Smart Anything Everywhere Initiative

Area 3: Advanced micro-electronics components and Smart System Integration Project: H2020–No 761809

Digital Innovation Hubs boosting European Microelectronics Industry

D3.2: Farm Monitoring and Control Application Experimentation

Author(s): Th. Zahariadis, P. Athanasoulis, K. Railis, P. Kapsalis

Status -Version: FF

Delivery Date (DoA): 31 May 2018

Actual Delivery Date: 30 May 2018

Distribution - Confidentiality: Public

Code: DIATOMIC_D3.2_SYN_FF-20180530

Abstract:

This manual provides a useful step-by-step guide for the precision agriculture integrated platform.

The integrated platform includes the Synfield platform, the Virtual Fort Knox Cloud platform and the plant-o-meter smartphone application. The Synfield nodes and the plant-o-meter devices collect measurements from the fields in this experiment.

Disclaimer

The DIATOMIC Consortium is the following:

Participant number	Participant organization name	Short name	Country
01	INTRASOFT International S.A.	INTRA	BE
02	F6S NETWORK LIMITED	F6S	UK
03	BioSense	BIOS	SRB
04	Synelixis Solutions	SYN	EL
05	Instituto Pedro Nunes	IPN	PT
06	Fraunhofer IPA	IPA	DE
07	InoSens	INO	SRB
08	Libelium Comunicaciones Distribuidas SL	LIB	ES
09	FastTrack	FASTT	PT

Document Revision History

Date	Issue	Author/Editor/Contributor	Summary of main changes
04.05.2018	V0.1	P. Athanasoulis, K. Railis (SYN)	Initial version of document
15.05.2018	V0.2	P. Athanasoulis, P. Kapsalis (SYN)	Add contributions from IPA and BIOS
17.05.2018	V0.3	Th. Zahariadis, P. Kapsalis (SYN)	SynField Analysis
21.05.2018	V0.4	P. Athanasoulis (SYN)	Finalize document
29.05.2018	V0.5	R. Hermoso (LIB)	Peer Review
30.05.2018	V0.6	P. Athanasoulis (SYN)	Pre-final version
30.05.2018	FF	B. Ipektsidis, D.Rublova (INTRA)	Final version

Table of Contents

1 Introduction ____ *10*

1.1 Goal ____ 10

1.2 Test scenario ____ 11

2 Modules description ____ *12*

2.1 The Plant-O-Metersmartphone application ____ 12

2.2 SynfieldPlatform ____ 16

2.3 Virtual Fort Knox Cloud Platform ____ 53

3 Step by step implementation of test scenario for Application Experiment: Agrifood ____ *77*

3.1 A general overview ____ **77**

4 Annex ____ *80*

4.1 Nitrogen rate experiment in maize ____ **80**

4.2 Smart irrigation experiment ____ **82**

List of Figures

Figure 1: Agro Application Experiment 11
Figure 2: Operating principle of the Plant-O-Meter 12
Figure 3: Plant-O-Meter (left) and specifically designed smartphone application (right) 12
Figure 4: Main menu of the Plant-O-Meter application 13
Figure 5: Point mode of measurement at an arbitrary location 13
Figure 6: Head node homocentric circles mode of measurement 14
Figure 7: Continuous mode of measurements 15
Figure 8: SynField ecosystem 16
Figure 9: SynField Head Node (left) and Peripheral Node (right) 16
Figure 10: Typical SynField installation 17
Figure 11: Registration in the Synfield platform 17
Figure 12: History of user subscriptions 18
Figure 13: AvailableSMS and remaining API calls 18
Figure 14: Selected SynControl Android app screen 19
Figure 15: Add a new field in your account 19
Figure 16: Register in the platform a new field 20
Figure 17: Manage your fields 20
Figure 18: Register a Synfield head node 21
Figure 19: Step 1- Type the basic details of the Synfield head node 21
Figure 20: Step 2 - Modify the head node's coordinates 22
Figure 21: Register a peripheral node as part a Synfield network 22
Figure 22: Manage the nodes in similar way 23
Figure 23: Add a new sensor in the head node 24
Figure 24: Follow the wizard to complete the sensor's registration 25
Figure 25: Manage the registered sensors 25
Figure 26: Add a new actuator in a Synfield node 26
Figure 27: Follow the wizard to register a new actuator 26
Figure 28: View the registered actuators 27
Figure 29: Add a new external device (e.g. Plant-O-Meter) 27
Figure 30: Register the Plant-O-Meter in Synfield platform 28
Figure 31: Managethe registered devices 28
Figure 32:Current status of sensing services in Synfield node 29
Figure 33: Select sensing services to be visualized 29
Figure 34: Visualize air temperature and relative humidity in one plot 30

Figure 35: Affected crop types by specific disease 30
Figure 36: Diseases Repository....31
Figure 37: Initialising disease prediction in a field 31
Figure 38: Setting disease prediction details 32
Figure 39: Basic step of irrigation rule 32
Figure 40: Define the metrics, necessary for the irrigation rule 33
Figure 41: Define the soil profile 33
Figure 42: Associate one or more actuators with the irrigation rule and define the irrigation plan 34
Figure 43: Submit and activate the irrigation rule....34
Figure 44:Overview of Synfield API in swagger 35
Figure 45: Authentication through user credentials....36
Figure 46: Description of the endpoint that fetches the list of fields of the user 36
Figure 47: List of fields 37
Figure 48: Description of the web service that fetches the list of sensing services of the node 38
Figure 49: List of sensing services per node....38
Figure 50: Description of the web service that fetches the list of measurements per sensing service 39
Figure 51: List of measurements per sensing service 40
Figure 52: Description of the web service that fetches the list of measurements per sensing service in date range 40
Figure 53: List of measurements per sensing service in date range 41
Figure 54: Description of the web service that fetches the list of measurements per node in date range 42
Figure 55: List of measurements per node in date range....43
Figure 56: Description of the web service that fetches latest measurement of each sensing service per node 44
Figure 57: List of latest measurements (per sensing service) per node 45
Figure 58: Description of the web service that fetches the list of actuators in a node....46
Figure 59: List of actuators per node 47
Figure 60: Description of the web service that sets the actuator in automatic mode 48
Figure 61: Set the actuator in automatic mode 48
Figure 62: Description of the web service that sets the actuator in manual mode 49
Figure 63: Set the actuator in manual mode 49
Figure 64: Description of the web service that activates the actuator (manual mode) 50
Figure 65: Activates an actuator 50
Figure 66: Description of the web service that allows the registration of metrics by device type and device serial number 51
Figure 67: Register measurements for a specific Plant-O-meter device....51
Figure 68: Retrieve the measurements from a Plant-O-Meter device 53
Figure 69: Virtual Fort Knox Concept....54

Figure 70: Virtual Fort Knox Architecture 54
Figure 71: VFK cell concept with locally hosted cells and publicly hosted cells in the main infrastructure 55
Figure 72: Communication Process of the Manufacturing Service Bus 56
Figure 73: Communication Pattern of the Manufacturing Services Bus 56
Figure 74: Self-description of services 57
Figure 75: Exemplary pattern for data transfer of smart object to an application 57
Figure 76: Extended self-description for RESTful API 59
Figure 77: REST endpoint to register application 60
Figure 78: REST endpoint to register Smart Object 60
Figure 79: Manual app creation wizard – Step 1: Basic Information 62
Figure 80: Manual app creation wizard – Step 2: Endpoints 63
Figure 81: Manual app creation wizard – Step 3: Functions 63
Figure 82: Manual app creation wizard – Step 4: Response Events 64
Figure 83: REST endpoint to send data 65
Figure 84: Activate new Component in GUI of MSB 65
Figure 85: Security Token and Visibility Setting (MSB GUI) 66
Figure 86: Detailed Information about Component (MSB GUI) 67
Figure 87: Detailed View on Component - Configurations Tab 67
Figure 88: Detailed View on Component - List of all associated Flows (MSB GUI) 68
Figure 89: Simple Information Flow modelled in the MSB GUI (only part of GUI is shown) 69
Figure 90: Steps to compete the Exchange of Information 69
Figure 91: First Step in Creating a new Integration Flow (MSB GUI) 70
Figure 92: Drag and Drop of Components to initiate the Creation of a new Integration Flow 70
Figure 93: Selection of Event or Function from Drop Down Menu 71
Figure 94: Selection of Return Event based on preselected Function 72
Figure 95: Successfully linked Components 72
Figure 96: Detail View for Mapping Event Data to Input Parameters for a Function 73
Figure 97: View to set Condition which incoming Events are forwarded to the next Component in the Flow. 74
Figure 98: Branching Integration Flow where one Event is forwarded to Two Components 75
Figure 99: Merging of Branches in an Integration Flow 75
Figure 100: Integration Flow after Saving 76
Figure 101: Integration steps 77
Figure 102: Synfield Platform as Application in MSB 78
Figure 103: POST request to register Synfield Platform as application to MSB 79
Figure 104: Integration flow to request data from Synfield Platform 79
Figure 105: Nitrogen trial setup 81

Figure 106: Sensors topology.....82

List of Abbreviations

API	Application programming interface
DIH	Digital Innovation Hubs
HN	Head node
PN	Peripheral node
JSON	JavaScript Object Notation
GPS	Global Positioning System
NDVI	Normalized Difference Vegetation Index
GUI	Graphical User Interface
VFK	Virtual Fort Knox
IPA	Institute for Manufacturing Engineering and Automation
SDK	Software Development Kit
MSB	Manufacturing Service Bus

1 Introduction

The DIATOMIC project vision is to establish a sustainable ecosystem, which will facilitate digital innovation in the health, agrifood and manufacturing sectors, all of which are under-digitized and of prime importance for society and the economy. It should be underlined that such achievement is quite important for the European industry, especially the SMEs, as most SMEs and midcaps in these sectors are characterised by a low Digital Innovation Index (we name these enterprises “non-tech” hereafter).

At the heart of the DIATOMIC ecosystem, three interconnected sector-specific Digital Innovation Hubs (DIH) pulsate to accelerate digitization in-beat with sector-specific needs, and to enable delivery of AME/SSI based applications to a critical mass of customers. DIHs first assist non-tech companies to find and couple with counterparts from across the EU with complementary technological competencies. Then, acting as a one-stop-shop, they offer the teams both (i) strong technological support to accelerate design, development, prototyping and manufacturing; and (ii) business support to develop solutions with a robust product/market fit, and chart the path to market growth and investment.

This document presents a thoughtful guide on how to integrate an existing solution with DIATOMIC farm monitoring platform. The platform consists of the BIOS Plant-o-meter system and the Synfield platform and contains several components. The users and the developers may integrate their solutions with all or a subset of the described components, which best suit their needs.

For the purpose of this document, a full integration with all the components is described, so users can have a broader view of all the potentialities of every component and the greater value added by integrating then together.

1.1 Goal

The objective of the Agro Application Experiments in DIATOMIC is to create a fully functional service that offers the opportunity to farmers/agronomists/farm logistics managers to access all important parameters to accomplish efficient and self-sustainable agricultural production and offer innovative and specialized services in the agricultural domain. DIATOMIC integrates and extends in this experiment Synelixis SynField, BIOS Plant-O-Meter and Fraunhofer VFK systems to offer unprecedented flexibility to prospective developers and an enhanced experience to technology adopters and users.

SynField offers solutions that help farmers control their farming procedures remotely, with high accuracy and at low cost to meet the needs, not only of large, but also, of small and medium sized farms. SynField allows users to monitor environmental conditions at one or multiple points on the farm (installing SynField Peripheral Nodes-SF PN and one SynField Head Node - SF HN) and use them to control (through relays or actuators) **irrigation**. The sensed data are delivered to the SynCloud (through SF HN) where commands for actuations are decided. The system (currently TRL-9) also alerts the farmer under specific user-defined conditions so that they may take appropriate action.

BIOS has developed a solution for **effective nutrient balance management** in precision agriculture and gardening, through the integration of an innovative instrument for on-the-fly nitrogen monitoring into a versatile and low-cost system for expert support to farmers through a combination of disciplines like soil science, agronomy, economics and information technology. The *Plant-O-Meter is a* handheld device (currently TRL 7): the optical sensor head contains a multispectral light source, photodetector and basic electronic circuitry for signal amplification and power supply. The sensor is connected with a smartphone using Bluetooth or Wi-Fi and thus supports the farmer/agronomist to define the needs for fertilizers. The sensor also contains a high-capacity, rechargeable, Li-Ion battery to support the smartphone via USB, thus ensuring 8-hours operation of the phone.

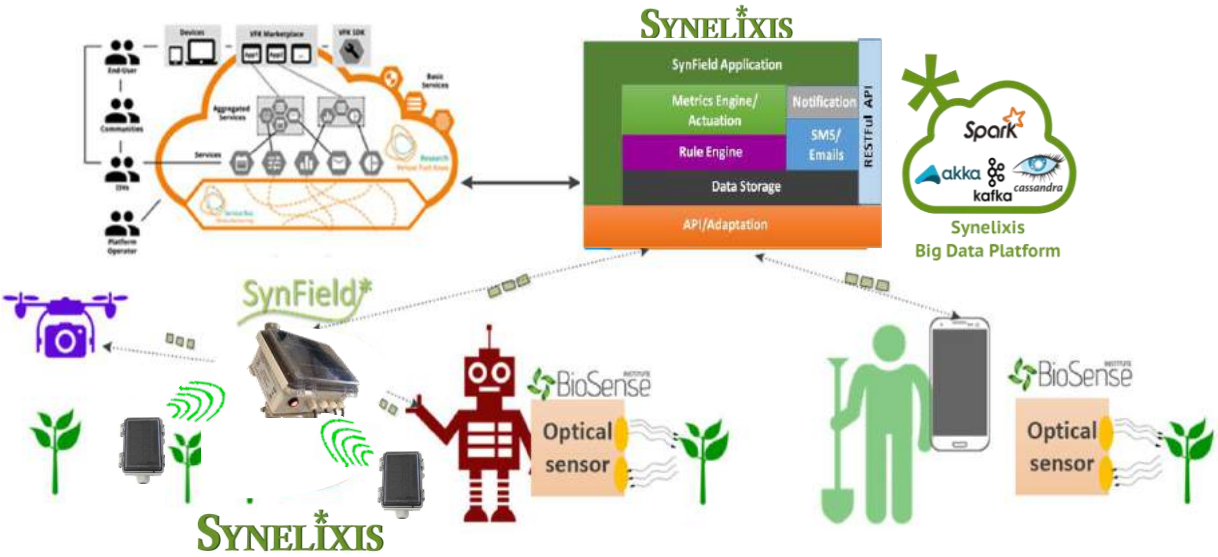


Figure 1: Agro Application Experiment

1.2 Test scenario

Taking into account that in any precision agriculture system, prospective developers are interested to experiment with different sensors and apply different logics and algorithms, in the framework of DIATOMIC, the following functionality could be considered as test scenarios:

- **Integration of Plant-O-meter with the SynField system** to offer an opportunity to farmers/agronomists/farm logistics managers to monitor both environmental and soil conditions from a single point of access. The readings from Plant-O-Meter are transferred and stored at the SynField Cloud through an interface that will be developed. By accessing the SynField platform through any device (laptop, smartphone, tablet), they will be able to monitor all conditions and control irrigation and fertilisation.
- **Enrich SynField with drones equipped with cameras** that fly over the fields and collect visual information enabling the farmers to inspect every square meter of their farm without physically visiting it. The collected information may be delivered to the farmers through the SynField front end.
- **Integrate an advanced version of the Plant-O-Meter instrument onto a mobile robotic platform** (to allow for automation of the data collection process and identify two pilot sites most suitable for testing and validation of the automated version of the solution.
- **Integrate SynField Cloud with Virtual FortKnox** towards delivering to developers a platform that accelerates significantly application development. While SynField enables the user to set rules for notifications and actuation (e.g. alert me when soil humidity is below a threshold), more advanced applications like disease prediction necessitate more advanced algorithms execution on the collected data and possible combination with 3rd party sources, using the VFK platform.

In the Annex of this deliverable some trials are described in detail.

2 Modules description

In this section, the different modules are presented and described. This includes the overall description of the functionalities that each module contains with all the necessary information that a user may need to understand and start working with them.

2.1 The Plant-O-Meter smartphone application

BioSense has developed a portable multispectral optical device called Plant-O-Meter (currently at level TRL 7) for precise plant stress measurement, which reconciles the requirements for a simple, cost-effective optical device and accuracy of applied optical method. The proposed device is based on a specially designed multispectral source that integrates the light sources of the four most indicative wavelengths in one optical module and enables simultaneous illumination of the whole plant. The four wavelengths emitted by the source, namely 850, 630, 535 and 465 nm, are chosen to cover the most indicative spectral bands and enable the calculation of multiple vegetation indices. Illumination and detection of reflected signals are performed in a sequential manner, which provides very quick measurements and raw data collection. The detector module sequentially receives the reflected spectrum and sends a raw data to the microcontroller. Each measurement comprises 128 excitation impulses for each wavelength, and the reflected signal is calculated as an average of the corresponding reflected pulse train. The raw data is transmitted to and processed in a smartphone using a specially developed application, which enables determination of vegetation indices. In particular, signifying plant stress caused by nitrogen deficiency and/or drought. The operating principle of the Plant-O-Meter is shown in Figure 2.

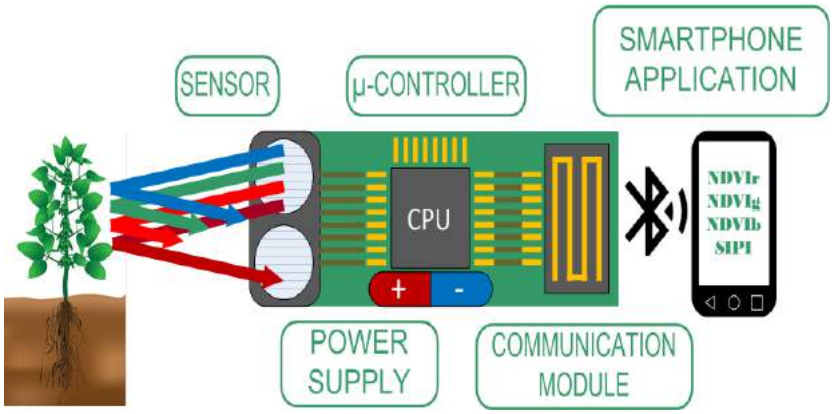


Figure 2: Operating principle of the Plant-O-Meter

In the frame of the DIATOMIC project, a new version of the device has been developed that is more immune to the ambient lighting and more ergonomic than the previous one. The updated version together with the smartphone application is shown in Figure 3.

Figure 3: Plant-O-Meter (left) and specifically designed smartphone application (right)

One of the key advantages of the proposed solution for plant stress and nutrient measurements is its coupling with the smartphone application. With the help of the application, one can initiate, gather, georeference and save data. Furthermore, the application uploads measurement results together with GPS coordinates to the SynField API (see section 2.2.11). The main menu of the application is shown in Figure 4. Below the figure, each menu item is explained separately.

Figure 4: Main menu of the Plant-O-Meter application

The first menu item *Find devices*, initiates Bluetooth connection between the Plant-O-Meter and the smartphone. Once the connection is established the user can choose between two single point modes and one automatic mode of measurements. The first mode, Figure 5, is made for point measurements at an arbitrary location. In this mode the user can initiate measurements, save them, and upload them to the SynField API if there is an internet connection. For safety reasons all measurements data are stored in the smartphone's memory until the upload to server is successful. Arbitrary location mode offers instant optical indices results.

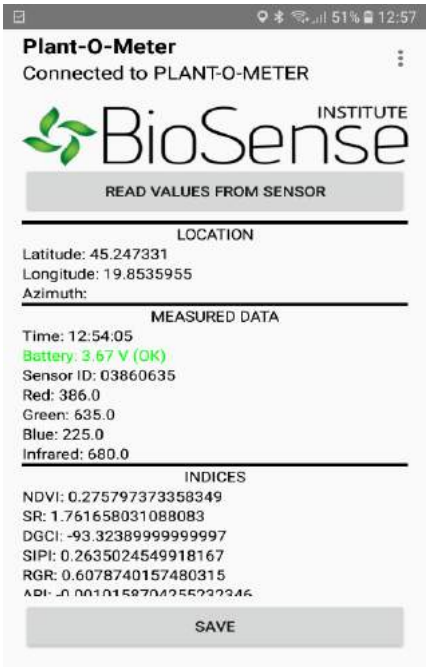


Figure 5: Point mode of measurement at an arbitrary location

The second point mode of measurement is related to the Head Node of SynField cloud platform. The user should be at the location of the node and initiate the "*Generate Points*" procedure which will create measurement points distributed over homocentric circles with a radius of 5, 10, 25 and 50 meters from the node. On each radius, a total of 8 point measurements will be done with the angle shift of 45°. It is also left as an option for the user to choose the number of circles, radius of circles and number of points per circles. These canopy measurements will estimate the health status of the plant surrounding the nodes with respect to nutrients levels based on vegetation indices provided by the device. All the point measurements are automatically related to the appropriate node and uploaded to the SynField cloud. In this mode the user is able to save the data of measurements.

Figure 6: Head node homocentric circles mode of measurement

The last measurement mode is called *Automatic measuring* and is dedicated but not restricted to automatic methods of measurements related to the mobile robotic platform Clearpath robotics Husky A200, tractors or similar agriculture machinery. In this mode the user is able to take continuous measurements which means having the ability to start logging by pressing START and automatically store the georeferenced measurements (points) when pressing STOP. Automatic measurements are done every second and stored in the smartphone's memory until the STOP button is pressed. It is possible to continue with the measurements by pressing START again. An option to initiate a new set of measurements by pressing NEW PLACE is provided to the user as well. The map with the measurements results is created in real time while the logging is done. Each point measurement is visualized with different colours according to the Normalized Difference Vegetation Index value.

The menu item named Send data is sending the results of measurements to the SynField API. After the successful sending, the data will be permanently deleted from the smartphone's memory.

The data exchange between Android application and SynField cloud platform is achieved thought the Synfield API. The data format is the JSON one while its structure is shown below:

```
[
    {
            "red":"string",
            "green":"string",
            "blue":"string",
            "infrared":"string",
            "latitude":"string",
            "longitude":"string",
            "date":"dd.mm.yyyy. hh:mm:ss"
        }
]
```

More details about the integration are depicted in section 2.2.11.11.

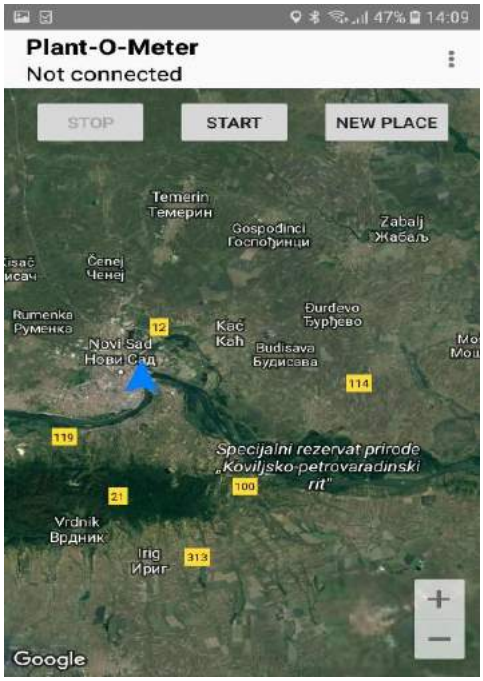


Figure 7: Continuous mode of measurements

The Android application can send on SynField API one or multiple measurements where each measurement is described by the above-mentioned attributes. Reflectance values of red, green, blue, infrared are presented by attributes “red”, “green”, “blue” and “infrared” respectively, while GPS coordinates are stated by “latitude” and “longitude” attributes. Finally the timestamp of measurement containing the date and time in the format <day.month.year hour:minute:second> is stated by “date” attribute.

The Settings option offers the user to choose a different language. For now, English and Serbian are available. The last option of the main menu is Exit which closes the application.

2.2 Synfield Platform

The Synfield platform is a cloud-based platform implemented by Synelixis. Given the Synfield platform, any user is able to register a field and setup the network of the installed Synfield nodes (devices) on it. Synelixis is the manufacturer of two types of nodes: the head node that is the gateway towards the Internet and the peripheral node that is able to communicate only with the head node that belongs in the same network. Figure 8 presents an indicative network of nodes (in the left side) and how these are connected. Typically, each field contains one head node and optionally, one or more peripheral nodes. The head-to-peripheral (and vice-versa) communication is performed using RF technologies while the head node is able to communicate (upload measurements, retrieve configuration and commands) with the Synfield platform (REST API) through WIFI or GPRS/GPS.

