"
    ],
    "value": 10,
    "start": "2015-01-01T00:00:00+00:00",
    "duration": "PT0H15M",
    "horizon": "-PT5M",
    "unit": "MW"
}
```

8.4.4 Single different values, multiple connections

Single different values for a 15-minute time interval for two connections, posted 5 minutes after realisation.

```
{
    "type": "PostMeterDataRequest",
    "groups": [
        {
            "connection": "ea1.2018-06.io.flexmeasures.company:1:1",
            "value": 220
        },
        {
            "connection": "ea1.2018-06.io.flexmeasures.company:1:2",
            "value": 300
        }
    ],
    "start": "2015-01-01T00:00:00+00:00",
    "duration": "PT0H15M",
    "horizon": "-PT5M",
    "unit": "MW"
}
```

8.4.5 Multiple values, multiple connections

Multiple values (indicating a univariate timeseries) for 15-minute time intervals for two connections, posted 5 minutes after realisation.

```
{
    "type": "PostMeterDataRequest",
    "groups": [
        {
            "connection": "ea1.2018-06.io.flexmeasures.company:1:1",
            "values": [
                220,
                210,
                200
            ]
        },
        {
            "connection": "ea1.2018-06.io.flexmeasures.company:1:2",
            "values": [
                300,
                303,
                306
            ]
        }
    ],
    "start": "2015-01-01T00:00:00+00:00",
    "duration": "PT0H45M",
    "horizon": "-PT5M",
    "unit": "MW"
}
```

8.5 Getting prognoses

Prognoses are power forecasts that are used by FlexMeasures to determine the best control signals to valorise on balancing opportunities. Researchers can check the accuracy of these forecasts by downloading the prognoses and comparing them against the meter data, i.e. the realised power measurements. A prognosis can be requested for a single asset at the following GET endpoint:

```
https://company.flexmeasures.io/api/<version>/getPrognosis
```

This example requests a prognosis with a rolling horizon of 6 hours before realisation.

```
{
    "type": "GetPrognosisRequest",
    "connection": "ea1.2018-06.io.flexmeasures.company:1:1",
    "start": "2015-01-01T00:00:00+00:00",
    "duration": "PT24H",
    "horizon": "R/PT6H",
    "resolution": "PT15M",
    "unit": "MW"
}
```

8.6 Posting flexibility constraints

Prosumers that have Active Demand & Supply can post the constraints of their flexible devices to FlexMeasures at the following POST endpoint:

```
https://company.flexmeasures.io/api/<version>/postUdiEvent
```

This example posts a state of charge value for a battery device (asset 10 of owner 7) as UDI event 203.

```
{
    "type": "PostUdiEventRequest",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203:soc",
    "value": 12.1,
    "datetime": "2015-06-02T10:00:00+00:00",
    "unit": "kWh"
}
```

Some devices also accept target values for their state of charge. As an example, consider the same UDI event as above with an additional target value.

```
{
    "type": "PostUdiEventRequest",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:204:soc-with-targets",
    "value": 12.1,
    "datetime": "2015-06-02T10:00:00+00:00",
    "unit": "kWh",
    "targets": [
        {
            "value": 25,
            "datetime": "2015-06-02T16:00:00+00:00"
        }
    ]
}
```

8.7 Getting control signals

A Prosumer can query FlexMeasures for control signals for its flexible devices using the following GET endpoint:

```
https://company.flexmeasures.io/api/<version>/getDeviceMessage
```

Control signals can be queried by UDI event for up to 1 week after the UDI event was posted. This example requests a control signal for UDI event 203 posted previously.

```
{
    "type": "GetDeviceMessageRequest",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203:soc"
}
```

The following example response indicates that FlexMeasures planned ahead 45 minutes. The list of consecutive power values represents the target consumption of the battery (negative values for production). Each value represents the average power over a 15 minute time interval.

```
{
    "type": "GetDeviceMessageResponse",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203",
    "values": [
        2.15,
        3,
        2
    ],
    "start": "2015-06-02T10:00:00+00:00",
    "duration": "PT45M",
    "unit": "MW"
}
```

One way of reaching the target consumption in this example is to let the battery start to consume with 2.15 MW at 10am, increase its consumption to 3 MW at 10.15am and decrease its consumption to 2 MW at 10.30am. However, because the targets values represent averages over 15-minute time intervals, the battery still has some degrees of freedom. For example, the battery might start to consume with 2.1 MW at 10.00am and increase its consumption to 2.25 at 10.10am, increase its consumption to 5 MW at 10.15am and decrease its consumption to 2 MW at 10.20am. That should result in the same average values for each quarter-hour.

8.8 Resetting the server

All power, price and weather data on the simulation server can be cleared using the following PUT endpoint (admin rights are required):

```
https://company.flexmeasures.io/api/<version>/restoreData
```

This example restores the database to a backup named demo_v0, which contains no timeseries data.

```
{
    "backup": "demo_v0"
}
```

CHAPTER

NINE

VERSION 2.0

9.1 Summary

Resource	Operation	Description
Asset	*DELETE /api/v2_0/asset/(id)*	Delete an asset, together with its existing data.
	DELETE /api/v2_0/asset/(id)	
	GET /api/v2_0/asset/(id)	Get an asset
	GET /api/v2_0/asset/(id)	
	PATCH /api/v2_0/asset/(id)	Patch data for an existing asset
	PATCH /api/v2_0/asset/(id)	
	GET /api/v2_0/assets	Download asset list
	GET /api/v2_0/assets	
	POST /api/v2_0/assets	Post a new asset
	POST /api/v2_0/assets	
Chart	*GET /api/v2_0/charts/power*	Get a power chart
	GET /api/v2_0/charts/power	
Public	*GET /api/*	List available API versions
	POST /api/requestAuthToken	Obtain an authentication token
	GET /api/v2_0/getService	Obtain a service listing for this version
	GET /api/v2_0/getService	
User	*GET /api/v2_0/getConnection*	Retrieve entity addresses of connections
	GET /api/v2_0/getConnection	
	GET /api/v2_0/getDeviceMessage	Download control signal from the platform
	GET /api/v2_0/getDeviceMessage	
	GET /api/v2_0/getMeterData	Download meter data from the platform
	GET /api/v2_0/getMeterData	
	GET /api/v2_0/getPrognosis	Download prognosis from the platform
	GET /api/v2_0/getPrognosis	
	POST /api/v2_0/postMeterData	Upload meter data to the platform
	POST /api/v2_0/postMeterData	
	POST /api/v2_0/postPriceData	Upload price data to the platform
	POST /api/v2_0/postPriceData	

continues on next page

Table 1 – continued from previous page

Resource	Operation	Description
	POST /api/v2_0/postUdiEvent	Upload flexibility constraints to the platform
	POST /api/v2_0/postUdiEvent	
	POST /api/v2_0/postWeatherData	Upload weather data to the platform
	POST /api/v2_0/postWeatherData	
	GET /api/v2_0/user/(id)	Get a user
	GET /api/v2_0/user/(id)	
	PATCH /api/v2_0/user/(id)	Patch data for an existing user
	PATCH /api/v2_0/user/(id)	
	PATCH /api/v2_0/user/(id)/password-reset	Password reset
	PATCH /api/v2_0/user/(id)/password-reset	
	GET /api/v2_0/users	Download user list
	GET /api/v2_0/users	

9.2 API Details

`GET /api/`

Public endpoint to list API versions.

`POST /api/requestAuthToken`

API endpoint to get a fresh authentication access token. Be aware that this fresh token has a limited lifetime (which depends on the current system setting SECURITY_TOKEN_MAX_AGE).

Pass the *email* parameter to identify the user. Pass the *password* parameter to authenticate the user (if not already authenticated in current session)

`DELETE /api/v2_0/asset/`(*id*)

`DELETE /api/v2_0/asset/`(*id*)

API endpoint to delete an asset, and its sensed data.

This endpoint deletes an existing asset, as well as all measurements recorded for it. Only users who own the asset are allowed to delete the asset.

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 204 No Content – DELETED
- 400 Bad Request – INVALID_REQUEST, REQUIRED_INFO_MISSING, UNEXPECTED_PARAMS
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER

`GET /api/v2_0/asset/`(*id*)

GET /api/v2_0/asset/(*id*)

API endpoint to get an asset.

This endpoint gets an asset. Only users who own the asset can use this endpoint.

Example response

```
{
    "asset_type": "battery",
    "capacity_in_mw": 2.0,
    "display_name": "Test battery",
    "event_resolution": 5,
    "id": 1,
    "latitude": 10,
    "longitude": 100,
    "market": 1,
    "max_soc_in_mwh": 5,
    "min_soc_in_mwh": 0,
    "name": "Test battery",
    "owner": 2,
    "soc_datetime": "2015-01-01T00:00:00+00:00",
    "soc_in_mwh": 2.5,
    "soc_udi_event_id": 203,
    "unit": "kW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_REQUEST, REQUIRED_INFO_MISSING, UNEXPECTED_PARAMS
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER

PATCH /api/v2_0/asset/(*id*)

PATCH /api/v2_0/asset/(*id*)

API endpoint to patch asset data.

