# FLEXERGY

## Deliverable 6 – Functional Specification and Requirements Identification of the Storage Energy Management System

Activity 3:

Specification of the Energy Storage Management Platform and Dashboard

| | |
|---|---|
| Editor: | Luis Miguel Costa |
| Dissemination level: (Confidentiality) | Public |
| Suggested readers: | |
| Version: | 04.00 |
| Total number of pages: | 68 |
| Keywords: | Energy Storage, System Architecture, Energy Management |


FLEXERGY ABSTRACT

The FLEXERGY project aims at the development of an advanced management solution, highly innovative and provided of artificial intelligence, for the management of assets of battery energy storage systems, integrated with renewable energy sources or for application within a microgrid



FLEXERGY is a project co-funded by:

COMPETE 2020 PORTUGAL 2020 UNIÃO EUROPEIA Fundo Europeu de Desenvolvimento Regional

# Document

| | | |
|---|---|---|
| **Name** | Deliverable 6 – Functional Specification and Requirements Identification of the Storage Energy Management System | |
| **Document No.** | AS19007921 | |
| **Revision and Date** | 4.0 | 2020-08-31 |
| **Project** | FLEXERGY | |
| **Made by** | INESC TEC – Luis Miguel Costa | |
| **Reviewed by** | Luís Azevedo Costa | |
| **Approved by** | Alberto Maia Bernardo | |
| **Total Pages** | 68 | |

## Language Requirements (for non-native English speakers)

In order to fully understand the content of this document, it is therefore recommended that the reader possesses a language proficiency equivalent to B1 level, according to European Language Levels

## Disclosure

# Revisions

| Rev. | Date | Comments | Author |
|---|---|---|---|
| 0.00 | 2020-03-20 | • First draft in accordance with INESC TEC's first proposal. | INESC TEC |
| 1.00 | 2020-04-30 | • Division of input data structure "configParams" into "settings" and "assets";<br>• Addition of relevant parameters for load curtailment;<br>• Addition of relevant parameters for conventional generation configuration, which will now be controllable by steps;<br>• Addition of $U_{dc}$ and $U_{cut\text{-}off}$ parameters to BESS assets' input data structures and measures $U_{dc}$ parameter to BESS measures data structure;<br>• Addition of reactive setpoints input structure. | INESC TEC |
| 2.00 | 2020-06-01 | • Inclusion of "systemID" on all major configuration and input structures to making it possible to define more than one system per ESManager;<br>• To accommodate conventional generation scheduling, a power output band of each asset must now be provided at each optimization call for each time step;<br>• A merit order must now be established between curtailable load assets and EV chargers to establish; a distinct merit order parameter is also available for establishing preference between load curtailment or conventional generation production for microgrid systems;<br>• The time step is now provided in minutes (provision in hours was prone to errors when using fractions of the hour);<br>• When operating on hybrid parks, it is now possible to define the imposition of a constant, hourly dependent, $\tan(\varphi)$ at the PCC;<br>• A new structure is provided for conveying information about eventual maintenance periods where BESS assets will be inactive during the optimization horizon. | INESC TEC |
| 3.00 | 2020-08-31 | • Included in the outputs structure the reactive power discharge set points for each BESS asset;<br>• Specified how missing/redundant data is handled for day-ahead bids, prices/tariffs, conventional generators' power limits and $\tan(\varphi)$ at the PCC.<br>• Clarification on how the time limits provided for curtailing flexible loads and EV chargers are adjusted to the time step provided. | |

# Executive Summary

This deliverable – Deliverable 6 – presents the technical and functional specifications of the Storage Energy Management main APIs, namely: the BESS optimization and BES life cycle assessment and parameter estimation APIs. The document describes the API main function, inputs and outputs. The optimization problem and formulation and battery models are also described.

# Table of Contents

# List of Figures

# List of Tables

# Glossary

API – application programming interface

BES – battery energy storage (unit)

BESS – battery energy storage system

EV – electric vehicle

HMI – human-machine interface

Li-ion – lithium-ion

MILP – mixed integer linear programming

MPC – model predictive control

PV - photovoltaic

RES – renewable energy sources

REST – Representational State Transfer

SCADA - supervisory control and data acquisition

SoC – state of charge

UC – use case

# 1. Functional Architecture of the Storage Energy Management APIs

The main objective of the Energy Storage Manager, ES Manager, is to control the Battery Energy Storage Systems (BESS) installed in hybrid power plants or microgrids, considering specified operation objectives and the estimation of their degradation. The Storage Energy Management system is composed of two main modules:

- Flexergy Optimizer API, responsible for the optimal dispatch of BESS considering a given operation objective (e.g. revenue maximization, arbitrage, self-consumption, etc.) and the minimization of the storage's degradation process. The BESS operation plan can be defined for a pre-configured time horizon (daily or intra daily) based on generation and load forecasting and updated to adapt its operation strategy to the real-time conditions of the hybrid park or microgrid, based on a predictive control strategy.

- BES Life Cycle Assessment and parameter estimation API, responsible for estimating the degradation of each BES unit that constitutes the BESS, considering the impact of the operation strategy defined by the optimization. In addition, the module will provide additional information regarding BES usage for a given period, namely: number of estimated cycles, energy charged/discharged and usage degree of intensity (i.e. indicate if the battery is being used conservatively or intensively).

The APIs will run in an application server, interacting with the main application layer through REST service. The REST interfaces support server and client services/functions allowing for requests "On Event", i.e. triggered by the client. The application server is also responsible for validating the data returned by the algorithms and can save a limited amount of data history, reducing the need to send in each request large amounts of data.

The general diagram presented in Figure 1 describes interactions between the BESS energy management APIs and other ES Manager applications. The Operation manager will be responsible for calling the APIs in case of any significant change in the input data or in study mode. It will also be responsible for sending all data required for running the APIs and for the implementation of the resulting set points.

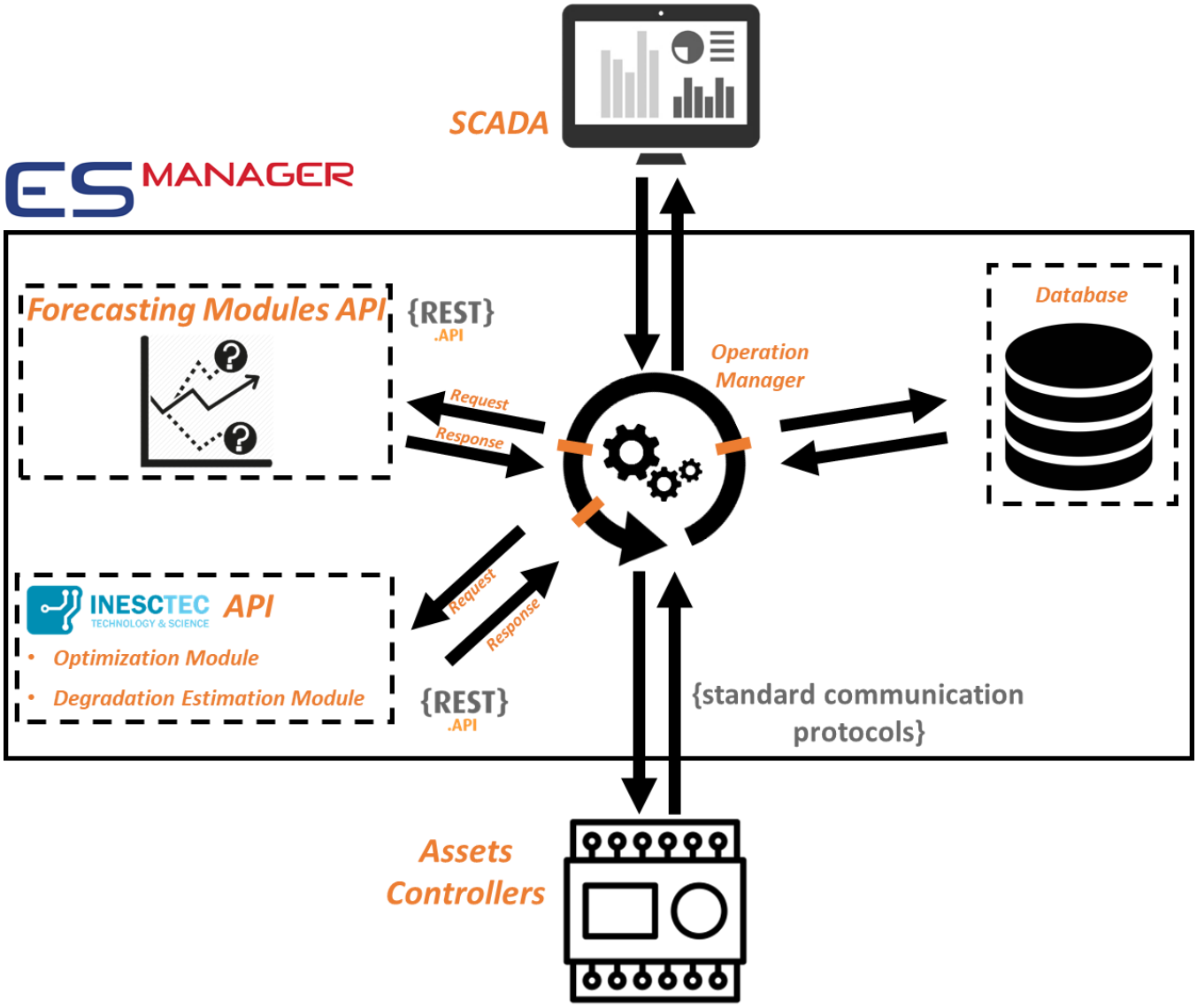



**Figure 1** - General diagram of the ES Manager energy optimization module and related systems.

## 1.1 Applicable Use Cases

Figure 2 represents the mapping between the project use cases, the BESS energy management APIs and its outputs. The BESS Optimization API defines the control of BESS when integrated in different systems. According to the FLEXERGY Use Cases the BESS will be installed to help manage hybrid parks (UC1) and microgrids (UC2, UC3).

The BESS Optimization API will be configured according to the system where the BESS is installed, its optimization objectives and additional energy services to meet the project use cases. As outputs, the API will provide the active and reactive power set points for the next time step, or in advance according to the ES Manager scheduler.

The BES Life Cycle Assessment and parameter estimation API implements use case 5, providing an estimation of BES usage and degradation considering the impact of the operation strategy defined by the optimization module.

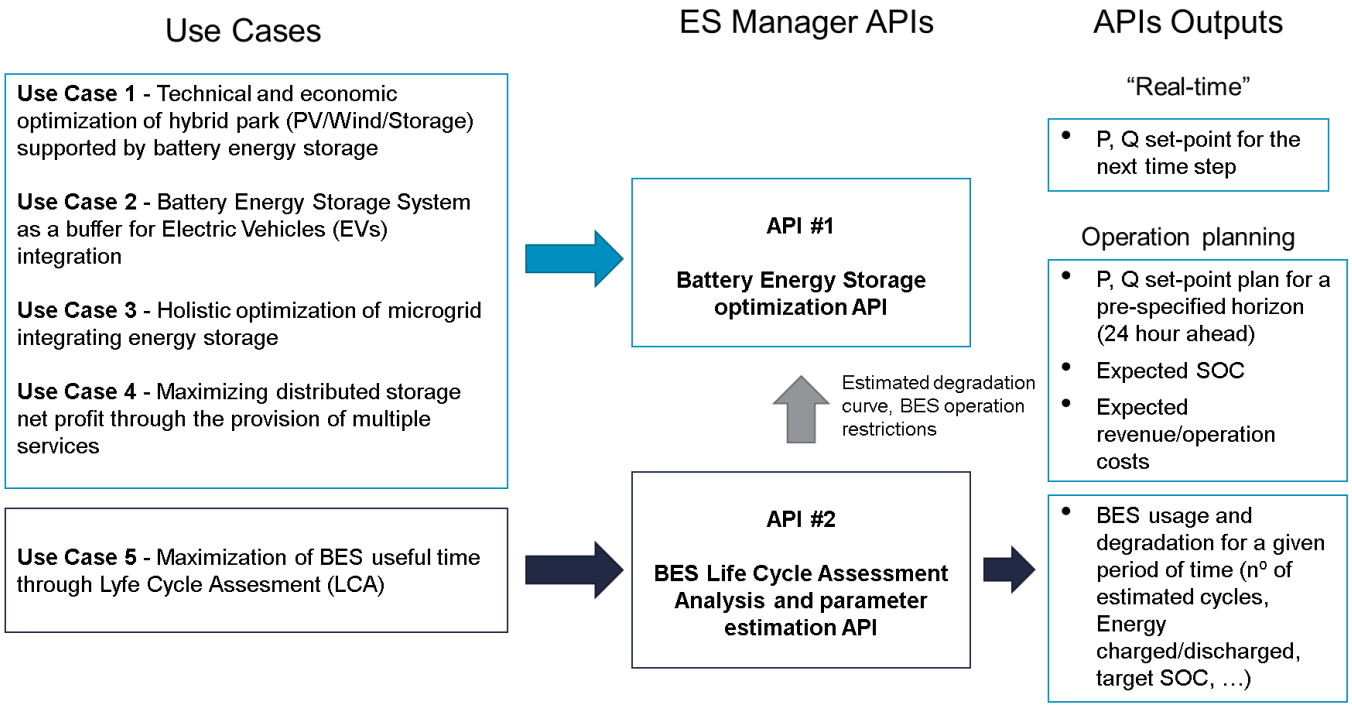



**Figure 2** - Mapping of use cases, API and outputs.

# 2. Optimization Module

## 2.1 General description

The BESS Optimization API is responsible for the optimal dispatch of BESS considering a given operation objective (e.g. revenue maximization, arbitrage, self-consumption, etc.) and the minimization of the storage's degradation process.

The BESS operation plan can be defined for a pre-configured time horizon (daily or intraday) based on a predictive control strategy. The operation plan is defined based on the real-time conditions of the hybrid park or microgrid and in the generation and load forecasting updates.

As shown in Figure 3, the algorithm runs "On Event", meaning that each run must be previously triggered by a client's request. Two main endpoints are provided, one for planning the operation for a given period (hours to one day) and another for real-time operation (providing set points to be implemented in the next time step). To provide an optimal control of a given system (microgrid or hybrid park), the API should run every 5-15 minutes in real-time operation mode. The planning mode can run every 24 hours or other pre-defined period in order to provide information for the next hours to be included in the HMI or to other operation planning activities, such as the activities related with market participation.

Real-time and planning operation modes will be supplied with short-term and medium-term forecast, respectively, ensuring optimization APIs input data to achieve a higher accuracy level for both optimization horizons. Every time there are any significant changes in the system, for example when a BESS unit (inverter plus battery system) or when a given PV string becomes out of service, the API reformulates the control strategy. In this case update of the input structure must be performed prior to any optimization request, through the available endpoints.

The API can also run in study mode for planning purposes, providing for example the charging/discharging power, expected SOC, battery system degradation for a given operation strategy, or expected revenues for a pre-defined period. Provided the sufficient computational power, this mode can run in parallel with the operation or planning modes running at the time.

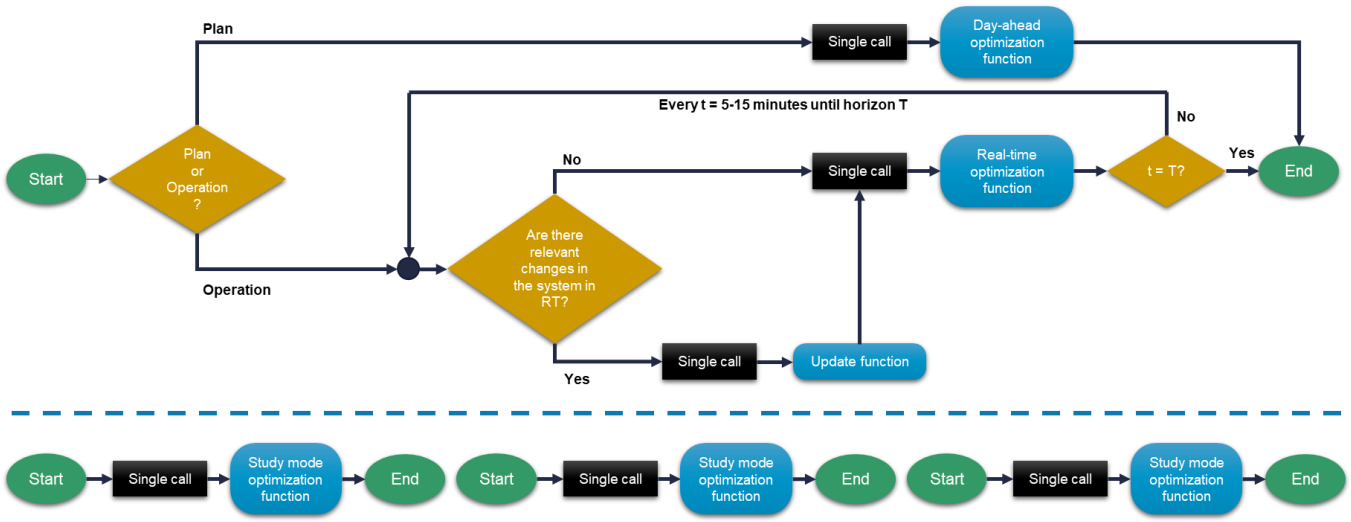



**Figure 3** - BESS Optimization API running modes.

Regarding the system, the API optimizes the BESS considering the system it is installed and the definition of the objective function. As shown in Table 1, the system configuration depends on the identification of the technologies involved. For example, if the BESS is installed in a microgrid system, load assets should be defined in addition to renewable based generation. Electric Vehicles' (EV) chargers and conventional generation units are optional. The nature of these technologies will dictate if they are controllable or not.