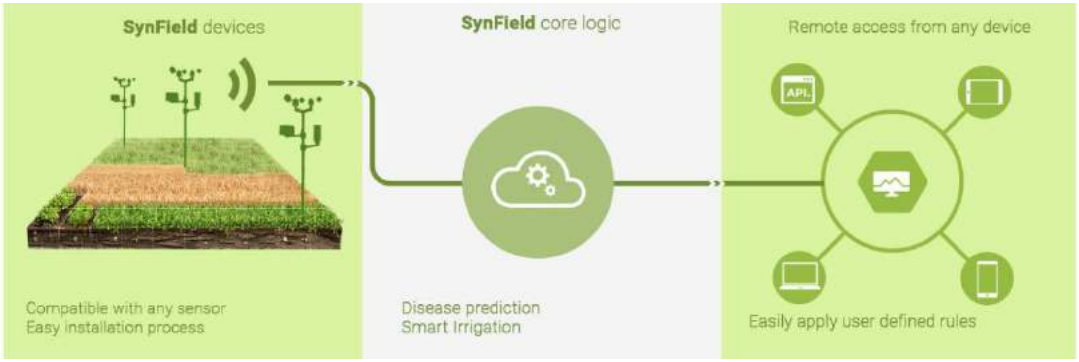


Figure 8: SynField ecosystem

Each node, head or peripheral one, can retrieve measures from one or more wired installed sensors. The exact number of the possible installed sensors on each node depends on its type, head versus peripheral, and its hardware version. Synfield nodes are compatible with several types of sensors; the only limitation is the interface of the sensors. Currently, ADC, PULSE, I2C and SERIAL sensors are fully supported using RJ connectors. Apart from the sensors, the Synfield nodes are compatible with a set of actuators. Also, in the frame of DIATOMIC project, the Synfield platform has been extended so as to support the manipulation of external devices such as the Plant-O-Meter device (provided by Biosense). As external devices we consider the devices that are not compatible with or not plugged in the Synfield nodes.

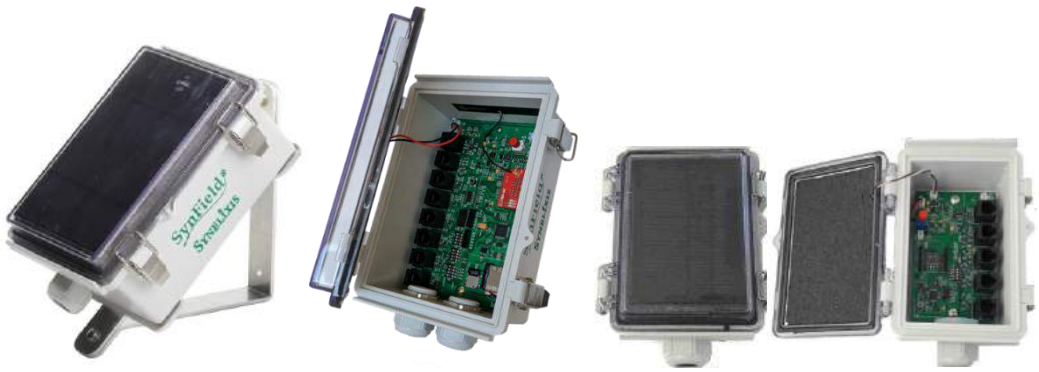


Figure 9: SynField Head Node (left) and Peripheral Node (right)

As shown in Figure 9, SynField nodes are compact self-contained. They feature a Photovoltaic panel in order to be autonomous and a Regulated Lead-Acid (Lithium) battery to operate for days even without being charged. Moreover, the electronic circuit is located in a waterproof box (IP-67), which protects the system from rain, snow and hail.

Figure 10 provides a typical SynField installation, where one headnote and multiple peripherals in ad-hoc networking architecture are installed.

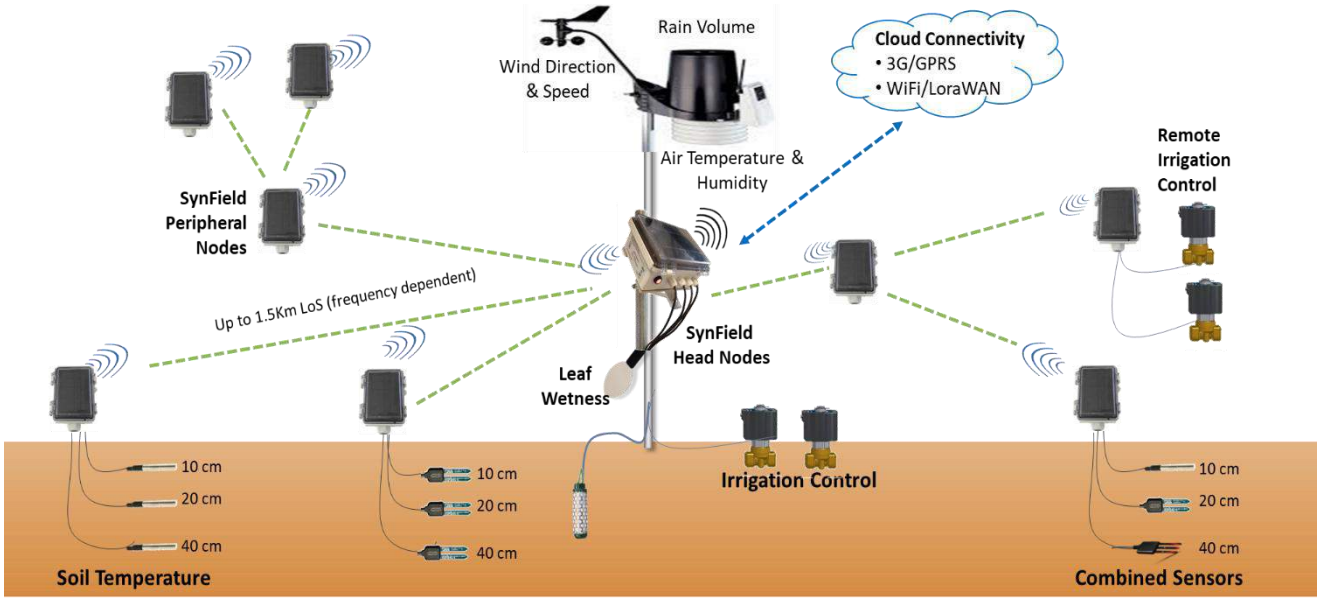


Figure 10: Typical SynField installation

2.2.1 Create a new account on the Synfield Account

The first step that a user must do is to create an account. This is offered by the DIATOMIC SynField platform instance free of charge for all users (both the beneficiaries accepted from the DIATOMIC Open Calls and for any interested party). To create the account, follow the link http://83.235.169.221:9990/en/register/ and type your personal information, check the box “I’m not a robot”, accept the terms of service and proceed on the account registration.

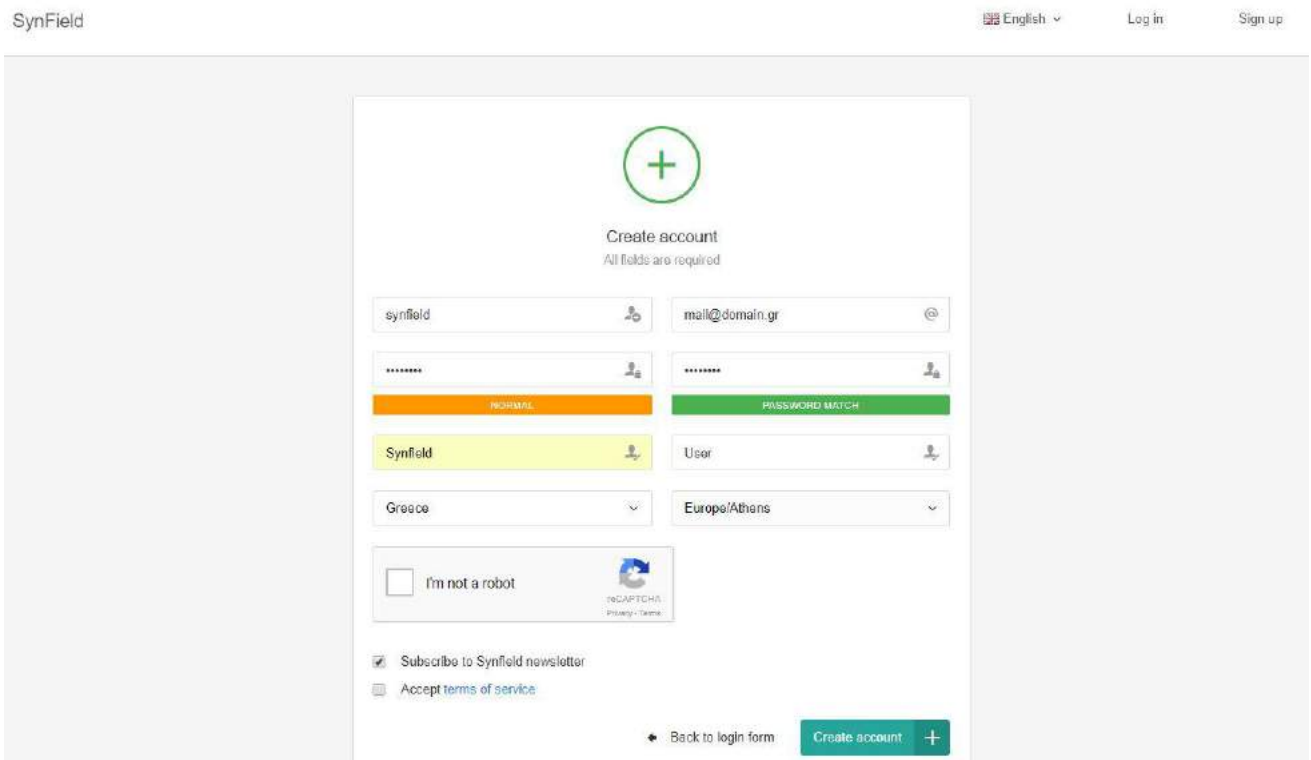


Figure 11: Registration in the Synfield platform

2.2.2 Account upgrade

When a user accesses for the first time the Synfield platform, they are assigned a free plan (no fee). That means that the user can not register any Synfield device in the platform, but just receives information from selected already available nodes.

Whenever needed, the beneficiaries of the DIATOMIC project can upgrade their SynField account (without fee) to a plan that allows the registration of the Synfield devices (including sensors and actuators) in the platform, use the API and enable a set of services that SynField platform provides. Contact us at synfield@synelixis.com in order to upgrade your account for free. After the completion of the DIATOMIC project, any upgrade will expire.

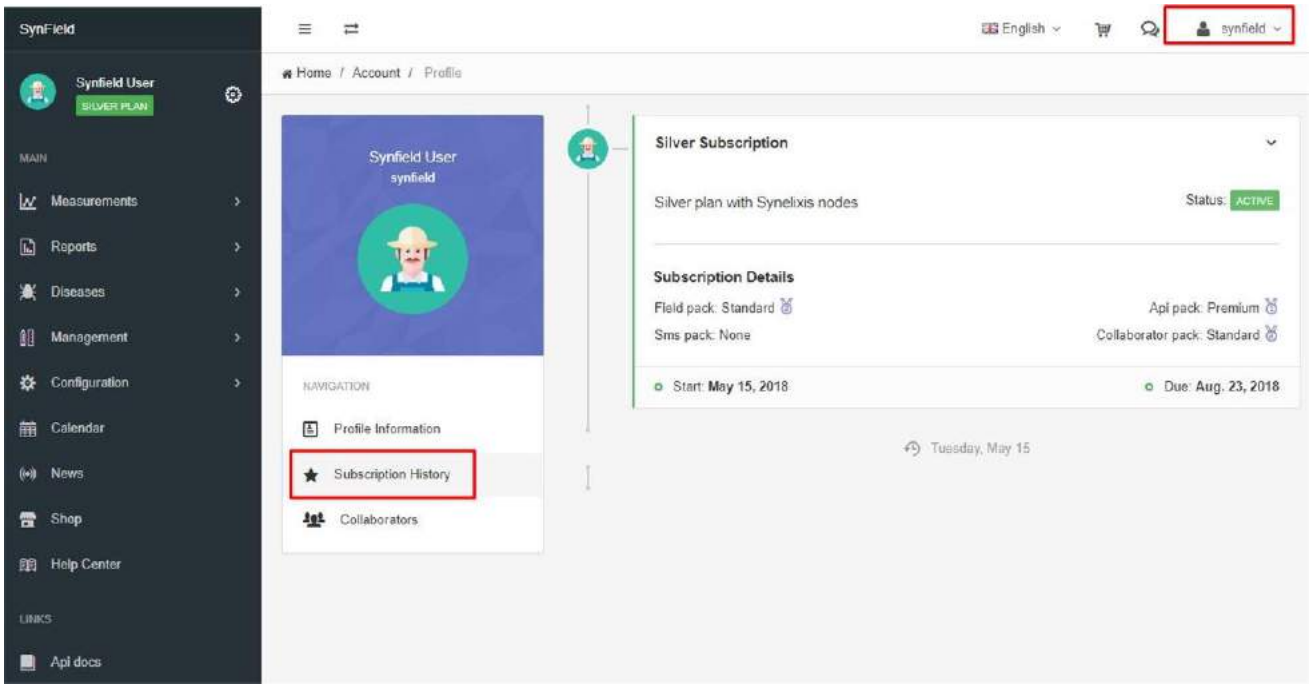


Figure 12: History of user subscriptions

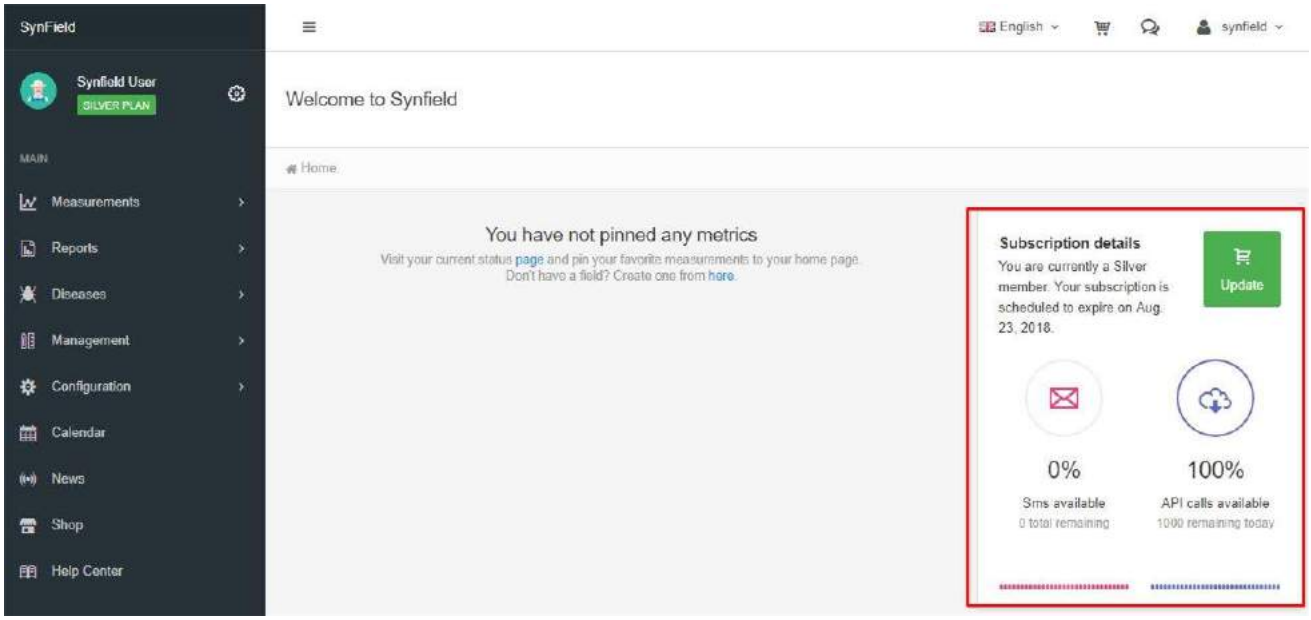


Figure 13: Available SMS and remaining API calls

2.2.3 Register a field

For the collection and the storage of any kind of measurements provided from Synfield (head and peripheral) nodes and external devices (e.g. Plant-O-Meter), their registration in the Synfield platform is necessary. Within DIATOMIC, the SynField nodes will be already installed, thus no further action will be needed. For completion, we also provide the relevant information.

The registration of a new field takes place in the Configuration>Fields page. There are two options to register a field including one or more Synfield nodes in the SynField platform. The first option is to configure the Synfield devices (that will be located in the area of interest) through the SynControl android application (https://play.google.com/store/apps/details?id=com.synelixis.SynControl) and upload the configuration file in the platform through the button *Import configuration*.

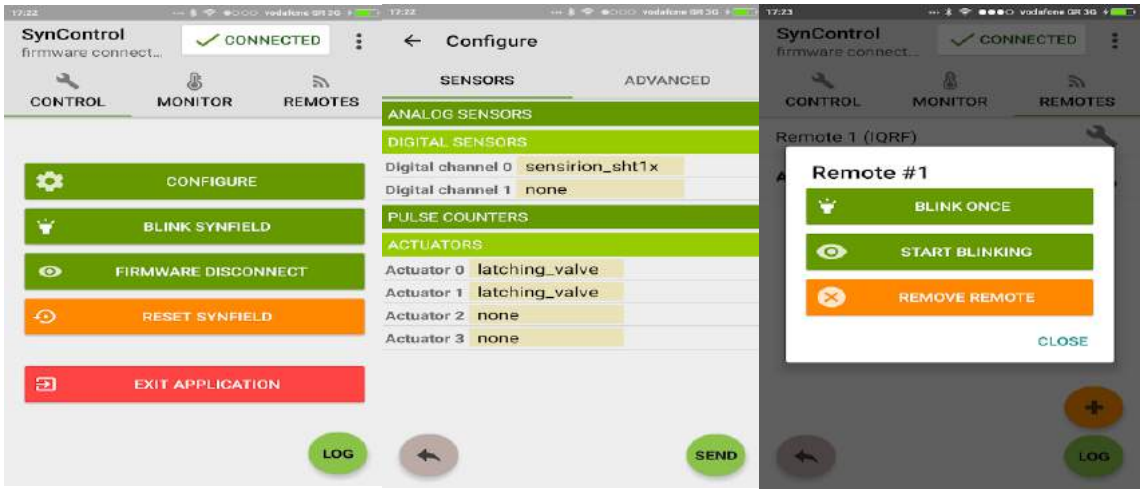


Figure 14: Selected SynControl Android app screen

The second option is to register by clicking on the button *New Field* as shown in Figure 15. The user needs to set the name of the field, its coordinates and the type of the crop as depicted in Figure 16.

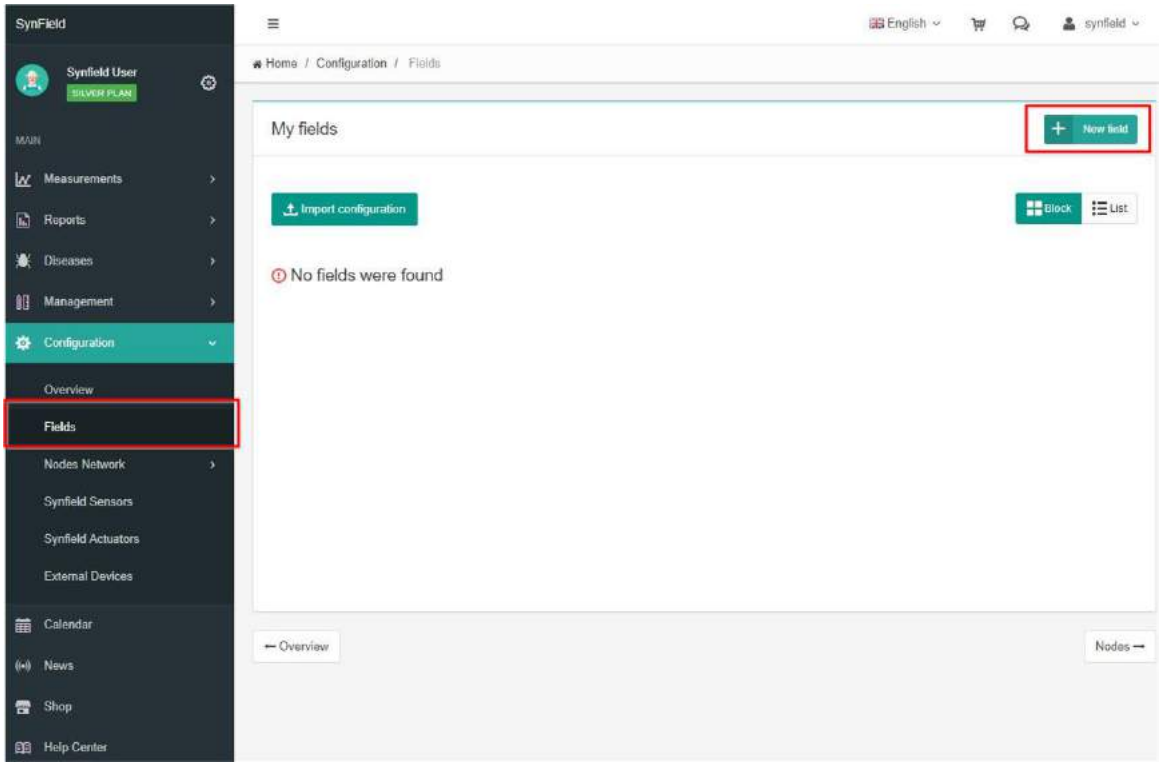


Figure 15: Add a new field in your account

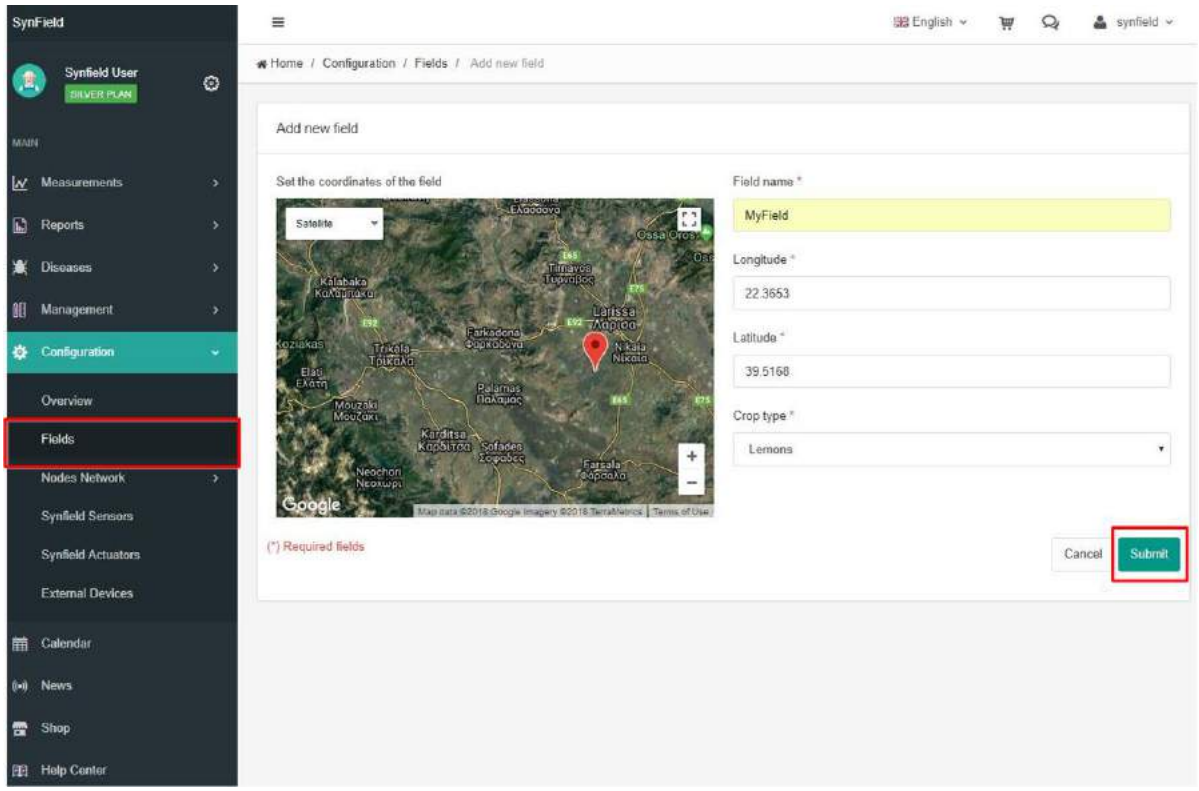


Figure 16: Register in the platform a new field

After the successful registration of the field, the field management as well as the registration of Synfield head nodes can be done as shown in Figure 17.