This endpoint sets data for an existing asset. Any subset of asset fields can be sent. Only users who own the asset are allowed to update its data.

Several fields are not allowed to be updated, e.g. id. They are ignored.

Example request

```
{
    "latitude": 11.1,
    "longitude": 99.9,
}
```

Note that event_resolution is expected as the number of minutes and soc_datetime is expected as ISO8601 datetime string.

Example response

The whole asset is returned in the response:

```
{
    "asset_type": "battery",
    "capacity_in_mw": 2.0,
    "display_name": "Test battery",
    "event_resolution": 5,
    "id": 1,
    "latitude": 11.1,
    "longitude": 99.9,
    "market": 1,
    "max_soc_in_mwh": 5,
    "min_soc_in_mwh": 0,
    "name": "Test battery",
    "owner": 2,
    "soc_datetime": "2015-01-01T00:00:00+00:00",
    "soc_in_mwh": 2.5,
    "soc_udi_event_id": 203,
    "unit": "kW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – UPDATED
- 400 Bad Request – INVALID_REQUEST, REQUIRED_INFO_MISSING, UNEXPECTED_PARAMS
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 422 Unprocessable Entity – UNPROCESSABLE_ENTITY

GET /api/v2_0/assets

GET /api/v2_0/assets

API endpoint to get assets.

This endpoint returns all accessible assets for a given owner. The *owner_id* query parameter can be used to set an owner. If no owner is set, all accessible assets are returned. A non-admin user can only access its own assets.

Example response

An example of one asset being returned:

```
[
    {
        "asset_type": "battery",
        "capacity_in_mw": 2.0,
        "display_name": "Test battery",
        "event_resolution": 10,
        "id": 1,
        "latitude": 10,
        "longitude": 100,
        "market": 1,
        "max_soc_in_mwh": 5,
        "min_soc_in_mwh": 0,
        "name": "Test battery",
        "owner": 2,
        "soc_datetime": "2015-01-01T00:00:00+00:00",
        "soc_in_mwh": 2.5,
        "soc_udi_event_id": 203,
        "unit": "MW"
    }
]
```

Note that event_resolution is returned as the number of minutes and soc_datetime is returned as ISO8601 datetime string.

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_REQUEST
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER

POST /api/v2_0/assets

POST /api/v2_0/assets

API endpoint to post a new asset.

This endpoint creates a new asset. Only users with the admin role are allowed to create assets.

Example request

The following example contains the required fields only, plus the two state of charge (soc) fields which a battery asset needs to specify:

```
{
    "name": "Test battery",
    "asset_type": "battery",
    "unit": "kW",
    "owner": 2,
    "market": 1,
```

(continues on next page)

(continued from previous page)

```
    "event_resolution": 5,
    "capacity_in_mw": 4.2,
    "latitude": 40,
    "longitude": 170.3,
    "max_soc_in_mwh": 5,
    "min_soc_in_mwh": 0
}
```

Note that event_resolution is expected as the number of minutes and soc_datetime is expected as ISO8601 datetime string.

Example response

The newly posted asset, including all fields, is returned in the response:

```
{
    "id": 1,
    "asset_type": "battery",
    "unit": "kW"
    "capacity_in_mw": 4.2,
    "display_name": "Test battery",
    "event_resolution": 5,
    "latitude": 40,
    "longitude": 170.3,
    "max_soc_in_mwh": 5,
    "min_soc_in_mwh": 0,
    "name": "Test battery",
    "owner": 2,
    "market": 1,
    "soc_datetime": null,
    "soc_in_mwh": null,
    "soc_udi_event_id": null
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 201 Created – CREATED
- 400 Bad Request – INVALID_REQUEST
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER

GET /api/v2_0/charts/power

GET /api/v2_0/charts/power

API endpoint to get a chart for power data which can be embedded in web pages.

This endpoint returns a Bokeh chart with power data which can be embedded in a website. It includes forecasts and even schedules, if available.

Example request

An example of a chart request:

```
{
    "resource": ""my-battery,
    "start_time": "2020-02-20:10:00:00UTC",
    "end_time": "2020-02-20:11:00:00UTC",
    "resolution": "PT15M",
    "consumption_as_positive": true
    "resolution": "PT6H",
}
```

On your webpage, you need to include the Bokeh libraries, e.g.:

<script src="https://cdn.pydata.org/bokeh/release/bokeh-1.0.4.min.js"></script>

(The version needs to match the version used by the FlexMeasures server, see requirements/app.txt)

Then you can call this endpoint and include the result like this:

```
<script>
    fetch('http://localhost:5000/api/v2_0/charts/power?' + urlData.toString(),
    {
        method: "GET",
        mode: "cors",
        headers:
            {
                "Content-Type": "application/json",
                "Authorization": "<users auth token>"
            },
    })
    .then(function(response) { return response.json(); })
    .then(function(item) { Bokeh.embed.embed_item(item, "<ID of the div >"); });
</script>
```

where *urlData* is a *URLSearchData* object and contains the chart request parameters (see above).

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_REQUEST
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 422 Unprocessable Entity – UNPROCESSABLE_ENTITY

GET /api/v2_0/getConnection

GET /api/v2_0/getConnection

API endpoint to get the user's connections as entity addresses ordered from newest to oldest.

Example request

```
{
    "type": "GetConnectionRequest",
}
```

Example response

This "GetConnectionResponse" message indicates that the user had access rights to retrieve four entity addresses owned by three different users.

```
{
    "type": "GetConnectionResponse",
    "connections": [
        "ea1.2018-06.io.flexmeasures.company:3:4",
        "ea1.2018-06.io.flexmeasures.company:8:3",
        "ea1.2018-06.io.flexmeasures.company:9:2",
        "ea1.2018-06.io.flexmeasures.company:3:1"
    ],
    "names": [
        "CS 4",
        "CS 3",
        "CS 2",
        "CS 1"
    ]
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_MESSAGE_TYPE
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

GET /api/v2_0/getDeviceMessage

GET /api/v2_0/getDeviceMessage

API endpoint to get device message.

Optional parameters

- "duration" (6 hours by default; can be increased to plan further into the future)

Example request

This "GetDeviceMessageRequest" message requests targeted consumption for UDI event 203 of device 10 of owner 7.

```
{
    "type": "GetDeviceMessageRequest",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203:soc"
}
```

Example response

This "GetDeviceMessageResponse" message indicates that the target for UDI event 203 is to consume at various power rates from 10am UTC onwards for a duration of 45 minutes.

```
{
    "type": "GetDeviceMessageResponse",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203:soc",
    "values": [
        2.15,
        3,
        2
    ],
    "start": "2015-06-02T10:00:00+00:00",
    "duration": "PT45M",
    "unit": "MW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_DOMAIN, INVALID_UNIT, UNKNOWN_SCHEDULE, UNRECOGNIZED_CONNECTION_GROUP, or UNRECOGNIZED_UDI_EVENT
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD
- 422 Unprocessable Entity – UNPROCESSABLE_ENTITY

GET /api/v2_0/getMeterData

GET /api/v2_0/getMeterData

API endpoint to get meter data.