Table 2 presents the objective functions implemented and its applicability to the systems where the BESS will be installed. As described in the table, the minimization of deviations costs is only applicable to the hybrid parks, while self-consumption maximization is exclusive of microgrids, since in each case the objective function only takes meaning in the respective system's context.

**Table 1** – Applicability of API considering system and technologies involved.

| Configuration | Options | Technologies | Controllable [1)] |
|---|---|---|---|
| Applicable System | Microgrid (UC 2 and UC3) | Load | No |
| | | Controllable Load | Yes[2)] |
| | | EV | Yes[2)] |
| | | PV | Yes[3)] |
| | | Wind | Yes[3)] |
| | | Other types of Generation (PQ controlled) | No[4)] |
| | Hybrid Park (UC1) | PV | Yes[3)] |
| | | Wind | Yes[3)] |

1) Control variable of BESS optimization algorithm
2) Optimization algorithm determines the number of load assets to curtail. It plans its reconnection but doesn't consider the rebound effect.
3) Models the PV generation as a single PQ generator
4) Modeled as a negative load.

**Table 2** – Objective Functions for the BESS optimization API and identification of applicable use cases.

| Configuration | Technologies | Applicable System/Use Cases |
|---|---|---|
| Optimization Objectives | Minimization of deviations costs | Hybrid Park (UC1) |
| | Minimization of energy deviations (from a given profile e.g. day-ahead bids) | Hybrid Park (UC1), Microgrid (UC2, UC3) |
| | Arbitrage (Maximization of net profit) | Hybrid Park (UC1), Microgrid (UC2, UC3) |
| | Maximization of self-consumption and minimization of operation costs | Microgrid (UC2, UC3) |

One of the main features of this API is the battery modeling. As shown in Figure 4, the user can select different modeling options, considering the battery V-I characteristic, charging and discharging efficiency as well as degradation. This will be detailed in section 2.5.3.

The API models considers battery degradation in the definition of the battery control strategy, as described also in UC 5- Maximizing Battery useful time. As detailed in more detail in section 2.5.3, the model developed estimates the BESS degradation based on the BESS degradation curves provided by the batteries manufacturers that can be updated by the Lyfe Cycle Assessment tool (API#2 in Figure 2).

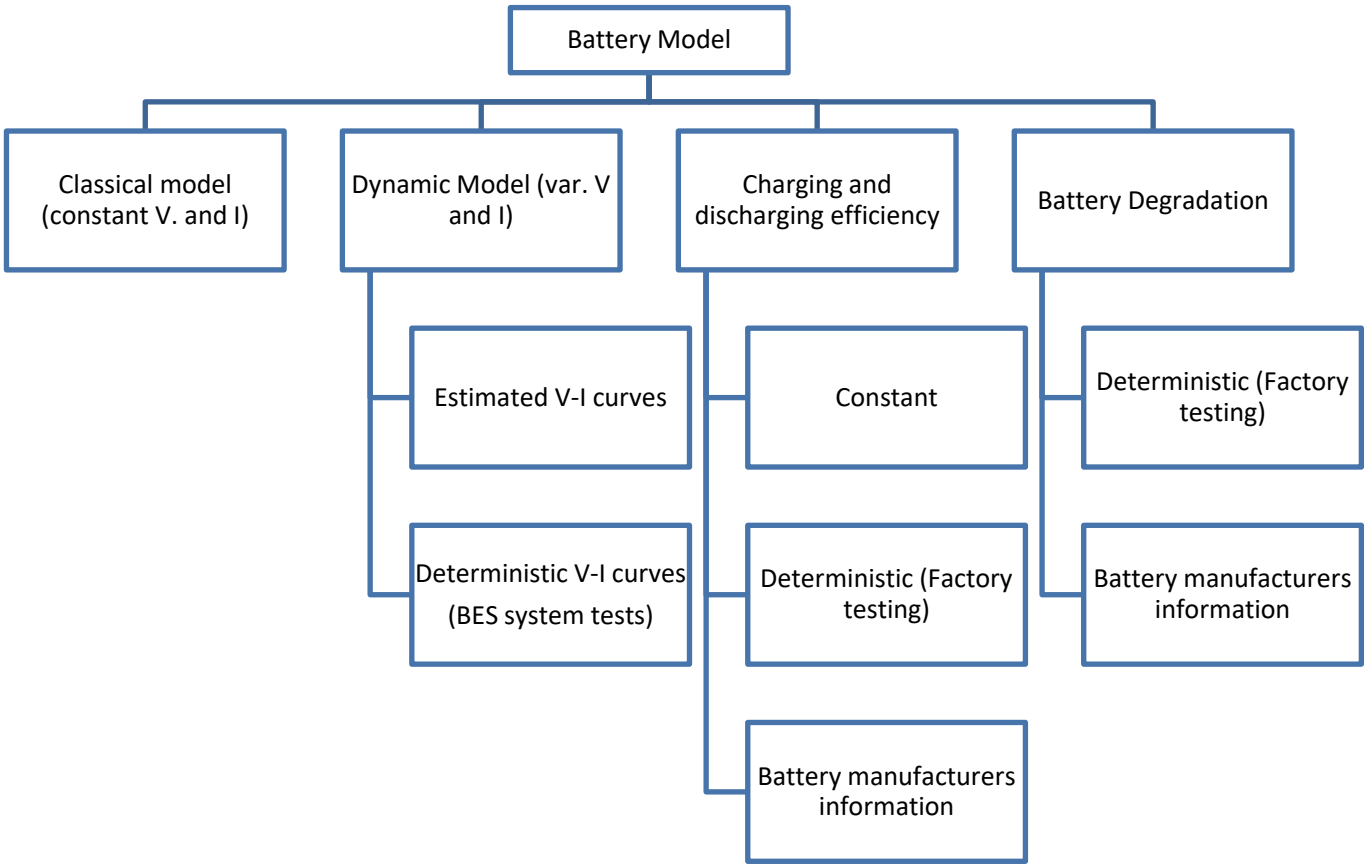



**Figure 4** - Schematic of Battery modelling options.

The next sections describe in more detail the BESS optimization API, regarding its inputs, outputs and functions. In section 2.5 is presented the optimization problem formulation.

## 2.2 Inputs/Data Structures

The optimization module will consider two types of data structures functionally divided into configuration parameters and measurements and forecasts.

Configuration parameters concern the system's and technologies' configuration, control objectives and modelling options. Different structures will be considered for the different optimization objective functions.

Measured inputs include the status and electric measurements provided in real-time. Forecasts will include generation, load demand and market prices to be considered for the optimization horizon. These measurements and forecasts must be provided every time the module is called.

The following subsections will describe functional input data models where data structures are named and described and hierarchical relationships are established. Note that for each structure, the variables' type is sometimes succeeded by a decorator, (*), indicating the required parameters. All other are considered optional.

### 2.2.1 Configuration parameters

In this chapter the configuration input data structures will be described. These include the structures that will convey the needed information for the initialization and further updates to the system, which must be provided before any request to the optimization general methods.

#### 2.2.1.1 Main configuration structure - **settings**

This structure defines a main system (hybrid park or microgrid) and its main characteristic parameters. Since more than one system can be stored in the module's internal database, an appropriate ID must be provided when configuring these general settings. Whenever (re)configuring the respective assets for the system (see sections 2.2.1.2 to 2.2.1.8) and at each API call (see section 2.2.2.1) the same ID must be provided to link those actions to the correct system.

**Table 3 -** Functional definition of the modules main static inputs structure: **settings**.

| Type | Identification - Description | Units |
|---|---|---|
| int (*) | **system** - Defines the system in which the BESS is installed:<br>• 1. Hybrid Park<br>• 2. Microgrid | - |
| string (*) | **systemID** - Designation of the system.<br>e.g. "hybridpark1", "microgrid2", etc. | - |
| double | **reNom** - Total renewable generation installed capacity.<br>• Default: 0.0 | MVA |
| double | **convNom** - Total conventional generation installed capacity.<br>• Default: 0.0 | MVA |
| int | **meritOrder** - In case conventional generation and curtailable/EV chargers' assets are configured, this parameter establishes which one is the preference of the user, when additional flexibility is needed:<br>• 0: it is preferable to start by resorting to conventional generation (default);<br>• 1: it is preferable to start by curtailing flexible load. | - |
| double | **pccLimitValue** - PCC power flow limits.<br>• Default: 1.00E+09 (equivalent to not considering any limit) | MVA |
| bool | **addOnSoc** - Indicates if the BES units' structures will include the add-on: dynamic SoC limits (see section 2.5.3.1):<br>• False - Do not include (default)<br>• True - Include | - |
| bool | **addOnInv** - Indicates if the BES units' structures will include the add-on: inverter efficiency modelling (see section 2.5.3.2):<br>• False - Do not include (default)<br>• True - Include | - |
| bool | **addOnDeg** - Indicates if the BES units' structures will include the add-on: degradation limit (see section 2.5.3.3):<br>• False - Do not include (default)<br>• True - Include | - |

#### 2.2.1.2 Technology structure – **pvPlant unitary structure**

The technology structures define the several components of the system. When more than one asset is to be configured, the endpoints are prepared to accept arrays of these unitary structures.
The pvPlant structure configures a solar generation plant.

**Table 4 -** Functional definition of the modules static inputs structures: **pvPlant unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **systemID** – Designation of the system for which the assets are being configured.<br>• e.g. "hybridpark1", "microgrid2", … etc.<br><br>Note: this designation must coincide with the one given for the respective system provided at the settings structure (section 2.2.1.1). | - |
| string (*) | **designation** - PV plant designation.<br>• e.g. "pv1", "pv2", … "pv10", … etc. | - |
| bool (*) | **status:**<br>• False – OFF<br>• True – ON | - |
| double (*) | **totalNom** - Maximum total capacity. | MVA |
| double | **powerFactor** - $\cos\varphi$ at which the PV plants operate.<br>• Default: 1.0 | - |
| double | **curtailPerc** - Curtailment percentage limit.<br>• Default: 0.0 (i.e., no curtailment is allowed) | % |

### 2.2.1.3 Technology structure – **windPlant unitary structure**

The windPlant structure configures a wind turbines' plant, rather than defining a single structure for each wind turbine.

**Table 5 -** Functional definition of the modules static inputs structures: **windPlant unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **systemID** – Designation of the system for which the assets are being configured.<br>• e.g. "hybridpark1", "microgrid2", … etc.<br><br>Note: this designation must coincide with the one given for the respective system provided at the settings structure (section 2.2.1.1). | - |
| string (*) | **designation** - Wind plant designation.<br>• e.g. "wind1", "wind2", … "wind10", … etc. | - |
| bool (*) | **status:**<br>• False – OFF<br>• True – ON | - |
| double (*) | **totalNom** - Maximum total capacity. | MW |
| double | **powerFactor** - $\cos\varphi$ at which the wind plants operate.<br>• Default: 1.0 | - |
| double | **curtailPerc** - Curtailment percentage limit.<br>• Default: 0.0 (i.e., no curtailment is allowed) | % |

### 2.2.1.4 Technology structure – **convGen unitary structure**

The convGens structure configures a single or a set of conventional generators that can be connected to the system.

**Table 6 -** Functional definition of the modules static inputs structures: **convGen unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **systemID** – Designation of the system for which the assets are being configured.<br>• e.g. "hybridpark1", "microgrid2", … etc.<br><br>Note: this designation must coincide with the one given for the respective system provided at the settings structure (section 2.2.1.1). | - |
| string (*) | **designation** - Conventional generators' designation.<br>• e.g. "conv1", "conv2", … "conv10", … etc. | - |
| bool (*) | **status**:<br>• False – OFF<br>• True – ON | - |
| double (*) | **pMax -** Maximum power deliverable by the unit when fully operational. | MVA |
| double (*) | **pMin -** Minimum power deliverable by the unit when fully operational. | MVA |
| double | **powerFactor** - $\cos\varphi$ at which the conventional generators operate.<br>• Default: 1.0 | - |

### 2.2.1.5 Technology structures – **inflexAsset unitary structure**

The inflexAsset structure is designed for individual load assets such as household domains that are not available for load curtailment.

**Table 7 -** Functional definition of the modules static inputs structures: **inflexAsset unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **systemID** – Designation of the system for which the assets are being configured.<br>• e.g. "hybridpark1", "microgrid2", … etc.<br><br>Note: this designation must coincide with the one given for the respective system provided at the settings structure (section 2.2.1.1). | - |
| string (*) | **designation** - Inflexible asset's designation.<br>• e.g. "inflex1", "inflex2", … "inflex10", … etc. | - |
| bool (*) | **status**:<br>• False – OFF<br>• True – ON | - |
| double (*) | **maxP** - Maximum active power limit. | MW |
| double (*) | **maxQ** - Maximum reactive power limit. | MVar |

### 2.2.1.6 Technology structure – **curtailAsset unitary structure**

The curtailAsset structure is designed for individual load assets that are available for load curtailment, excluding EVs.

**Table 8 -** Functional definition of the modules static inputs structures: **curtailAsset unitary structure.**

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **systemID** – Designation of the system for which the assets are being configured.<br>• e.g. "hybridpark1", "microgrid2", … etc.<br><br>Note: this designation must coincide with the one given for the respective system provided at the settings structure (section 2.2.1.1). | - |
| string (*) | **designation** - Curtailable asset's designation.<br>• e.g. "curtail1", "curtail2", … "curtail10", … etc. | - |
| bool (*) | **status**:<br>• False – OFF<br>• True – ON | - |
| double (*) | **maxP** - Maximum active power limit. | MW |
| double (*) | **maxQ** - Maximum reactive power limit. | MVar |
| double | **curtailPerc** - Percentage of the asset's demand available for curtailment.<br>• Default: 100.0 | % |
| int | **minTCycle -** Minimum time between curtailment events (i.e., minimum operation time).<br>• Default: 0 (meaning no minimum amount of time is to be considered between curtailment events) | minutes |
| int | **maxTCut -** Maximum period of time that the load can remain curtailed.<br>• Default: equal to the optimization horizon. | minutes |
| int | **meritOrder** – Relative importance of the curtailable asset. The higher the priority number assigned, the higher the cost of curtailing this asset in relation to other curtailable assets (including the EV chargers).<br>• Options: [1-10]<br>• Default: 1 | - |

### 2.2.1.7 Technology structure – **evAsset unitary structure**

Structure for initializing EV charging managers in the microgrid.

Electric vehicles' charging managers should be all aggregated in this structure. The curtailment set points output by the algorithm relate to the total power to be cut by the chargers as a whole, and must be distributed afterwards by a general controller, providing each individual charger with its own curtailment set point.

**Table 9 -** Functional definition of the modules static inputs structures: **evAsset unitary structure.**

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **systemID** – Designation of the system for which the assets are being configured.<br>• e.g. "hybridpark1", "microgrid2", … etc.<br><br>Note: this designation must coincide with the one given for the respective system provided at the settings structure (section 2.2.1.1). | - |
| string (*) | **designation** - EV charging manager designation.<br>• e.g. "ev_manager" | - |
| bool (*) | **status**:<br>• False – OFF<br>• True – ON | - |
| double (*) | **maxP** - Maximum active power limit. | MW |
| double | **curtailPerc** - Percentage of the asset's demand available for curtailment.<br>• Default: 100.0 | % |
| **int** | **minTUncut -** Minimum time between curtailment events (i.e., minimum operation time).<br>• Default: 0 (meaning no minimum amount of time is to be considered between curtailment | minutes |

| | | |
|---|---|---|
| | events) | |
| int | **maxTCut -** Maximum consecutive time the load can remain curtailed.<br>• Default: equal to the optimization horizon. | minutes |
| **int** | **meritOrder** – Relative importance of the EV chargers. The higher the priority number assigned, the higher the cost of curtailing the EV chargers in relation to the other curtailable assets.<br>• Options: [1-10]<br>• Default: 1 | - |

### 2.2.1.8 Technology structure – **bessAsset unitary structure**

The **bessAssets unitary structure** is designed to harbour the specific characteristics of each individual unit of Li-ion battery and its respective inverter. Optional parameters are *None* by default.

**Table 10** - Functional definition of the modules static inputs structures: **bessAsset unitary structure.**

| Type | Identification - Description | Units |
|---|---|---|
| **string (*)** | **systemID** – Designation of the system for which the assets are being configured.<br>• e.g. "hybridpark1", "microgrid2", … etc.<br><br>Note: this designation must coincide with the one given for the respective system provided at the settings structure (section 2.2.1.1). | - |
| string (*) | **designation** - Storage system's designation.<br>• e.g. "bess1", "bess2", … "bess10", … etc. | - |
| bool (*) | **status**:<br>• False – OFF<br>• True – ON | - |
| double (*) | **vNom** - Battery unit's nominal voltage. | V |
| double (*) | **eNom** - Battery unit's nominal energy capacity. | MWh |
| double | **maxSoc** - Maximum SoC limit of battery unit.<br>• Default: 90.0 | % |
| double | **minSoc** - Minimum SoC limit of battery unit.<br>• Default: 10.0 | % |
| double | **reserveSoc -** Minimum reserve capacity that must be maintained in the battery. If set, this value must be ≥ minSoc.<br>• Default: same as minSoC | % |
| double | **maxCCh** - Maximum C-rate admissible for charging.<br>• Default: 1.0 | - |
| double | **maxCDch** - Maximum C-rate admissible for discharging.<br>• Default: 1.0 | - |
| double | **minPCh** - Minimum power rate admissible for charging.<br>• Default: 2.0 | % |
| double | **minPDch** - Minimum power rate admissible for discharging.<br>• Default: 2.0 | % |
| double | **chEff** - Fixed charging efficiency of the storage system (battery unit + inverter).<br>Note: If only roundtrip efficiency is available, charging efficiency should be provided as the square root of the first.<br>• Default: 100.0 | % |
| double | **dischEff** - Fixed discharging efficiency of the storage system (battery unit + inverter). | % |

| | | |
|---|---|---|
| | Note: If only roundtrip efficiency is available, discharging efficiency should be provided as the square root of the first.<br>• Default: 100.0 | |
| double | **eolCriterion** – End-of-life criterion (e.g. if 30% degradation is to be considered, EOL should be given as 70%)<br>• Default: 70.0 | % |
| int | **lifetime** - Battery units' desired lifetime in years<br>• Default: 0 (equivalent to setting addOnDeg = False) | Years |
| double (*) | **invSNom** - Inverter's nominal power | MVA |
| double (*) | **invVNom** - Inverter's nominal voltage | V |
| double (*) | **invMaxIDC** – inverter's maximum DC voltage | A |
| testData | **testData** – Single testData structure (section 2.2.1.8.1).<br>• Default: None | - |

The introduction of relational data from the laboratorial tests performed on the storage systems requires the definition of specific additional data structures that will be presented on the following subsections.