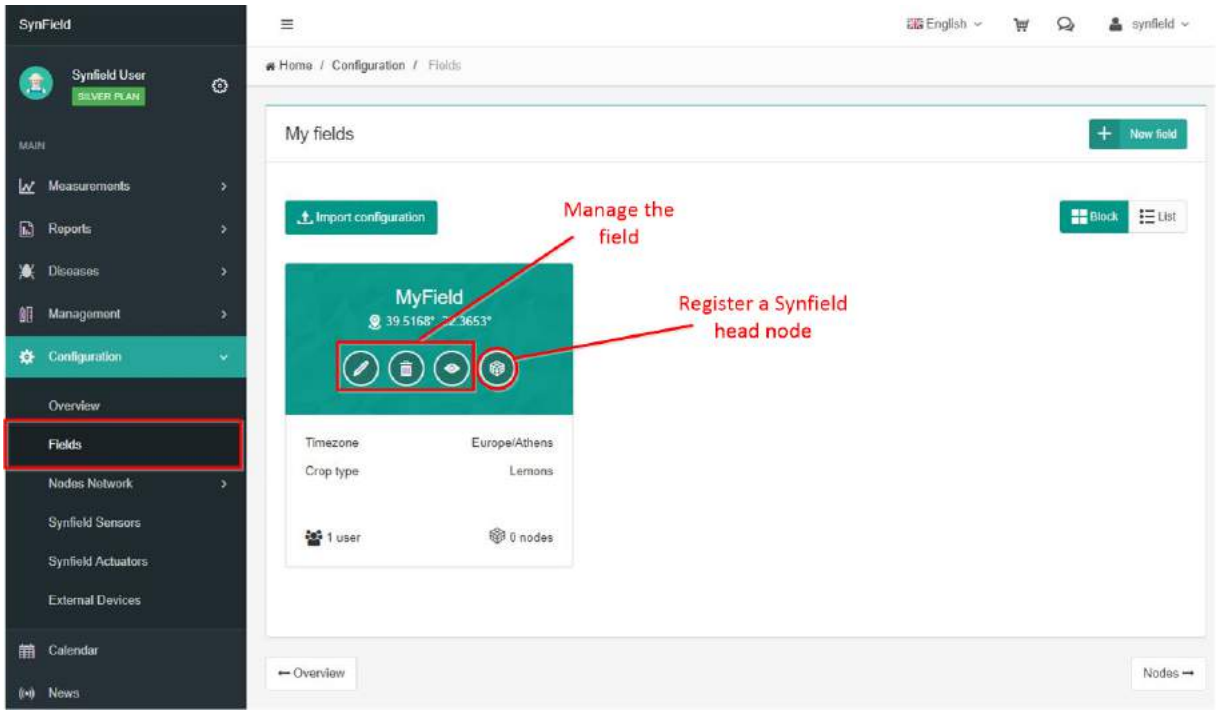


Figure 17: Manage your fields

2.2.4 Define your network of Synfield devices

Navigate in the *Configuration>Nodes Network>Head nodes* to register a new head node assigning in a field or manage the existing ones. Click on the *New node* button to process on the registration of the head node (see Figure 18).

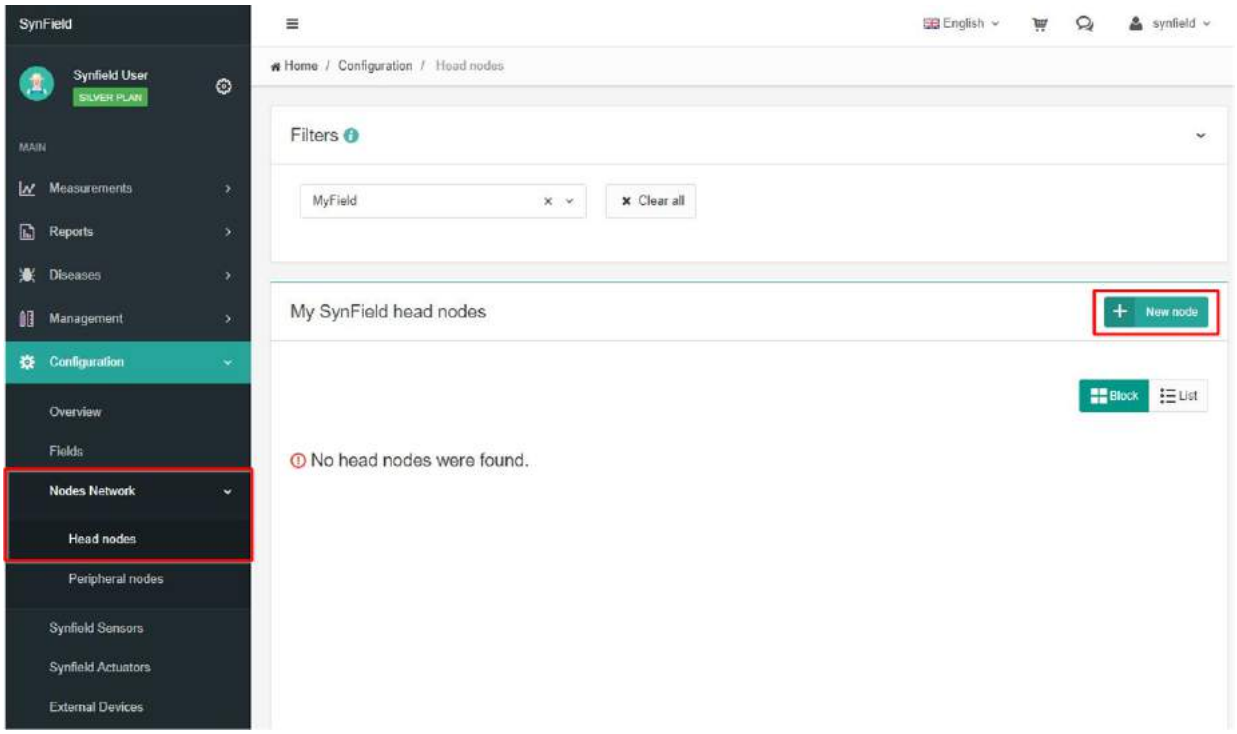


Figure 18: Register a Synfield head node

To complete the registration, fill in the required details in the wizard. In the tab Description, select the field of interest, type the serial number of the node (provided from Synelixis), select the hardware version of the node and type other details as shown in Figure 19.

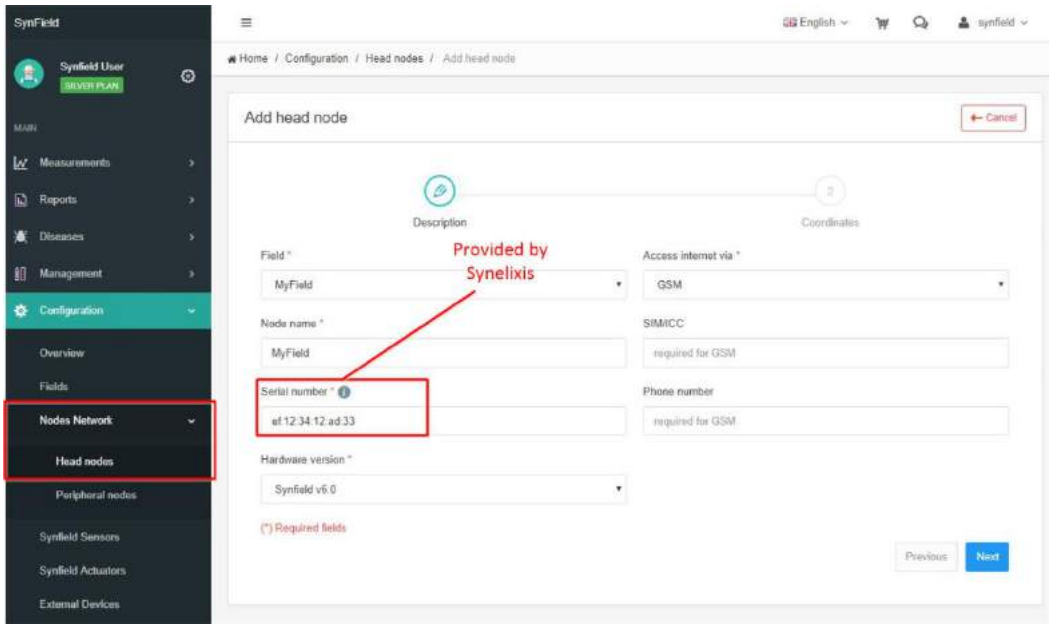


Figure 19: Step 1- Type the basic details of the Synfield head node

In the tab Coordinates, define the exact coordinates in which the head node has been installed and press submit (Figure 20). Repeat the aforementioned steps to register more head nodes.

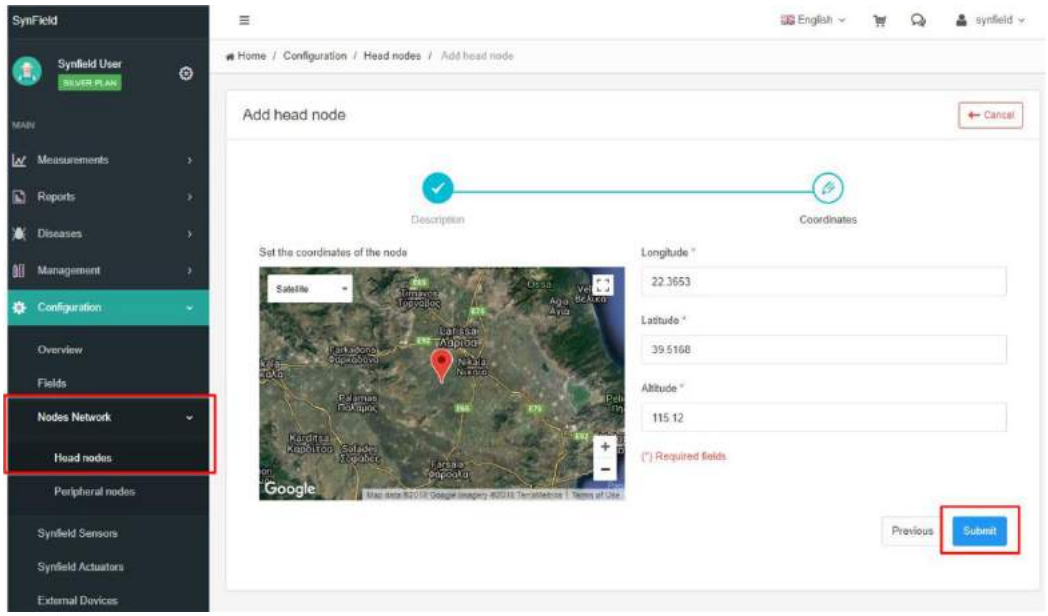


Figure 20: Step 2 - Modify the head node's coordinates

In analogous manner, the user can register a Synfield peripheral node as part of the network of the Synfield nodes. The only difference here is the fact that the user needs to associate the peripheral node with one of the existing head nodes (see Figure 21). In this way, the user defines a network of Synfield devices that covers the area of interest. The head node is the representer of the network since it is the one that has access to the internet.

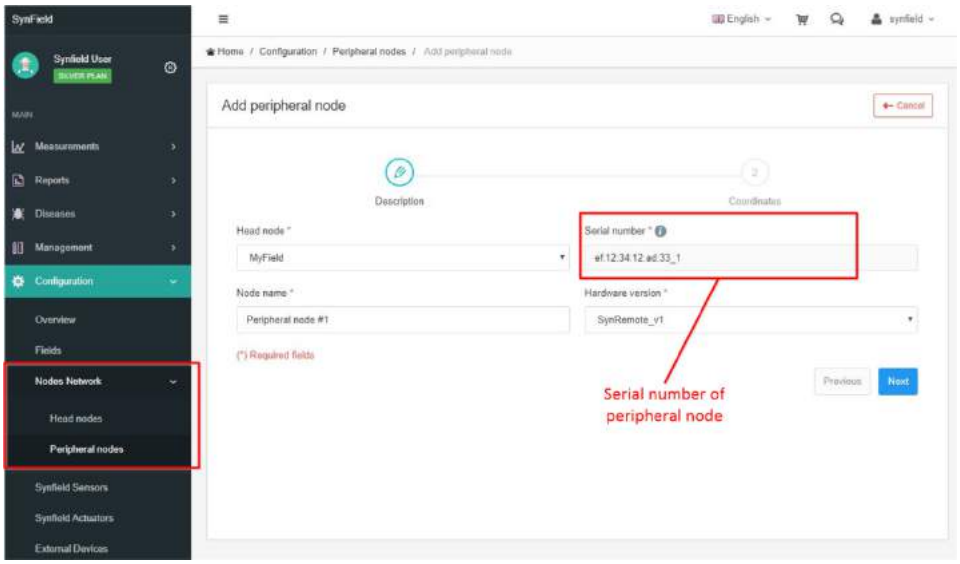


Figure 21: Register a peripheral node as part a Synfield network

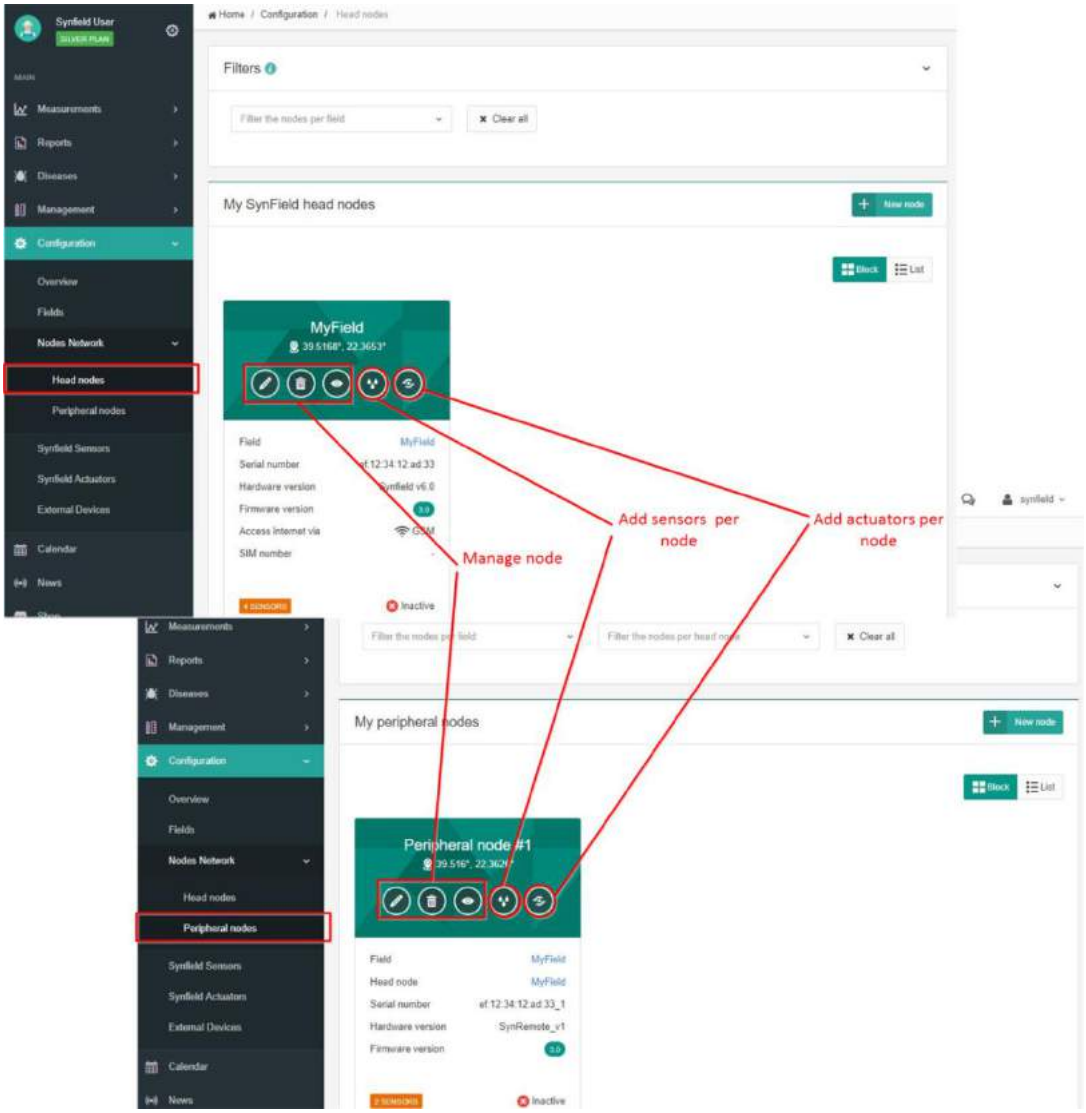


Figure 22: Manage the nodes in similar way

After the registration of the head and peripheral nodes in the platform, the user is able to manage (view, edit, delete) them as depicted in Figure 22.

2.2.5 Register sensors in existing Synfield nodes

The next step includes the registration of the plugged-in sensors in each Synfield node. To do this, navigate on the *Configuration>Synfield sensors* area and press the button *New sensor* (Figure 23). By default, a set of sensors, that reflects the health of the node (battery and so on), is registered automatically.

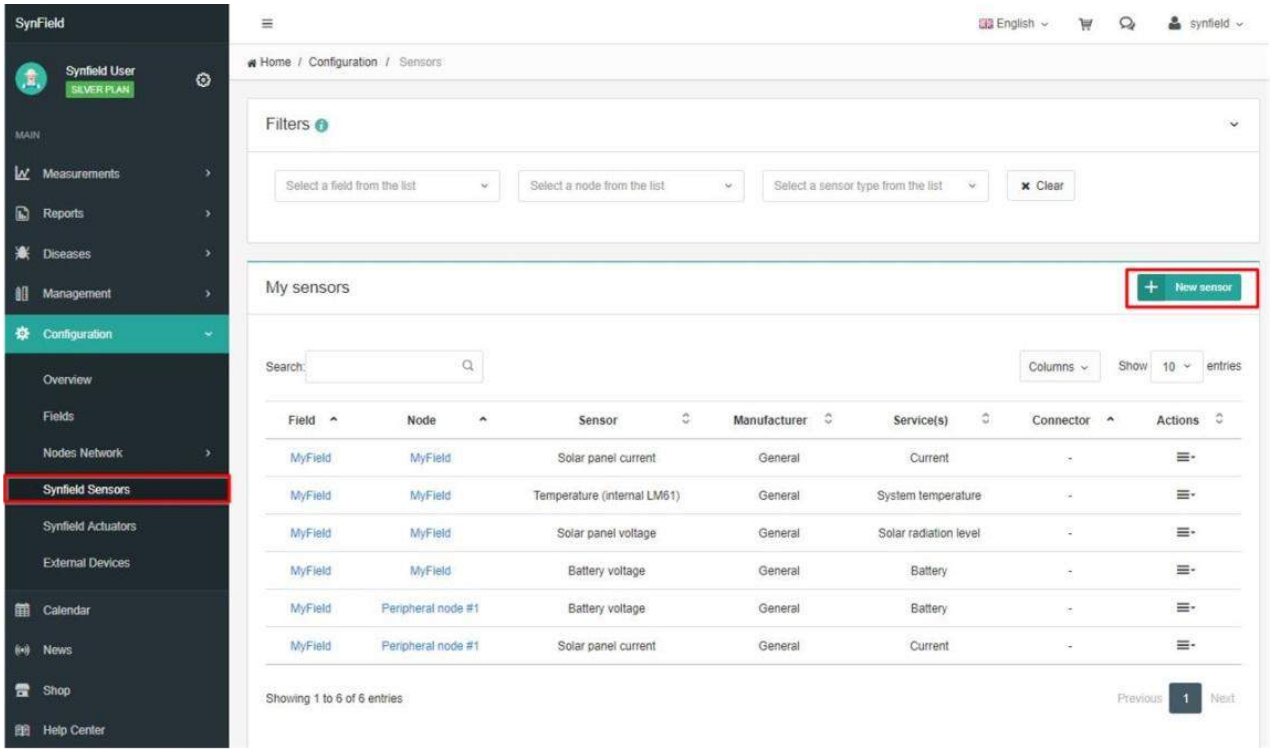


Figure 23: Add a new sensor in the head node

The registration of a sensor includes the selection of its type, the associated (head or peripheral) node, the connector in which the sensor is plugged and optionally the adjustment of some metadata (e.g. soil type and sensor depth in case of soil moisture sensor) as depicted in Figure 24. Figure 25 depicts the registered sensors with further details such as the sensing services that each sensor measures; the management actions of the sensors include view, edit, rename sensing service and delete while the filtering of sensors per field, node or type is available.

For the agrifood experiment, the collection of sensing services such as air temperature, relative humidity, soil moisture, wind speed, wind direction, rainfall, flow meter will be meaningful.

2.2.6 Register actuators in existing Synfield nodes

The next step includes the registration of the plugged-in actuators in each Synfield node. To do this, navigate on the *Configuration>Synfield Actuators* area and press the button *New actuator* (Figure 26). The registration of an actuator includes the selection of its type, the associated (head or peripheral) node, the connector in which the actuator is plugged on (see Figure 27).Figure 28 depicts the registered actuators with further; the management actions of the actuators include view, edit, rename service and delete while the filtering of actuators per field, node or type is available.

A kind of actuator that will provide added value in the experiment is the solenoid valve.