Optional parameters

- "resolution" (see *Resolutions*)
- "horizon" (see *Beliefs*)
- "prior" (see *Beliefs*)
- "source" (see *Sources*)

Example request

This “GetMeterDataRequest” message requests measured consumption between 0.00am and 1.30am for charging station 1.

```
{
    "type": "GetMeterDataRequest",
    "connection": "CS 1",
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Example response

This “GetMeterDataResponse” message indicates that consumption for charging station 1 was measured in 15-minute intervals.

```
{
    "type": "GetMeterDataResponse",
    "connection": "CS 1",
    "values": [
        306.66,
        306.66,
        0,
        0,
        306.66,
        306.66
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_SOURCE, INVALID_TIMEZONE, INVALID_UNIT, UNRECOGNIZED_ASSET, or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

GET /api/v2_0/getPrognosis

GET /api/v2_0/getPrognosis
API endpoint to get prognosis.

Optional parameters

- "resolution" (see *Resolutions*)
- "horizon" (see *Beliefs*)
- "prior" (see *Beliefs*)
- "source" (see *Sources*)

Example request

This "GetPrognosisRequest" message requests prognosed consumption between 0.00am and 1.30am for charging station 1, with a rolling horizon of 6 hours before the end of each 15 minute time interval.

```
{
    "type": "GetPrognosisRequest",
    "connection": "CS 1",
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "horizon": "PT6H",
    "resolution": "PT15M",
    "unit": "MW"
}
```

Example response

This "GetPrognosisResponse" message indicates that a prognosis of consumption for charging station 1 was available 6 hours before the start of each 15 minute time interval.

```
{
    "type": "GetPrognosisResponse",
    "connection": "CS 1",
    "values": [
        306.66,
        306.66,
        0,
        0,
        306.66,
        306.66
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_MESSAGE_TYPE, INVALID_SOURCE, INVALID_TIMEZONE, INVALID_UNIT, UNRECOGNIZED_ASSET, or UNRECOGNIZED_CONNECTION_GROUP

- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

GET /api/v2_0/getService

GET /api/v2_0/getService

API endpoint to get a service listing for this version.

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED

POST /api/v2_0/postMeterData

POST /api/v2_0/postMeterData

API endpoint to post meter data.

Optional parameters

- "horizon" (see *Prognoses*)

Example request

This "PostMeterDataRequest" message posts measured consumption for 15-minute intervals between 0.00am and 1.30am for charging stations 1, 2 and 3 (negative values denote production).

```
{
    "type": "PostMeterDataRequest",
    "groups": [
        {
            "connections": [
                "CS 1",
                "CS 3"
            ],
            "values": [
                306.66,
                306.66,
                0,
                0,
                306.66,
                306.66
            ]
        },
        {
            "connections": [
                "CS 2"
            ],
            "values": [
                306.66,
                0,
                0,
                0,
                306.66,
                306.66
            ]
```

(continues on next page)

(continued from previous page)

```
        }
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

It is allowed to send higher resolutions (in this example for instance, 30 minutes) which will be upsampled.

Example response

This "PostMeterDataResponse" message indicates that the measurement has been processed without any error.

```
{
    "type": "PostMeterDataResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

POST /api/v2_0/postPriceData

POST /api/v2_0/postPriceData

API endpoint to post price data.

Optional parameters

- "horizon" (see *Prognoses*)

Example request

This "PostPriceDataRequest" message posts prices for hourly intervals between midnight and midnight the next day for the EPEX SPOT day-ahead auction. The horizon indicates that the prices were published at 1pm on December 31st 2014 (i.e. 35 hours ahead of midnight the next day).

```
{
    "type": "PostPriceDataRequest",
    "market": "ea1.2018-06.localhost:epex_da",
```

(continues on next page)

(continued from previous page)

```
    "values": [
        52.37,
        51.14,
        49.09,
        48.35,
        48.47,
        49.98,
        58.7,
        67.76,
        69.21,
        70.26,
        70.46,
        70,
        70.7,
        70.41,
        70,
        64.53,
        65.92,
        69.72,
        70.51,
        75.49,
        70.35,
        70.01,
        66.98,
        58.61
    ],
    "start": "2015-01-01T15:00:00+09:00",
    "duration": "PT24H",
    "horizon": "PT35H",
    "unit": "EUR/MWh"
}
```

Example response

This “PostPriceDataResponse” message indicates that the prices have been processed without any error.

```
{
    "type": "PostPriceDataResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_MARKET

- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

POST /api/v2_0/postUdiEvent

POST /api/v2_0/postUdiEvent

API endpoint to post UDI event.

Example request A

This "PostUdiEventRequest" message posts a state of charge (soc) of 12.1 kWh at 10.00am as UDI event 203 of device 10 of owner 7.

```
{
    "type": "PostUdiEventRequest",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203:soc",
    "value": 12.1,
    "unit": "kWh",
    "datetime": "2015-06-02T10:00:00+00:00"
}
```

Example request B

This "PostUdiEventRequest" message posts a state of charge (soc) of 12.1 kWh at 10.00am, and a target state of charge of 25 kWh at 4.00pm, as UDI event 204 of device 10 of owner 7.

```
{
    "type": "PostUdiEventRequest",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:204:soc-with-targets",
    "value": 12.1,
    "unit": "kWh",
    "datetime": "2015-06-02T10:00:00+00:00",
    "targets": [
        {
            "value": 25,
            "datetime": "2015-06-02T16:00:00+00:00"
        }
    ]
}
```

Example response

This "PostUdiEventResponse" message indicates that the UDI event has been processed without any error.

```
{
    "type": "PostUdiEventResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INCOMPLETE_UDI_EVENT, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_DATETIME, INVALID_DOMAIN, INVALID_UNIT, OUTDATED_UDI_EVENT, PTUS_INCOMPLETE, OUTDATED_UDI_EVENT or UNRECOGNIZED_UDI_EVENT
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

POST /api/v2_0/postWeatherData

POST /api/v2_0/postWeatherData

API endpoint to post weather data, such as:

- "radiation" (with kW/m^2 as unit)
- "temperature" (with °C as unit)
- "wind_speed" (with m/s as unit)

The sensor type is part of the unique entity address for each sensor, together with the sensor's latitude and longitude.

Optional parameters

- "horizon" (see *Prognoses*)

Example request

This "PostWeatherDataRequest" message posts temperature forecasts for 15-minute intervals between 3.00pm and 4.30pm for a weather sensor located at latitude 33.4843866 and longitude 126.477859. The forecasts were made at noon.

```
{
    "type": "PostWeatherDataRequest",
    "groups": [
        {
            "sensor": "ea1.2018-06.localhost:temperature:33.4843866:126.477859",
            "values": [
                20.04,
                20.23,
                20.41,
                20.51,
                20.55,
                20.57
            ]
        }
    ],
    "start": "2015-01-01T15:00:00+09:00",
    "duration": "PT1H30M",
    "horizon": "PT3H",
    "unit": "°C"
}
```

It is allowed to send higher resolutions (in this example for instance, 30 minutes) which will be upsampled.

Example response

This "PostWeatherDataResponse" message indicates that the forecast has been processed without any error.

```
{
    "type": "PostWeatherDataResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_SENSOR
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

GET /api/v2_0/user/(*id*)

GET /api/v2_0/user/(*id*)

API endpoint to get a user.

This endpoint gets a user. Only admins or the user themselves can use this endpoint.

Example response

```
{
    'active': True,
    'email': 'test_prosumer@seita.nl',
    'flexmeasures_roles': [1, 3],
    'id': 1,
    'timezone': 'Europe/Amsterdam',
    'username': 'Test Prosumer'
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED

- 400 Bad Request – INVALID_REQUEST, REQUIRED_INFO_MISSING, UNEXPECTED_PARAMS
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER

PATCH /api/v2_0/user/(*id*)

PATCH /api/v2_0/user/(*id*)

API endpoint to patch user data.

This endpoint sets data for an existing user. Any subset of user fields can be sent. Only the user themselves or admins are allowed to update its data, while a non-admin can only edit a few of their own fields.

Several fields are not allowed to be updated, e.g. id. They are ignored.

Example request

```
{
    "active": false,
}
```

Example response

The whole user is returned in the response:

```
{
    'active': True,
    'email': 'test_prosumer@seita.nl',
    'flexmeasures_roles': [1, 3],
    'id': 1,
    'timezone': 'Europe/Amsterdam',
    'username': 'Test Prosumer'
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – UPDATED
- 400 Bad Request – INVALID_REQUEST, REQUIRED_INFO_MISSING, UNEXPECTED_PARAMS
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 422 Unprocessable Entity – UNPROCESSABLE_ENTITY

PATCH /api/v2_0/user/(*id*)**/password-reset**

PATCH /api/v2_0/user/(*id*)**/password-reset**

API endpoint to reset the user password. They'll get an email to choose a new password.