#### 2.2.1.8.1 Test data main substructure – **testData**

Main substructure to register the test data performed over the storage system considered. It is important to note that the inclusion of the add-ons to the storage model is dependent on the data available through this structure. Although none of the parameter structures defined are considered as required, note that:

- In order to activate addOnSoc, structures vNomD, vNomC, dLim and cLim must be properly provided;
- In order to achieve a better approximation when activating addOnInv, structures effD and effC or, in return, structure roundEff must be properly provided. If effD and effC structures are properly provided, roundEff will not be considered.

**Table 11** - Functional definition of the modules main test data structure: **testData.**

| Type | Identification - Description | Units |
|---|---|---|
| array (vNomD) | **vNomD** - Array of vNomD unitary structures (section 2.2.1.8.2).<br>Array size: number of available test set points; a single set point is enough for the structure to be considered. | - |
| array (vNomC) | **vNomC** - Array of vNomC unitary structures (section 2.2.1.8.3).<br>Array size: number of available test set points; a single set point is enough for the structure to be considered. | - |
| array (dLim) | **dLim** - Array of dLim unitary structures (section 2.2.1.8.4).<br>Array size: number of available test set points; a single set point is enough for the structure to be considered. | - |
| array (cLim) | **cLim** - Array of cLim unitary structures (section 2.2.1.8.5).<br>Array size: number of available test set points; a single set point is enough for the structure to be considered. | - |
| array (effD) | **effD** - Array of effD unitary structures (section 2.2.1.8.6).<br>Array size: number of available test set points; a single set point is enough for the structure to be considered. | - |
| array (effC) | **effC** - Array of effC unitary structures (section 2.2.1.8.7).<br>Array size: number of available test set points; a single set point is enough for the structure to be considered. | - |

| | | |
|---|---|---|
| array (roundEff) | **roundEff** - Array of roundEff unitary structures (section 2.2.1.8.8).<br><br>Array size: number of available test set points; a single set point is enough for the structure to be considered. | - |

#### 2.2.1.8.2 Test data substructure – **vNomD unitary structure**

Substructure to register the average measured voltage during a storage asset's full discharge test at a specified C-rate.

**Table 12** - Functional definition of the modules test data structures: **vNomD unitary structure.**

| Type | Identification - Description | Units |
|---|---|---|
| double (*) | **cRate** - C-rate at which the full discharge test was performed. | - |
| double (*) | **vAvg** - Average voltage registered. | V |
| int | **trial** - Trial number (used to distinguish between different trials, in case more than one test was performed over the same unit).<br>• Default: None | - |

#### 2.2.1.8.3 Test data substructure – **vNomC unitary structure**

Substructure to register the average measured voltage during a storage asset's full charge test at a specified C-rate.

**Table 13** - Functional definition of the modules test data structures: **vNomC unitary structure.**

| Type | Identification - Description | Units |
|---|---|---|
| double (*) | **cRate** - C-rate at which the full charge test was performed. | - |
| double (*) | **vAvg** - Average voltage registered. | V |
| int | **trial** - Trial number (used to distinguish between different trials, in case more than one test was performed over the same unit).<br>• Default: None | - |

#### 2.2.1.8.4 Test data substructure – **dLim unitary structure**

Substructure to register the measured remaining energy content at the end of a storage asset's full discharge test at a specified C-rate.

**Table 14** - Functional definition of the modules test data structures: **dLim unitary structure.**

| Type | Identification - Description | Units |
|---|---|---|
| double (*) | **cRate** - C-rate at which the full discharge test was performed. | - |
| double (*) | **eRemain** - Remaining energy content registered. | % |
| int | **trial** - Trial number (used to distinguish between different trials, in case more than one test was performed over the same unit).<br>• Default: None | - |

#### 2.2.1.8.5 Test data substructure – **cLim unitary structure**

Substructure to register the measured remaining energy content at the end a storage asset's full charge test at a specified C-rate.

**Table 15** - Functional definition of the modules test data structures: **cLim unitary structure.**

| Type | Identification - Description | Units |
|---|---|---|
| double (*) | **cRate** - C-rate at which the full charge test was performed. | - |
| double (*) | **eRemain** - Remaining energy content registered. | % |
| int | **trial** - Trial number (used to distinguish between different trials, in case more than one test was performed over the same unit).<br>• Default: None | - |

#### 2.2.1.8.6 Test data substructure – **effD unitary structure**

Substructure to register the average measured discharge efficiency during a storage asset's full discharge test at a specified C-rate.

**Table 16** - Functional definition of the modules test data structures: **effD unitary structure.**

| Type | Identification - Description | Units |
|---|---|---|
| double (*) | **cRate** - C-rate at which the full discharge test was performed. | - |
| double (*) | **effDchAvg** - Average discharge efficiency measured. | % |
| int | **trial** - Trial number (used to distinguish between different trials, in case more than one test was performed over the same unit).<br>• Default: None | - |

#### 2.2.1.8.7 Test data substructure – **effC unitary structure**

Substructure to register the average measured charge efficiency during a storage asset's full discharge test at a specified C-rate.

**Table 17** - Functional definition of the modules test data structures: **effC unitary structure.**

| Type | Identification - Description | Units |
|---|---|---|
| double (*) | **cRate** - C-rate at which the full charge test was performed. | - |
| double (*) | **effChAvg** - Average charge efficiency measured. | % |
| int | **trial** - Trial number (used to distinguish between different trials, in case more than one test was performed over the same unit).<br>• Default: None | - |

#### 2.2.1.8.8 Test data substructure – **roundEff unitary structure**

Substructure to register the average measured roundtrip efficiency during a storage asset's full cycle test at a specified C-rate.

**Table 18** - Functional definition of the modules test data structures: **roundEff unitary structure.**

| Type | Identification - Description | Units |
|---|---|---|
| double (*) | **cRate** - C-rate at which the full cycle test was performed.<br>• Default: None | - |
| double (*) | **roundEffAvg** - Average roundtrip efficiency measured. | % |
| int | **trial** - Trial number (used to distinguish between different trials, in case more than one test was performed over the same unit).<br>• Default: None | - |

### 2.2.2 Optimization parameters

This section is dedicated to the input data structures constituting the request body of the optimization methods.

#### 2.2.2.1 General optimization structure

**Table 19 -** Functional definition of the modules main optimization structure: **milpInputs.**

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **requestID** – Identification of the request. Although each request is followed by an immediate response, the outputs of the optimization algorithm may take some time to be computed (seconds to minutes). In order to close the request connection, avoiding possible errors if it was kept on hold, the outputs are sent through a request via the INESC TEC API to a server on the client's side. This strategy raises the need to create an identification tag for each optimization request, which is defined in this parameter.<br>• e.g. MD5 HASH of millisecond "c06c37631c60c6d067345d670909a328" | - |
| string (*) | **systemID** – Designation of the system for which the optimization is to be performed.<br>• e.g. "hybridpark1", "microgrid2", … etc.<br><br>Note: this designation must coincide with the one given for the respective system provided at the settings structure (section 2.2.1.1). | - |
| milp (*) | **milp** – milp structure (section 2.2.2.2). | - |
| measures (*) | **measures** – measures structure (section 2.2.2.3). | - |
| forecasts (*) | **forecasts** – forecasts structure(2.2.2.12). | - |

#### 2.2.2.2 Main MILP structure – **milp**

The milp structure stores the configurable parameters that will rule the base MILP of the module.

**Table 20 -** Functional definition of the modules main milp parameters structure: **milp.**

| Type | Identification - Description | Units |
|---|---|---|
| int | **step** - Optimization step given in minutes<br>• Available options: [5, 10, 15, 20, 30, 60, 120]<br>• Default: 60 | Minutes |
| int | **horizon** - MILP temporal horizon.<br>• Available options: [6, 12, 24, 48]<br>• Default: 24 | Hours |
| string (*) | **init** - Starting date and time of the optimization horizon in ISO 8601 format.<br>e.g. "2020-01-01T01:00:00Z" | - |

| | | |
|---|---|---|
| int | **obj** - Desired objective function to consider given the system structure selected:<br>• For Hybrid Park:<br>  ○ 1 - Arbitrage (default)<br>  ○ 2 - Minimization of energy deviations<br>  ○ 3 - Minimization of deviations costs<br>• For Microgrid:<br>  ○ 1 - Arbitrage<br>  ○ 2 - Minimization of energy deviations<br>  ○ 3 - Maximization of self-consumption and minimization of operation costs (default) | - |
| int | **meritOrder** - merit order between load curtailment or conventional generation's production, if such flexibility is necessary to comply with the operational constraints.<br>• Available option:<br>[1 - conventional generation's production preferred over load curtailment;<br>2 - load curtailment preferred over conventional generation's production]<br>• Default: 1 | - |
| double | **mipgap** - solver's tolerance when searching for the optimal solution.<br>• Minimum: 0.0<br>• Maximum: 1.0<br>• Default: 0.001 | - |
| int | **timeout** - solver's temporal limit to find optimal solution, in seconds<br>• Default: 10 | s |

### 2.2.2.3 Main measurements structure – **measures**

This structure serves the purpose of being a placeholder for all measurements' structures, i.e. all the registered values in real-time that are required to characterize the current state of the system.

Note: To increase the robustness of the measured data provided, every "current observable" measurement should be considered, whenever possible, as an average of the records of the past 5-15 minutes.

**Table 21-** Functional definition of the modules main measured inputs structure: **measures.**

| Type | Identification - Description | Units |
|---|---|---|
| array (bessMeasures) (*) | **bessMeasures** - Array of bessMeasures unitary structures (section 2.2.2.4).<br>Array size: number of respective assets defined (section 2.2.1.8). | - |
| array (daBids) | **daBids** - Array of daBids unitary structures (section 2.2.2.5).<br>Array size: Number of day-ahead time steps.<br>Note: only required, when applicable (see section 2.2.2.5). | - |
| array (pvMeasures) | **pvMeasures** - Array of pvMeasures unitary structures (section 2.2.2.6).<br>Array size: number of respective assets defined (section 2.2.1.2).<br>Note: only required for the realTimeOpt method. | - |
| array (windMeasures) | **windMeasures** - Array of windMeasures unitary structures (section 2.2.2.7).<br>Array size: number of respective assets defined (section 2.2.1.3).<br>Note: only required for the realTimeOpt method. | - |
| array (convMeasures) | **convMeasures** - Array of convMeasures unitary structures (section 2.2.2.8).<br>Array size: number of respective assets defined (section 2.2.1.4).<br>Note: only required for the realTimeOpt method. | |
| array (inflexMeasures) | **inflexMeasures** - Array of inflexMeasures unitary structures (2.2.2.9). | - |

| | | |
|---|---|---|
| | Array size: number of respective assets defined (section 2.2.1.5).<br><br>Note: only required for the realTimeOpt method. | |
| array (curtailMeasures) | **curtailMeasures** - Array of curtailMeasures unitary structures (section 2.2.2.10).<br><br>Array size: number of respective assets defined (section 2.2.1.6).<br><br>Note: only required for the realTimeOpt method. | - |
| array (evMeasures) | **evMeasures** - Array of evMeasures unitary structure (section 2.2.2.11).<br><br>Array size: number of respective assets defined (section 2.2.1.7).<br><br>Note: only required for the realTimeOpt method. | |
| float | **tanFi** - $\tan(\varphi)$ defined by the System Operator to be imposed at the PCC for power injection at the current time step.<br>• Default: None (i.e., no restriction involving a constant $\tan(\varphi)$ shall be assumed.<br><br>Note: only considered for hybrid parks. | - |

### 2.2.2.4 Measurements structure – **bessMeasures unitary structure**

Structure designed to harbour the SoC measured for each individual battery unit plus the respective inverter.

Note: When operating the module at day-ahead planning mode, the SoC value introduced should be the expected SoC value at the beginning of the optimization horizon considered.

**Table 22-** Functional definition of the modules measured inputs structures: **bessMeasures unitary structure.**

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - Storage system's designation.<br>• e.g. "bess1", "bess2", … "bess10", … etc.<br><br>Note: this designation must coincide with the one given for the respective bessAssets unitary structure (section 2.2.1.8). | - |
| double (*) | **soc** - Measured SoC. | % |
| bool | **availability** - a True value means that the asset is available and not in maintenance.<br>• Default: True | - |

### 2.2.2.5 Measurements structure – **daBids unitary structure**

Structure designed to convey information about the energy bids made at day-ahead market by the hybrid park / microgrid and that will be confronted with the forecasted injection at the PCC.
The module will confront the information provided in this structure with the information conveyed at the milp structure (section 2.2.2.1) regarding the optimization horizon and step. Missing values or insufficient data provided will be considered as 0.0 and excess data (beyond the range defined by the horizon parameter) will be discarded.

Note: This structure will only be considered for objective functions directed at minimizing deviations.

**Table 23 -** Functional definition of the modules measured inputs structures: **daBids unitary structure.**

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **datetime** - Date and time to which the bid relates in ISO 8601 format.<br>• e.g. "2020-01-01T01:00:00Z" | - |
| double (*) | **daBid** - Day-ahead energy bid with hourly frequency. | MWh |

### 2.2.2.6 Measurements structure – **pvMeasures unitary structure**

Structure to store the observable generation of each solar panels' plant at the beginning of the optimization horizon.

**Table 24 -** Functional definition of the modules measured inputs structures: **pvMeasures unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - PV panels' pool designation.<br>• e.g. "pv1", "pv2", … "pv10", … etc.<br><br>Note: this designation must coincide with the one given for the respective pvPlants unitary structure (section 2.2.1.2). | - |
| double (*) | **pGen** - Current observable power generation. | MW |

### 2.2.2.7 Measurements structure – **windMeasures unitary structure**

Structure to store the observable generation of each wind turbines' plant at the beginning of the optimization horizon.

**Table 25 -** Functional definition of the modules measured inputs structures: **windMeasures unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - Wind turbines' pool designation.<br>• e.g. "wind1", "wind2", … "wind10", … etc.<br><br>Note: this designation must coincide with the one given for the respective windPlants unitary structure (section 2.2.1.3). | - |
| double (*) | **pGen** - Current observable power generation. | MW |

### 2.2.2.8 Measurements structure – **convMeasures unitary structure**

Structure to store the observable power limits that constitute the generation band of each conventional generation plant at the beginning of the optimization horizon.

**Table 26 -** Functional definition of the modules measured inputs structures: **convMeasures unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - Conventional generators' pool designation.<br>• e.g. "conv1", "conv2", … "conv10", … etc.<br><br>Note: this designation must coincide with the one given for the respective convGens unitary structure (section 2.2.1.4). | - |
| double (*) | **pMin** - Current minimum power output that can be asked from the generator(s). | MW |
| double (*) | **pMax** - Current maximum power output that can be asked of the generator(s). | MW |

### 2.2.2.9 Measurements structure – **inflexMeasures unitary structure**

Structure to store the observable inflexible load demand of each asset at the beginning of the optimization horizon.

**Table 27 -** Functional definition of the modules measured inputs structures: **inflexMeasures unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - Inflexible asset's designation.<br>• e.g. "inflex1", "inflex2", … "inflex10", … etc.<br><br>Note: this designation must coincide with the one given for the respective inflexAssets unitary structure (section 2.2.1.5). | - |
| double (*) | **pLoad** - Current observable inflexible load asset power demand. | MW |

### 2.2.2.10 Measurements structure – **curtailMeasures unitary structure**

Structure to store the observable curtailable load demand of each asset at the beginning of the optimization horizon.

**Table 28 -** Functional definition of the modules measured inputs structures: **curtailMeasures unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - Curtailable asset's designation.<br>• e.g. "curtail1", "curtail2", … "curtail10", … etc.<br><br>Note: this designation must coincide with the one given for the respective curtailAssets unitary structure (section 2.2.1.6). | - |
| double (*) | **pLoad** - Current observable curtailable load asset's power demand. | MW |
| int | **uncutTime** – For how long has the asset's load been provided without being curtailed.<br>• Default: 1E9 (meaning the asset has been provided for a sufficient amount of time that, if necessary, it can be curtailed in this time step) | minutes |
| int | **cutTime** – For how long has the asset's load been curtailed up to this point.<br>• Default: 0 (meaning the asset has not been curtailed in the last time step and, if necessary, can be curtailed in this time step) | minutes |
| bool | **availability** - a True value means that the asset is available for curtailment.<br>• Default: True | - |

### 2.2.2.11 Measurements structure – **evMeasures unitary structure**

Structure to store the observable demand of the EV chargers at the beginning of the optimization horizon.