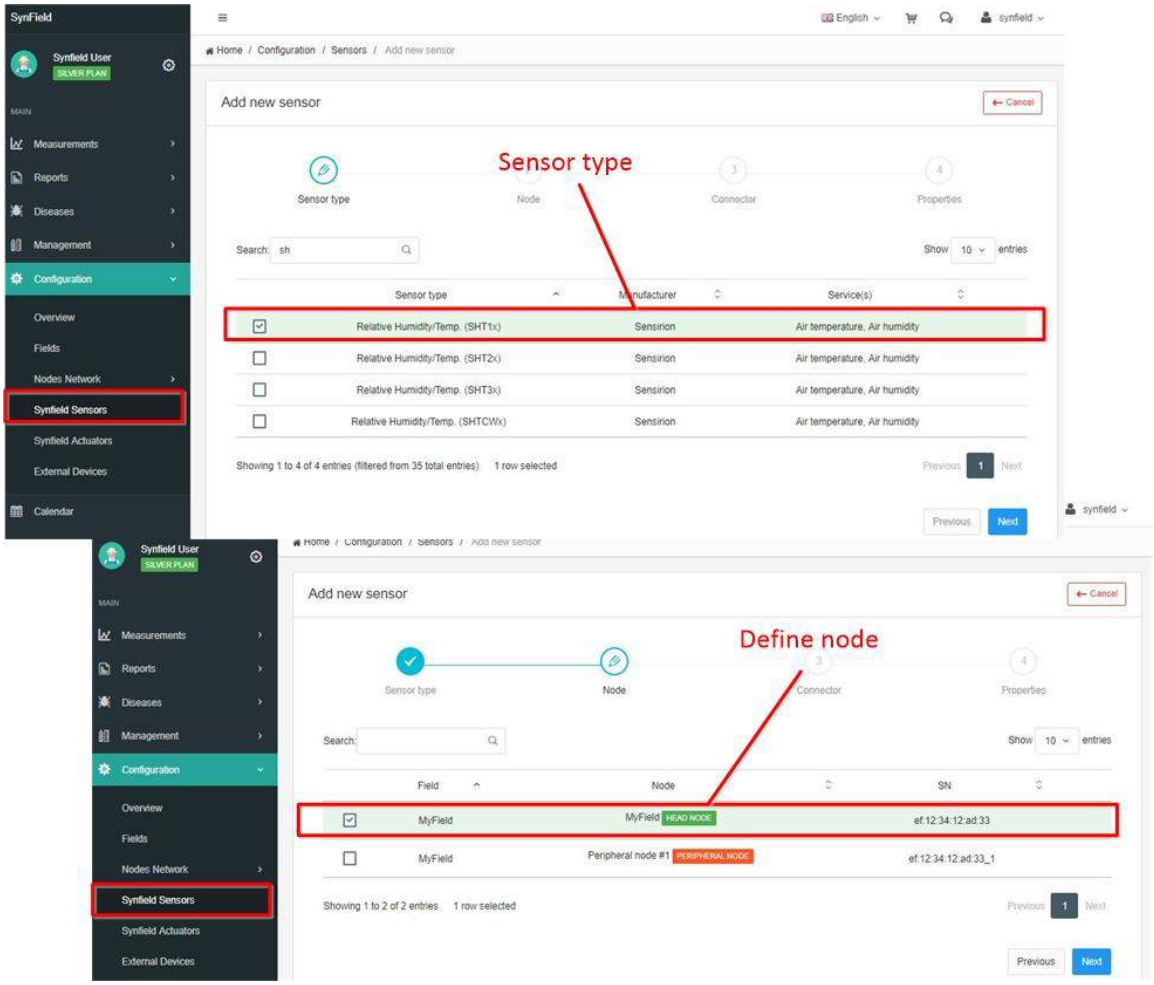


Figure 24: Follow the wizard to complete the sensor's registration

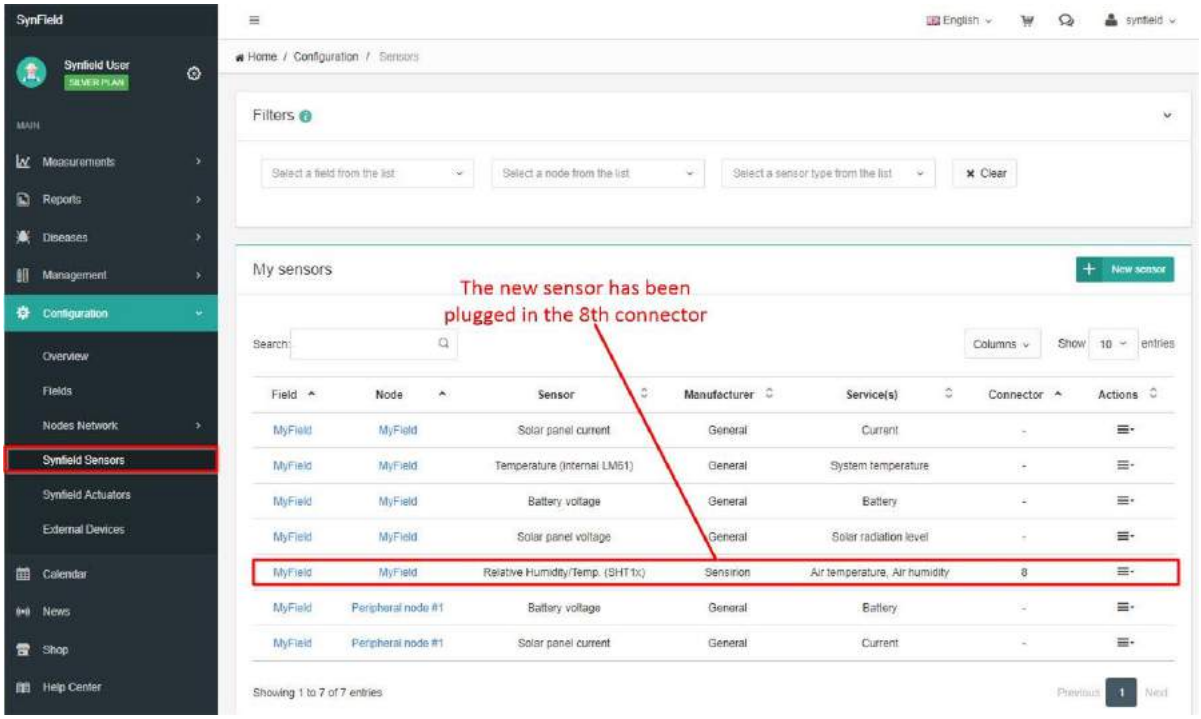


Figure 25: Manage the registered sensors

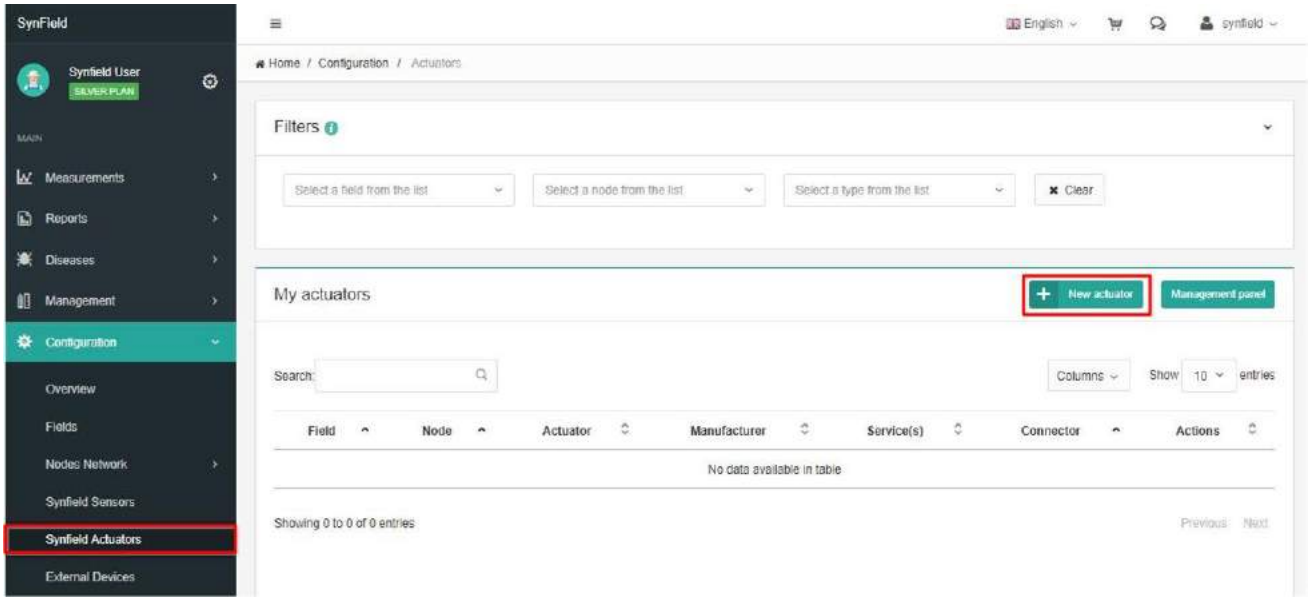


Figure 26: Add a new actuator in a Synfield node

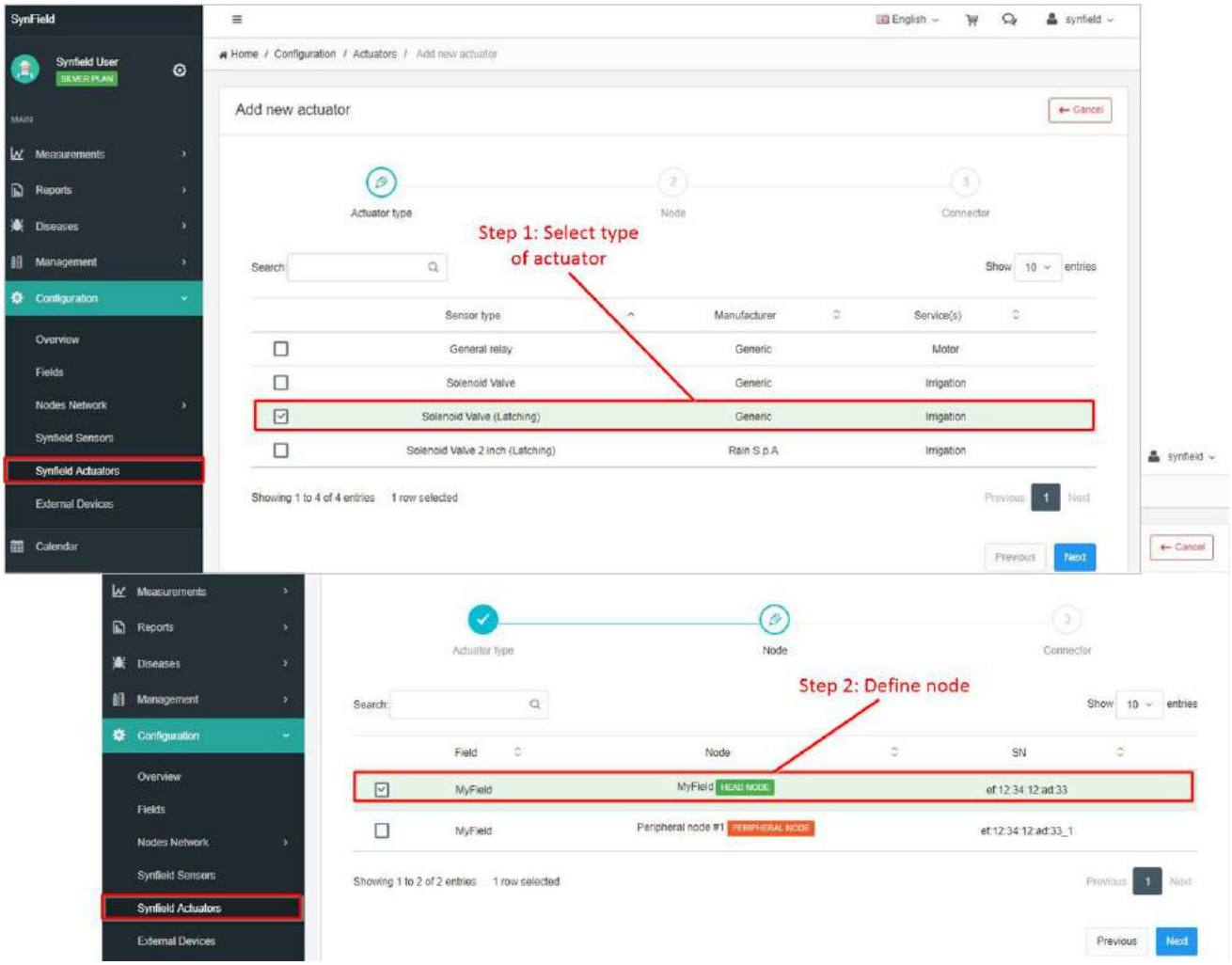


Figure 27: Follow the wizard to register a new actuator

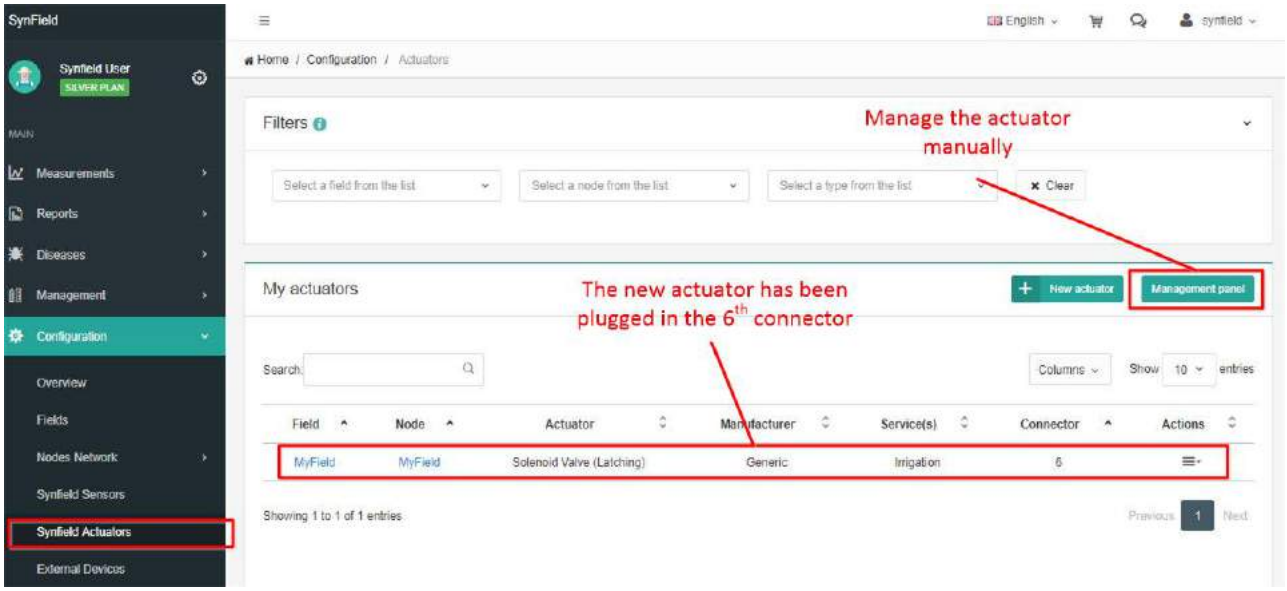


Figure 28: View the registered actuators

2.2.7 Register external devices

Synfield platform supports the management of external devices by clicking on the tab *External Devices* in the left menu. A new device can be registered in the platform by clicking on the *Add device* button.

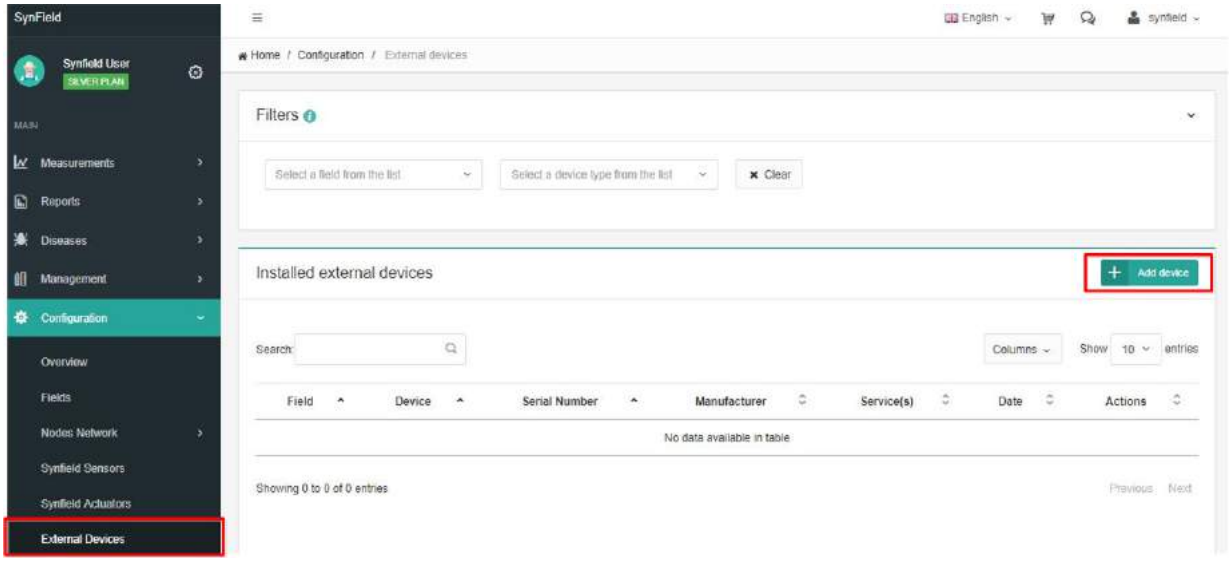


Figure 29: Add a new external device (e.g. Plant-O-Meter)

A device can be registered easily and rapidly by selecting the field in which the device is located or planning to be installed, the type of the device and by typing the serial or product number as shown in Figure 30.

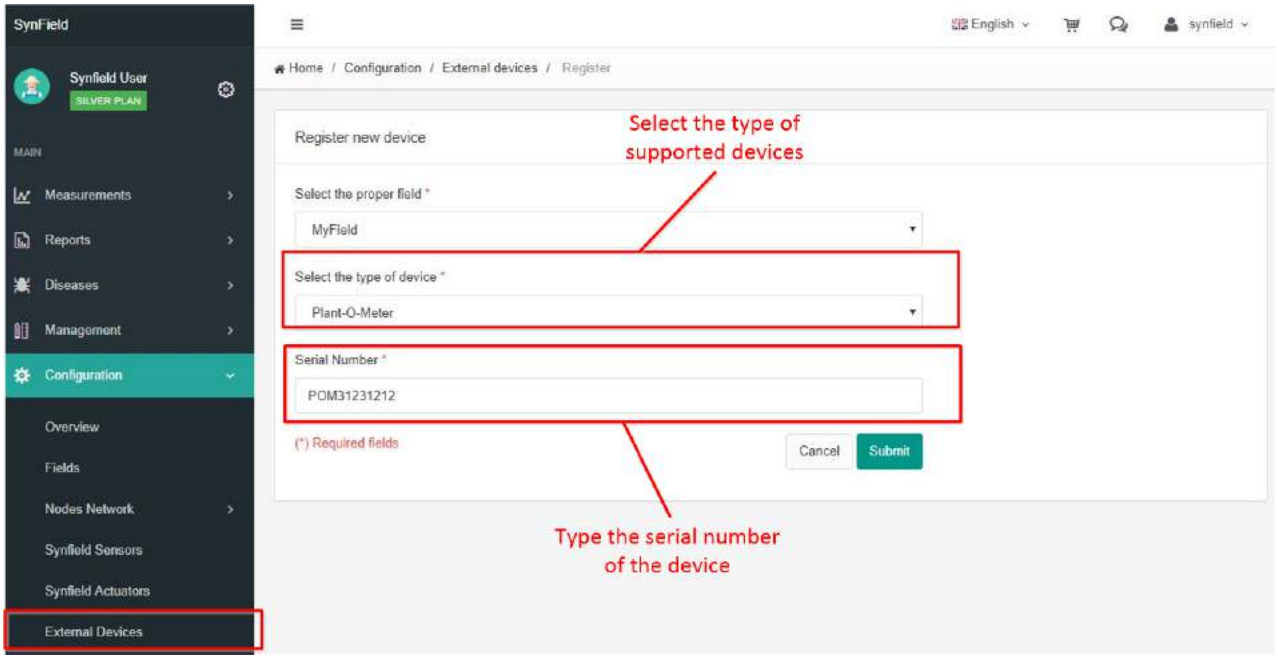


Figure 30: Register the Plant-O-Meter in Synfield platform

After the successful registration of the device, more details per device are depicted in the management page (see Figure 31).The preview, edit and deletion actions of each device are supported through the Actions drop-down list.

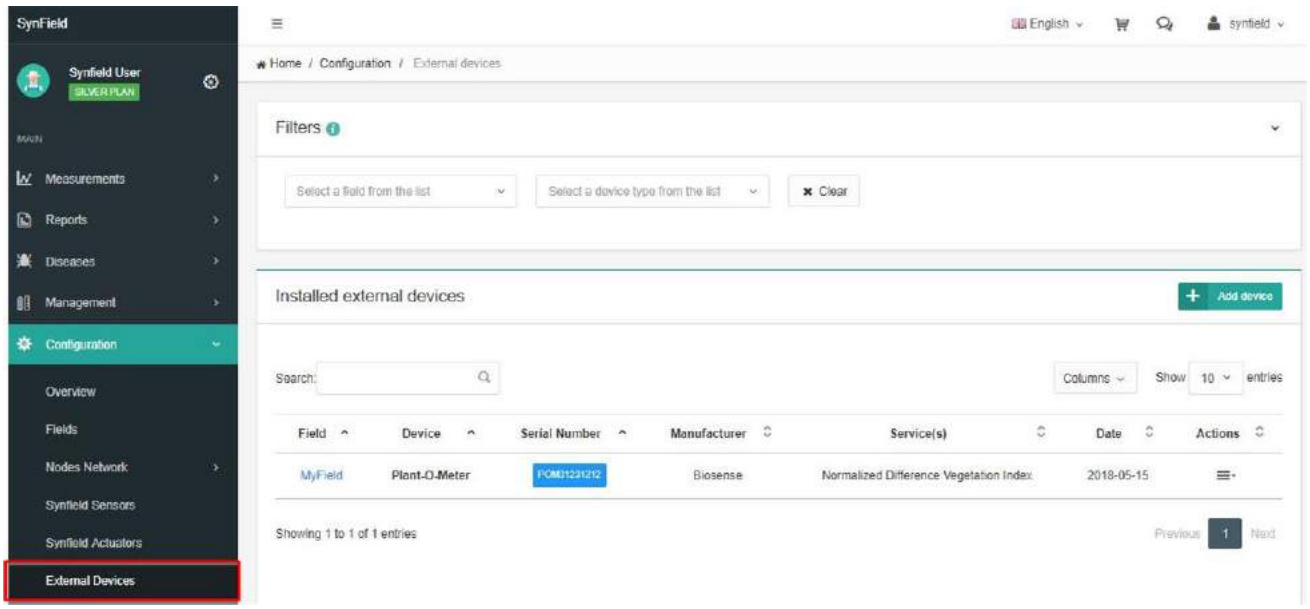


Figure 31: Manage the registered devices

2.2.8 Data visualization

SynField allows for viewing of the current measurements of all the sensors attached on the user's nodes from the *Measurements> Current status* on the left menu. The node can be chosen on the left of the page and the last recorded values will appear on the right in separate frames per origin: sensors' measurements, calculated metrics, actuators, internal system measurements. Normal values appear green while red colour denotes an issue.

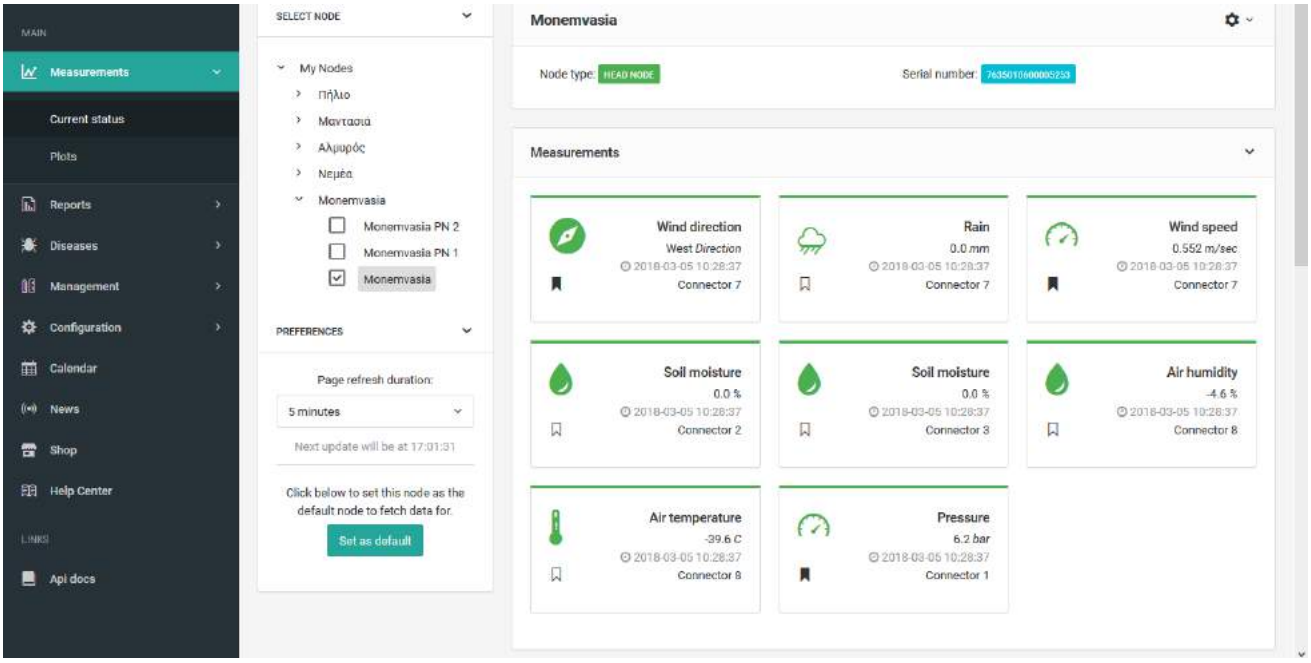


Figure 32: Current status of sensing services in Synfield node

Each panel carries specific information: the name of the service, the value, the date and timestamp of the recorded value, the connector the sensor is attached to the node. Each service is depicted by a certain icon found in the left top corner of the panel and on the little flag icon each service may be pinned on the user's dashboard.

SynField also carries a powerful visualization tool in the *Measurements>Plots* page. Clicking on the *Add new* button, the user is able to compose a plot of interest by combining sensing services that are measured from devices installed in the user's fields.

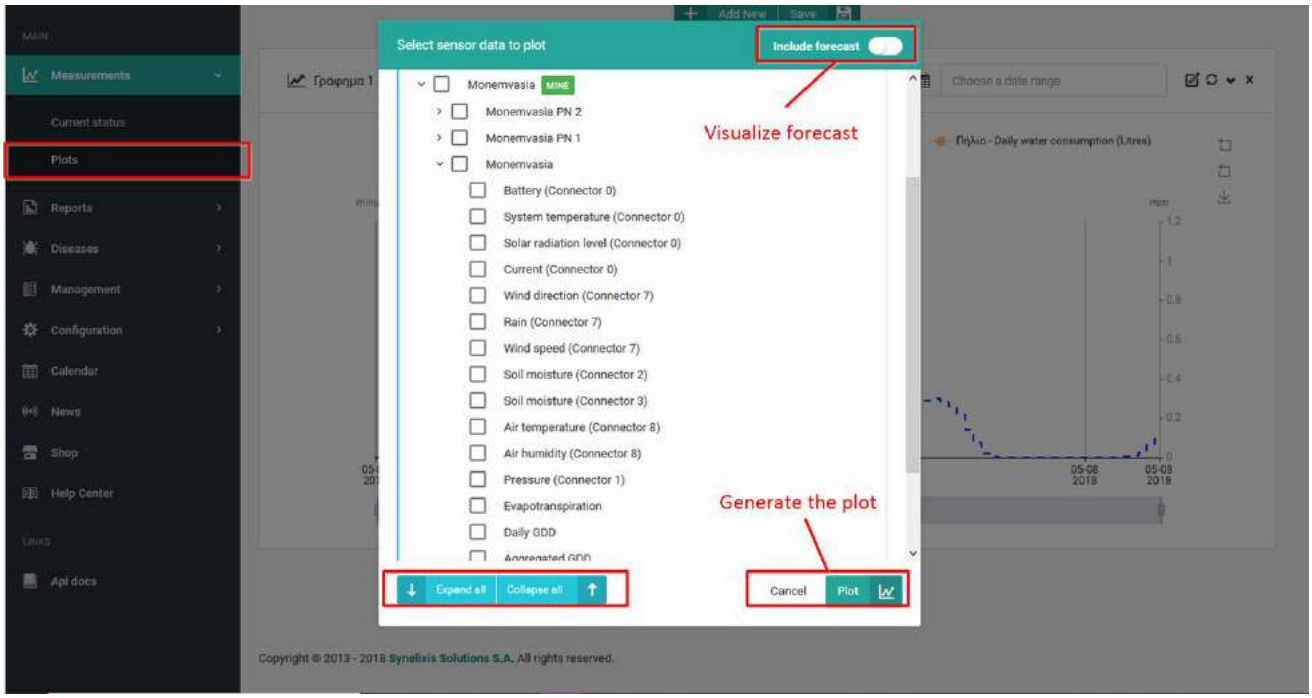


Figure 33: Select sensing services to be visualized

As soon as the node of interest is expanded all the measurements referring to it appear. SynField also allows for viewing a forecast of the corresponding weather conditions by activating the *Include forecast* option on the top right corner of the pop-up window. The Plot button on the bottom right corner of the pop-up window creates the new plot.