Reset the user's password, and send them instructions on how to reset the password. This endoint is useful from a security standpoint, in case of worries the password might be compromised. It sets the current password to something random, invalidates cookies and auth tokens, and also sends an email for resetting the password to the user.

Only admins can use this endpoint.

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_REQUEST, REQUIRED_INFO_MISSING, UNEXPECTED_PARAMS
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER

GET /api/v2_0/users

GET /api/v2_0/users

API endpoint to get users.

This endpoint returns all accessible users. By default, only active users are returned. The *include_inactive* query parameter can be used to also fetch inactive users. Only admins can use this endpoint.

Example response

An example of one user being returned:

```
[
    {
        'active': True,
        'email': 'test_prosumer@seita.nl',
        'flexmeasures_roles': [1, 3],
        'id': 1,
        'timezone': 'Europe/Amsterdam',
        'username': 'Test Prosumer'
    }
]
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_REQUEST

- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER

CHAPTER

TEN

VERSION 1.3

10.1 Summary

Resource	Operation	Description
Public	*GET /api/*	List available API versions
	POST /api/requestAuthToken	Obtain an authentication token
	GET /api/v1_3/getService	Obtain a service listing for this version
User	*GET /api/v1_3/getConnection*	Retrieve entity addresses of connections
	GET /api/v1_3/getDeviceMessage	Download control signal from the platform
	GET /api/v1_3/getMeterData	Download meter data from the platform
	GET /api/v1_3/getPrognosis	Download prognosis from the platform
	POST /api/v1_3/postMeterData	Upload meter data to the platform
	POST /api/v1_3/postPriceData	Upload price data to the platform
	POST /api/v1_3/postUdiEvent	Upload flexibility constraints to the platform
	POST /api/v1_3/postWeatherData	Upload weather data to the platform

10.2 API Details

`GET /api/`

Public endpoint to list API versions.

`POST /api/requestAuthToken`

API endpoint to get a fresh authentication access token. Be aware that this fresh token has a limited lifetime (which depends on the current system setting SECURITY_TOKEN_MAX_AGE).

Pass the *email* parameter to identify the user. Pass the *password* parameter to authenticate the user (if not already authenticated in current session)

`GET /api/v1_3/getConnection`

API endpoint to get the user's connections as entity addresses ordered from newest to oldest.

Example request

```
{
    "type": "GetConnectionRequest",
}
```

Example response

This "GetConnectionResponse" message indicates that the user had access rights to retrieve four entity addresses owned by three different users.

```
{
    "type": "GetConnectionResponse",
    "connections": [
        "ea1.2018-06.io.flexmeasures.company:3:4",
        "ea1.2018-06.io.flexmeasures.company:8:3",
        "ea1.2018-06.io.flexmeasures.company:9:2",
        "ea1.2018-06.io.flexmeasures.company:3:1"
    ],
    "names": [
        "CS 4",
        "CS 3",
        "CS 2",
        "CS 1"
    ]
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_MESSAGE_TYPE
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

GET /api/v1_3/getDeviceMessage

API endpoint to get device message.

Optional parameters

- "duration" (6 hours by default; can be increased to plan further into the future)

Example request

This "GetDeviceMessageRequest" message requests targeted consumption for UDI event 203 of device 10 of owner 7.

```
{
    "type": "GetDeviceMessageRequest",
```

(continues on next page)

(continued from previous page)

```
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203:soc"
}
```

Example response

This "GetDeviceMessageResponse" message indicates that the target for UDI event 203 is to consume at various power rates from 10am UTC onwards for a duration of 45 minutes.

```
{
    "type": "GetDeviceMessageResponse",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203:soc",
    "values": [
        2.15,
        3,
        2
    ],
    "start": "2015-06-02T10:00:00+00:00",
    "duration": "PT45M",
    "unit": "MW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_DOMAIN, INVALID_UNIT, UNKNOWN_SCHEDULE, UNRECOGNIZED_CONNECTION_GROUP, or UNRECOGNIZED_UDI_EVENT
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD
- 422 Unprocessable Entity – UNPROCESSABLE_ENTITY

GET /api/v1_3/getMeterData

API endpoint to get meter data.

Optional parameters

- "resolution" (see *Resolutions*)
- "horizon" (see *Beliefs*)
- "prior" (see *Beliefs*)
- "source" (see *Sources*)

Example request

This "GetMeterDataRequest" message requests measured consumption between 0.00am and 1.30am for charging station 1.

```
{
    "type": "GetMeterDataRequest",
    "connection": "CS 1",
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Example response

This "GetMeterDataResponse" message indicates that consumption for charging station 1 was measured in 15-minute intervals.

```
{
    "type": "GetMeterDataResponse",
    "connection": "CS 1",
    "values": [
        306.66,
        306.66,
        0,
        0,
        306.66,
        306.66
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_SOURCE, INVALID_TIMEZONE, INVALID_UNIT, UNRECOGNIZED_ASSET, or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

GET /api/v1_3/getPrognosis

API endpoint to get prognosis.

Optional parameters

- "resolution" (see *Resolutions*)

- "horizon" (see *Beliefs*)
- "prior" (see *Beliefs*)
- "source" (see *Sources*)

Example request

This "GetPrognosisRequest" message requests prognosed consumption between 0.00am and 1.30am for charging station 1, with a rolling horizon of 6 hours before the end of each 15 minute time interval.

```
{
    "type": "GetPrognosisRequest",
    "connection": "CS 1",
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "horizon": "PT6H",
    "resolution": "PT15M",
    "unit": "MW"
}
```

Example response

This "GetPrognosisResponse" message indicates that a prognosis of consumption for charging station 1 was available 6 hours before the start of each 15 minute time interval.

```
{
    "type": "GetPrognosisResponse",
    "connection": "CS 1",
    "values": [
        306.66,
        306.66,
        0,
        0,
        306.66,
        306.66
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_MESSAGE_TYPE, INVALID_SOURCE, INVALID_TIMEZONE, INVALID_UNIT, UNRECOGNIZED_ASSET, or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER

- 405 Method Not Allowed – INVALID_METHOD

GET /api/v1_3/getService

API endpoint to get a service listing for this version.

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED

POST /api/v1_3/postMeterData

API endpoint to post meter data.

Optional parameters

- "horizon" (see *Prognoses*)

Example request

This "PostMeterDataRequest" message posts measured consumption for 15-minute intervals between 0.00am and 1.30am for charging stations 1, 2 and 3 (negative values denote production).

```
{
    "type": "PostMeterDataRequest",
    "groups": [
        {
            "connections": [
                "CS 1",
                "CS 3"
            ],
            "values": [
                306.66,
                306.66,
                0,
                0,
                306.66,
                306.66
            ]
        },
        {
            "connections": [
                "CS 2"
            ],
            "values": [
                306.66,
                0,
                0,
                0,
                306.66,
                306.66
            ]
        }
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

It is allowed to send higher resolutions (in this example for instance, 30 minutes) which will be upsampled.

Example response

This “PostMeterDataResponse” message indicates that the measurement has been processed without any error.

```
{
    "type": "PostMeterDataResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

`POST /api/v1_3/postPriceData`

API endpoint to post price data.