**Table 29 -** Functional definition of the modules measured inputs structures: **evMeasures unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - EV charger's designation.<br>• e.g. "ev1", "ev2", … "ev10", … etc.<br><br>Note: this designation must coincide with the one given for the respective evAssets unitary structure (section 2.2.1.7). | - |
| double (*) | **pLoad** - Current observable EV charger power demand. | MW |
| int | **uncutTime** – For how long has the assets' load been provided without being curtailed.<br>• Default: 1E9 (meaning the asset has been provided for a sufficient amount of time that, if necessary, it can be curtailed in this time step) | minutes |
| int | **cutTime** – For how long has the assets' load been curtailed up to this point.<br>• Default: 0 (meaning the asset has not been curtailed in the last time step and, if necessary, can be curtailed in this time step) | minutes |
| bool | **availability** - a True value means that the assets are available for curtailment.<br>• Default: True | - |

Note that if uncutTime >0, cutTime must be 0 and vice-versa. If both parameters are provided as >0, default values will be considered instead.

### 2.2.2.12 Main forecasts structure – **forecasts**

Forecasted and scheduled data has dedicated structures that will be listed below. The array sizes of each structure are variable, depending on the optimization running mode and on the MPC mode. Address section 0 for further details.

As it happened for the daBids structure (section 2.2.2.5), the module will confront the information provided in this structure with the information conveyed at the milp structure (section 2.2.2.2) regarding the optimization horizon and step. Missing values or insufficient data provided will be considered as 0.0 and excess data (beyond the range defined by the horizon parameter) will be discarded.

This structure serves the purpose of being a placeholder for all forecasts' structures.

**Table 30 -** Functional definition of the modules main forecasted inputs structure: **forecasts**.

| Type | Identification - Description | Units |
|---|---|---|
| array (pvForecasts) (*) | **pvForecasts** - Array of pvForecasts unitary structures (section 2.2.2.13).<br><br>Array size: number of respective assets defined (section 2.2.1.2). | - |
| array (windForecasts) (*) | **windForecasts** - Array of windForecasts unitary structures (section 2.2.2.14).<br><br>Array size: number of respective assets defined (section 2.2.1.3). | - |
| array (convLimits) (*) | **convLimits** - Array of convLimits unitary structures (section 2.2.2.15).<br><br>Array size: number of respective assets defined (section 2.2.1.4). | |

| | | |
|---|---|---|
| array (inflexForecasts) (*) | **inflexForecasts** - Array of inflexForecasts unitary structures (section 2.2.2.16).<br>Array size: number of respective assets defined (section 2.2.1.5). | - |
| array (curtailForecasts) (*) | **curtailForecasts** - Array of curtailForecasts unitary structures (section 2.2.2.17).<br>Array size: number of respective assets defined (section 2.2.1.6). | - |
| array (evForecasts) (*) | **evForecasts** - Array of evForecasts unitary structures (section 2.2.2.18).<br>Array size: number of respective assets defined (section 2.2.1.7). | - |
| array (bessMaintenance) | **bessMaintenance** – Array of bessMaintenance unitary structures (section 2.2.2.23).<br>Array size: number of respective assets defined (section 2.2.1.8). | |
| tanFi | **tanFi** – Single tanFi structure (section 2.2.2.24).<br>Note: required when applicable (see section 2.2.2.24). | |
| devPricesPositive (*) | **devPricesPositive** - Single devPricesPositive structure (section 2.2.2.19).<br>Note: required when applicable (see section 2.2.2.19). | - |
| devPricesNegative (*) | **devPricesNegative** - Single devPricesNegative structure (section 2.2.2.20).<br>Note: required when applicable (see section 2.2.2.20). | - |
| feedinTariffs (*) | **feedinTariffs** – Single feedinTariffs structure (section 2.2.2.21).<br>Note: required when applicable (see section 2.2.2.21). | - |
| marketPrices (*) | **marketPrices** - Single marketPrices structure (section 2.2.2.22).<br>Note: required when applicable (see section 2.2.2.22). | - |

#### 2.2.2.12.1 Main forecasts substructure – **forecastsSingle**

Since all forecasts structures will rely on arrays of data, a common structure for each element of each array can be defined to unify and simplify all.

**Table 31 -** Functional definition of the modules main forecasted inputs substructure – **forecastsSingle**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **datetime** - Date and time to which the forecast relates in ISO 8601 format.<br>• e.g. "2020-01-01T01:00:00Z" | - |
| double (*) | **forecast** - Value of the forecast. | MW (for power limits)<br>or<br>MWh (for energy forecasts)<br>or<br>€/MWh (for price forecasts)<br>or<br>None (for tg(φ) set points) |

#### 2.2.2.12.2 Main forecasts substructure – **availabilitySingle**

This substructure is used for when Boolean states are to be conveyed for each time step.

**Table 32 -** Functional definition of the modules main forecasted inputs substructure – **availabilitySingle**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **datetime** - Date and time to which the forecast relates in ISO 8601 format.<br>• e.g. "2020-01-01T01:00:00Z" | - |
| bool (*) | **available** – If True, the asset or the asset's property is available for the corresponding time step.<br>• Default: True<br><br>Notes:<br>• for BESS maintenance, a True value means that the BESS asset is active and not in maintenance, being available for charge or discharge control signals; time steps where the asset is expected to be in maintenance must be signalled with False;<br>• for curtailable assets and EV chargers a True value means that the asset is available for curtailment | - |

### 2.2.2.13 Forecasts structure – **pvForecasts unitary structure**

Structure to store the forecasted generation of each solar panels' plant for the optimization horizon to be considered.

**Table 33 -** Functional definition of the modules forecasted inputs structures – **pvForecasts unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - PV panels' pool designation.<br>• e.g. "pv1", "pv2", … "pv10", … etc.<br><br>Note: this designation must coincide with the one given for the respective pvPlants unitary structure (section 2.2.1.2). | - |
| array (forecastsSingle) (*) | **forecasts** - Array of forecastsSingle structures (section 2.2.2.12.1).<br>Array size: Number of optimization time steps. | MWh |

### 2.2.2.14 Forecasts structure – **windForecasts unitary structure**

Structure to store the forecasted generation of each wind turbines' plant for the optimization horizon to be considered.

**Table 34 -** Functional definition of the modules forecasted inputs structures –**windForecasts unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - Wind turbines' pool designation.<br>• e.g. "wind1", "wind2", … "wind10", … etc.<br><br>Note: this designation must coincide with the one given for the respective windPlants unitary structure (section 2.2.1.3). | - |
| array (forecastsSingle) (*) | **forecasts** - Array of forecastsSingle structures (section 2.2.2.12.1).<br>Array size: Number of optimization time steps. | MWh |

### 2.2.2.15 Forecasts structure – **convLimits unitary structure**

Structure to store the scheduled power limits that constitute the generation band of each conventional generation plant at the beginning of the optimization horizon.

**Table 35 -** Functional definition of the modules forecasted inputs structures –**convLimits unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - Conventional generators' pool designation.<br>• e.g. "conv1", "conv2", … "conv10", … etc.<br><br>Note: this designation must coincide with the one given for the respective convGens unitary structure (section 2.2.1.4). | - |
| array (forecastsSingle) (*) | **pMin** - Array of forecastsSingle structures (section 2.2.2.12.1).<br>Array size: Number of optimization time steps. | MWh |
| array (forecastsSingle) (*) | **pMax** - Array of forecastsSingle structures (section 2.2.2.12.1).<br>Array size: Number of optimization time steps. | MWh |

### 2.2.2.16 Forecasts structure – **inflexForecasts unitary structure**

Structure to convey the **total** forecasted inflexible load demand for the optimization horizon.

Note: This structure will only be considered for microgrid systems.

**Table 36 -** Functional definition of the modules forecasted inputs structures –**inflexForecasts unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - Inflexible asset's designation.<br>• e.g. "inflex1", "inflex2", … "inflex10", … etc.<br><br>Note: this designation must coincide with the one given for the respective inflexAssets unitary structure (section 2.2.1.5). | - |
| array (forecastsSingle) (*) | **forecasts** - Array of forecastsSingle structures (section 2.2.2.12.1).<br>Array size: Number of optimization time steps. | MWh |

### 2.2.2.17 Forecasts structure – **curtailForecasts unitary structure**

Structure to convey the forecasted demand and the availability for curtailment of each curtailable load asset for the optimization horizon.

Note: This structure will only be considered for microgrid systems.

**Table 37 -** Functional definition of the modules forecasted inputs structures -**curtailForecasts unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - Curtailable asset's designation.<br>• e.g. "curtail1", "curtail2", … "curtail10", … etc.<br><br>Note: this designation must coincide with the one given for the respective curtailAssets unitary structure (section 2.2.1.6). | - |
| array (forecastsSingle) (*) | **forecasts** - Array of forecastsSingle structures (section 2.2.2.12.1).<br>Array size: Number of optimization time steps. | MWh |
| array (availabilitySingle) | **availability** - Array of availabilitySingle structures (section 2.2.2.12.2).<br>Array size: Number of optimization time steps. | - |

### 2.2.2.18 Forecasts structure – **evForecasts unitary structure**

Structure to convey the forecasted demand and the availability for curtailment of each EV charger for the optimization horizon.

Note: This structure will only be considered for microgrid systems.

**Table 38 -** Functional definition of the modules forecasted inputs structures -**evForecasts unitary structure**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - EV charger's designation.<br>• e.g. "ev1", "ev2", … "ev10", … etc.<br><br>Note: this designation must coincide with the one given for the respective evAssets unitary structure (section 2.2.1.7). | - |
| array (forecastsSingle) (*) | **forecasts** - Array of forecastsSingle structures (section 2.2.2.12.1).<br>Array size: Number of optimization time steps. | MWh |
| array (availabilitySingle) | **availability** - Array of availabilitySingle structures (section 2.2.2.12.2).<br>Array size: Number of optimization time steps. | - |

### 2.2.2.19 Forecasts structure – **devPricesPositive**

Structure to convey the forecasted positive imbalance costs for the selected optimization horizon.

Note: This structure will only be considered for objective functions directed at minimizing deviations.

**Table 39 -** Functional definition of the modules forecasted inputs structures -**devPricesPositive**.

| Type | Identification - Description | Units |
|---|---|---|
| array (forecastsSingle) (*) | Array of forecastsSingle structures (section 2.2.2.12.1).<br>Array size: Number of optimization time steps. | €/MWh |

### 2.2.2.20 Forecasts structure – **devPricesNegative**

Structure to convey the forecasted negative imbalance costs for the selected optimization horizon.

Note: This structure will only be considered for objective functions directed at minimizing deviations.

**Table 40 -** Functional definition of the modules forecasted inputs structures -**devPricesNegative**.

| Type | Identification - Description | Units |
|---|---|---|
| array (forecastsSingle) (*) | Array of forecastsSingle structures (section 2.2.2.12.1).<br>Array size: Number of optimization time steps. | €/MWh |

### 2.2.2.21 Forecasts structure – **feedinTariffs**

Structure to convey the forecasted feed-in tariffs for the selected optimization horizon.

Note: This structure will only be considered for objective functions directed at arbitrage.

**Table 41 -** Functional definition of the modules forecasted inputs structures – **feedinTariffs**.

| Type | Identification - Description | Units |
|---|---|---|
| array (forecastsSingle) (*) | Array of forecastsSingle structures (section 2.2.2.12.1).<br>Array size: Number of optimization time steps. | €/MWh |

### 2.2.2.22 Forecasts structure – **marketPrices**

Structure to convey the forecasted market electricity prices for the selected optimization horizon.

Note: This structure will only be considered for objective functions directed at arbitrage or minimization of operation costs / maximization of self-consumption.

**Table 42 -** Functional definition of the modules forecasted inputs structures – **marketPrices**.

| Type | Identification - Description | Units |
|---|---|---|
| array (forecastsSingle) (*) | Array of forecastsSingle structures (section 2.2.2.12.1).<br>Array size: Number of optimization time steps. | €/MWh |

### 2.2.2.23 Forecasts structure – **bessMaintenance**

Structure to convey scheduled maintenances to the bess assets, within the optimization horizon, during which, the algorithm must consider them to be unavailable for controlling.

**Table 43 -** Functional definition of the modules forecasted inputs structures – **bessMaintenance**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **designation** - Storage system's designation.<br>• e.g. "bess1", "bess2", … "bess10", … etc.<br><br>Note: this designation must coincide with the one given for the respective bessAssets unitary structure (section 2.2.1.8). | - |
| array (availabilitySingle) (*) | Array of availabilitySingle structures (section 2.2.2.12.1).Array size: Number of optimization time steps. | - |

#### 2.2.2.24 Forecasts structure – **tanFi**

Structure to convey the tan(φ) defined by the System Operator to be imposed at the PCC for power injection for each time step.
This structure will only be taken into account when a hybrid park has been defined as the operating system and, if not provided, a constant tan(φ) restriction will not be imposed. For example, in Portugal, for installed capacities ≤ 6MW, at periods of increased consumption, the tg(φ) must be 0.3 either for absorption as for injection. In that case, the array should be filled with 0.3 for those periods and with 0 for the rest.

**Table 44 -** Functional definition of the modules forecasted inputs structures – **tanFi**.

| Type | Identification - Description | Units |
|---|---|---|
| array (availabilitySingle) (*) | Array of forecastSingle structures (section 2.2.2.12.1)<br>Note: only considered for hybrid parks. | - |

## 2.3 Outputs

The outputs provided by the API modules will be sent via REST interface to a server on the client´s side. The respective data structure will depend essentially on the system structure considered and on the objective function selected.

### 2.3.1 Main outputs structure - **outputs**

**Table 45** - Functional definition of the modules main outputs structure – **outputs**.

| Type | Identification - Description | Units |
|---|---|---|
| string (*) | **requestID** – Identification of the request. Matches the identification of the client's request to the optimization method that originated the outputs.<br>e.g. MD5 HASH of millisecond "c06c37631c60c6d067345d670909a328" | - |
| integer (*) | **milpStatus** – Status of the solution achieved by the optimization algorithm.<br>• Available options:<br>• 1: Optimal<br>• 0: Not Solved<br>• -1: Infeasible<br>• -2: Unbounded<br>• -3: Undefined | - |
| string (*) | **systemID** – Designation of the system.<br>e.g. "hybridpark1", "microgrid2", … etc. | - |
| array (pvPlants) | **pvPlants** - Array of pvPlants unitary structures (section 2.3.2).<br>Array size: number of respective assets defined (section 2.2.1.2). | - |
| array (windPlants) | **windPlants** – Array of windPlants unitary structures (section 2.3.2).<br>Array size: number of respective assets defined (section 2.2.1.3). | - |
| array (convGens) | **convGens** - Array of convGens unitary structures (section 2.3.2).<br>Array size: number of respective assets defined (section 2.2.1.4). | |
| array (curtailAssets) | **curtailAssets** - Array of curtailAssets unitary structures (section 2.3.2).<br>Array size: number of respective assets defined (section 2.2.1.6). | - |
| array (evAssets) | **evAssets** - Array of evAssets unitary structures (section 2.3.2).<br>Array size: number of respective assets defined (section 2.2.1.7). | - |
| array (bessAssets) | **bessAssets** - Array of bessAssets unitary structures (section).<br>Array size: number of respective assets defined (section 2.2.1.8). | - |
| array (expectedRevenues) | **expectedRevenues** – Array of expectedRevenues unitary structures (section 2.3.4).<br>Array size: number of time steps (see Section 0) | - |

### 2.3.2 Outputs structures – **pvPlants, windPlants, convGens, curtailAssets, evAssets**

General structure for providing the set points for all controllable assets, excluding BESS. Regarding renewable generation assets (PV and wind plants) the set points convey the power curtailment for each time step and for conventional generators, the output power set points, that are to be scaled afterwards by a centralized controller (ES Manager). Regarding the controllable load assets (curtailable assets and electric vehicle chargers), the set points convey the power curtailment per time step.

**Table 46 -** Functional definition of the modules outputs structures – **pvPlants, windPlants, curtailAssets, evAssets**.

| Type | Identification - Description | Units |
|---|---|---|
| string | **designation** – Asset's designation.<br>• e.g. "pv1", "ev2", "curtail3", etc.<br><br>Note: this designation must coincide with the one given for the respective assets' unitary configuration structure (sections 2.2.1.2, 2.2.1.3, 2.2.1.6 and 2.2.1.7**Error! Reference source not found.**). | - |
| array (generalSetPoints) | **generalSetPoints** - Array of generalSetPoints structures (section 2.3.2.1) regarding the expected average power curtailed for the time step.<br><br>Array size: number of time steps (see Section 0) | MW |

### 2.3.2.1 Outputs substructure – **generalSetPoints**

Since some of the output structures will rely on arrays of data, a common structure for each element of each array can be defined to unify and simplify all. Each generalSetPoints structure will refer to a setpoint to be implemented/ to be expected at the respective time step.

**Table 47 -** Functional definition of the modules outputs substructure – **generalSetPoints**.

| Type | Identification - Description | Units |
|---|---|---|
| string | **datetime** - Date and time to which the set point refers in ISO 8601 format.<br>• e.g. "2020-01-01T01:00:00Z" | - |
| double | **setPoint** - Set point for the correspondent time step in MW. | MW |

## 2.3.3 Outputs structure – **bessAssets**

General structure for providing the set points for all BESS assets.

**Table 48 -** Functional definition of the modules outputs structure – **bessAssets**.

| Type | Identification - Description | Units |
|---|---|---|
| string | **designation** – Asset's designation.<br>• e.g. "bess1"<br><br>Note: this designation must coincide with the one given for the respective assets' unitary configuration structure (section 2.2.1.8). | - |
| array (bessSetPoints) | **bessSetPoints** - Array of bessSetPoints structures (section 2.3.3.12.3.2.1).<br><br>Array size: number of time steps (see Section 0) | MW |

#### 2.3.3.1 Outputs substructure – **bessSetPoints**

**Table 49 -** Functional definition of the modules outputs substructure – **bessSetPoints**.

| Type | Identification - Description | Units |
|---|---|---|
| string | **datetime** - Date and time to which the set point refers in ISO 8601 format.<br>• e.g. "2020-01-01T01:00:00Z" | - |
| double | **pCharge** – Active charge power set point for the correspondent time step in MW. | MW |
| double | **pDischarge** – Active discharge power set point for the correspondent time step in MW. | MW |
| double | **qDischarge - Reactive discharge power set point for the correspondent time step in Mvar.** | Mvar |
| double | **soc** - Expected SoC for the correspondent time step in MWh. | MWh |
| double | **degradation** - Expected energy capacity degradation for the correspondent time step in Wh. | Wh |

### 2.3.4 Outputs structure – **expectedRevenues**

Structure providing the expected revenue for each time step. Total expected revenue can be obtained by summing all steps' values.