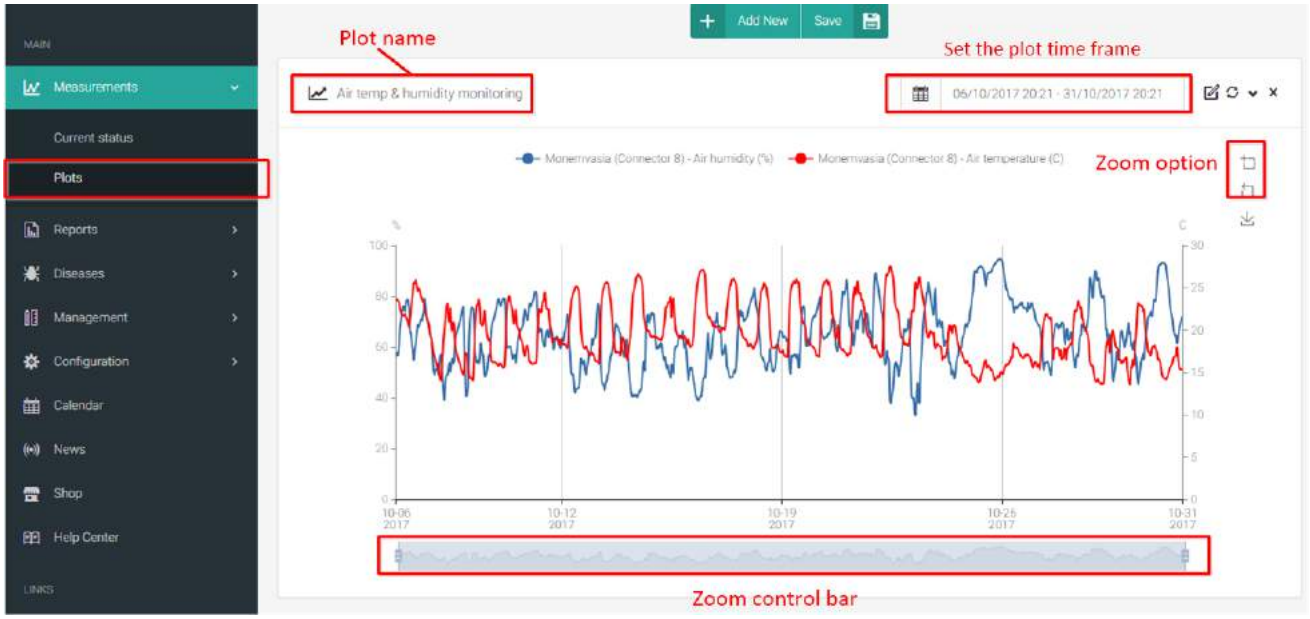


Figure 34: Visualize air temperature and relative humidity in one plot

2.2.9 Diseases

SynField provides disease prediction for numerous crops though the monitoring of soil characteristics, leaf moisture and local weather conditions and their correlation to the type of crop and stage of growth. In the Disease Repository there is an extensive list of diseases SynField supports. Each appears in a separate panel where apart from a brief description, by clicking on *Affected crop types*, a list of all the crops affected by the disease appears.

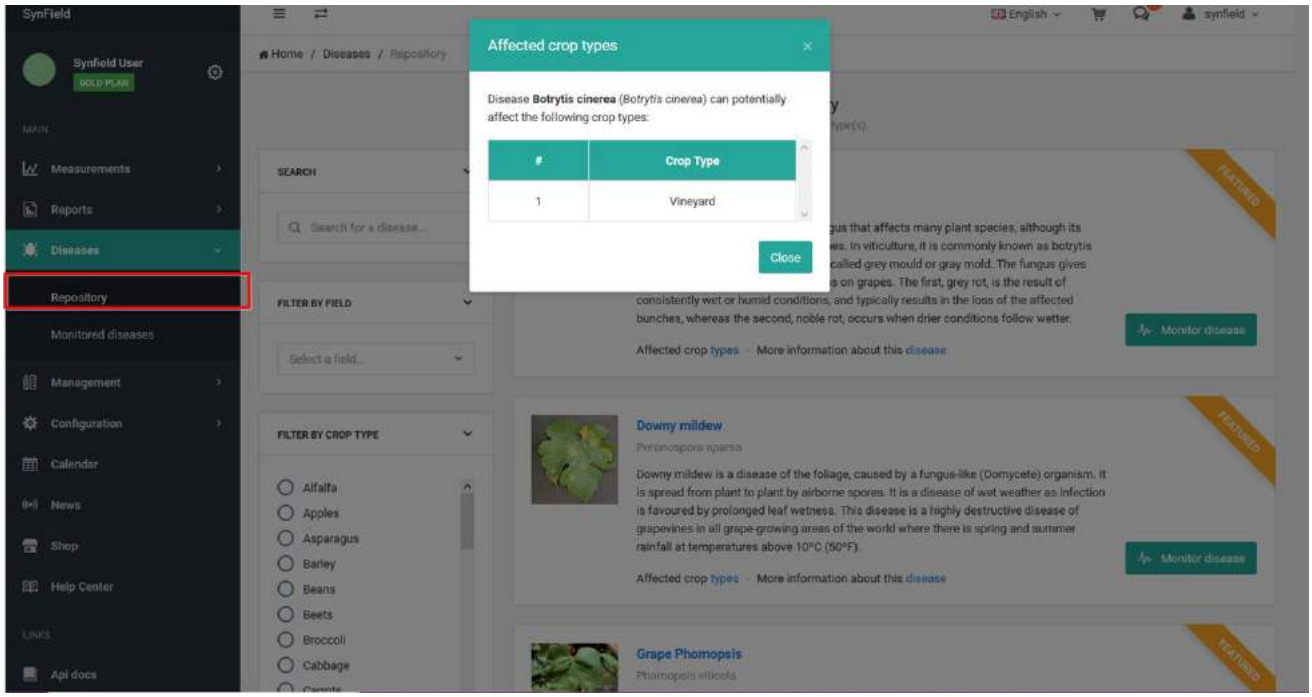


Figure 35: Affected crop types by specific disease

There is a filtering tool which allows search in the repository per species or per field.

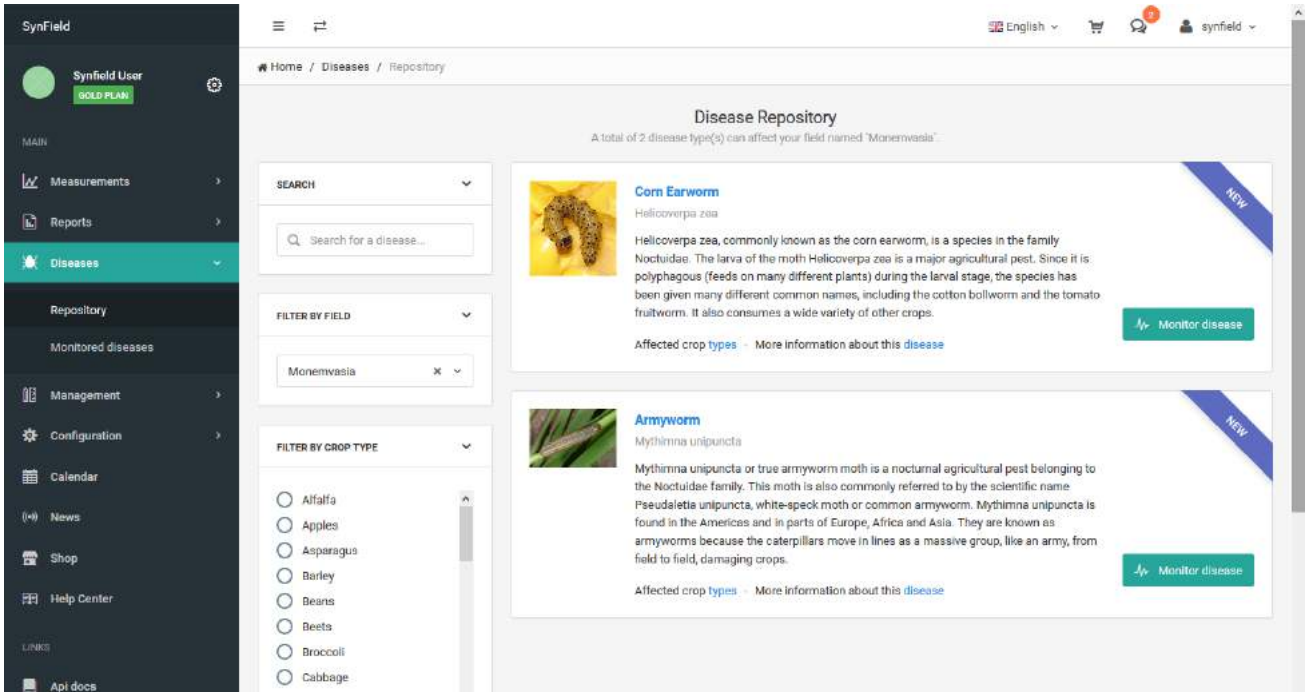


Figure 36: Diseases Repository

The *Monitor disease* button starts the procedure of adding the prediction of the chosen disease to your field. Clicking *Select* the page where all the necessary information to be defined appears. When all is set the Monitor button creates the disease prediction.

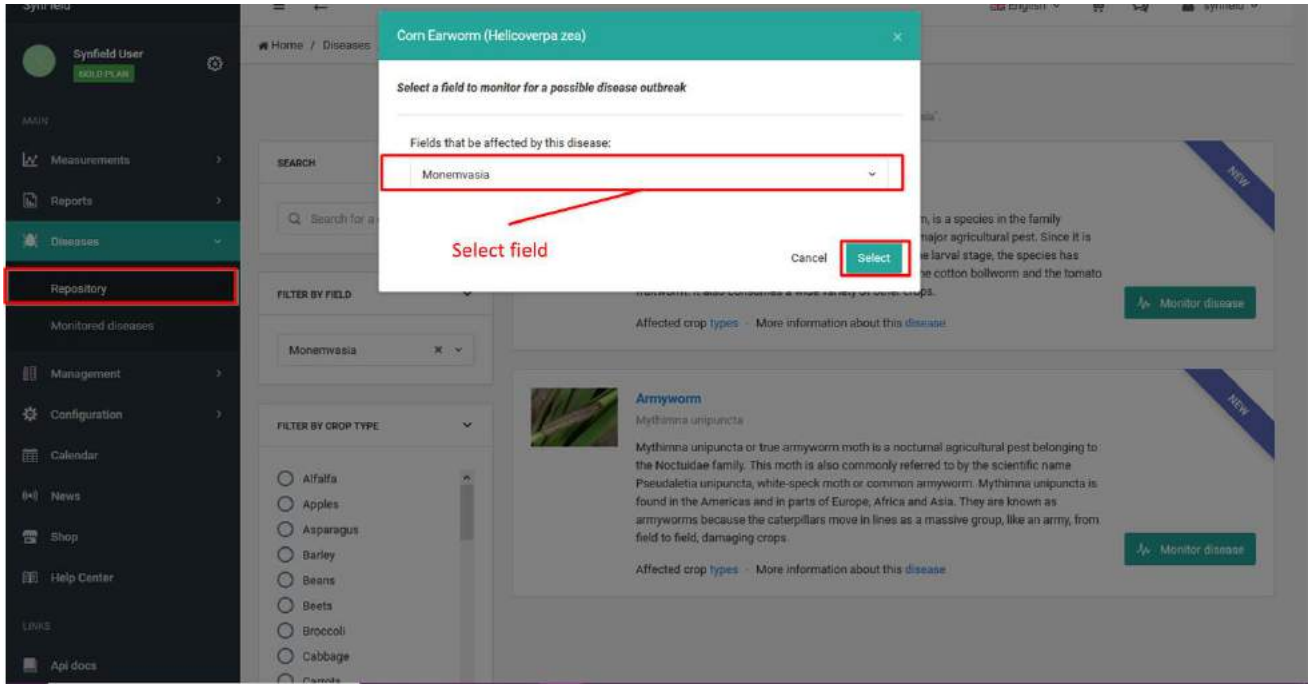


Figure 37: Initialising disease prediction in a field

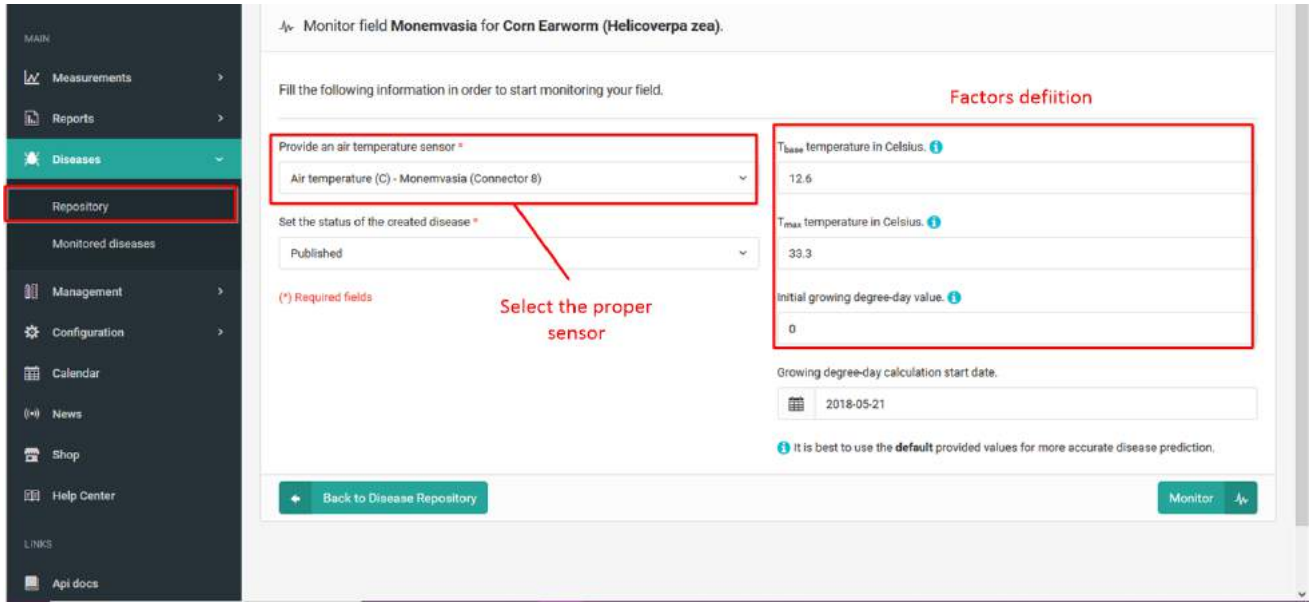


Figure 38: Setting disease prediction details

2.2.10 Smart and simple rules

SynField also allows the creation of rules, smart or custom ones, exploiting measurements of the plugged sensors as well as plugged actuators. These rules can manage an actuator, enable alerts/notifications or calculate a kind of metadata (e.g. evapotranspiration, GDD and so on).

In the *Rules repository>Smart rules* are SynField's predefined rules. Focusing on the smart irrigation, the user is able to create a smart irrigation rule by clicking on *My smart rules* button on the Irrigation panel. SynField then creates a draft version of the smart rule which you can be adapted to the user's needs by clicking *Edit*. Figure 39 up to Figure 43 depicts the completion of the irrigation rule based on user's preferences.

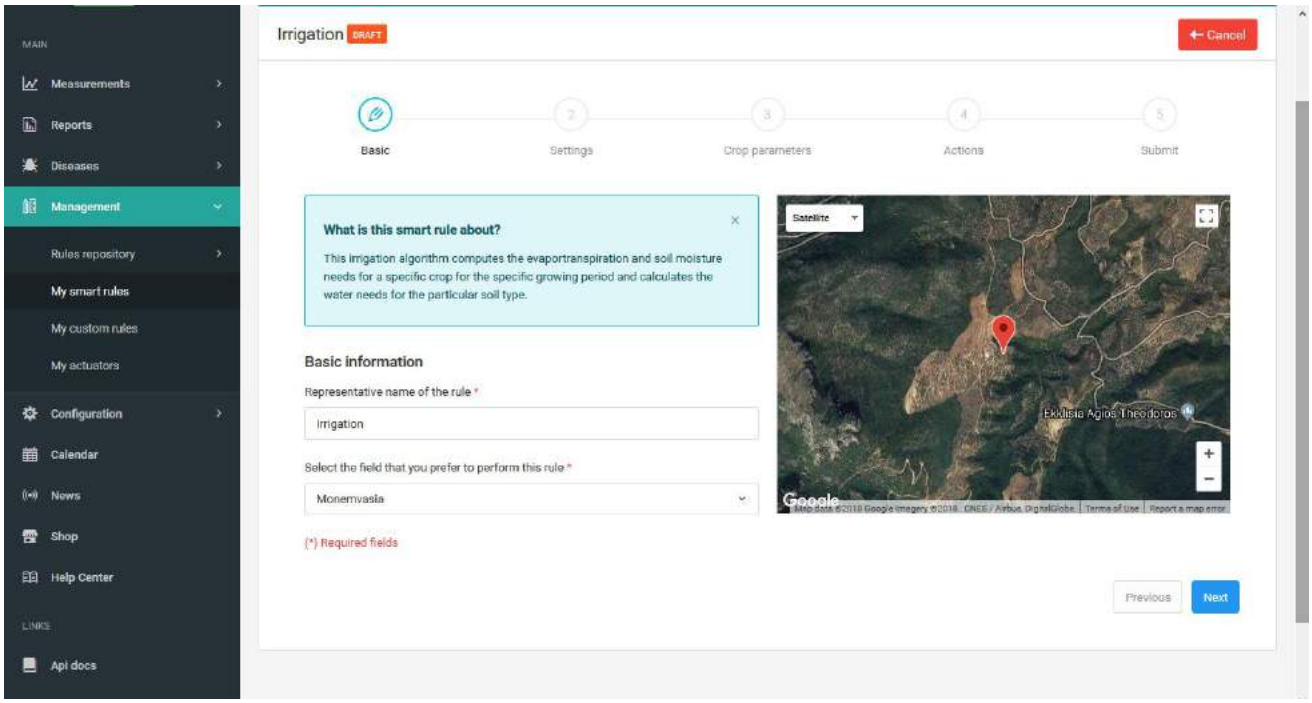


Figure 39: Basic step of irrigation rule

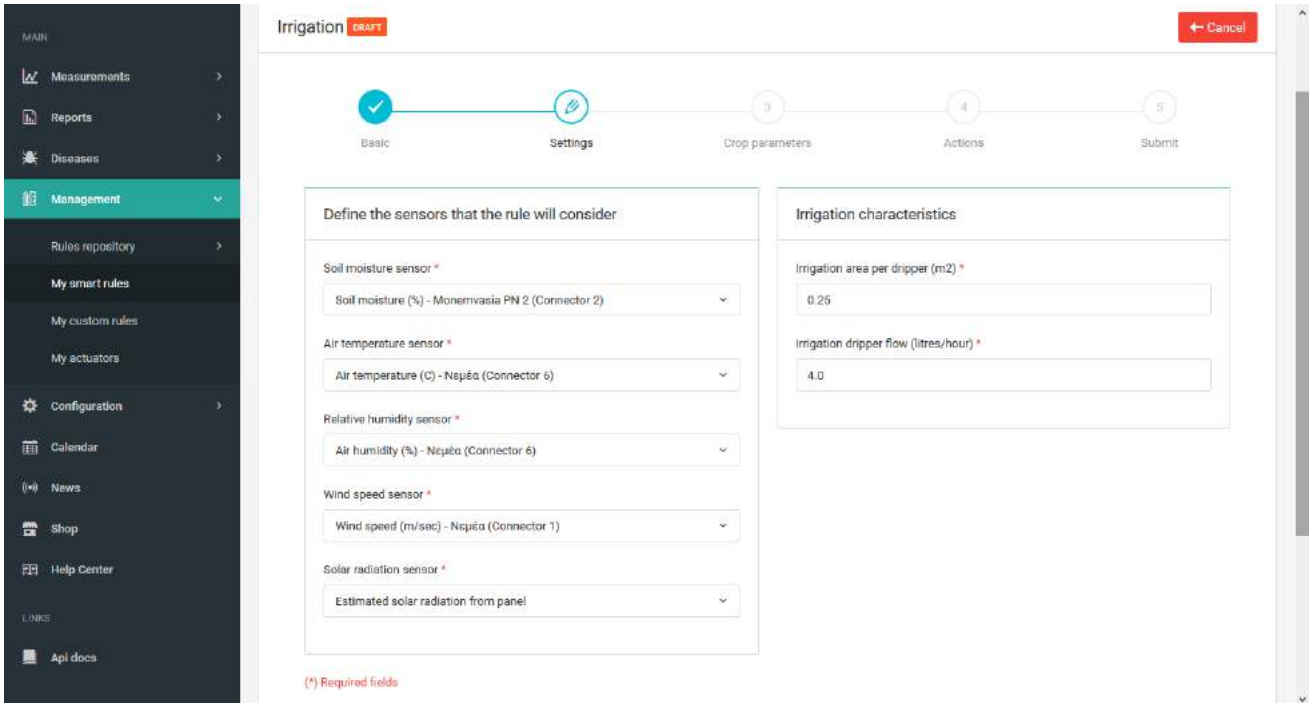


Figure 40: Define the metrics, necessary for the irrigation rule

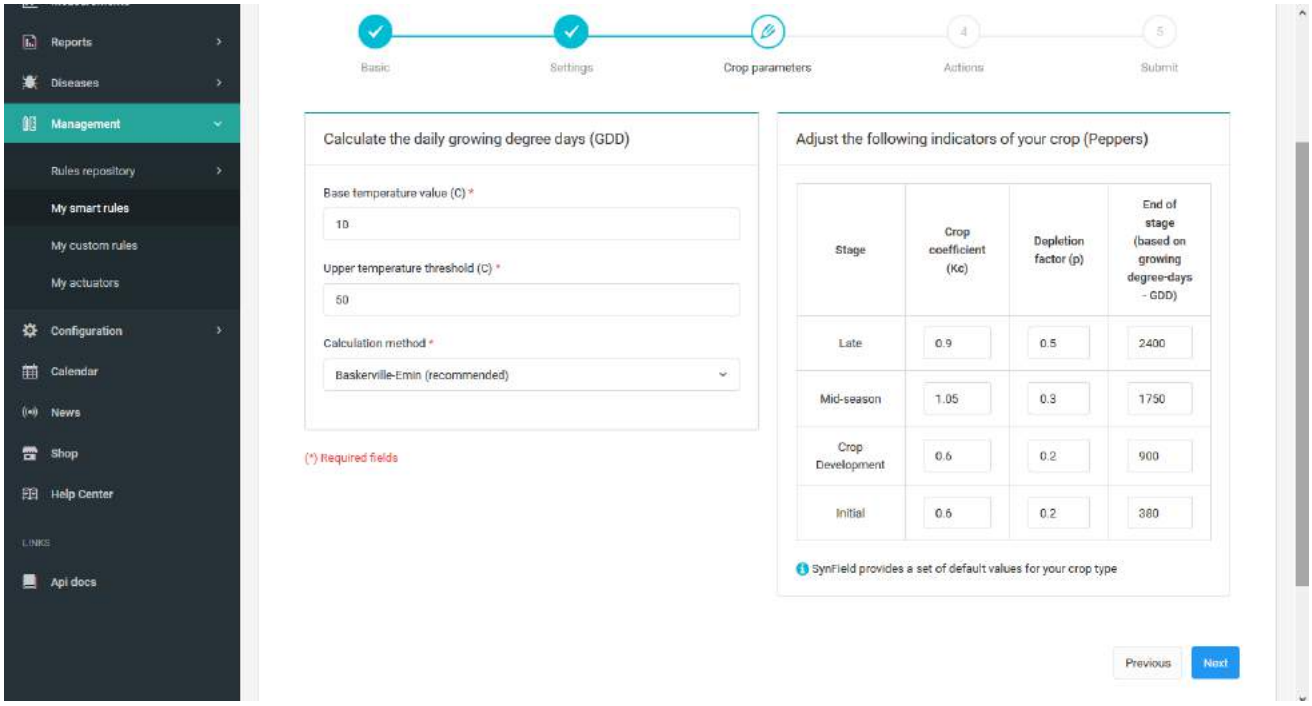


Figure 41: Define the soil profile

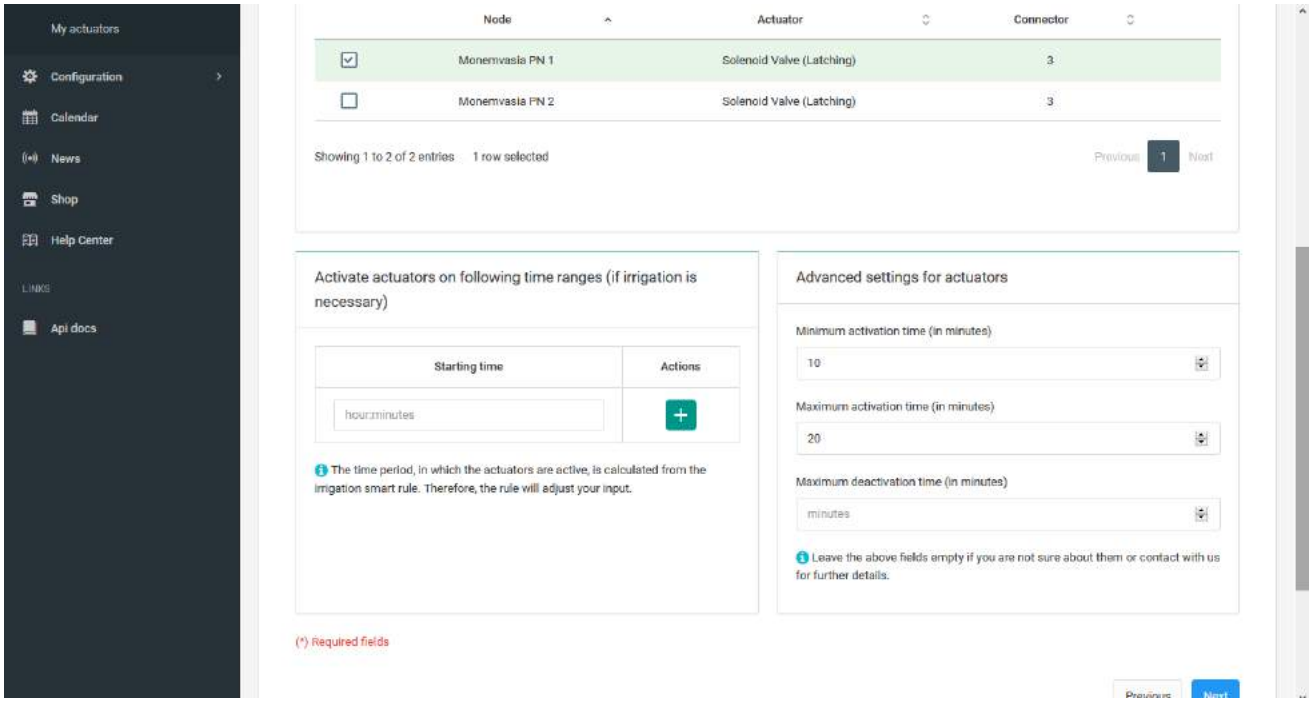


Figure 42: Associate one or more actuators with the irrigation rule and define the irrigation plan