Optional parameters

- “horizon” (see *Prognoses*)

Example request

This “PostPriceDataRequest” message posts prices for hourly intervals between midnight and midnight the next day for the EPEX SPOT day-ahead auction. The horizon indicates that the prices were published at 1pm on December 31st 2014 (i.e. 35 hours ahead of midnight the next day).

```
{
    "type": "PostPriceDataRequest",
    "market": "ea1.2018-06.localhost:epex_da",
    "values": [
        52.37,
        51.14,
        49.09,
        48.35,
        48.47,
        49.98,
        58.7,
        67.76,
        69.21,
        70.26,
```

(continues on next page)

(continued from previous page)

```
        70.46,
        70,
        70.7,
        70.41,
        70,
        64.53,
        65.92,
        69.72,
        70.51,
        75.49,
        70.35,
        70.01,
        66.98,
        58.61
    ],
    "start": "2015-01-01T15:00:00+09:00",
    "duration": "PT24H",
    "horizon": "PT35H",
    "unit": "EUR/MWh"
}
```

Example response

This "PostPriceDataResponse" message indicates that the prices have been processed without any error.

```
{
    "type": "PostPriceDataResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_MARKET
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

POST /api/v1_3/postUdiEvent

API endpoint to post UDI event.

Example request A

This "PostUdiEventRequest" message posts a state of charge (soc) of 12.1 kWh at 10.00am as UDI event 203 of device 10 of owner 7.

```
{
    "type": "PostUdiEventRequest",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203:soc",
    "value": 12.1,
    "unit": "kWh",
    "datetime": "2015-06-02T10:00:00+00:00"
}
```

Example request B

This "PostUdiEventRequest" message posts a state of charge (soc) of 12.1 kWh at 10.00am, and a target state of charge of 25 kWh at 4.00pm, as UDI event 204 of device 10 of owner 7.

```
{
    "type": "PostUdiEventRequest",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:204:soc-with-targets",
    "value": 12.1,
    "unit": "kWh",
    "datetime": "2015-06-02T10:00:00+00:00",
    "targets": [
        {
            "value": 25,
            "datetime": "2015-06-02T16:00:00+00:00"
        }
    ]
}
```

Example response

This "PostUdiEventResponse" message indicates that the UDI event has been processed without any error.

```
{
    "type": "PostUdiEventResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INCOMPLETE_UDI_EVENT, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_DATETIME, INVALID_DOMAIN, INVALID_UNIT, OUTDATED_UDI_EVENT, PTUS_INCOMPLETE, OUTDATED_UDI_EVENT or UNRECOGNIZED_UDI_EVENT
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER

- 405 Method Not Allowed – INVALID_METHOD

POST /api/v1_3/postWeatherData

API endpoint to post weather data, such as:

- “radiation” (with kW/m^2 as unit)
- “temperature” (with °C as unit)
- “wind_speed” (with m/s as unit)

The sensor type is part of the unique entity address for each sensor, together with the sensor’s latitude and longitude.

Optional parameters

- “horizon” (see *Prognoses*)

Example request

This “PostWeatherDataRequest” message posts temperature forecasts for 15-minute intervals between 3.00pm and 4.30pm for a weather sensor located at latitude 33.4843866 and longitude 126.477859. The forecasts were made at noon.

```
{
    "type": "PostWeatherDataRequest",
    "groups": [
        {
            "sensor": "ea1.2018-06.localhost:temperature:33.4843866:126.477859",
            "values": [
                20.04,
                20.23,
                20.41,
                20.51,
                20.55,
                20.57
            ]
        }
    ],
    "start": "2015-01-01T15:00:00+09:00",
    "duration": "PT1H30M",
    "horizon": "PT3H",
    "unit": "°C"
}
```

It is allowed to send higher resolutions (in this example for instance, 30 minutes) which will be upsampled.

Example response

This “PostWeatherDataResponse” message indicates that the forecast has been processed without any error.

```
{
    "type": "PostWeatherDataResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_SENSOR
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

CHAPTER

ELEVEN

VERSION 1.2

11.1 Summary

Resource	Operation	Description
Public	*GET /api/*	List available API versions
	POST /api/requestAuthToken	Obtain an authentication token
	GET /api/v1_2/getService	Obtain a service listing for this version
User	*GET /api/v1_2/getConnection*	Retrieve entity addresses of connections
	GET /api/v1_2/getDeviceMessage	Download control signal from the platform
	GET /api/v1_2/getMeterData	Download meter data from the platform
	GET /api/v1_2/getPrognosis	Download prognosis from the platform
	POST /api/v1_2/postMeterData	Upload meter data to the platform
	POST /api/v1_2/postPriceData	Upload price data to the platform
	POST /api/v1_2/postUdiEvent	Upload flexibility constraints to the platform
	POST /api/v1_2/postWeatherData	Upload weather data to the platform

11.2 API Details

`GET /api/`

Public endpoint to list API versions.

`POST /api/requestAuthToken`

API endpoint to get a fresh authentication access token. Be aware that this fresh token has a limited lifetime (which depends on the current system setting SECURITY_TOKEN_MAX_AGE).

Pass the *email* parameter to identify the user. Pass the *password* parameter to authenticate the user (if not already authenticated in current session)

`GET /api/v1_2/getConnection`

API endpoint to get the user's connections as entity addresses ordered from newest to oldest.

Example request

```
{
    "type": "GetConnectionRequest",
}
```

Example response

This "GetConnectionResponse" message indicates that the user had access rights to retrieve four entity addresses owned by three different users.

```
{
    "type": "GetConnectionResponse",
    "connections": [
        "ea1.2018-06.io.flexmeasures.company:3:4",
        "ea1.2018-06.io.flexmeasures.company:8:3",
        "ea1.2018-06.io.flexmeasures.company:9:2",
        "ea1.2018-06.io.flexmeasures.company:3:1"
    ],
    "names": [
        "CS 4",
        "CS 3",
        "CS 2",
        "CS 1"
    ]
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_MESSAGE_TYPE
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

GET /api/v1_2/getDeviceMessage

API endpoint to get device message.

Optional parameters

- "duration" (6 hours by default; can be increased to plan further into the future)

Example request

This "GetDeviceMessageRequest" message requests targeted consumption for UDI event 203 of device 10 of owner 7.

```
{
    "type": "GetDeviceMessageRequest",
```

(continues on next page)

(continued from previous page)

```
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203:soc"
}
```

Example response

This "GetDeviceMessageResponse" message indicates that the target for UDI event 203 is to consume at various power rates from 10am UTC onwards for a duration of 45 minutes.

```
{
    "type": "GetDeviceMessageResponse",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203:soc",
    "values": [
        2.15,
        3,
        2
    ],
    "start": "2015-06-02T10:00:00+00:00",
    "duration": "PT45M",
    "unit": "MW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, UNKNOWN_PRICES, UNRECOGNIZED_CONNECTION_GROUP, or UNRECOGNIZED_UDI_EVENT
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD
- 422 Unprocessable Entity – UNPROCESSABLE_ENTITY

GET /api/v1_2/getMeterData

API endpoint to get meter data.

Optional parameters

- "resolution" (see *Resolutions*)
- "horizon" (see *Beliefs*)
- "prior" (see *Beliefs*)
- "source" (see *Sources*)

Example request

This “GetMeterDataRequest” message requests measured consumption between 0.00am and 1.30am for charging station 1.

```
{
    "type": "GetMeterDataRequest",
    "connection": "CS 1",
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Example response

This “GetMeterDataResponse” message indicates that consumption for charging station 1 was measured in 15-minute intervals.

```
{
    "type": "GetMeterDataResponse",
    "connection": "CS 1",
    "values": [
        306.66,
        306.66,
        0,
        0,
        306.66,
        306.66
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_SOURCE, INVALID_TIMEZONE, INVALID_UNIT, UNRECOGNIZED_ASSET, or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

GET /api/v1_2/getPrognosis

API endpoint to get prognosis.

Optional parameters

- “resolution” (see *Resolutions*)

- “horizon” (see *Beliefs*)
- “prior” (see *Beliefs*)
- “source” (see *Sources*)

Example request

This “GetPrognosisRequest” message requests prognosed consumption between 0.00am and 1.30am for charging station 1, with a rolling horizon of 6 hours before the end of each 15 minute time interval.