**Table 50 -** Functional definition of the modules outputs structure – **expectedRevenues**.

| Type | Identification - Description | Units |
|---|---|---|
| string | **datetime** - Date and time to which the set point refers in ISO 8601 format.<br>e.g. "2020-01-01T01:00:00Z" | - |
| double | **setPoint** - Expected revenue for the correspondent time step in €. If negative, consider as a cost. | € |

## 2.4 API main methods

This section will list the available functions of the optimization module. Three main methods are defined for the Flexergy Optimizer API. The system and asset methods, where all system settings and assets can be configured and updated and the optimize method, where the real time operation and day-ahead planning method are available.

### 2.4.1 System configuration methods - system

Before running the optimization modes, the system must be configured at first use or reconfigured if any relevant changes were observed such the ON/OFF state of an asset. The following subsections will describe the two endpoints for both interactions.

#### 2.4.1.1 Configure system settings endpoint: **/api/system/ (POST)**

The first interaction with the optimization module must be through the **/api/system/** POST HTTP method, where the settings structure (section 2.2.1.1) is provided and validated.

#### 2.4.1.2 Update system settings endpoint: **/api/system/ (PUT)**

At any moment, the settings can be updated with relevant changes in the system such as the change in the PCC limit value. Those changes will need to be reflected at the next iteration of the optimization algorithm. The PUT method applied to endpoint **/api/system/** is intended for updating one or more parameters of the settings structure (section 2.2.1.1).

If the validation tests of the input structure fail, the module will simply not reflect the changes to the original settings configuration and retain only the previous information.

#### 2.4.1.3 Configure system assets endpoint: **/api/asset/ (GET)**

Method for retrieving the complete set of assets configured for all system IDs.

#### 2.4.1.4 Configure system assets endpoint: **/api/asset/{system_id} (GET)**

Method for retrieving the complete set of assets configured for the specified system ID.

#### 2.4.1.5 Configure system assets endpoint: **/api/asset/{asset_type}/ (POST)**

Following the configuration of the system settings, the system assets must be characterized. This POST HTTP method is intended for configuring any specified asset type structure (sections **Error! Reference source not found.** to 2.2.1.8). A validation step is also naturally included in this method. {**asset_type**} can be one of the following:

- ✓ pv_plant
- ✓ wind_plant
- ✓ conv_gen
- ✓ inflex

- ✓ curtail
- ✓ ev
- ✓ bess

each corresponding to each of the available asset types.

#### 2.4.1.6 Update system assets endpoint: **/api/asset/{asset_type}/{system_id}/{designation} (PUT)**

As for the system settings, updates to assets should be communicated through the corresponding structure, superimposing the previously saved information in the module. For example, if a PV panel or string is lost, the corresponding technology structure, pvPlant (section 2.2.1.2), should be updated indicating the new maximum total capacity and/or status. The univocal key {system_id}/{designation} is used to pinpoint the exact asset to be reconfigured.

For structures that are designed for individual equipment and assets, a parameter "status" is provided in order to indicate if the structure should be considered in "ON" or in "OFF" state. Therefore, and for example, if for some reason a battery was not removed from the system but has been switched off, the operator should simply change its status in the correspondent structure (2.2.1.82.2.1.8) to "OFF", avoiding a new configuration of the same asset when it is turned on again.

If the validation tests of the input structure fail, the module will simply not reflect the changes to the original assets configuration and retain only the previous information.

Once again, {**asset_type}** can be one of the following:

- ✓ pv_plant
- ✓ wind_plant
- ✓ conv_gen
- ✓ inflex
- ✓ curtail
- ✓ ev
- ✓ bess

each corresponding to each of the available asset types.

#### **2.4.1.7** Delete system assets endpoint: /**api/asset/{asset_type}/{system_id}/{designation} (DELETE)**

Method to delete assets that were permanently removed from the system and are not expected to be reintegrated in the future. This method permanently deletes an asset from the system's configuration so, if the asset is expected to be reinstalled, it is advised to simply set its status to "OFF", avoiding a reconfiguration of the asset in the future.

The univocal key {system_id}/{designation} is used to pinpoint the exact asset to be reconfigured.

Once again, {**asset_type}** can be one of the following:

- ✓ pv_plant
- ✓ wind_plant
- ✓ conv_gen
- ✓ inflex
- ✓ curtail

- ✓ ev
- ✓ bess

each corresponding to each of the available asset types.

### 2.4.2 System optimization methods - optimize

The optimization function represents the core of the module being responsible for formulating and solving the problem by calling the solver. The interaction with this function depends on the running mode selected. Apart from choosing and providing the necessary initial configurations (see section 2.4.1) and optional MILP parameters (section 2.2.2), each mode will require the provision of measurements (section 2.2.2.3) and forecasts (section 2.2.2.12) that will be enumerated in the following subsections describing the two endpoints regarding the real-time operation and day-ahead planning methods.

#### 2.4.2.1 Day-ahead planning endpoint: /api/**optimize/dayAheadOpt (POST)**

The dayAheadOpt method is intended for up to 24 hours ahead planning or as a study mode. The optimization function will run a single MILP requiring the day-ahead forecasts for the selected horizon and for the desired objective function, providing for each time step the set points for each asset. Table 51 specifies the required structures for running each optimization function in this mode.

**Table 51** - Definition of structures to be filled when running the optimization function in day-ahead planning mode

| System Structure | Objective Function | Measurements Structures | Forecasts Structures |
|---|---|---|---|
| Hybrid Park | Arbitrage | bessMeasures (2.2.2.4); | pvForecasts (2.2.2.13);<br>windForecasts (2.2.2.14);<br>convLimits (2.2.2.15);<br>feedinTariffs (2.2.2.21);<br>marketPrices (2.2.2.22) |
| Microgrid | Arbitrage | bessMeasures (2.2.2.4); | pvForecasts (2.2.2.13);<br>windForecasts (2.2.2.14);<br>convLimits (2.2.2.15);<br>inflexForecasts (2.2.2.16);<br>curtailForecasts (2.2.2.17);<br>evForecasts(2.2.2.18);<br>feedinTariffs (2.2.2.21);<br>marketPrices (2.2.2.22) |
| Microgrid | Minimization of operation costs | bessMeasures (2.2.2.4); | pvForecasts (2.2.2.13);<br>windForecasts (2.2.2.14);<br>convLimits (2.2.2.15);<br>inflexForecasts (2.2.2.16);<br>curtailForecasts (2.2.2.17);<br>evForecasts(2.2.2.18);<br>marketPrices (2.2.2.22) |

The forecasts' (and day-ahead bids') structures shall be provided as arrays with no strict size limitation other than having a valid, unique timestamp associated to each forecasted value. Each forecasted value represents a power setpoint. It is assumed that each setpoint provided represents the average power/energy observed for the respective step's duration, i.e., the module assumes the same power setpoint for the whole step.

During the validation step of the data provided, the module will test the array size of each forecasts structure and shall raise an internal warning flag if it doesn't coincide with the value of horizon (in hours) divided by the time step (also in hours).

At the beginning of the optimization procedure, being the model prepared for temporally equidistant time steps, the numerical data is always resampled according to the time step provided at the milp structure (section 2.2.2.2). Since the power.is assumed to not change during a time step, the power setpoint will characterize the resampled time step where it befalls. If, after resampling, more than one value befalls the same time step, the numerical average of the values will be taken (an example is given in Figure 5), exceptions made for generation power limits of the conventional generators (section 2.2.2.15) and the $\tan(\varphi)$ at the PCC (section 2.2.2.24). In the first case, if after resampling, different values for the maximum and minimum output power are provided, the module will assume the ones that produce the more restrictive band (i.e. the smallest maximum power value and the highest minimum power value). Conflicting values of $\tan(\varphi)$ at the PCC for a same time step will be considered as not provided.

For the sake of flexibility, the module shall admit as few as a single forecast and shall simply discard excessive information (i.e., after sorting by date and time, if forecasts beyond the time horizon are provided, they shall not be considered). The only requirement is that the forecast has a valid datetime value within the optimization horizon, starting from the defined datetime value at the milp structure's parameter "init" (see 2.2.2.2).

When data is missing for a time step, the module will apply a linear interpolation between the available data surrounding that time step or broadcast the value of the last available data point until the last step in the optimization horizon is reached. The only exception is made for missing data regarding the first time step where the broadcasting shall be made in reverse fashion (i.e., the initial steps with missing data shall assume the same value as the first available forecast).

Once again, exceptions are made for generation power limits of the conventional generators (section 2.2.2.15), day-ahead bids (section 2.2.2.5), prices/tariffs (sections 2.2.2.19-2.2.2.22) and the $\tan(\varphi)$ at the PCC (section 2.2.2.24). The conventional generators' power limits shall only characterize the time step where they befall while for the remaining steps, for which no power band has been provided, the generators will be considered as unavailable. Regarding the $\tan(\varphi)$ at the PCC, they will only be imposed for the time steps where they were explicitly defined. The filling of missing data for prices and tariffs follows a different rationale. Instead of applying a linear interpolation, the missing values will be forward filled with the last available set point, followed by a backwards filling (only effective if the first time step(s) are missing data).

If measurements are available for the configured generation pools and load assets (section 2.2.2.3) these will substitute any respective forecast provided for the first time step, prior to any other operation. Figure 6 clarifies the method for dealing with missing excessive data. It is extremely important to notice, although, that the quality of the optimization schedules is closely dependent of the provision of the most complete and well-dated forecasts possible.

The module will also evaluate cases where the granularity of the forecast step doesn't match the optimization step's. If the first is lesser than the latter's step, the same forecasted data will be broadcasted for optimization steps sharing the same temporal slot (Figure 7). Conversely, if the granularity of the forecast step is higher, an average of the values with a timestamp within the corresponding optimization step will be considered (Figure 8).

As outputs, the function will provide an operational plan for the select optimization horizon, for example, the next day. The list of outputs that are expected to be filled in the main output structure (2.3.1) can be consulted at Table 52.

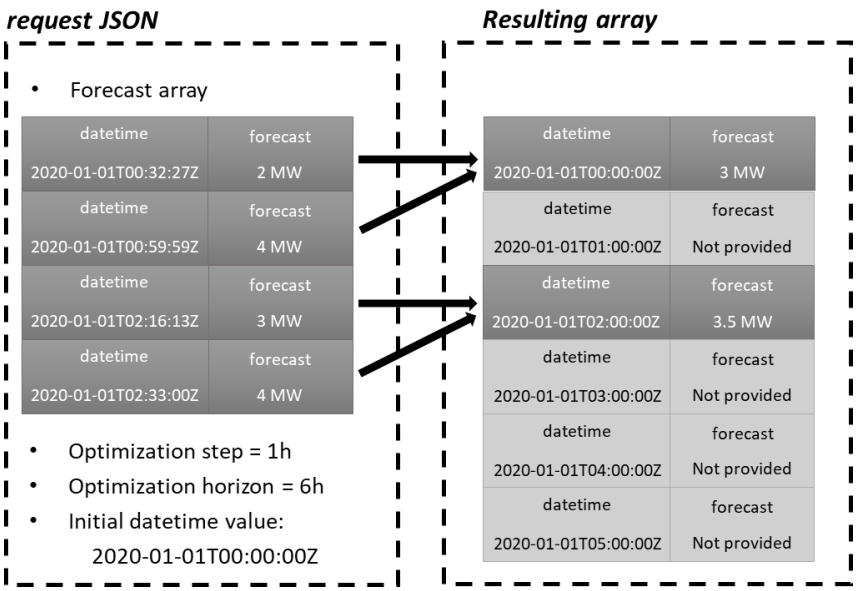



**Figure 5** – Resampling strategy for non-temporally equidistant forecasts. Notice that for forecasts befalling the same time step in the resulting array, the forecast value stored is calculated as the average of those forecasts.

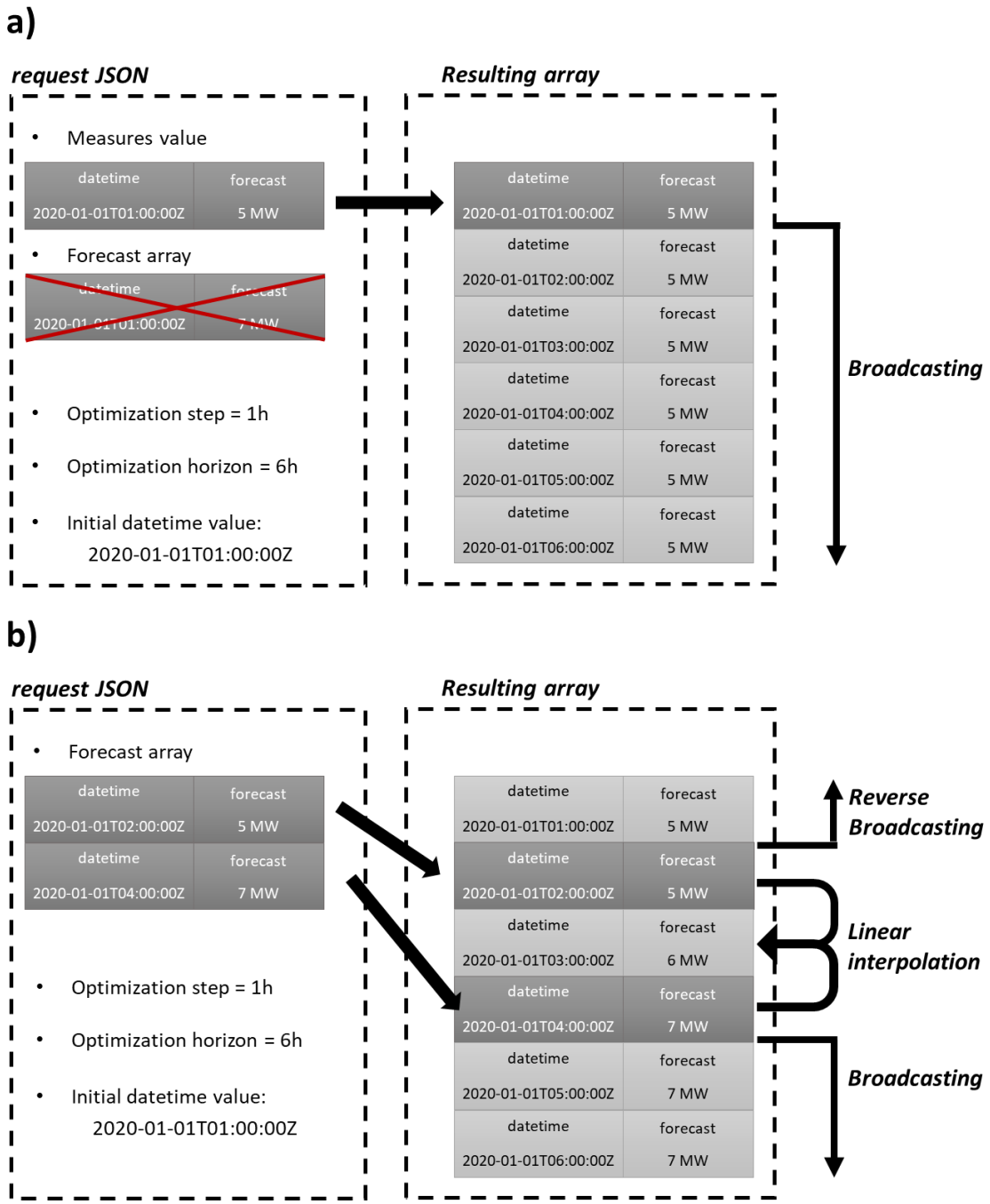



**Figure 6** – Framework for dealing with missing data in forecasts (and day-ahead bids) data.

After sorting the data provided in the forecasts structure (section 2.2.2.12), the module will confront the provided set points with the optimization step and horizon provided at the milp parameters (section 2.2.2). The first step shall be provided by the user through the parameter "init" (section 2.2.2.2). After that, and for the example provided, a total of 5 more steps need to be characterized. In case a), the module simply broadcasts the same value provided for the first time step until all 6 steps are filled. Notice that, when a measurement is available (section 2.2.2.3), it takes precedence over the forecast for the first time step. In case b), since two forecasts were provided, the set points that should have been provided between them are filled by linear interpolation. Notice that, for the example above, if a set point with a timestamp of "2020-01-01T07:00:00Z" is provided it will be discarded since it exceeds 6 hours (the desired horizon), starting at the first data point provided. Since no data has been provided for the defined initial time step, it is filled with by reverse broadcasting.