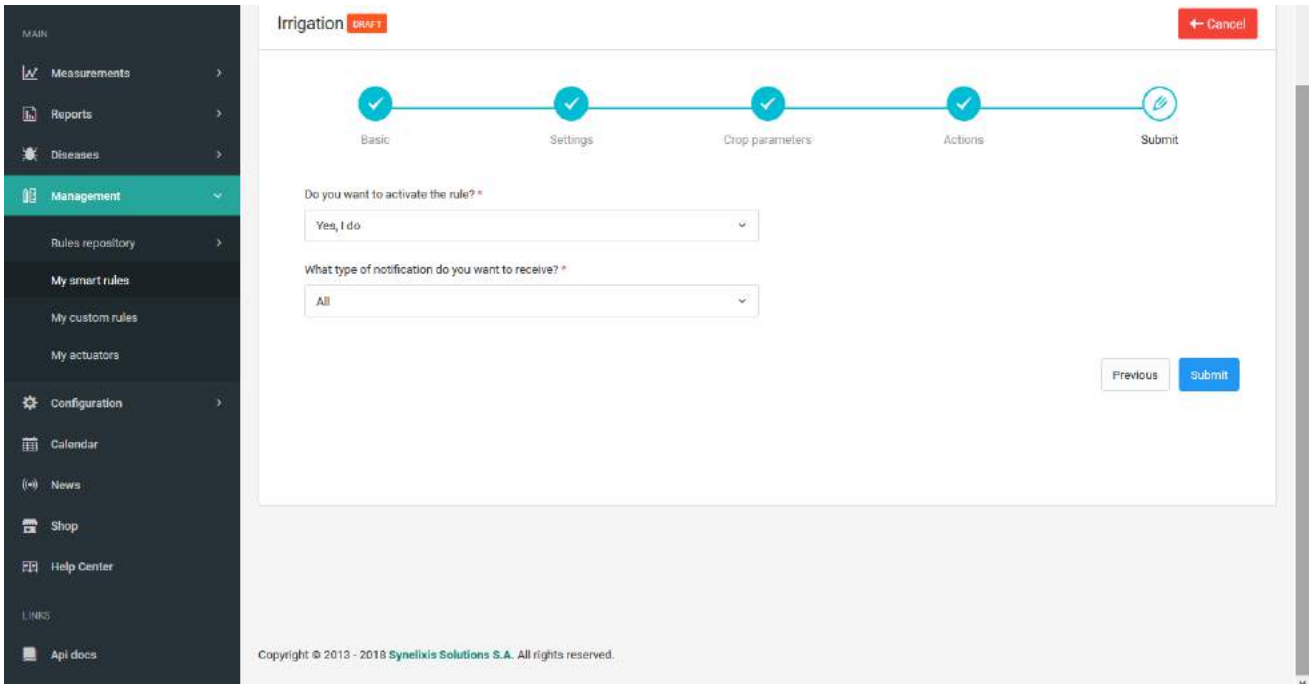


Figure 43: Submit and activate the irrigation rule

As soon as the *Submit* button is pressed the rule will appear in the smart rules page.

2.2.11 Synfield REST API

Synfield provides a RESTful API so as any user is able to retrieve data related to his account. Also, the Synfield API facilitates the integration among the Synfield platform and other platforms. The format of the timestamp, included in each web service, is based on the ISO 8061. Synelixis provides an interactive API documentation for the Synfield API that enables the users to try out the API calls directly in the

browser. The API documentation is based on the swagger framework. The Synfield API documentation is public available in the link http://83.235.169.221:9990/en/api/docs/.

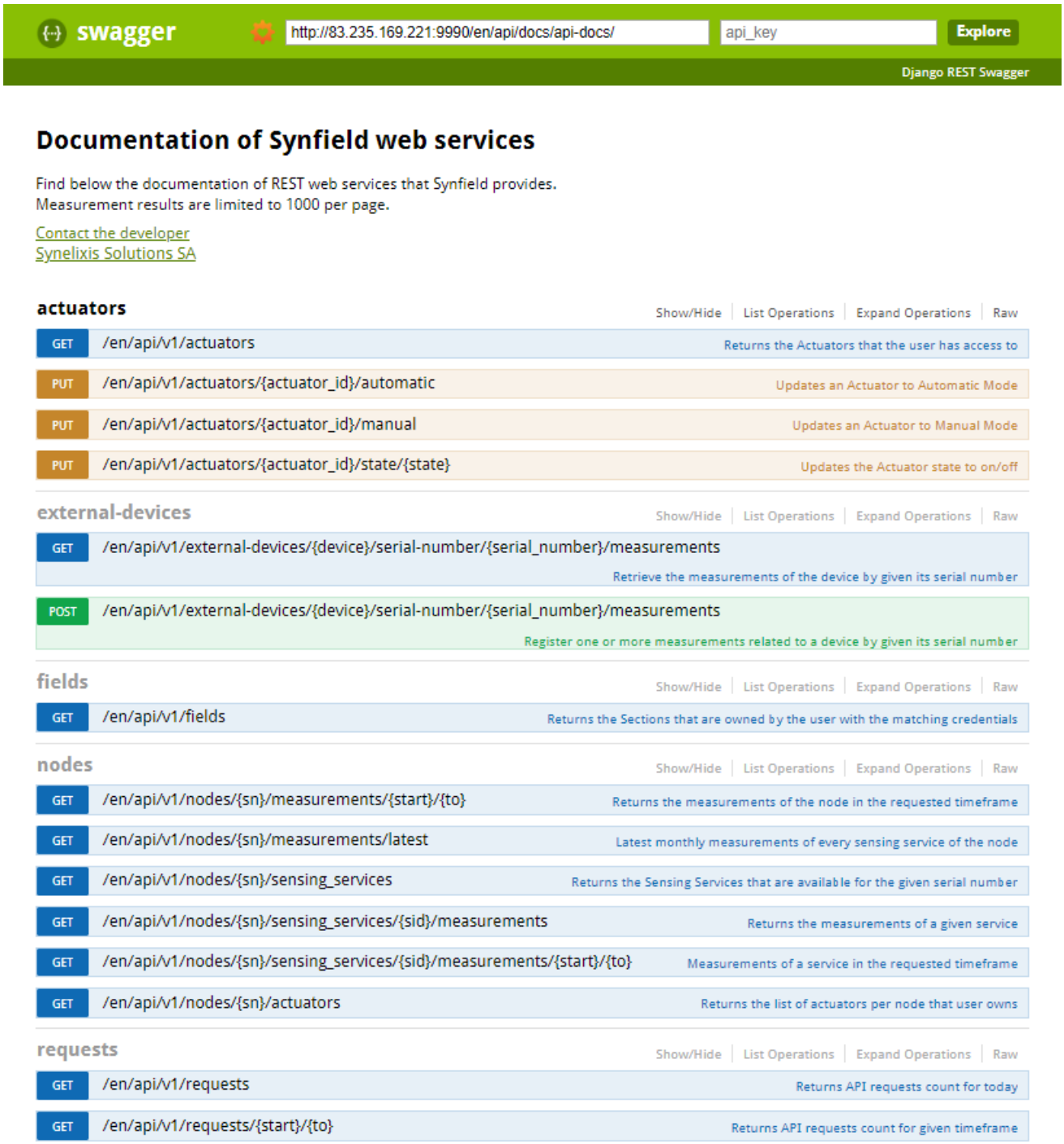


Figure 44: Overview of Synfield API in swagger

2.2.11.1 User authentication and authorization

The first time that the user will try to invoke a web service, the user must type his personal credentials (see Figure 45).Actually, in any web service call, the user's credentials must be used in the HTTP Authorization (Basic type) after the encoding in base64 format. To generate this information, the conversion *base64_encode(username:password)* must be performed. After that, each call will refer to

user resources. It is worth to stress here that any user is not able to access resources that owns to another user.

Figure 45: Authentication through user credentials

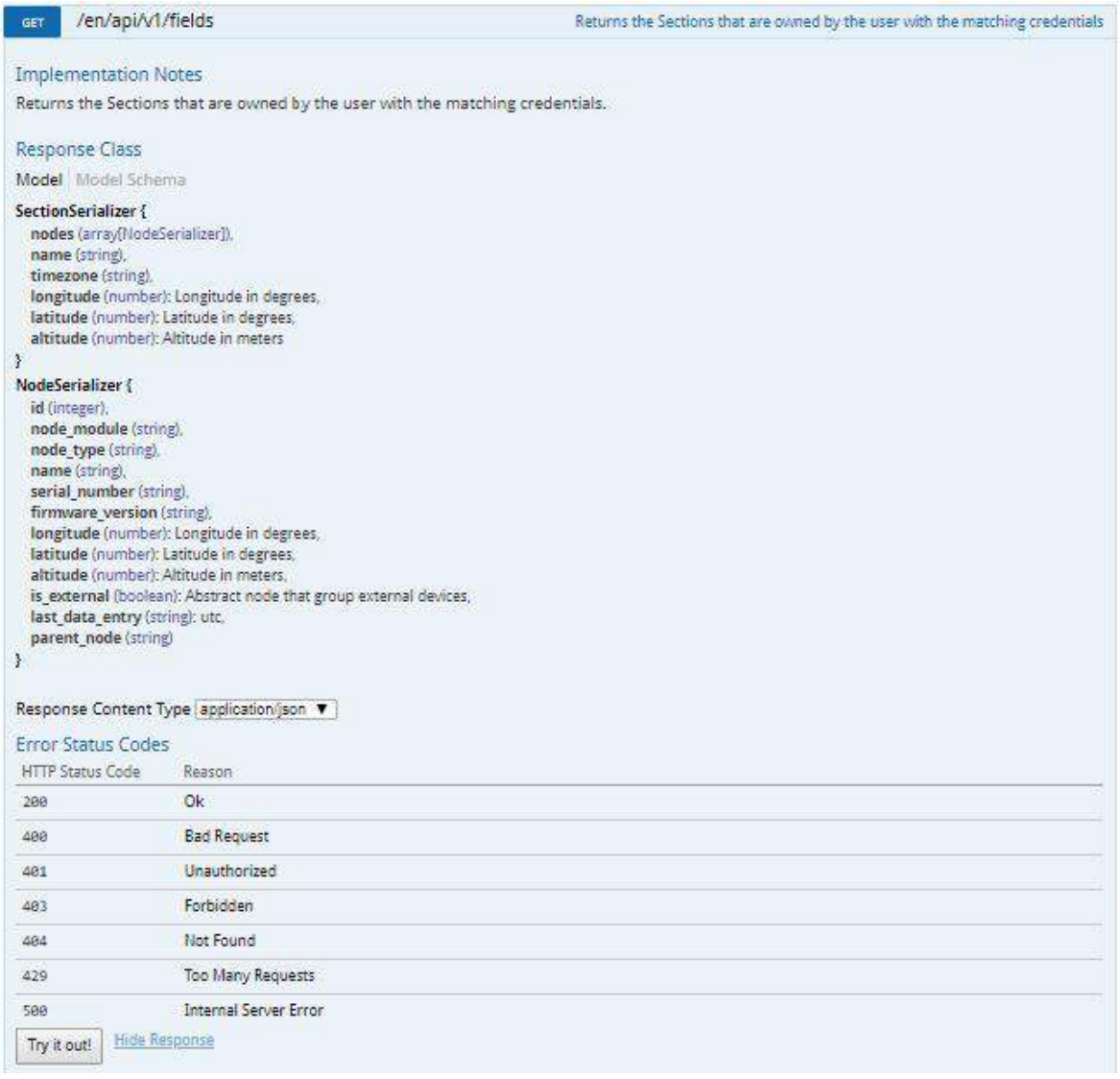


Figure 46: Description of the endpoint that fetches the list of fields of the user

2.2.11.2 Get the list of the fields

The user is able to retrieve the list of fields that owns using the web service as described in Figure 46. Each field includes its name, its coordinates, its time zone and a list of associated Synfield (head and peripheral) nodes (see *Section Serializer*). Each node contains its serial number, its coordinates, its

hardware and firmware version and its name (see *Node Serializer*).Focus on the response model (serializers) and list of the potential HTTP status codes. Figure 47 depicts the response of the web service after the user clicks on “Try it out!”.

Figure 47: List of fields

2.2.11.3 Get the list of sensing services per node

The user is able to retrieve the list of sensing services per node (the services that are measured through the plugged sensors) by given Synfield node’s serial number as described in Figure 48. Each sensing service includes its name, its unit, the type of the sensor and the timestamp of the latest measurement. Focus on the response model (serializers) and list of the potential HTTP status codes. Figure 49 depicts the response of the web service after the user clicks on “Try it out!”.

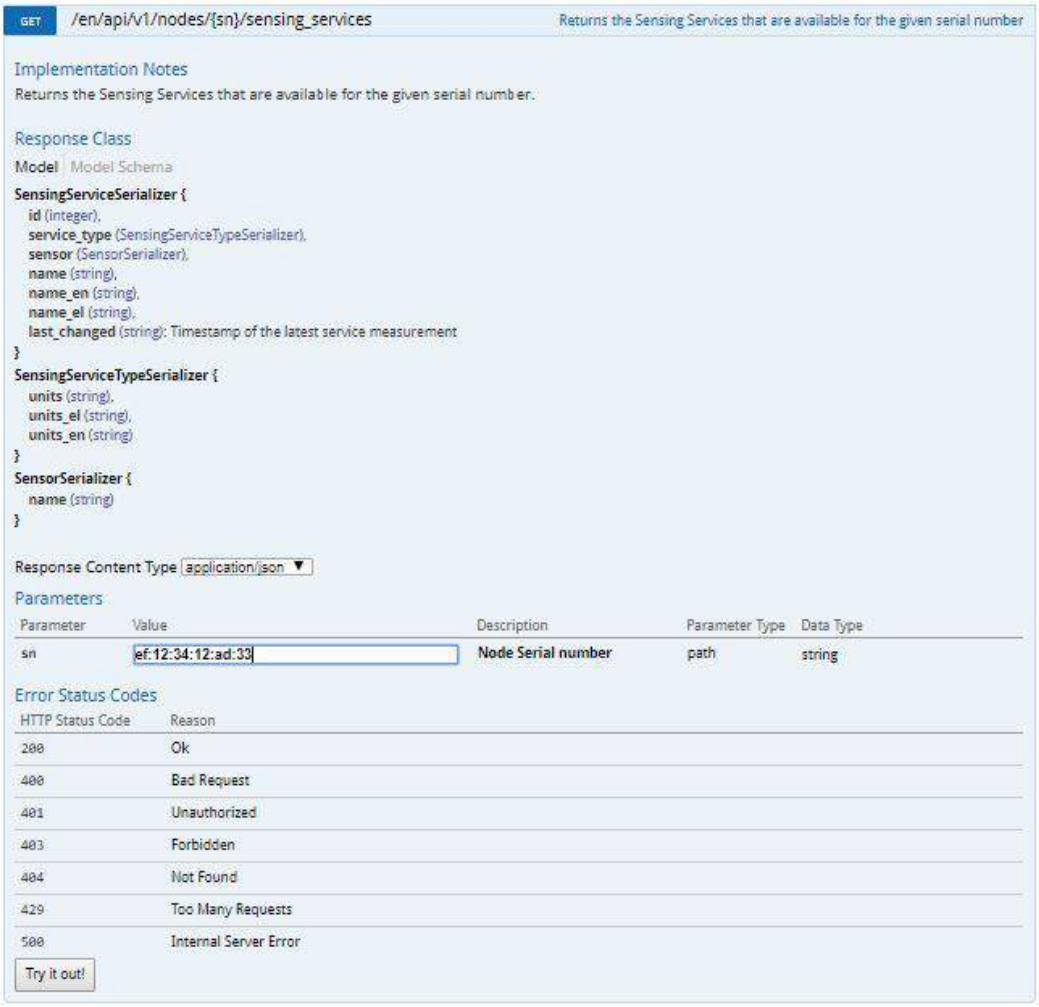


Figure 48: Description of the web service that fetches the list of sensing services of the node

Figure 49: List of sensing services per node

2.2.11.4 Get the list of measurements per sensing service

The user is able to retrieve the list of measurements by given the node's serial number and the sensing service id as described in Figure 50. Each measurement includes the id of the sensing service, its value and the timestamp. Figure 51 depicts the response of the web service after the user clicks on "Try it out!".

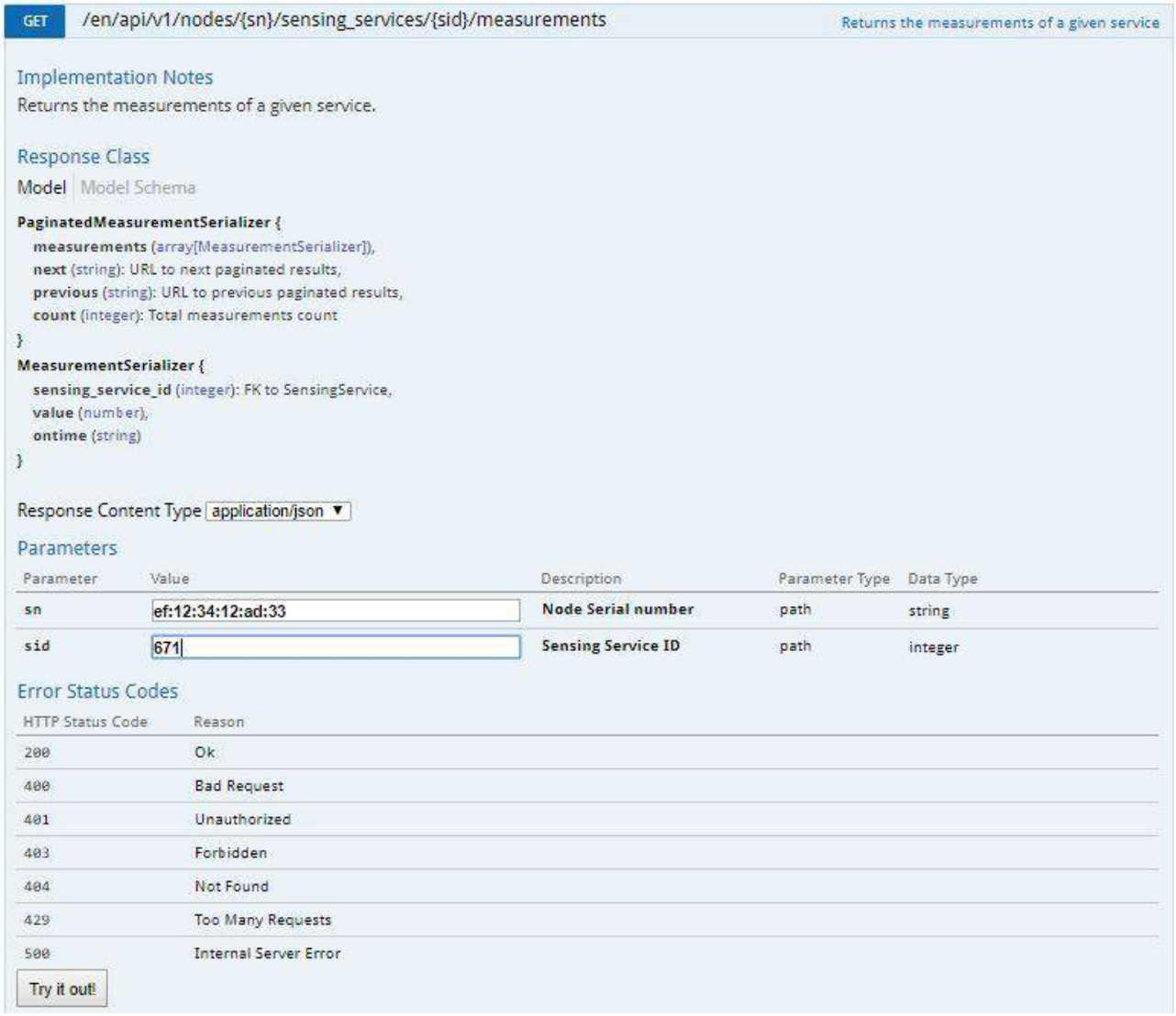


Figure 50: Description of the web service that fetches the list of measurements per sensing service

Request URL

```
http://83.235.169.221:9990/en/api/v1/nodes/ef%3A12%3A34%3A12%3Aad%3A33/sensing_services/671/measurements
```

Response Body


```
{
  "count": 2,
  "previous": null,
  "measurements": [
    {
      "sensing_service_id": 671,
      "value": 18.9,
      "ontime": "2018-05-15T22:14:35.253984Z"
    },
    {
      "sensing_service_id": 671,
      "value": 22.9,
      "ontime": "2018-05-15T22:13:36.523073Z"
    }
  ],
  "next": null
}
```


Response Code

```
200 OK
```

Figure 51: List of measurements per sensing service

GET /en/api/v1/nodes/{sn}/sensing_services/{sid}/measurements/{start}/{to} — Measurements of a service in the requested timeframe

Implementation Notes

Measurements of a service in the requested timeframe.

Response Class

Model | Model Schema

PaginatedMeasurementSerializer {
  measurements (array[MeasurementSerializer]),
  next (string): URL to next paginated results,
  previous (string): URL to previous paginated results,
  count (integer): Total measurements count
}
MeasurementSerializer {
  sensing_service_id (integer): FK to SensingService,
  value (number),
  ontime (string)
}

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
sn	ef:12:34:12:ad:33	**Node Serial number**	path	string
sid	671	**Sensing Service ID**	path	integer
start	2018-03-27T12:30:00Z	**Valid ISO 8601 date (yyyy-mm-dd) or datetime (yyyy-mm-ddThh:mm:ssZ)**	path	string
to	2018-05-16T16:30:00Z	**Valid ISO 8601 date (yyyy-mm-dd) or datetime (yyyy-mm-ddThh:mm:ssZ)**	path	string
page	1	Page number	query	integer
page_size	50	Maximum page size	query	integer

Error Status Codes

HTTP Status Code	Reason
200	Ok
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
429	Too Many Requests
500	Internal Server Error

Try it out!