```
{
    "type": "GetPrognosisRequest",
    "connection": "CS 1",
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "horizon": "PT6H",
    "resolution": "PT15M",
    "unit": "MW"
}
```

Example response

This “GetPrognosisResponse” message indicates that a prognosis of consumption for charging station 1 was available 6 hours before the start of each 15 minute time interval.

```
{
    "type": "GetPrognosisResponse",
    "connection": "CS 1",
    "values": [
        306.66,
        306.66,
        0,
        0,
        306.66,
        306.66
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_MESSAGE_TYPE, INVALID_SOURCE, INVALID_TIMEZONE, INVALID_UNIT, UNRECOGNIZED_ASSET, or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER

- 405 Method Not Allowed – INVALID_METHOD

GET /api/v1_2/getService

API endpoint to get a service listing for this version.

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED

POST /api/v1_2/postMeterData

API endpoint to post meter data.

Optional parameters

- "horizon" (see *Prognoses*)

Example request

This "PostMeterDataRequest" message posts measured consumption for 15-minute intervals between 0.00am and 1.30am for charging stations 1, 2 and 3 (negative values denote production).

```
{
    "type": "PostMeterDataRequest",
    "groups": [
        {
            "connections": [
                "CS 1",
                "CS 3"
            ],
            "values": [
                306.66,
                306.66,
                0,
                0,
                306.66,
                306.66
            ]
        },
        {
            "connections": [
                "CS 2"
            ],
            "values": [
                306.66,
                0,
                0,
                0,
                306.66,
                306.66
            ]
        }
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

It is allowed to send higher resolutions (in this example for instance, 30 minutes) which will be upsampled.

Example response

This "PostMeterDataResponse" message indicates that the measurement has been processed without any error.

```
{
    "type": "PostMeterDataResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

POST /api/v1_2/postPriceData

API endpoint to post price data.

Optional parameters

- "horizon" (see *Prognoses*)

Example request

This "PostPriceDataRequest" message posts prices for hourly intervals between midnight and midnight the next day for the EPEX SPOT day-ahead auction. The horizon indicates that the prices were published at 1pm on December 31st 2014 (i.e. 35 hours ahead of midnight the next day).

```
{
    "type": "PostPriceDataRequest",
    "market": "ea1.2018-06.localhost:epex_da",
    "values": [
        52.37,
        51.14,
        49.09,
        48.35,
        48.47,
        49.98,
        58.7,
        67.76,
        69.21,
        70.26,
```

(continues on next page)

(continued from previous page)

```
        70.46,
        70,
        70.7,
        70.41,
        70,
        64.53,
        65.92,
        69.72,
        70.51,
        75.49,
        70.35,
        70.01,
        66.98,
        58.61
    ],
    "start": "2015-01-01T15:00:00+09:00",
    "duration": "PT24H",
    "horizon": "PT35H",
    "unit": "EUR/MWh"
}
```

Example response

This "PostPriceDataResponse" message indicates that the prices have been processed without any error.

```
{
    "type": "PostPriceDataResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_MARKET
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

POST /api/v1_2/postUdiEvent

API endpoint to post UDI event.

Example request

This "PostUdiEventRequest" message posts a state of charge (soc) of 12.1 kWh at 10.00am as UDI event 203 of device 10 of owner 7.

```
{
    "type": "PostUdiEventRequest",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203:soc",
    "value": 12.1,
    "unit": "kWh",
    "datetime": "2015-06-02T10:00:00+00:00",
}
```

Example response

This "PostUdiEventResponse" message indicates that the UDI event has been processed without any error.

```
{
    "type": "PostUdiEventResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_DATETIME, INVALID_UNIT, PTUS_INCOMPLETE, OUTDATED_UDI_EVENT or UNRECOGNIZED_UDI_EVENT
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

POST /api/v1_2/postWeatherData

API endpoint to post weather data, such as:

- "radiation" (with kW/m^2 as unit)
- "temperature" (with °C as unit)
- "wind_speed" (with m/s as unit)

The sensor type is part of the unique entity address for each sensor, together with the sensor's latitude and longitude.

Optional parameters

- "horizon" (see *Prognoses*)

Example request

This "PostWeatherDataRequest" message posts temperature forecasts for 15-minute intervals between 3.00pm and 4.30pm for a weather sensor located at latitude 33.4843866 and longitude 126.477859. The forecasts were made at noon.

```
{
    "type": "PostWeatherDataRequest",
    "groups": [
        {
            "sensor": "ea1.2018-06.localhost:temperature:33.4843866:126.477859",
            "values": [
                20.04,
                20.23,
                20.41,
                20.51,
                20.55,
                20.57
            ]
        }
    ],
    "start": "2015-01-01T15:00:00+09:00",
    "duration": "PT1H30M",
    "horizon": "PT3H",
    "unit": "°C"
}
```

It is allowed to send higher resolutions (in this example for instance, 30 minutes) which will be upsampled.

Example response

This "PostWeatherDataResponse" message indicates that the forecast has been processed without any error.

```
{
    "type": "PostWeatherDataResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_SENSOR
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

CHAPTER TWELVE

VERSION 1.1

12.1 Summary

Resource	Operation	Description
Public	*GET /api/*	List available API versions
	POST /api/requestAuthToken	Obtain an authentication token
	GET /api/v1_1/getService	Obtain a service listing for this version
User	*GET /api/v1_1/getConnection*	Retrieve entity addresses of connections
	GET /api/v1_1/getMeterData	Download meter data from the platform
	GET /api/v1_1/getPrognosis	Download prognosis from the platform
	POST /api/v1_1/postMeterData	Upload meter data to the platform
	POST /api/v1_1/postPriceData	Upload price data to the platform
	POST /api/v1_1/postPrognosis	Upload prognosis to the platform
	POST /api/v1_1/postWeatherData	Upload weather data to the platform

12.2 API Details

GET /api/

Public endpoint to list API versions.

POST /api/requestAuthToken

API endpoint to get a fresh authentication access token. Be aware that this fresh token has a limited lifetime (which depends on the current system setting SECURITY_TOKEN_MAX_AGE).

Pass the *email* parameter to identify the user. Pass the *password* parameter to authenticate the user (if not already authenticated in current session)

GET /api/v1_1/getConnection

API endpoint to get the user's connections as entity addresses ordered from newest to oldest.

Example request

```
{
    "type": "GetConnectionRequest",
}
```

Example response

This "GetConnectionResponse" message indicates that the user had access rights to retrieve four entity addresses owned by three different users.

```
{
    "type": "GetConnectionResponse",
    "connections": [
        "ea1.2018-06.io.flexmeasures.company:3:4",
        "ea1.2018-06.io.flexmeasures.company:8:3",
        "ea1.2018-06.io.flexmeasures.company:9:2",
        "ea1.2018-06.io.flexmeasures.company:3:1"
    ],
    "names": [
        "CS 4",
        "CS 3",
        "CS 2",
        "CS 1"
    ]
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_MESSAGE_TYPE
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

GET /api/v1_1/getMeterData

API endpoint to get meter data.

Optional parameters

- "resolution" (see *Resolutions*)
- "horizon" (see *Beliefs*)
- "prior" (see *Beliefs*)
- "source" (see *Sources*)

Example request

This "GetMeterDataRequest" message requests measured consumption between 0.00am and 1.30am for charging station 1.

```
{
    "type": "GetMeterDataRequest",
    "connection": "CS 1",
    "start": "2015-01-01T00:00:00Z",
```

(continues on next page)

(continued from previous page)

```
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Example response

This "GetMeterDataResponse" message indicates that consumption for charging station 1 was measured in 15-minute intervals.

```
{
    "type": "GetMeterDataResponse",
    "connection": "CS 1",
    "values": [
        306.66,
        306.66,
        0,
        0,
        306.66,
        306.66
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_SOURCE, INVALID_TIMEZONE, INVALID_UNIT, UNRECOGNIZED_ASSET, or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

GET /api/v1_1/getPrognosis

API endpoint to get prognosis.