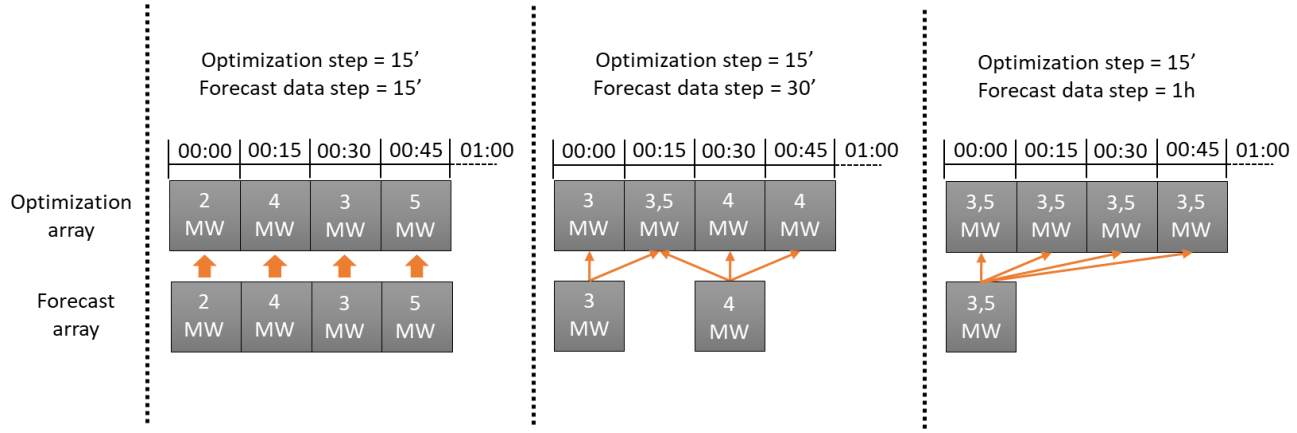



**Figure 7** - Visual representation of the linear interpolation and broadcasting procedure applied to a hypothetical PV generation forecast data array in order to form the corresponding optimization array, for different forecast data steps.

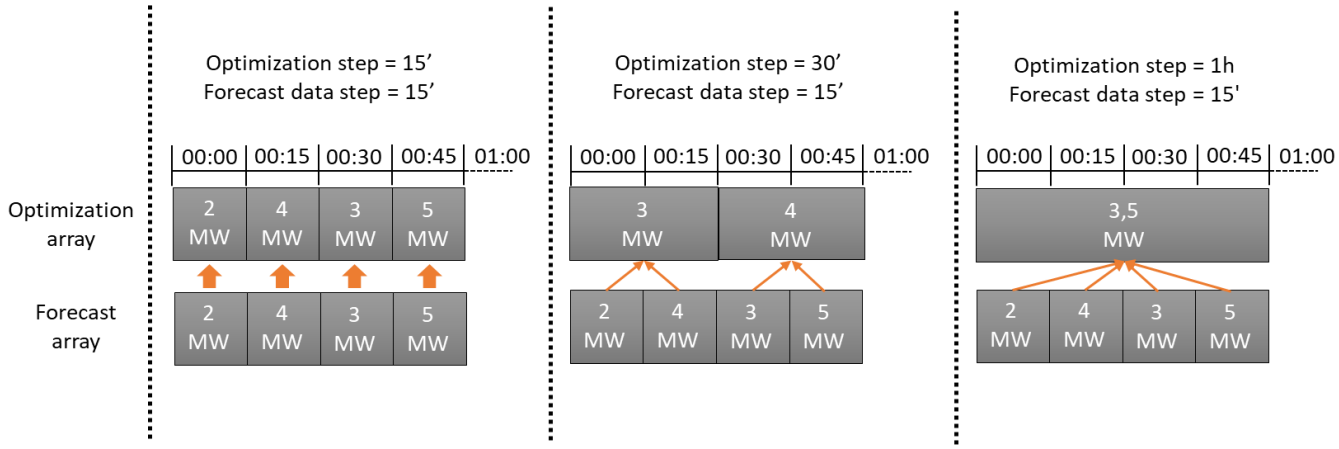



**Figure 8** - Visual representation of the averaging procedure applied to a hypothetical PV generation forecast data array in order to form the corresponding optimization array, for different forecast data steps.

Regarding the availability of the load assets for curtailment (traduced by the availability parameter in sections 2.2.2.17 and 2.2.2.18) and maintenance schedules for the BESS assets (traduce In the bessMaintenance forecast structure at section 2.2.2.23), since the conveyed data is not numerical but Boolean, the broadcasting process needs to be distinct.

A resampling takes place as explained for all other numerical forecast arrays, but in this case, empty time steps will be considered by default as True, i.e., the load assets are available for curtailment and the BESS is operational and not in maintenance. Time steps with conflicting values will always give priority to False values. This means that all Boolean values befalling the same time step, after resampling, are subjected to a logical AND operation. An exception is naturally made for the first time step when running the module in realTimeOpt mode if a corresponding measures structure was provided. In that case the value conveyed in the measures structure (sections 2.2.2.10 for curtailable load assets, 2.2.2.11 for EV chargers and 2.2.2.4 for BESS assets) takes precedence over the forecasted value(s).

Also regarding all curtailable load assets and EV chargers, the parameters “minTCycle” and “maxTCut” (sections 2.2.1.6 and 2.2.1.7) will always be adjusted regarding the step provided for each run. For example, if a step of 15 minutes is defined, both parameters will be fitted into 15 minute blocks, with the “minTCycle” being rounded

by excess and the “maxTCut” by default. This means that if, for example, both parameters are defined as 46 minutes, “minTCycle” will be converted into four 15 minute blocks and “maxTCut” into three blocks only.

**Table 52** – Output structure content of the day-ahead planning mode

| Structure | Description |
|---|---|
| requestID | The ID of the matching request made for this solution by the client through the optimize/dayAheadOpt method. |
| milpStatus | The status of the solution found by the optimization algorithm. |
| pvPlants | Operational plan with the power curtailment set points per asset and per time step for the whole optimization horizon.<br>Number of time steps per asset: optimization horizon / optimization time step<br>Example: Optimization horizon = 24h, time step = 1/4 h: array size = 96 |
| windPlants | Operational plan with the power curtailment set points per asset and per time step for the whole optimization horizon.<br>Number of time steps per asset: optimization horizon / optimization time step |
| convGens | Operational plan with the power set points per power level and per time step for the whole optimization horizon.<br>Number of time steps per asset: optimization horizon / optimization time step |
| curtailSetpoints<br>NOTE: only provided for microgrid systems | Operational plan with the power curtailment set points per asset and per time step for the whole optimization horizon.<br>Number of time steps per asset: optimization horizon / optimization time step |
| evSetpoints<br>NOTE: only provided for microgrid systems | Operational plan with the power curtailment set points per asset and per time step for the whole optimization horizon.<br>Number of time steps per asset: optimization horizon / optimization time step |
| bessSetpoints | Operation plan per BES unit, with the operational plan for the optimization horizon considered (power charge and discharge set points), the respective energy content at the end of the time step and the expected degradation of the total energy content suffered at each time step of the optimization horizon.<br>Number of time steps per asset: optimization horizon / optimization time step |
| expectedRevenues | Expected revenue per time step in € of the total system for the entire optimization horizon. If negative, consider as a cost.<br>Number of time steps: optimization horizon / optimization time step |

#### 2.4.2.2 Real-time operation endpoint: **optimize/realTimeOpt (POST)**

The realTimeOpt method is intended for real-time update of the assets set points. The user shall call the function at every 5-15 minutes where a MILP will be run following an MPC approach, i.e. it only provides the assets' set points for the next time step. At each iteration, new measurements and forecasts must be provided, for the considered time horizon (see Table 53 for details). Table 54 specifies the required structures for running each optimization function in this mode.

**Table 53** – Framework for calling the optimization function in the realTimeOpt method.

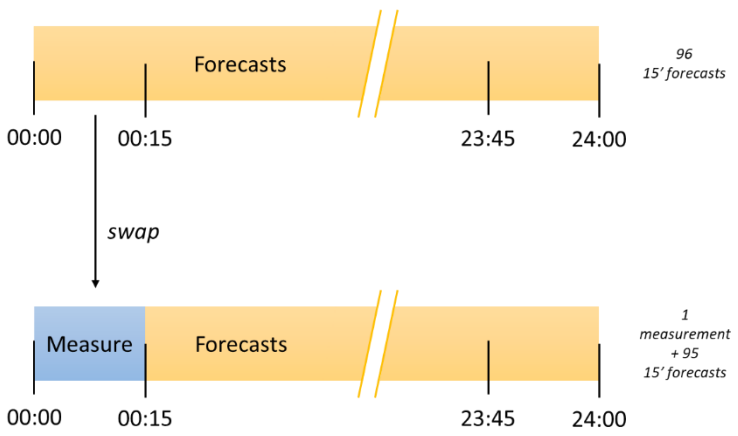
| MPC mode | Instructions |
|---|---|
| Sliding horizon | At each iteration of the MPC (i.e. at every 5 – 15 minutes), all the applicable measurements' and forecasts' structures must be provided;<br><br>The forecasts' structures array size is **fixed**:<br><br>• Iteration 1: Array size = optimization horizon / time step<br>• Iteration 2: Array size = optimization horizon / time step<br>• (…)<br>• Iteration N: Array size = optimization horizon / time step<br>• (…)<br>• Final iteration: Array size = optimization horizon / time step<br><br>Example: For an optimization horizon of 24h and a time step of 15 minutes (0.25 hours) starting at 00:00 of a certain day:<br><br>• Iteration 1 (at 00:00): Forecasts array size = 96 (spanning from 00:00 to 24:00)<br>• Iteration 2 (at 00:15): Forecasts array size = 96 (spanning from 00:15 to 24:15)<br>• (…)<br>• Final iteration (at 23:45): Forecasts array size = 96 (spanning from 23:45 to 47:45)<br><br>Note: The first time step will not consider forecasts but real-time measurements. Nonetheless a forecast for the first time step must be provided in each structure which can be filled with 0.0:<br><br>  |

**Table 54 -** Definition of structures to be filled when running the optimization function in MPC Mode.

| System Structure | Objective Function | Measurements Structures | Forecasts Structures |
|---|---|---|---|
| Hybrid Park | Minimization of deviations costs | bessMeasures (2.2.2.4);<br>daBids (2.2.2.5);<br>pvMeasures (2.2.2.6);<br>windMeasures(2.2.2.7);<br>convMeasures (2.2.2.8) | pvForecasts (2.2.2.13);<br>windForecasts (2.2.2.14);<br>convLimits (2.2.2.15);<br>devPricesPositive (2.2.2.19);<br>devPricesNegative (2.2.2.20) |
| Hybrid Park | Minimization of energy deviations | bessMeasures (2.2.2.4);<br>daBids (2.2.2.5);<br>pvMeasures (2.2.2.6);<br>windMeasures(2.2.2.7);<br>convMeasures (2.2.2.8) | pvForecasts (2.2.2.13);<br>windForecasts (2.2.2.14);<br>convLimits (2.2.2.15) |
| Hybrid Park | Arbitrage | bessMeasures (2.2.2.4);<br>pvMeasures (2.2.2.6);<br>windMeasures(2.2.2.7);<br>convMeasures (2.2.2.8) | pvForecasts (2.2.2.13);<br>windForecasts (2.2.2.14);<br>convLimits (2.2.2.15);<br>feedinTariffs (2.2.2.21);<br>marketPrices (2.2.2.22) |
| Microgrid | Minimization of energy deviations | bessMeasures (2.2.2.4);<br>daBids (2.2.2.5);<br>pvMeasures (2.2.2.6);<br>windMeasures(2.2.2.7);<br>convMeasures (2.2.2.8);<br>inflexMeasures (2.2.2.9);<br>curtailMeasures (2.2.2.10):<br>evMeasures (2.2.2.11) | pvForecasts (2.2.2.13);<br>windForecasts (2.2.2.14);<br>convLimits (2.2.2.15);<br>inflexForecasts (2.2.2.16);<br>curtailForecasts (2.2.2.17);<br>evForecasts(2.2.2.18) |
| Microgrid | Arbitrage | bessMeasures (2.2.2.4);<br>pvMeasures (2.2.2.6);<br>windMeasures(2.2.2.7);<br>convMeasures (2.2.2.8);<br>inflexMeasures (2.2.2.9);<br>curtailMeasures (2.2.2.10):<br>evMeasures (2.2.2.11) | pvForecasts (2.2.2.13);<br>windForecasts (2.2.2.14);<br>convLimits (2.2.2.15);<br>inflexForecasts (2.2.2.16);<br>curtailForecasts (2.2.2.17);<br>evForecasts(2.2.2.18);<br>feedinTariffs (2.2.2.21);<br>marketPrices (2.2.2.22) |
| Microgrid | Minimization of operation costs | bessMeasures (2.2.2.4);<br>pvMeasures (2.2.2.6);<br>windMeasures(2.2.2.7);<br>convMeasures (2.2.2.8);<br>inflexMeasures (2.2.2.9);<br>curtailMeasures (2.2.2.10):<br>evMeasures (2.2.2.11) | pvForecasts (2.2.2.13);<br>windForecasts (2.2.2.14);<br>convLimits (2.2.2.15);<br>inflexForecasts (2.2.2.16);<br>curtailForecasts (2.2.2.17);<br>evForecasts(2.2.2.18);<br>marketPrices (2.2.2.22) |

Without detriment of the rules expressed at Table 53, the same logic presented at section 2.4.2.1. is applied for the MPC mode, in case the granularity of the forecasted data array does not match the granularity imposed by the optimization step.

The output structures will not provide operational plans as in the dayAheadOpt mode, but individual set points for the next time step within the MPC framework. The suite of output structures can be consulted at Table 55.

**Table 55 -** Output structures of the real-time operation mode.

| Structure | Description |
|---|---|
| requestID | The ID of the matching request made for this solution by the client through the optimize/realTimeOpt method. |
| milpStatus | The status of the solution found by the optimization algorithm. |
| pvPlants | Power curtailment set point per asset for the next time step. |
| windPlants | Power curtailment set point per asset for the next time step. |
| convGens | Power generation set point per power level for the next time step. |
| curtailSetpoints<br>NOTE: only provided for microgrid systems | Power curtailment set point per asset for the next time step. |
| evSetpoints<br>NOTE: only provided for microgrid systems | Power curtailment set point per asset for the next time step. |
| bessSetpoints | Power charge and discharge set point per BESS asset for the next time step, and the respective energy content and energy content degradation at the end of the time step. |
| expectedRevenues | Expected revenue in € of the total system for the next time step. If negative, consider as a cost. |

All the resampling strategies presented for the dayAheadOpt endpoint are also applied in the realTimeOpt endpoint.

#### 2.4.2.3 Additional functions

##### 2.4.2.3.1 Post-operational analysis function

A post-operational analysis function is provided in order to confront the estimations considered and provided by the optimization function and the actual observed measurements and costs/revenues.

Outputs from this function will be available regarding the last performed optimization procedure, either in day-ahead or in real-time mode.

##### 2.4.2.3.2 BESS parameters calculation function / BESS modelling function

This function provides the BESS parameters provided the data necessary to characterize the storage system.

A bessAssets unitary structure, presented at section 2.2.1.8, should be completely filled for each BES unit (battery and respective inverter) for this function to be accessible either independently or by the optimization function.

##### 2.4.2.3.3 Inputs parser and verifier function

This function serves as an entry point to both endpoints, verifying the adequacy of the inputs provided upstream to the called functions, namely measurements and static parameters.

##### 2.4.2.3.4 Forecasts parser and verifier function

This function has a similar purpose to the one presented at section 2.4.2.3.3, but specifically directed at evaluating the forecasts provided to the optimization function.

##### 2.4.2.3.5 Error diagnosis function

This function provides hints about any error that may have occurred during each call, ranging from insufficient/inadequate data to the cause for the solver not reaching an optimal result at a particular run.

## 2.5 BESS energy Optimization problem formulation and modelling

### 2.5.1 BESS energy optimization problem formulation

The BESS optimization algorithm is formulated as a Mixed Integer Linear Problem (MILP) with the objective of defining the optimal dispatch multiple BESS.

Table 56 lists the complete set of objective functions implemented in the module and their applicability to the systems and project use cases as discussed in section 1.

**Table 56** - Objective functions implemented in the optimization module respectively framed in the defined use cases and temporal context of application.

| Objective function | Applicable system/Use Cases |
|---|---|
| 1. Minimization of deviation cost from a given profile (e.g. day-ahead bids) | Hybrid Park (UC1) |
| 2. Minimization of absolute deviations from a given profile (e.g. day-ahead bids) | Hybrid Park (UC1)<br>Microgrid (UC2, UC3) |
| 3. Maximization of net profit through energy arbitrage in a market pricing scheme | Hybrid Park (UC1)<br>Microgrid (UC2, UC3) |
| 4. Minimization of operation costs | Microgrid (UC2, UC3) |

Table 57 summarizes the constraints applied in the base MILP problem as well as complementary constraints that can be added in order to capture BESS dynamics more realistically. The constraints will be enabled and disabled according to the module configuration options, namely: the applicable system, objective function and BESS and inverter model.