Figure 52: Description of the web service that fetches the list of measurements per sensing service in date range

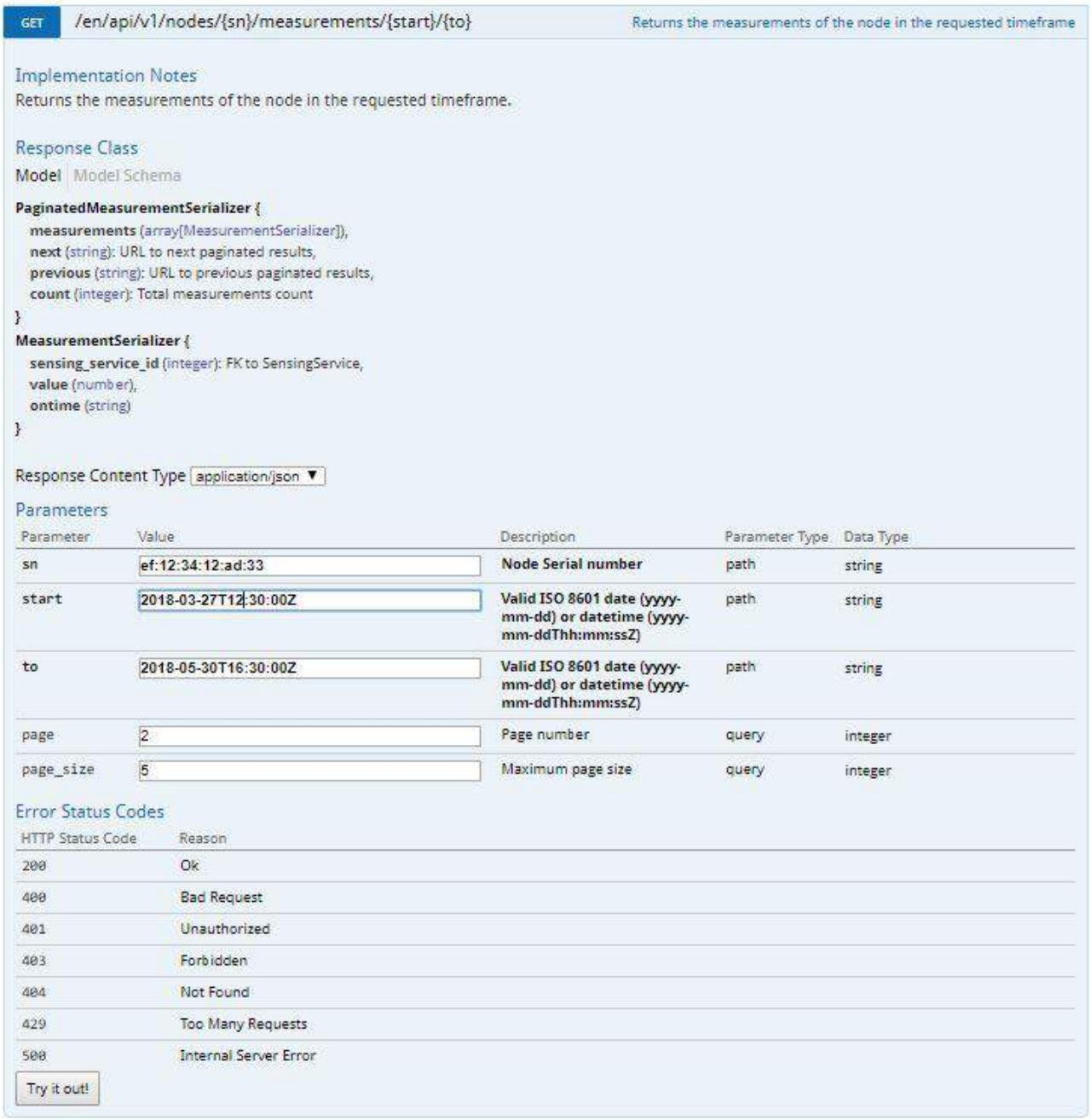
2.2.11.5 Get the list of measurements per sensing service in date range

The user can retrieve the list of measurements by given the node's serial number and the sensing service id in the requested date range as described in Figure 52. The structure of the response is descripted through the serializers. This web service provides also pagination query parameters (maximum page size and number of page). Figure 53 depicts the response of the web service after the user clicks on "Try it out!".

Figure 53: List of measurements per sensing service in date range

2.2.11.6 Get the list of measurement per node in date range

The user can retrieve the list of measurements by given the node's serial number in the requested date range as described in Figure 54. The structure of the response is descripted through the serializers. This web service provides also pagination query parameters (maximum page size and number of page). Figure 55 depicts the response of the web service after the user clicks on "Try it out!".



GET /en/api/v1/nodes/{sn}/measurements/{start}/{to} — Returns the measurements of the node in the requested timeframe

Implementation Notes

Returns the measurements of the node in the requested timeframe.

Response Class

Model | Model Schema

PaginatedMeasurementSerializer {
  measurements (array[MeasurementSerializer]),
  next (string): URL to next paginated results,
  previous (string): URL to previous paginated results,
  count (integer): Total measurements count
}
MeasurementSerializer {
  sensing_service_id (integer): FK to SensingService,
  value (number),
  ontime (string)
}

Response Content Type: application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
sn	ef:12:34:12:ad:33	**Node Serial number**	path	string
start	2018-03-27T12:30:00Z	**Valid ISO 8601 date (yyyy-mm-dd) or datetime (yyyy-mm-ddThh:mm:ssZ)**	path	string
to	2018-05-30T16:30:00Z	**Valid ISO 8601 date (yyyy-mm-dd) or datetime (yyyy-mm-ddThh:mm:ssZ)**	path	string
page	2	Page number	query	integer
page_size	5	Maximum page size	query	integer

Error Status Codes

HTTP Status Code	Reason
200	Ok
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
429	Too Many Requests
500	Internal Server Error

Try it out!

Figure 54: Description of the web service that fetches the list of measurements per node in date range

Figure 55: List of measurements per node in date range

2.2.11.7 Get the latest measurement of each sensing service per node

The user can retrieve the latest measurement of each sensing service that are associated with the given node as described in Figure 56. The structure of the response is descripted through the serializers. Figure 57 depicts the response of the web service after the user clicks on “Try it out!”.

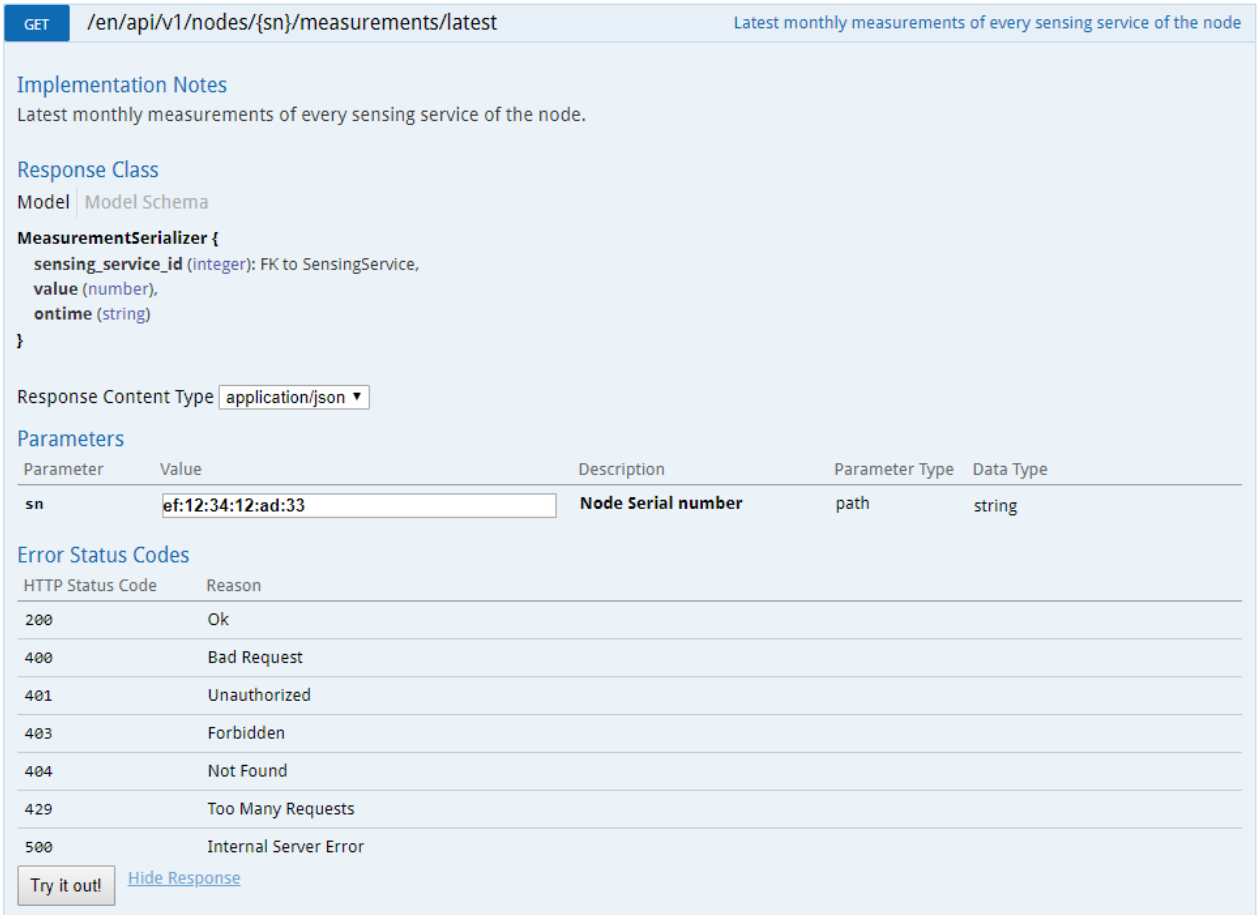


Figure 56: Description of the web service that fetches latest measurement of each sensing service per node

Request URL

```
http://83.235.169.221:9990/en/api/v1/nodes/ef%3A12%3A34%3A12%3Aad%3A33/measurements/latest
```

Response Body


```
[
  {
    "sensing_service_id": 673,
    "value": 0,
    "ontime": "2018-05-16T06:29:18.507363Z"
  },
  {
    "sensing_service_id": 672,
    "value": 56.3,
    "ontime": "2018-05-16T06:29:18.507363Z"
  },
  {
    "sensing_service_id": 671,
    "value": 22.4,
    "ontime": "2018-05-16T06:29:18.507363Z"
  },
  {
    "sensing_service_id": 668,
    "value": 21.1,
    "ontime": "2018-05-16T06:29:18.507363Z"
```


Response Code

```
200 OK
```

Figure 57: List of latest measurements (per sensing service) per node

2.2.11.8 Get the list of actuators per node

The user can retrieve the list of actuators by given the node's serial as described in Figure 58. The structure of the response is descripted through the serializers. Figure 59 depicts the response of the web service after the user clicks on "Try it out!".

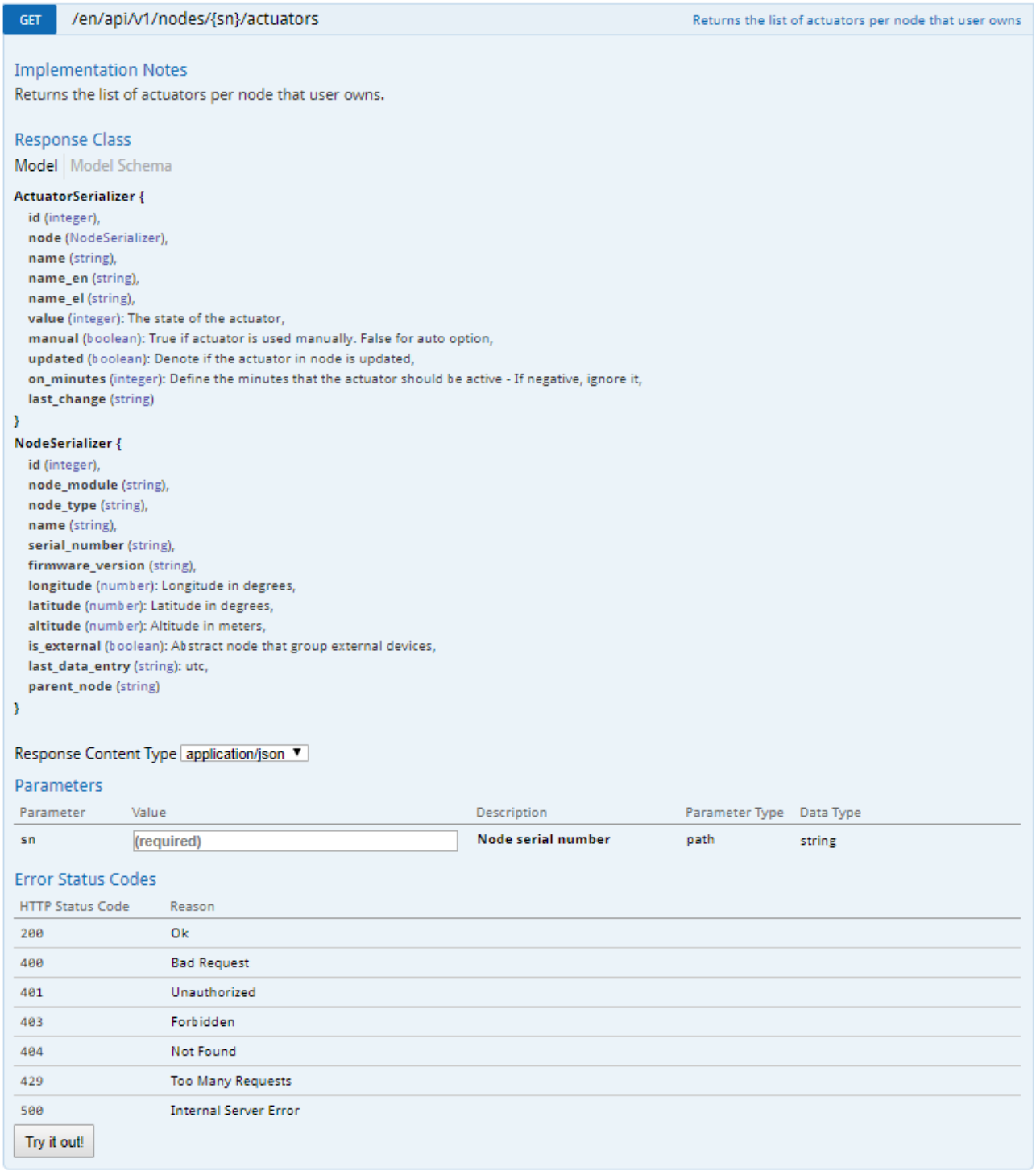


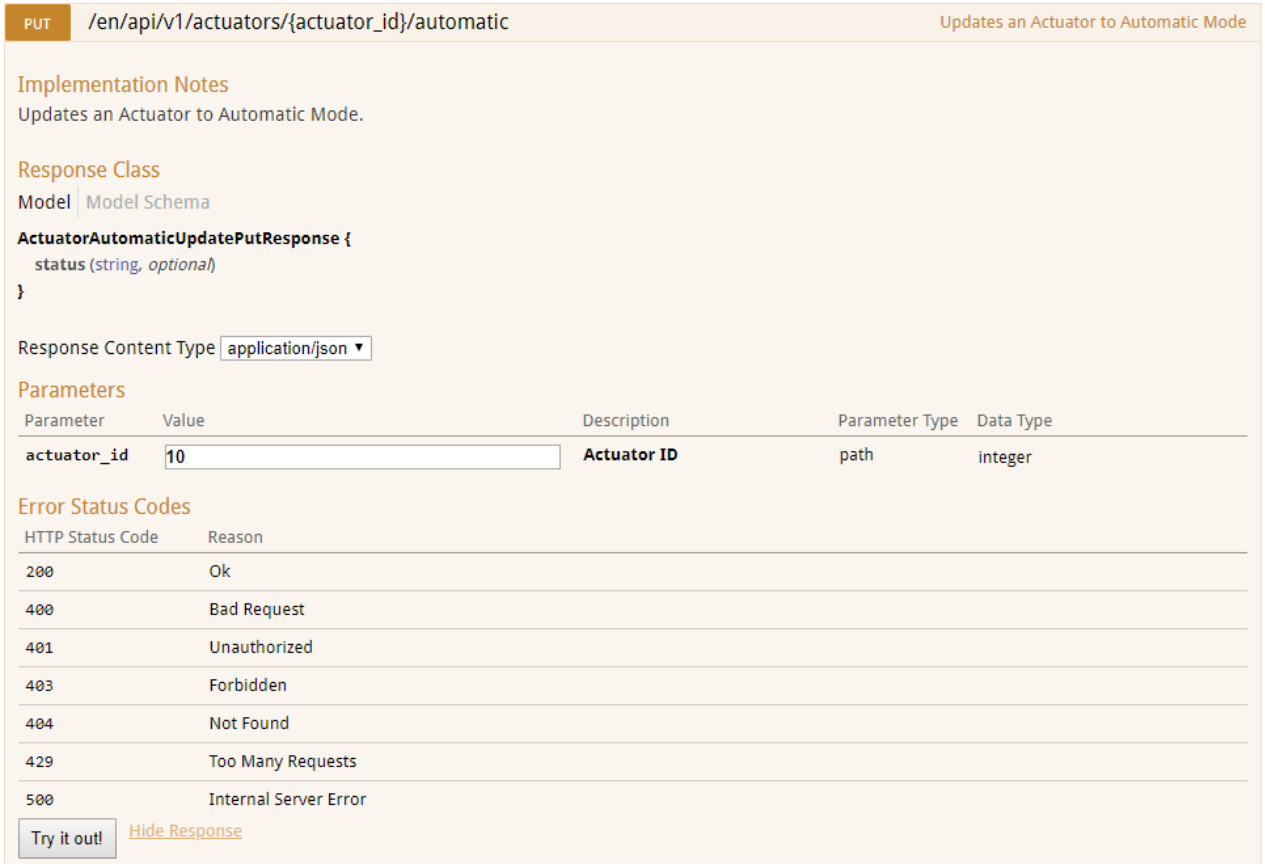
Figure 58: Description of the web service that fetches the list of actuators in a node

Figure 59: List of actuators per node

2.2.11.9 Update the mode of the actuator

By default, a new actuator is working on manual mode (the user is able to set its state directly from the GUI). The user is able to set the actuator in automatic mode by given its id as described in Figure 60. The automatic mode means that the actuator's state depends on the defined rules in the Synfield platform. No payload is required. The structure of the response is descripted through the serializers. Figure 61 depicts the response of the web service after the user clicks on "Try it out!".

Another web service is available that swaps the mode of an actuator from automatic to manual one. Its description is depicted in Figure 62 while the response after its call by given actuator id is depicted in Figure 63.



PUT /en/api/v1/actuators/{actuator_id}/automatic — Updates an Actuator to Automatic Mode

Implementation Notes

Updates an Actuator to Automatic Mode.

Response Class

Model | Model Schema

```
ActuatorAutomaticUpdatePutResponse {
  status (string, optional)
}
```

Response Content Type: application/json

Parameters

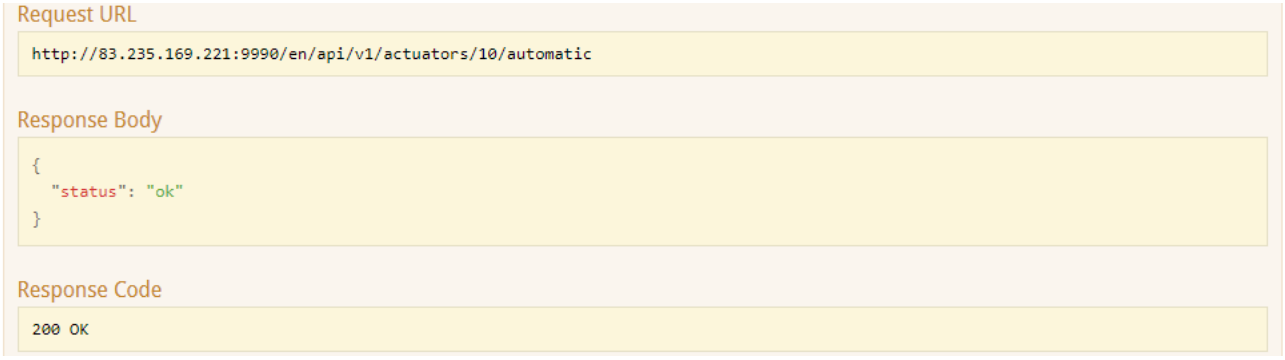
Parameter	Value	Description	Parameter Type	Data Type
actuator_id	10	**Actuator ID**	path	integer

Error Status Codes

HTTP Status Code	Reason
200	Ok
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
429	Too Many Requests
500	Internal Server Error

Try it out! Hide Response

Figure 60: Description of the web service that sets the actuator in automatic mode



Request URL

```
http://83.235.169.221:9990/en/api/v1/actuators/10/automatic
```

Response Body

```
{
  "status": "ok"
}
```

Response Code

```
200 OK
```

Figure 61: Set the actuator in automatic mode

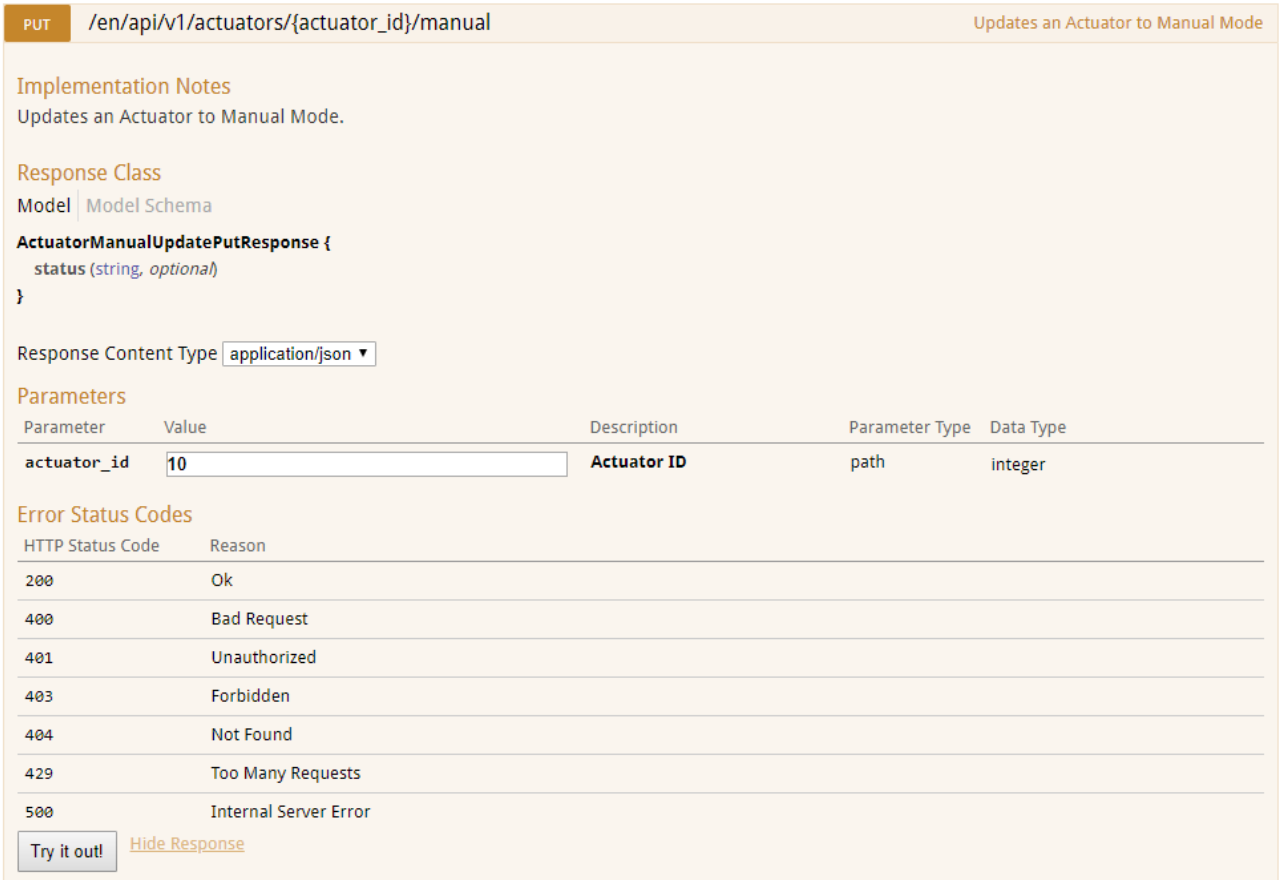
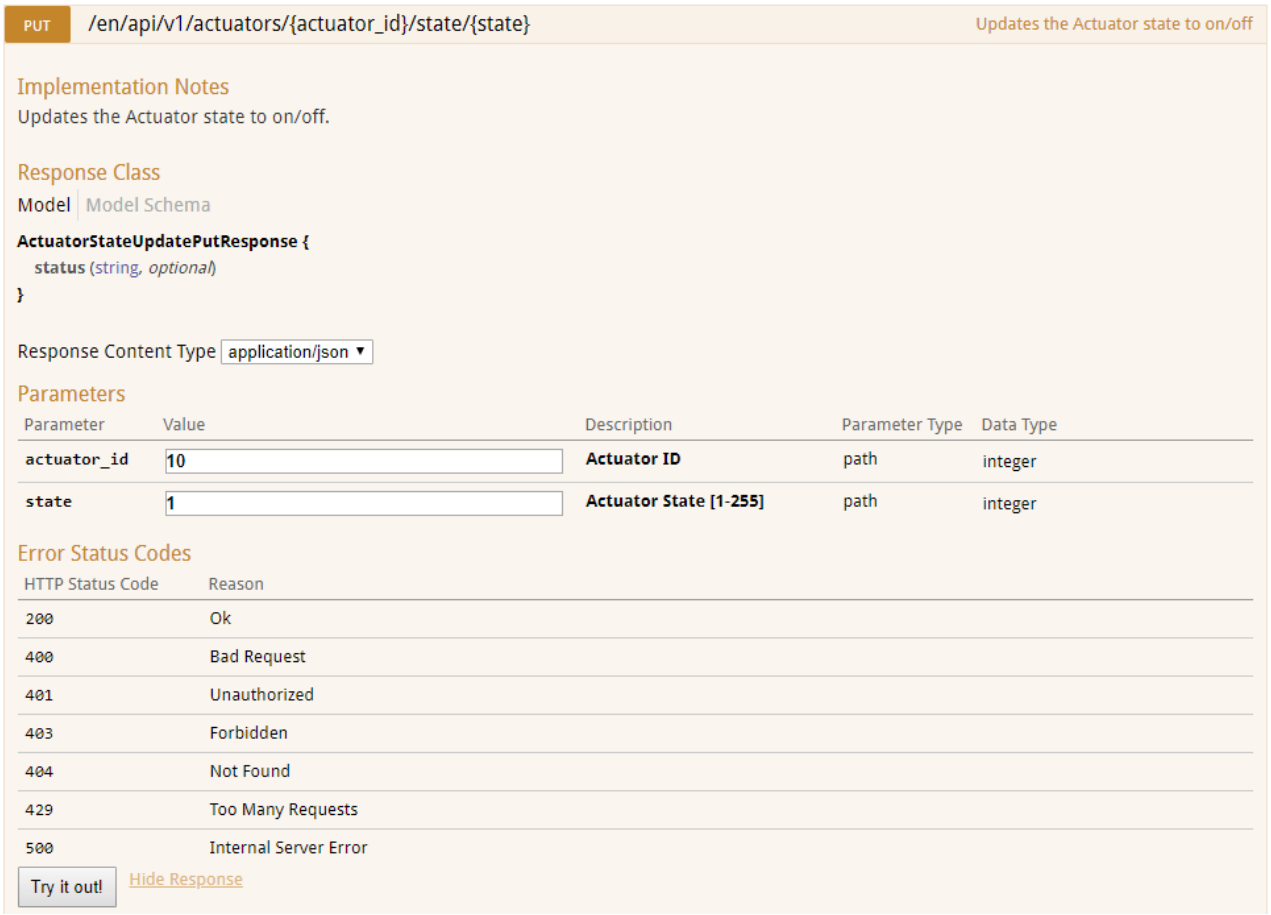


Figure 62: Description of the web service that sets the actuator in manual mode

Figure 63: Set the actuator in manual mode

2.2.11.10 Update the state of the actuator

Synfield API includes a web service that sets the state of an actuator. Figure 64 describes how to activate an actuator while its response (after its call by given actuator id) is depicted in Figure 65. It is worth to illustrate here that the state of an actuator can be modified only if the actuator operates in manual mode.