Optional parameters

- "resolution" (see *Resolutions*)
- "horizon" (see *Beliefs*)
- "prior" (see *Beliefs*)
- "source" (see *Sources*)

Example request

This "GetPrognosisRequest" message requests prognosed consumption between 0.00am and 1.30am for charging station 1, with a rolling horizon of 6 hours before the end of each 15 minute time interval.

```
{
    "type": "GetPrognosisRequest",
    "connection": "CS 1",
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "horizon": "PT6H",
    "resolution": "PT15M",
    "unit": "MW"
}
```

Example response

This "GetPrognosisResponse" message indicates that a prognosis of consumption for charging station 1 was available 6 hours before the start of each 15 minute time interval.

```
{
    "type": "GetPrognosisResponse",
    "connection": "CS 1",
    "values": [
        306.66,
        306.66,
        0,
        0,
        306.66,
        306.66
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_MESSAGE_TYPE, INVALID_SOURCE, INVALID_TIMEZONE, INVALID_UNIT, UNRECOGNIZED_ASSET, or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

GET /api/v1_1/getService

API endpoint to get a service listing for this version.

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED

POST /api/v1_1/postMeterData

API endpoint to post meter data.

Optional parameters

- "horizon" (see *Prognoses*)

Example request

This "PostMeterDataRequest" message posts measured consumption for 15-minute intervals between 0.00am and 1.30am for charging stations 1, 2 and 3 (negative values denote production).

```
{
    "type": "PostMeterDataRequest",
    "groups": [
        {
            "connections": [
                "CS 1",
                "CS 3"
            ],
            "values": [
                306.66,
                306.66,
                0,
                0,
                306.66,
                306.66
            ]
        },
        {
            "connections": [
                "CS 2"
            ],
            "values": [
                306.66,
                0,
                0,
                0,
                306.66,
                306.66
            ]
        }
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

It is allowed to send higher resolutions (in this example for instance, 30 minutes) which will be upsampled.

Example response

This "PostMeterDataResponse" message indicates that the measurement has been processed without any error.

```
{
    "type": "PostMeterDataResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

POST /api/v1_1/postPriceData

API endpoint to post price data.

Optional parameters

- "horizon" (see *Prognoses*)

Example request

This "PostPriceDataRequest" message posts prices for hourly intervals between midnight and midnight the next day for the EPEX SPOT day-ahead auction. The horizon indicates that the prices were published at 1pm on December 31st 2014 (i.e. 35 hours ahead of midnight the next day).

```
{
    "type": "PostPriceDataRequest",
    "market": "ea1.2018-06.localhost:epex_da",
    "values": [
        52.37,
        51.14,
        49.09,
        48.35,
        48.47,
        49.98,
        58.7,
        67.76,
        69.21,
        70.26,
        70.46,
        70,
        70.7,
        70.41,
```

(continues on next page)

(continued from previous page)

```
        70,
        64.53,
        65.92,
        69.72,
        70.51,
        75.49,
        70.35,
        70.01,
        66.98,
        58.61
    ],
    "start": "2015-01-01T15:00:00+09:00",
    "duration": "PT24H",
    "horizon": "PT35H",
    "unit": "EUR/MWh"
}
```

Example response

This "PostPriceDataResponse" message indicates that the prices have been processed without any error.

```
{
    "type": "PostPriceDataResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_MARKET
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

POST /api/v1_1/postPrognosis

API endpoint to post prognoses about meter data.

Optional parameters

- "horizon" (see *Prognoses*)

Example request

This “PostPrognosisRequest” message posts prognosed consumption for 15-minute intervals between 0.00am and 1.30am for charging stations 1, 2 and 3 (negative values denote production), prognosed at 6pm the previous day.

```
{
    "type": "PostPrognosisRequest",
    "groups": [
        {
            "connections": [
                "CS 1",
                "CS 3"
            ],
            "values": [
                300,
                300,
                300,
                0,
                0,
                300
            ]
        },
        {
            "connections": [
                "CS 2"
            ],
            "values": [
                300,
                0,
                0,
                0,
                300,
                300
            ]
        }
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "horizon": "PT7H30M",
    "unit": "MW"
}
```

It is allowed to send higher resolutions (in this example for instance, 30 minutes) which will be upsampled.

Example response

This “PostPrognosisResponse” message indicates that the prognosis has been processed without any error.

```
{
    "type": "PostPrognosisResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

POST /api/v1_1/postWeatherData

API endpoint to post weather data, such as:

- "radiation" (with kW/m^2 as unit)
- "temperature" (with °C as unit)
- "wind_speed" (with m/s as unit)

The sensor type is part of the unique entity address for each sensor, together with the sensor's latitude and longitude.

Optional parameters

- "horizon" (see *Prognoses*)

Example request

This "PostWeatherDataRequest" message posts temperature forecasts for 15-minute intervals between 3.00pm and 4.30pm for a weather sensor located at latitude 33.4843866 and longitude 126.477859. The forecasts were made at noon.

```
{
    "type": "PostWeatherDataRequest",
    "groups": [
        {
            "sensor": "ea1.2018-06.localhost:temperature:33.4843866:126.477859",
            "values": [
                20.04,
                20.23,
                20.41,
                20.51,
                20.55,
                20.57
            ]
        }
    ],
    "start": "2015-01-01T15:00:00+09:00",
    "duration": "PT1H30M",
    "horizon": "PT3H",
    "unit": "°C"
}
```

It is allowed to send higher resolutions (in this example for instance, 30 minutes) which will be upsampled.

Example response

This "PostWeatherDataResponse" message indicates that the forecast has been processed without any error.

```
{
    "type": "PostWeatherDataResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_SENSOR
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

CHAPTER THIRTEEN

VERSION 1.0

13.1 Summary

Resource	Operation	Description
Public	*GET /api/*	List available API versions
	POST /api/requestAuthToken	Obtain an authentication token
	GET /api/v1/getService	Obtain a service listing for this version
User	*POST /api/v1/getMeterData*	Download meter data from the platform
	GET /api/v1/getMeterData	
	POST /api/v1/postMeterData	Upload meter data to the platform

13.2 API Details

`GET /api/`

Public endpoint to list API versions.

`POST /api/requestAuthToken`

API endpoint to get a fresh authentication access token. Be aware that this fresh token has a limited lifetime (which depends on the current system setting SECURITY_TOKEN_MAX_AGE).

Pass the *email* parameter to identify the user. Pass the *password* parameter to authenticate the user (if not already authenticated in current session)

`POST /api/v1/getMeterData`

API endpoint to get meter data.

Optional parameters

- "resolution" (see *Resolutions*)
- "horizon" (see *Beliefs*)
- "prior" (see *Beliefs*)
- "source" (see *Sources*)

Example request

This "GetMeterDataRequest" message requests measured consumption between 0.00am and 1.30am for charging station 1.

```
{
    "type": "GetMeterDataRequest",
    "connection": "CS 1",
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Example response

This "GetMeterDataResponse" message indicates that consumption for charging station 1 was measured in 15-minute intervals.

```
{
    "type": "GetMeterDataResponse",
    "connection": "CS 1",
    "values": [
        306.66,
        306.66,
        0,
        0,
        306.66,
        306.66
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_SOURCE, INVALID_TIMEZONE, INVALID_UNIT, UNRECOGNIZED_ASSET, or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

GET /api/v1/getMeterData

API endpoint to get meter data.

Optional parameters

- "resolution" (see *Resolutions*)
- "horizon" (see *Beliefs*)

- “prior” (see *Beliefs*)
- “source” (see *Sources*)

Example request

This “GetMeterDataRequest” message requests measured consumption between 0.00am and 1.30am for charging station 1.

```
{
    "type": "GetMeterDataRequest",
    "connection": "CS 1",
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Example response

This “GetMeterDataResponse” message indicates that consumption for charging station 1 was measured in 15-minute intervals.

```
{
    "type": "GetMeterDataResponse",
    "connection": "CS 1",
    "values": [
        306.66,
        306.66,
        0,
        0,
        306.66,
        306.66
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_SOURCE, INVALID_TIMEZONE, INVALID_UNIT, UNRECOGNIZED_ASSET, or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

GET /api/v1/getService

API endpoint to get a service listing for this version.

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED

POST /api/v1/postMeterData

API endpoint to post meter data.