**Table 57** – Constraints implemented in the optimization module. Each constraint is classified as being fundamental for the base model or optional given the available data or the user's preferences.

| Description | Framework |
|---|---|
| (1.1) Charging and discharging non-simultaneity imposition per BESS and per time step<br>(2.1) Charging and discharging maximum power rates imposition per BESS and per time step | Base model |
| (1.2) Charging and discharging non-simultaneity imposition per BESS, per time step and per piecewise segment of inverter efficiency modelling<br>(2.2) Charging and discharging maximum power rates imposition per BESS, per time step and per piecewise segment of inverter efficiency modelling | Add-on: inverter efficiency modelling (optional) |
| (3.1) SoC calculation at the end of each time step, per BESS and per time step | Base model |
| (3.2) SoC calculation at the end of each time step, per BESS and per time step, considering variable inverter efficiencies | Add-on: inverter efficiency modelling (optional) |
| (4.1) Static SoC minimum and maximum percentages imposition per BESS and per time step, evaluated by SoC limits and voltage ranges. | Base model |
| (4.2) Dynamic SoC minimum and maximum percentages imposition per BESS and per time step | Add-on: dynamic SoC limits (optional) |
| (4.3) Dynamic SoC minimum and maximum percentages imposition per BESS and per time step, considering variable inverter efficiencies | Add-on: dynamic SoC limits (optional)<br>Add-on: inverter efficiency modelling (optional) |
| (5) Minimum reserve SoC content imposition per BESS. | Base model (optional) |
| (6) Unavailability of BESS flexibility for maintenance periods. | Base model (optional) |
| (7.1) Maximum admissible power injection at the point of common coupling imposition per time step | Base model (optional) |
| (7.2) Maximum admissible power injection at the point of common coupling imposition per time step, considering variable inverter efficiencies | Add-on: inverter efficiency modelling (optional) |
| (8) Constraints to establish a reactive power set point at PCC. | Base model (optional). |
| (9) Constraint limiting the conventional generators' power output to the band provided for each time step. | Base model. |
| (10.1) Limit imposition to the daily degradation related to each discharge of each BESS | Add-on: degradation limit (optional) |
| (10.2) Limit imposition to the daily degradation related to each discharge of each BESS, considering variable inverter efficiencies | Add-on: degradation limit (optional)<br>Add-on: inverter efficiency modelling (optional) |
| (11) Auxiliary charge and discharge power variables for inclusion of variable inverter efficiencies without compromising the problems linearity, defined per BESS and per time step | Add-on: inverter efficiency modelling (optional) |
| (12) Constraints regarding conventional generators power and time limits. | Base model. |
| (13) Constraints regarding maximum load and generation curtailment, load curtailment maximum consecutive time steps and minimum number of time steps between curtailment episodes and no curtailment periods. | Base model. |
| (14) Positive/negative deviation calculation per time step | Objective function 1 (Base model) |

### 2.5.2 MPC framework

As explained in section 2.1 the BESS energy optimization API can run in day-ahead or in real-time mode. In order to control the BESS in real-time and be able to adapt the control strategy to the real-time status of the system, the optimization API needs to run cyclically every 5-15 minute.

In this case, the optimization is based in a Model Predictive Control (MPC) strategy, being able to plan the control strategy of the BESS considering the real-time status of the system as well as the future status, based on accurate load, generation and price forecasts. As represented in Figure 9, the most recent data is fed to the optimization problem and the solution that is outputted serves as new information to be inputted as updated data, creating a feedback loop that minimizes the impact of forecasting and state estimation errors.

The optimization horizon can either be receding (concerns only a given day) or rolling (meaning that will consider the next 24 hours each time step).

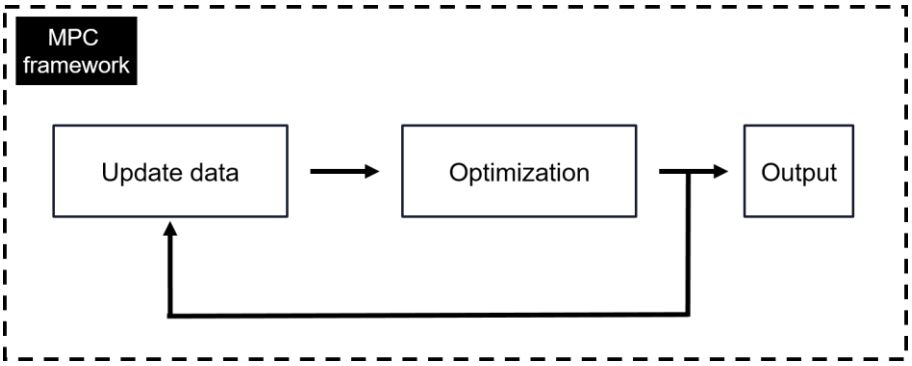



**Figure 9** –Model Predictive Control framework.

### 2.5.3 BESS models

The base MILP model embodies a simplistic BESS model that captures the fundamental aspects of the resource's constraints, namely:

- nominal energy (provided in MWh);
- nominal voltage (provided in V);
- fixed energy content limits ($SoC_{max}$ and $SoC_{min}$);
- reserve SoC ($\geq SoC_{min}$);
- the available energy content (current SoC);
- maximum and minimum C-rates;
- charge and discharge or roundtrip efficiencies (provided in %);
- respective inverter's nominal power (provided in MVA);
- respective inverter's nominal voltage (provided in V);
- respective inverter's efficiency (provided in %);

These characteristics sums into a very simple but functional model, illustrated in Figure 10, whose parameters are usually available in BESS manufacturer's datasheet.

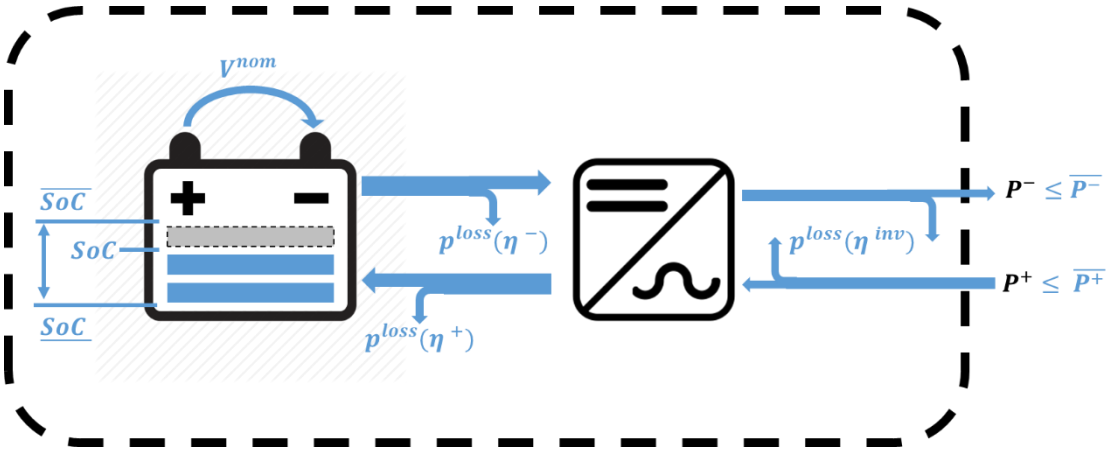



**Figure 10** - BESS base model, comprised of: the BESS current $\boldsymbol{SoC}$, bounded by its upper $\overline{\boldsymbol{SoC}}$ and lower $\underline{\boldsymbol{SoC}}$ limits; the nominal voltage $\boldsymbol{V^{nom}}$; the charge $\boldsymbol{\eta^{+}}$, discharge $\boldsymbol{\eta^{-}}$ and inverter's roundtrip $\boldsymbol{\eta^{inv}}$ efficiencies and the maximum C-rates for charging and discharging the BES unit, depicted as power limits $\overline{\boldsymbol{P^{-}}}$, $\overline{\boldsymbol{P^{+}}}$.

Complementarily, a more realistic and adaptive model can also be configured, aiming to minimize gross errors between expected and actual BESS behavior. This model is built on top of the base model taking the form of three add-ons, each focused on three different objectives:

- dynamic SoC limits' calculation;
- inverter's efficiency linear approximation;
- maximum degradation limitation.

All three add-ons are optional and data-dependent, meaning that their consideration will only take place, provided the required additional parametrization of each BES unit. The following paragraphs will describe each add-on in further detail, as well as the required parameters for their implementation.

#### 2.5.3.1 Dynamic SoC limits

This add-on models the fact that a battery cannot be fully charged/discharged at high currents due to the voltage jump/drop. This notion is modelled as complementary constraints while formulating the MILP problem, that adjust the SoC limits ($\overline{SoC}$ and $\underline{SoC}$ in Figure 10) as a function of the power set points imposed to the BESS at each time step ($P^{-}$ and $P^{+}$).

Restriction 4.2 of Table 57 sums two additional constraints that are introduced by this add-on, that linearly approximate the maximum and minimum SoC of each BESS asset. The optimization module, when provided with the measured data points, performs a least-squares approximation in order to linearize both sets of data (for charge and for discharge), computing the slopes and origins that will be used by these two constraints. An example of measured data points for a li-ion battery is illustrated at Figure 11. Each dot in the upper plot represents the measured energy fraction measured when maximum/minimum voltage was reached after performing a continuous charging/discharging procedure at the correspondent C-rate. The lower plot illustrates the least-squares approximation of the curves, whose parameters are fed into the optimization problem for dynamic SoC limits calculation at each step.

### 2.5.3.2 Inverter efficiency modelling

Typically, an inverter's roundtrip efficiency is provided as a constant value in the manufacturer's data. This approximation is valid for the majority of use cases it applies to, and could be sufficient for modelling were it not for the parameter's behaviour at low C-rates, where its variation is highly non-linear, as can be observed in Figure 12.

To account for this non-linearity, the inverter efficiency modelling add-on considers a simple piecewise linearization of the curve, with two segments, assuming the efficiency to vary linearly for lower output power fractions (≤0.1) and to be constant for higher ones (>0.1). More segments could be considered for lower power fractions but the trade-off between better accuracy and problem's decreased complexity weighs in favour of the latter for the purpose at hands.

This is the add-on with the highest computational weight since it requires the addition and modification of several constraints in order to ensure the linearity of the modelled problem, along with the introduction of new variables representing the power set points imposed at BESS level.

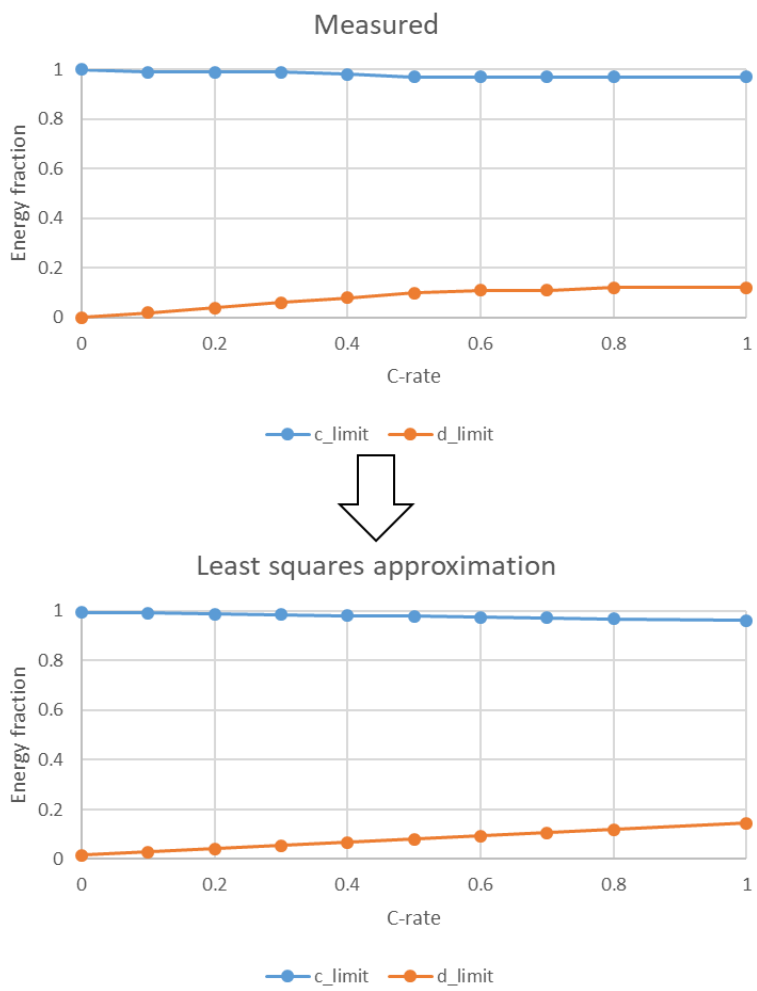



**Figure 11 -** Example plot of charging and discharging "C-rate – energy fraction" curves for a 100.9MWh stationary battery.

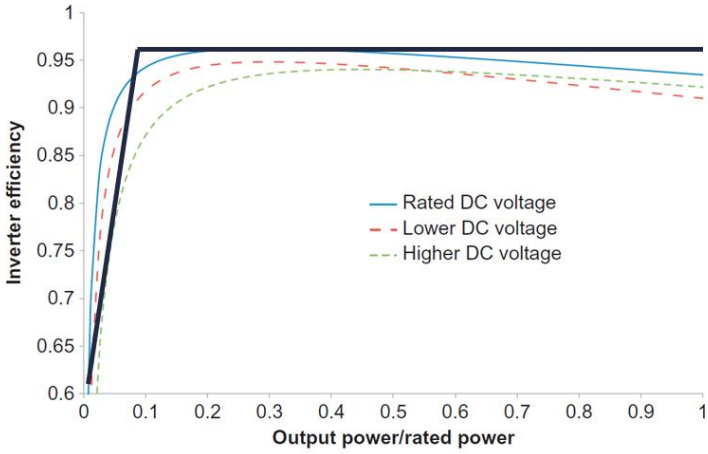



**Figure 12** - Example of a typical inverter's charge/discharge efficiency curve.

### 2.5.3.3 BESS Degradation Model

The charging/discharging strategy defined by the optimization problem will have a significant impact on the degradation of the BESS. The model developed estimates the BESS degradation based on the BESS degradation curves represented in Figure 13.

Besides the information from the batteries manufacturers, the model also requires the definition of the maximum years the BESS is expected to operate, usually resultant from the investment plan performed for the system. To ensure that the battery will reach its EOL (end-of-life) capacity (typically 70-80% of the original - also configurable) not before a specified threshold, the a restriction is added to the optimization problem (see Table 57) for limiting the total degradation allowed for the specified time horizon

The metric is calculated based on the discharging cycles imposed. For each discharge the degradation is estimated through a linear relationship between depth of discharge (DOD) and degradation percentage (Figure 13).

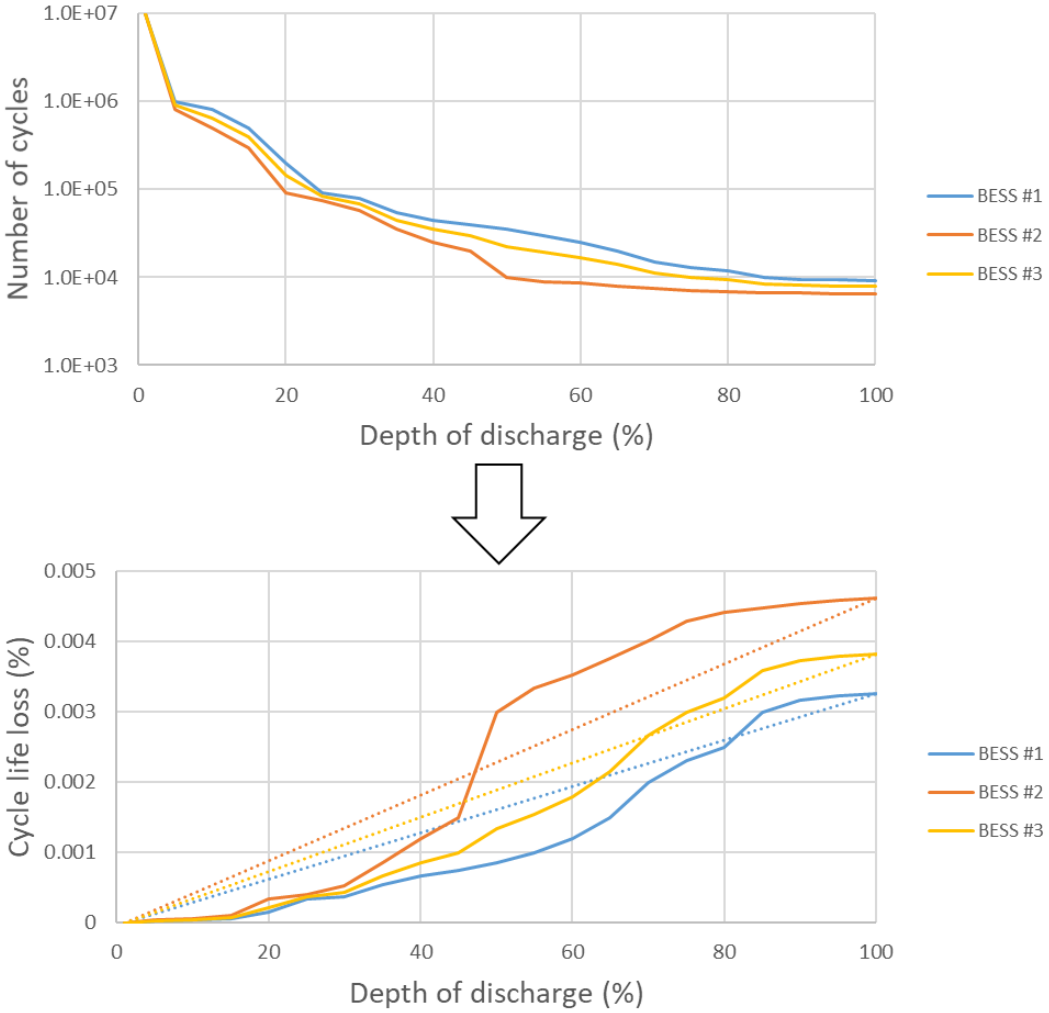



**Figure 13** - Typical curves relating total number of cycles until EOL (70%) with DOD percentage can be obtained at the manufacturer's datasheets (top). The add-on will accept the data of each BESS's curve, convert it into degradation percentage per single discharge event (cycle life loss) and use the resulting curve's linearization in the new constraint (**bottom**). The conversion between curves is made following the equation: $\frac{(100 - EOL(\%))}{number\ of\ cycles}$.

# 3. References


[1] A. A. Akhil, G. Huff, A. B. Currier, B. C. Kaun, D. M. Rastler, S. B. Chen, et al., in *DOE/EPRI 2013 Electricity Storage Handbook in Collaboration with NRECA*, Albuquerque, Sandia National Laboratories, 2015, p. 347.

[2] "European Energy Storage Technology Development Roadmap Towards 2030," Alliance, European Energy Research, European Association for Storage of Energy (EASE), 2013.

[3] "Storage as a Strategic Tool for Managing Variability and Capacity Concerns in the Modern Grid," Electricity Advisory Committee, 2008.

[4] N. Jenkins, R. Allan, P. Crossley, D. Kirschen and G. Strbac, Embebed Generation, The Institution of Electrical Engineers, 2000.

[5] J. A. P. Lopes, "Integration of dispersed generation on distribution networks-impact studies," in *Power Engineering Society Winter Meeting IEEE*, 2002.

[6] D. Pudjianto, M. Aunedi, P. Djapic, and G. Strbac, "Whole-systems assessment of the value of energy storage in low-carbon electricity systems," em *Smart Grid, IEEE Transactions*, 2014.

[7] "Energy Storage Trends and Opportunities in Emerging Markets," ESMAP, IFC, 2017.

[8] Holger C. Hesse, Michael Schimpe, Daniel Kucevic and Andreas Jossen, "Lithium-Ion Battery Storage for the Grid—A Review of Stationary Battery Storage System Design Tailored for Applications in Modern Power Grids," em *energies*, 2017.

[9] I. T. S. Miranda, "Integration of Battery Energy Storage Systems in the planning and operation of distribution networks," Porto, 2017.

[10] M. G. Molina, "Energy Storage and Power Electronics Technologies: A Strong Combination to Empower the transformation to the Smart Grid," in *IEEE*, 2017.

[11] "Battery University," 21 03 2017. [Online]. Available: https://batteryuniversity.com/learn/archive/whats_the_best_battery. [Accessed 20 Oct 2018].

[12] Dunn B1, Kamath H, Tarascon JM., "Electrical energy storage for the grid: a battery of choices," em *Science*, 2011.

[13] I. T. S. Miranda, "Otimização do Planeamento de Sistemas de Armazenamento Distribuído de Energia em Redes com Elevada Produção Dispersa," 2012.

[14] J. G. A. P. d. Fonseca, "Análise térmica de um sistema contentorizado de armazenamento de energia em baterias," Porto, 2018.

[15] "Sistemas ecológicos de extinção de incêndios," Saphire.

[16] ABB, "Smart Grid - Power to Control Energy - Converters & Battery Energy Storage Systems".

[17] General Electric, [Online]. Available: http://www.geautomation.com/system/files/files/gea-s1300.pdf. [Accessed 7 11 2018].

[18] "American Solar Energy Society," Solar Today Magazine, [Online]. Available: https://www.ases.org/schneider-releases-conext-insight/. [Accessed 8 11 2018].

[19] "Solar Quarter," [Online]. Available: https://solarquarter.com/index.php/solarquarter-blog/103-company-blogs/13889-schneider-electric-s-new-conext-xw-pro-hybrid-inverter-will-meet-california-rule-21. [Acedido em 9 11 2018].

[20] Greensmith, "Greensmith Energy Management Systems," in *IEEE Supersession on Energy Storage*, 2011.

[21] "DocSlide," [Online]. Available: https://docslide.us/documents/features-and-benefits-ge-renewable-energy-robust-plant-level-control-outdoor.html. [Accessed 8 11 2018].

[22] "NEC Energy Solution's Distributed Storage Solution (DSSTM) enables advanced energy management and resiliency services for commercial & industrial customers and the utilities that serve them," NEC Energy Solutions.

[23] Siemens, [Online]. Available: https://www.siemens.com/global/en/home/products/energy/energy-automation-and-smart-grid/microgrid/spectrum-power-mgms.html. [Accessed 6 11 2018].

[24] Siemens, 2016. [Online]. Available: https://www.siemens.com/content/dam/webassetpool/mam/tag-siemens-com/smdb/energy-management/energy-automation-and-smart-grid/microgrids/mgms-advanced-control-brochure.pdf. [Accessed 6 11 2018].

[25] Siemens, 2015. [Online]. Available: https://w3.usa.siemens.com/smartgrid/us/en/microgrid/Documents/Spectrum_Power_7_MGMS_Application_Guide.pdf. [Accessed 6 11 2018].

[26] ABB, [Online]. Available: https://library.e.abb.com/public/eefb9712ad0243ff8d38a7671bbd9529/EssPro-Grid_Brochure_2017.pdf. [Acedido em 7 11 2018].

[27] General Electric, [Online]. Available: https://www.gegridsolutions.com/multilin/catalog/mcs.htm. [Accessed 7 11 2018].

[28] Schneider, [Online]. Available: https://41j5tc3akbrn3uezx5av0jj1bgm-wpengine.netdna-ssl.com/wp-content/uploads/2018/09/DS20180919-PPC-CA2-Datasheet.pdf. [Accessed 8 11 2018].

[29] Schneider, [Online]. Available: https://41j5tc3akbrn3uezx5av0jj1bgm-wpengine.netdna-ssl.com/wp-content/uploads/2017/10/BR20171013_Conext-Advisor-2-Brochure.pdf. [Accessed 07 11 2018].

[30] Scneider electric, [Online]. Available: https://41j5tc3akbrn3uezx5av0jj1bgm-wpengine.netdna-ssl.com/wp-content/uploads/2018/06/BR20180615-Conext-Insight.pdf. [Accessed 8 11 2018].

[31] Schneider Electric, [Online]. Available: https://solar.schneider-electric.com/schneider-electric-announces-new-monitoring-and-edge-control-solutions-for-commercial-industrial-solar-and-storage-projects/. [Accessed 9 11 2018].

[32] Schneider Electric, 9 11 2018. [Online]. Available: https://solar.schneider-electric.com/product/conext-gateway/.

[33] NEC Energy Solutions, [Online]. Available: https://www.neces.com/products-services/grid-energy-storage-products/aeros-r-control-system/. [Accessed 9 11 2018].

[34] NEC Energy Solutions, [Online]. Available: https://www.neces.com/products-services/grid-energy-storage-products/aeros-r-control-system/aeros-display/. [Acedido em 9 11 2018].

[35] Nidec, [Online]. Available: https://www.aeit.it/man/CA2015/atti/SessioniPlenarie/03_Brocca.pdf. [Accessed 10 11 2018].

[36] Nidec, [Online]. Available: https://www.nidec-industrial.com/wp-content/uploads/2017/07/DEP2015.07.01.00EN.ARTICS-Smart-Energy.pdf. [Accessed 10 11 2018].

[37] Younicos, [Online]. Available: https://www.energy-storage-online.de/vis-content/event-energy2017/exh-energy2017.2510057/Energy-Storage-Europe-2017-Younicos-AG-Paper-energy2017.2510057-O4SKur0WQDiCKB8pSuL1MQ.pdf. [Accessed 12 11 2012].

[38] Greensmith, [Online]. Available: https://www.greensmithenergy.com/sites/default/files/greensmith-energy-brochure.pdf. [Accessed 12 11 2018].

[39] O. Schmidt, A. Hawkes, A. Gambhir e I. Staffell, "The future cost of electrical energy storage based on experience rates," *Nature Energy*, vol. 2, nº 8, 2017.

[40] B. Liu, Z. Lu, K. Yao e F. Gao, "A MPC Operation Method for A Photovoltaic System with Batteries," *IFAC-PapersOnLine*, vol. 48, nº 8, pp. 807-812, 2015.

[41] K. Yuasa, T. Shimakage, N. Takeuchi e Y. Sugiyama, "Optimized storage battery control in hybrid power distribution system for improving energy self-consumption," em *2016 IEEE International Telecommunications Energy Conference (INTELEC)*, 2016.

[42] G. Henri, N. Lu e C. Carrejo, "Design of a novel mode-based energy storage controller for residential PV systems," em *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, 2017.

[43] G. Henri, N. Lu e C. Carrejo, "A Machine Learning Approach for Real-time Battery Optimal Operation Mode Prediction and Control," em *2018 IEEE/PES Transmission and Distribution Conference and Exposition (T&D)*, 2018.

[44] R. Martins, P. Musilek e H. C. Hesse, "Optimization of photovoltaic power self-consumption using linear programming," em *2016 IEEE 16th International Conference on Environment and Electrical Engineering (EEEIC)*, 2016.

[45] E. Namor, F. Sossan, R. Cherkaoui e M. Paolone, "Control of Battery Storage Systems for the Simultaneous Provision of Multiple Services," *IEEE Transactions on Smart Grid*, pp. 1-1, 2018.

[46] Y. Parvini, A. Vahidi e S. A. Fayazi, "Heuristic Versus Optimal Charging of Supercapacitors, Lithium-Ion, and Lead-Acid Batteries: An Efficiency Point of View," *IEEE Transactions on Control Systems Technology*, vol. 26, nº 1, pp. 167-180, 2018.

[47] C. Zou, C. Manzie e D. Nesic, "Model Predictive Control for Lithium-Ion Battery Optimal Charging," *IEEE/ASME Transactions on Mechatronics*, vol. 23, nº 2, pp. 947-957, 2018.

[48] J. M. Reniers, G. Mulder, S. Ober-Blöbaum e D. A. Howey, "Improving optimal control of grid-connected lithium-ion batteries through more accurate battery and degradation modelling," *Journal of Power Sources*, vol. 379, pp. 91-102, 2018.

[49] B. Heymann, J. F. Bonnans, P. Martinon, F. J. Silva, F. Lanas e G. Jiménez-Estévez, "Continuous optimal control approaches to microgrid energy management," *Energy Systems*, vol. 9, nº 1, pp. 59-77, 2017.

[50] M. Kazemi e H. Zareipour, "Long-Term Scheduling of Battery Storage Systems in Energy and Regulation Markets Considering Battery's Lifespan," *IEEE Transactions on Smart Grid,* vol. 9, nº 6, pp. 6840-6849, 2018.

[51] W. Zhi, Z. Xiao-Ping, J. Brandt, Z. Su-Yang e L. Jia-Ning, "Three Control Approaches for Optimized Energy Flow With Home Energy Management System," *IEEE Power and Energy Technology Systems Journal,* vol. 2, nº 1, pp. 21-31, 2015.

[52] J. Meunier, J. Rohmer, D. Knittel e P. Collet, "Metaheuristic based battery control optimization of a photovoltaic system with grid connection," em *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016.

[53] G. M. Kaci, A. Mahrane, M. Chikh e K. Ghedamsi, "PV self-consumption improvements with energy flow management and storage — Case of solar home in the north of algeria," em *2017 5th International Conference on Electrical Engineering - Boumerdes (ICEE-B)*, 2017.

[54] A. S. M. Shah, Y. Ishikawa, S. Odakura e N. Kakimoto, "Power Control Modelling for Future Energy Management Based on Photovoltaic Integrated System with Lithium-Ion Storage Batteries," em *2014 8th Asia Modelling Symposium*, 2014.

[55] C. Zou, C. Manzie e D. Nesic, "A Framework for Simplification of PDE-Based Lithium-Ion Battery Models," *IEEE Transactions on Control Systems Technology,* vol. 24, nº 5, pp. 1594-1609, 2016.

[56] I. Ranaweera, O. Midtgård, M. Korpås e H. Farahmand, "Control strategies for residential battery energy storage systems coupled with PV systems," em *2017 IEEE International Conference on Environment and Electrical Engineering and 2017 IEEE Industrial and Commercial Power Systems Europe (EEEIC / I&CPS Europe)*, 2017.

[57] R. L. Fares e M. E. Webber, "A flexible model for economic operational management of grid battery energy storage," *Energy,* vol. 78, pp. 768-776, 2014.

[58] A. Parisio e L. Glielmo, "Energy efficient microgrid management using Model Predictive Control," em *2011 50th IEEE Conference on Decision and Control and European Control Conference*, 2011.

[59] P. Fortenbacher, J. L. Mathieu e G. Andersson, "Modeling, identification, and optimal control of batteries for power system applications," em *2014 Power Systems Computation Conference*, 2014.

[60] P. Fortenbacher, G. Andersson e J. L. Mathieu, "Optimal real-time control of multiple battery sets for power system applications," em *2015 IEEE Eindhoven PowerTech*, 2015.

[61] F. R. S. Sevilla, V. Knazkins, C. Park e P. Korba, "Advanced Control of Energy Storage Systems for PV Installation Maximizing Self-Consumption," *IFAC-PapersOnLine,* vol. 48, nº 30, pp. 524-528, 2015.

[62] P. Fortenbacher, J. L. Mathieu e G. Andersson, "Modeling and Optimal Operation of Distributed Battery Storage in Low Voltage Grids," *IEEE Transactions on Power Systems,* vol. 32, nº 6, pp. 4340-4350, 2017.

[63] Y. Riffonneau, S. Bacha, F. Barruel e S. Ploix, "Optimal Power Flow Management for Grid Connected PV Systems With Batteries," *IEEE Transactions on Sustainable Energy,* vol. 2, nº 3, pp. 309-320, 2011.

[64] K. Abdulla, J. De Hoog, V. Muenzel, F. Suits, K. Steer, A. Wirth e S. Halgamuge, "Optimal operation of energy storage systems considering forecasts and battery degradation," *IEEE Transactions on Smart Grid,* vol. 9, nº 3, pp. 2086-2096, 2018.

[65] Y. Jun, Zhang, C. Zhao, W. Tang e S. H. Low, "Profit-Maximizing Planning and Control of Battery Energy Storage Systems for Primary Frequency Control," *ArXiv e-prints,* 1 April 2016.

[66] T. Morstyn, B. Hredzak, R. P. Aguilera e V. G. Agelidis, "Model Predictive Control for Distributed Microgrid Battery Energy Storage Systems," *ArXiv e-prints,* 01 February 2017.

[67] A.-a. Mamun, I. Narayanan, D. Wang, A. Sivasubramaniam e H. K. Fathy, "A Stochastic Optimal Control Approach for Exploring Tradeoffs between Cost Savings and Battery Aging in Datacenter Demand Response," *IEEE Transactions on Control Systems Technology,* vol. 26, nº 1, pp. 360-367, 2018.

[68] X. Wu, X. Hu, X. Yin e S. J. Moura, "Stochastic Optimal Energy Management of Smart Home With PEV Energy Storage," *IEEE Transactions on Smart Grid,* vol. 9, nº 3, pp. 2065-2075, 2018.

[69] J. Qin, R. Sevlian, D. Varodayan e R. Rajagopal, "Optimal electric energy storage operation," em *2012 IEEE Power and Energy Society General Meeting*, 2012.

[70] Q. Wei, G. Shi, R. Song e Y. Liu, "Adaptive Dynamic Programming-Based Optimal Control Scheme for Energy Storage Systems With Solar Renewable Energy," *IEEE Transactions on Industrial Electronics,* vol. 64, nº 7, pp. 5468-5478, 2017.

[71] B. Mbuwir, F. Ruelens, F. Spiessens e G. Deconinck, "Battery Energy Management in a Microgrid Using Batch Reinforcement Learning," *Energies,* vol. 10, nº 11, p. 1846, 2017.

[72] E. Kuznetsova, Y.-F. Li, C. Ruiz, E. Zio, G. Ault e K. Bell, "Reinforcement learning for microgrid energy management," *Energy,* vol. 59, pp. 133-146, 2013.

[73] D. R. Jiang, T. V. Pham, W. B. Powell, D. F. Salas e W. R. Scott, "A comparison of approximate dynamic programming techniques on benchmark energy storage problems: Does anything work?," em *Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), 2014 IEEE Symposium on*, 2014.

[74] D. Ernst, M. Glavic, F. Capitanescu e L. Wehenkel, "Reinforcement learning versus model predictive control: a comparison on a power system problem," *IEEE Trans Syst Man Cybern B Cybern,* vol. 39, nº 2, pp. 517-529, 2009.

[75] X. Xi, R. Sioshansi e V. Marano, "A stochastic dynamic programming model for co-optimization of distributed energy storage," *Energy Systems,* vol. 5, nº 3, pp. 475-505, 2014.

[76] D. Manz, R. Piwko e N. Miller, "Look Before You Leap: The Role of Energy Storage in the Grid," *IEEE Power and Energy Magazine,* vol. 10, nº 4, pp. 75-84, 2012.

[77] Y. Shi, C. Træholt e B. Poulsen, "Economic dispatch of electric energy storage with multi-service provision," em *2010 Conference Proceedings IPEC*, 2010.

[78] F. Teng e G. Strbac, "Business cases for energy storage with multiple service provision," *Journal of Modern Power Systems and Clean Energy,* vol. 4, nº 4, pp. 615-625, 2016.

[79] A. Perez, R. Moreno, R. Moreira, M. Orchard e G. Strbac, "Effect of Battery Degradation on Multi-Service Portfolios of Energy Storage," *IEEE Transactions on Sustainable Energy,* vol. 7, nº 4, pp. 1718-1729, 2016.

[80] E. Namor, F. Sossan, R. Cherkaoui e M. Paolone, "Control of Battery Storage Systems for the Simultaneous Provision of Multiple Services," *IEEE Transactions on Smart Grid,* 2018.

[81] F. Saidani, F. X. Hutter, R.-G. Scurtu, W. Braunwarth e J. N. Burghartz, "Lithium-ion battery models: a comparative study and a model-based powerline communication," *Advances in Radio Science,* vol. 15, pp. 83-91, 2017.

[82] M. Chen e G. A. Rincón-Mora, "Accurate Electrical Battery Model Capable of Predicting Runtime and I-V Performance," *IEEE Transactions on Energy Conversion,* vol. 21, nº 2, pp. 504-511, 2006.

[83] M. Rampazzo, M. Luvisotto, N. Tomasone, I. Fastelli e M. Schiavetti, “Modelling and simulation of a Li-ion energy storage system: Case study from the island of Ventotene in the Tyrrhenian Sea,” *Journal of Energy Storage*, vol. 15, pp. 57-68, 2018.

[84] H. Chaoui e H. Gualous, “Adaptive State of Charge Estimation of Lithium-Ion Batteries With Parameter and Thermal Uncertainties,” *IEEE Transactions on Control Systems Technology*, vol. 25, nº 2, pp. 752-759, 2017.