Optional parameters

- "horizon" (see *Prognoses*)

Example request

This "PostMeterDataRequest" message posts measured consumption for 15-minute intervals between 0.00am and 1.30am for charging stations 1, 2 and 3 (negative values denote production).

```
{
    "type": "PostMeterDataRequest",
    "groups": [
        {
            "connections": [
                "CS 1",
                "CS 3"
            ],
            "values": [
                306.66,
                306.66,
                0,
                0,
                306.66,
                306.66
            ]
        },
        {
            "connections": [
                "CS 2"
            ],
            "values": [
                306.66,
                0,
                0,
                0,
                306.66,
                306.66
            ]
        }
    ],
    "start": "2015-01-01T00:00:00Z",
    "duration": "PT1H30M",
    "unit": "MW"
}
```

It is allowed to send higher resolutions (in this example for instance, 30 minutes) which will be upsampled.

Example response

This "PostMeterDataResponse" message indicates that the measurement has been processed without any error.

```
{
    "type": "PostMeterDataResponse",
    "status": "PROCESSED",
    "message": "Request has been processed."
}
```

Request Headers

- Authorization – The authentication token
- Content-Type – application/json

Response Headers

- Content-Type – application/json

Status Codes

- 200 OK – PROCESSED
- 400 Bad Request – INVALID_DOMAIN, INVALID_MESSAGE_TYPE, INVALID_TIMEZONE, INVALID_UNIT, REQUIRED_INFO_MISSING, UNRECOGNIZED_ASSET or UNRECOGNIZED_CONNECTION_GROUP
- 401 Unauthorized – UNAUTHORIZED
- 403 Forbidden – INVALID_SENDER
- 405 Method Not Allowed – INVALID_METHOD

CHAPTER

FOURTEEN

API CHANGE LOG

14.1 v2.0 | 2020-11-14

- REST endpoints for managing assets: */assets/* (GET, POST) and */asset/<id>* (GET, PATCH, DELETE)

14.2 v1.3-7 | 2020-12-16

Affects all versions since v1.0.

- Separated the dual purpose of the "horizon" parameter in the *getMeterData* and *getPrognosis* endpoints by introducing the "prior" parameter:
  - The "horizon" parameter in GET endpoints is now always interpreted as a rolling horizon, regardless of whether it is stated as an ISO 8601 repeating time interval.
  - The *getMeterData* and *getPrognosis* endpoints now accept an optional "prior" parameter to select only data recorded before a certain ISO 8601 timestamp (replacing the unintuitive usage of the horizon field for specifying a latest time of belief).

14.3 v1.3-6 | 2020-12-11

Affects all versions since v1.0.

- The *getMeterData* and *getPrognosis* endpoints now return the INVALID_SOURCE status 400 response in case the optional "source" parameter is used and no relevant sources can be found.

14.4 v1.3-5 | 2020-10-29

Affects all versions since v1.0.

- Endpoints to POST meter data will now check incoming data to see if the required asset's resolution is being used — upsampling is done if possible. These endpoints can now return the REQUIRED_INFO_MISSING status 400 response.
- Endpoints to GET meter data will return data in the asset's resolution — downsampling to the "resolution" parameter is done if possible.
- As they need to determine the asset, all of the mentioned POST and GET endpoints can now return the UNRECOGNIZED_ASSET status 4000 response.

14.5 v1.3-4 | 2020-06-18

- Improved support for use cases of the *getDeviceMessage* endpoint in which a longer duration, between posting UDI events and retrieving device messages based on those UDI events, is required; the default *time to live* of UDI event identifiers is prolonged from 500 seconds to 7 days, and can be set as a config variable (*FLEXMEASURES_PLANNING_TTL*)

14.6 v1.3-3 | 2020-06-07

- Changed backend support (API specifications unaffected) for scheduling charging stations to scheduling Electric Vehicle Supply Equipment (EVSE), in accordance with the Open Charge Point Interface (OCPI).

14.7 v1.3-2 | 2020-03-11

- Fixed example entity addresses in simulation section

14.8 v1.3-1 | 2020-02-08

- Backend change: the default planning horizon can now be set in FlexMeasures's configuration (*FLEXMEASURES_PLANNING_HORIZON*)

14.9 v1.3-0 | 2020-01-28

- Introduced new event type "soc-with-targets" to support scheduling charging stations (see extra example for the *postUdiEvent* endpoint)
- The *postUdiEvent* endpoint now triggers scheduling jobs to be set up (rather than scheduling directly triggered by the *getDeviceMessage* endpoint)
- The *getDeviceMessage* now queries the job queue and database for an available schedule

14.10 v1.2-3 | 2020-01-28

- Updated endpoint descriptions with additional possible status 400 responses:
  - INVALID_DOMAIN for invalid entity addresses
  - UNKNOWN_PRICES for infeasible schedules due to missing prices

14.11 v1.2-2 | 2018-10-08

- Added a list of registered types of weather sensors to the Simulation section and *postWeatherData* endpoint
- Changed example for the *postPriceData* endpoint to reflect Korean situation

14.12 v1.2-1 | 2018-09-24

- Added a local table of contents to the Simulation section
- Added a description of the *postPriceData* endpoint in the Simulation section
- Added a description of the *postWeatherData* endpoint in the Simulation section
- Revised the subsection about posting power data in the Simulation section
- Revised the entity address for UDI events to include the type of the event

```
i.e.

{
    "type": "PostUdiEventRequest",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203:soc",
}

rather than the erroneously double-keyed:

{
    "type": "PostUdiEventRequest",
    "event": "ea1.2018-06.io.flexmeasures.company:7:10:203",
    "type": "soc"
}
```

14.13 v1.2-0 | 2018-09-08

- Added a description of the *postUdiEvent* endpoint in the Prosumer and Simulation sections
- Added a description of the *getDeviceMessage* endpoint in the Prosumer and Simulation sections

14.14 v1.1-5 | 2020-06-18

- Fixed the *getConnection* endpoint where the returned list of connection names had been unnecessarily nested

14.15 v1.1-4 | 2020-03-11

- Added support for posting daily and weekly prices for the *postPriceData* endpoint

14.16 v1.1-3 | 2018-09-08

- Added the Simulation section:
  - Added information about setting up a new simulation
  - Added examples for calling the *postMeterData* endpoint
  - Added example for calling the *getPrognosis* endpoint

14.17 v1.1-2 | 2018-08-15

- Added the *postPrognosis* endpoint
- Added the *postPriceData* endpoint
- Added a description of the *postPrognosis* endpoint in the Aggregator section
- Added a description of the *postPriceData* endpoint in the Aggregator and Supplier sections

14.18 v1.1-1 | 2018-08-06

- Added the *getConnection* endpoint
- Added the *postWeatherData* endpoint
- Changed the Introduction section:
  - Added information about the sign of power values (production is negative)
  - Updated information about horizons (now anchored to the end of each time interval rather than to the start)
- Added an optional horizon to the *postMeterData* endpoint

14.19 v1.1-0 | 2018-07-15

- Added the *getPrognosis* endpoint
- Changed the *getMeterData* endpoint to accept an optional resolution, source, and horizon
- Changed the Introduction section:
  - Added information about timeseries resolutions
  - Added information about sources
  - Updated information about horizons
- Added a description of the *getPrognosis* endpoint in the Supplier section

14.20 v1.0-1 | 2018-07-10

- Moved specifications to be part of the platform's Sphinx documentation:
  - Each API service is now documented in the docstring of its respective endpoint
  - Added sections listing all endpoints per version
  - Documentation includes specifications of **all** supported API versions (supported versions have a registered Flask blueprint)

14.21 v1.0-0 | 2018-07-10

- Started change log
- Added Introduction section with notes regarding:
  - Authentication
  - Relevant roles for the API
  - Key notation
  - The addressing scheme for assets
  - Connection group notation
  - Timeseries notation
  - Prognosis notation
  - Units of timeseries data
- Added a description of the *getService* endpoint in the Introduction section
- Added a description of the *postMeterData* endpoint in the MDC section
- Added a description of the *getMeterData* endpoint in the Prosumer section

CHAPTER

FIFTEEN

CODE DOCUMENTATION

Go To source.

HTTP ROUTING TABLE

/api