PUT /en/api/v1/actuators/{actuator_id}/state/{state} — Updates the Actuator state to on/off

Implementation Notes

Updates the Actuator state to on/off.

Response Class

Model | Model Schema

ActuatorStateUpdatePutResponse {
status (string, *optional*)
}

Response Content Type application/json

Parameters

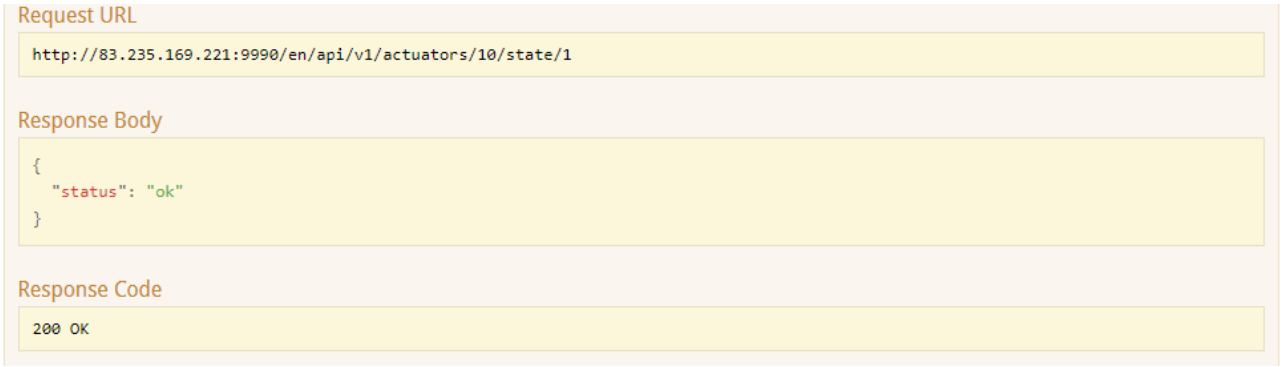
Parameter	Value	Description	Parameter Type	Data Type
actuator_id	10	**Actuator ID**	path	integer
state	1	**Actuator State [1-255]**	path	integer

Error Status Codes

HTTP Status Code	Reason
200	Ok
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
429	Too Many Requests
500	Internal Server Error

Try it out! Hide Response

Figure 64: Description of the web service that activates the actuator (manual mode)



Request URL

```
http://83.235.169.221:9990/en/api/v1/actuators/10/state/1
```

Response Body

```
{
  "status": "ok"
}
```

Response Code

```
200 OK
```

Figure 65: Activates an actuator

2.2.11.11 Register the measurements of the external devices

The Synfield REST API provides two web services that achieve the integration among the Synfield platform and the external devices such as the Plant-O-Meter (provided from BIOS). Especially, a web service supports the insertion of the external devices measurements and the other provides the retrieval of these measurements. Both consider the type of the external device and the serial number as well. By given the type of external device (e.g. plant-o-meter-biosense) and its serial number (e.g.POM31231212), the measurements of the Plant-O-Meter device can be inserted in the Synfield platform through the specific web service as depicted in Figure 66 and Figure 67.

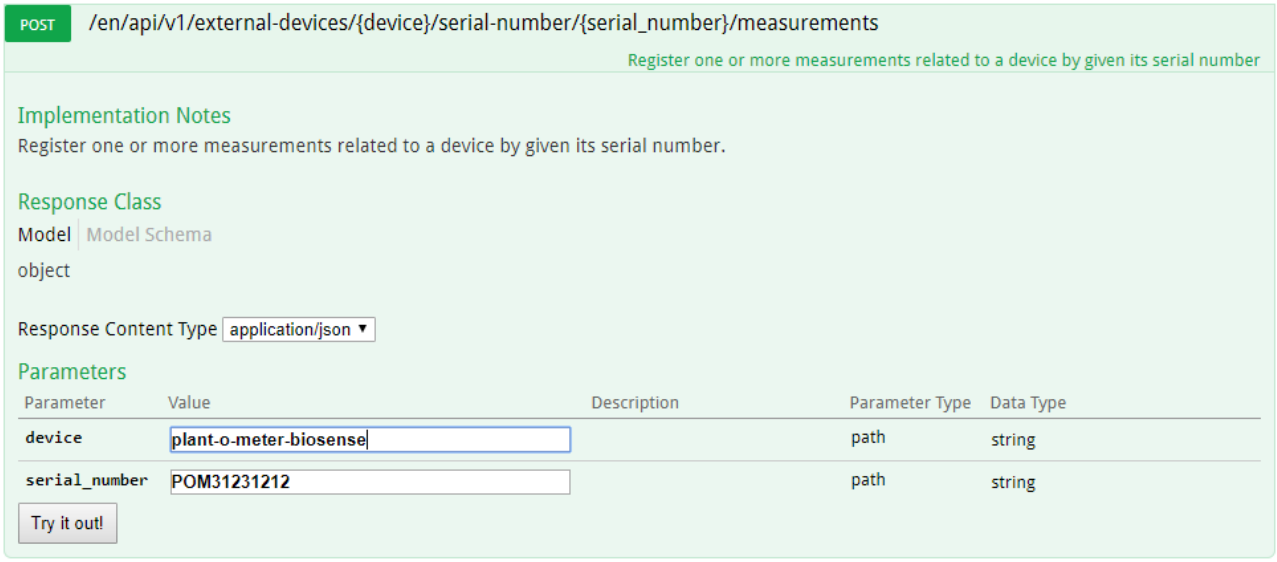


Figure 66: Description of the web service that allows the registration of metrics by device type and device serial number

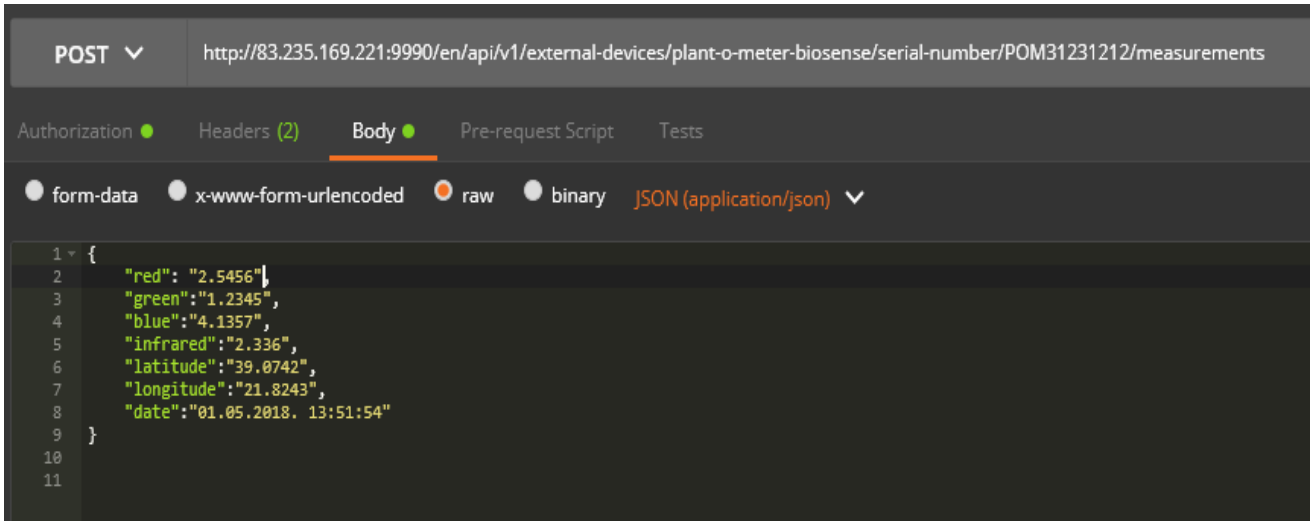


Figure 67: Register measurements for a specific Plant-O-meter device

The above web service is able to consume both an array of measurements as well as one measurement per request. Currently, since the Synfield platform supports only the Plant-O-Meter, the attributes of the payload are depicted in Table 1.

Table 1: Description of Plant-O-Meter attributes

Name	Type	Description
red	String	The value of the red colour as provided from the Plant-O-Meter
green	String	The value of the green colour as provided from the Plant-O-Meter
blue	String	The value of the blue colour as provided from the Plant-O-Meter
infrared	String	The value of the infrared colour as provided from the Plant-O-Meter
latitude	String	The latitude of the device
longitude	String	The longitude of the device
date	string	The timestamp of the measurement

The potential error status codes related to the above web service are listed the following table.

Table 2: List of potential HTTP status codes in registration of the device measurements

Name	Type	Description
201	Created	Successful insertion of measurements
400	Bad Request	Erroneous payload
401	Unauthorized	Invalid credentials
404	Not Found	Unsupported type of device, non-existing serial number
500	Internal Server Error	Unhandled exception

2.2.11.12 Get the measurements of the external devices

The retrieval of the measurements can be performed in similar way. By given the type of external device (plant-o-meter-biosense) and its serial number (POM31231212), the measurements of the Plant-O-Meter device can be retrieved through the Synfield API as depicted in Figure 68.

The response of this web service provides an array of measurements, where each measurement includes the attributes is described in Table 3.

Table 3: Description of Plant-O-Meter attributes in the retrieval of measurements

Name	Type	Description
red	String	The value of the red colour as provided from the Plant-O-Meter
green	String	The value of the green colour as provided from the Plant-O-Meter
blue	String	The value of the blue colour as provided from the Plant-O-Meter
infrared	String	The value of the infrared colour as provided from the Plant-O-Meter
latitude	String	The latitude of the device
longitude	String	The longitude of the device
ndvi	float	The Normalized Difference Vegetation Index as calculated in the Synfield platform
timestamp	string	The timestamp of the measurement (time zone-aware)

The potential error status codes related to the aforementioned web service are listed in Table 4. The measurements related to external devices can be also retrieved through the standard web services that retrieve the measurements related to the Synfield nodes.

Table 4: List of potential HTTP status codes in retrieval of the device measurements

Name	Type	Description
200	OK	Successful retrieval of measurements
401	Unauthorized	Invalid credentials
404	Not Found	Unsupported type of device
500	Internal Server Error	Unhandled exception

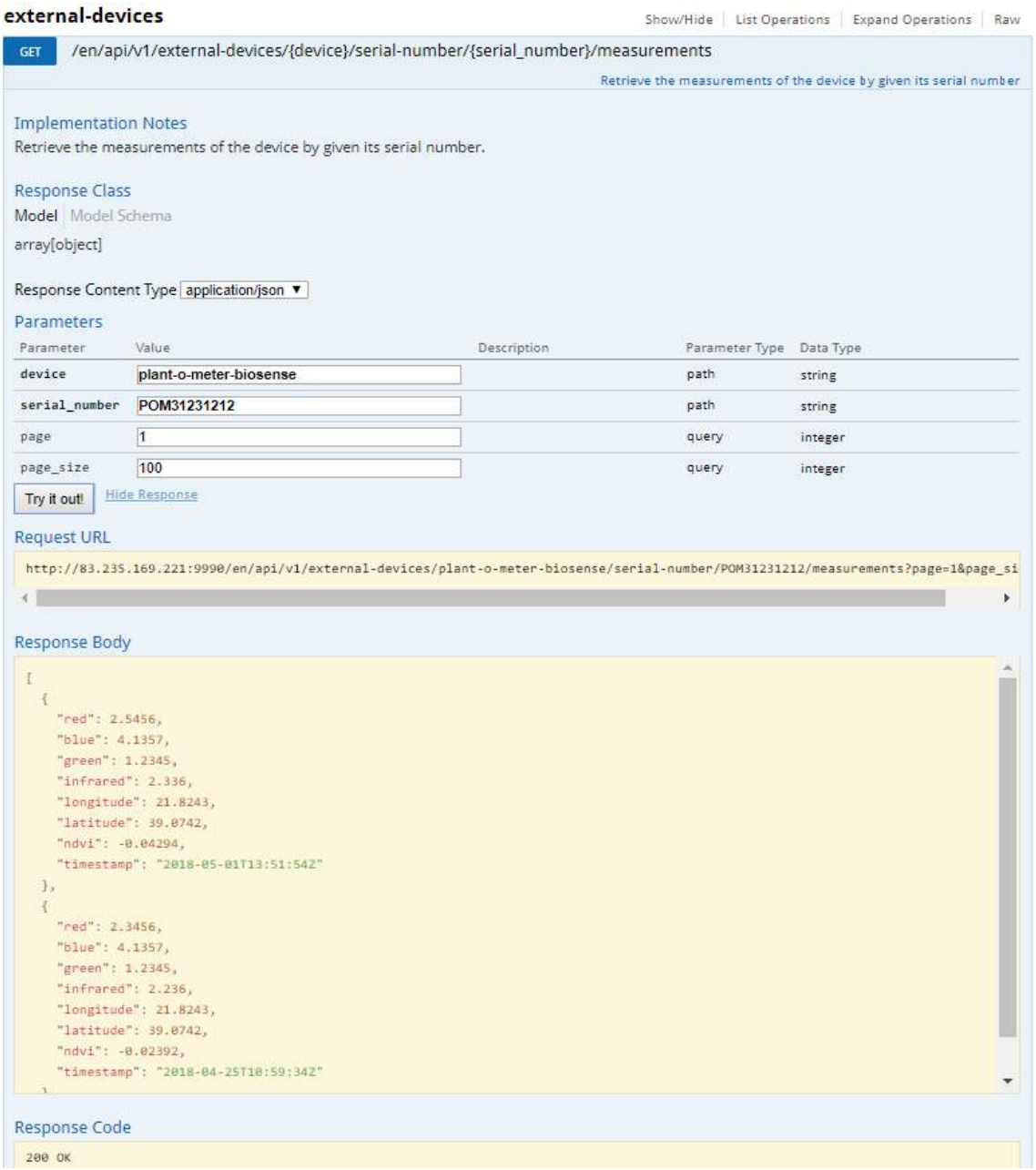


Figure 68: Retrieve the measurements from a Plant-O-Meter device

2.3 Virtual Fort Knox Cloud Platform

Virtual Fort Knox (VFK) is a federative platform for the manufacturing industry developed by the Fraunhofer Institute for Manufacturing Engineering and Automation (Fraunhofer IPA). It will offer manufacturing companies an IT strategy that is cost-efficient, agile and scalable. Companies will be provided with efficient access to Industry 4.0 software solutions which are independent of manufacturers, to make advances in the digitalisation and optimisation of their production processes. Figure 69 depicts the concept of VFK. It is based on a cell structure and follows the “security-by-design” principle. Each VFK cell is a securely encapsulated environment for service users and service providers.

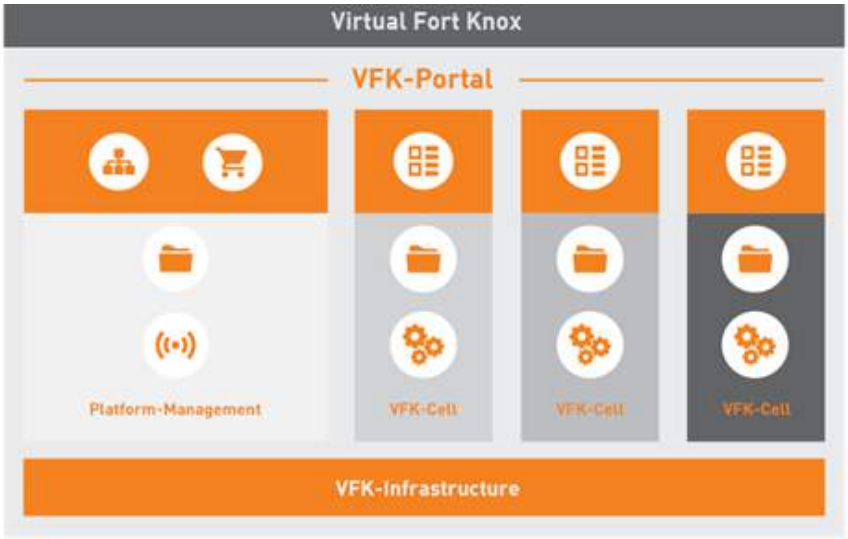


Figure 69: Virtual Fort Knox Concept

Figure 70 illustrates the VFK architecture. The physical devices are on the show floor level and are called Smart Objects (e.g. equipment or cyber-physical systems (CPS)). Due to the substantial number of communication protocols, a middleware is used for the communication. This middleware is called Manufacturing Services Bus (MSB) and is described in section 2.3.1.

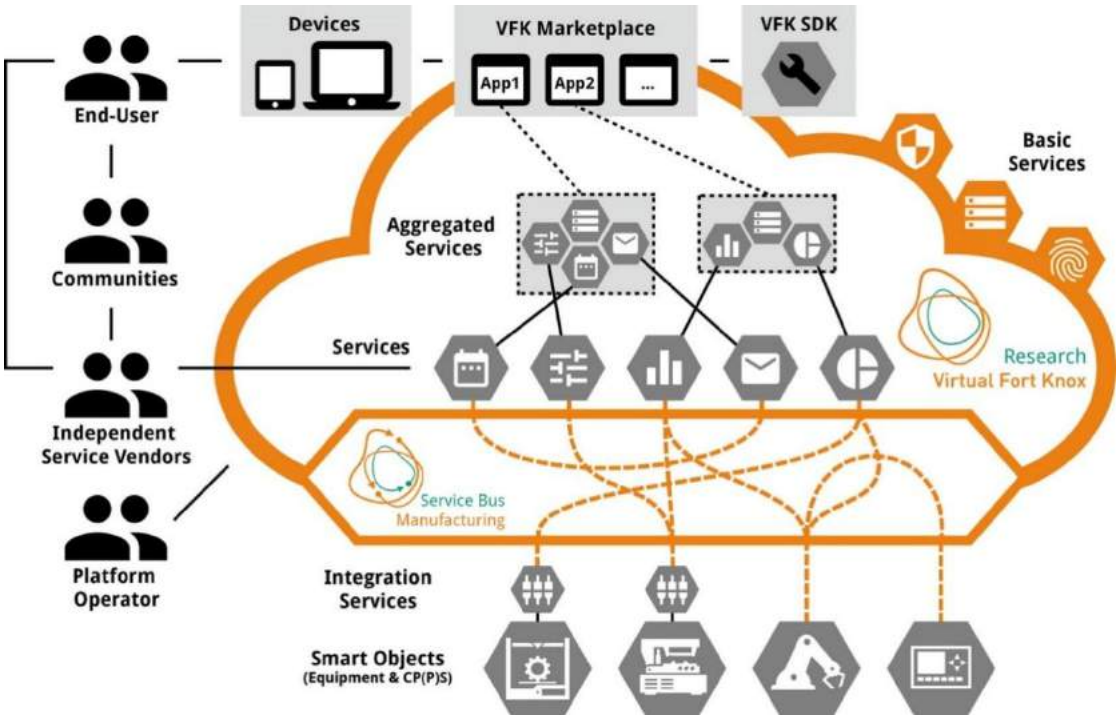


Figure 70: Virtual Fort Knox Architecture

To communicate with IT services running in the cloud these services are also connected the MSB. Following a service-oriented approach, the services can be aggregated to new services that provide new functionalities. Economically relevant will be the opportunity for Independent Service Vendors (ISVs) to offer their services in the VFK marketplace where the end users are able to purchase the services they need. E.g. an equipment manufacturer can offer some special services for its equipment and the customers can purchase the services which they need. From the technical side VFK offers a software development kit (SDK) which is available in all common programming languages. Applications can be hosted in the cloud infrastructure in form of virtual machines and docker containers. Additionally, the platform provides a flexible middleware as abstraction layer between components which allows changes to the flow of information at run-time.

2.3.1 Cell Concept of Virtual Fort Knox

Each organization, using VFK, operates within an encapsulated environment, referred to as a *cell*. These cells can be publicly hosted or run on a local machine in the network infrastructure of the organization, as shown in Figure 71. Data cannot be transferred between cells, unless applications or smart objects are specifically set up to do so (e.g. a bridge interface). Generally, it is advised to use the middleware accompanying VFK to set up communication within each cell. Each organization with its own cell may consist of multiple users. Users can deploy virtual machines or will be able to download preset software from the shop. Such preset software will be available in the centralized shop, which operates like other app-shops. By default, components which are deployed by a user are only visible, and therefore useable, to him. However, they can be made visible to other users within the organization as well.

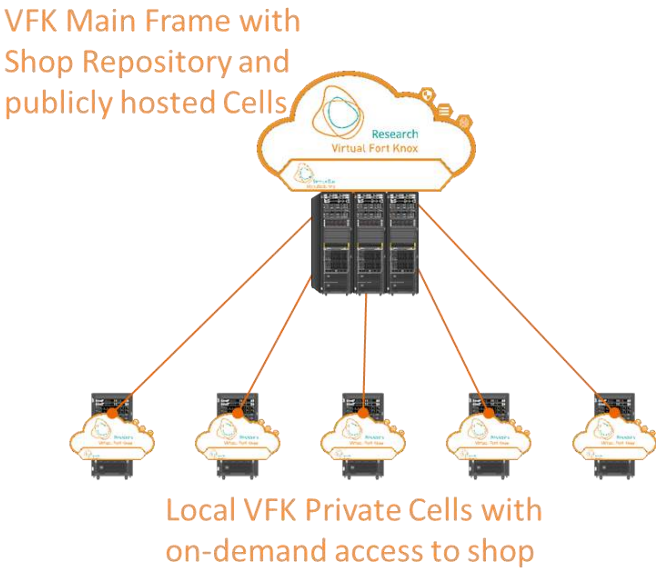


Figure 71: VFK cell concept with locally hosted cells and publicly hosted cells in the main infrastructure

2.3.2 Middleware - Manufacturing Service Bus

The Manufacturing Service Bus (MSB) enables a fast and low-effort integration of smart objects or IT-Services, because it provides the integration between various communication protocols such as RESTful Web Service or WebSocket API and various communication standards, for instance OPC UA. For this purpose the MSB provides common interfaces which allow the communication between smart objects and IT-services. The communication process is shown in Figure 72. The data are transferred in an encrypted channel. All send data are transformed to a common data format which ensures that all communication participants can communicate with each other. Received data are added to a queue to allow communication between communication partners with different communication cycles. The routing of the data is done using so-called integration flows, which allow the users to flexibly define where data is forwarded to. Integration flows can be defined without programming skills in the web-based user interface of the MSB. Alternatively, a RESTful API is also available for automation purposes.

Figure 72: Communication Process of the Manufacturing Service Bus

2.3.3 Communication pattern

The communication follows the pattern depicted in Figure 73. At the start of the communication, the client registers itself with the appropriate interface (depending on the used communication protocol). When registering, the client sends its self-description, so that the MSB knows who is registering and what capabilities are available and which data can be expected. After registration is done the client can send data by throwing an event that contains the data. To send data to the client the MSB calls the appropriate function on the client with the data as function parameters.

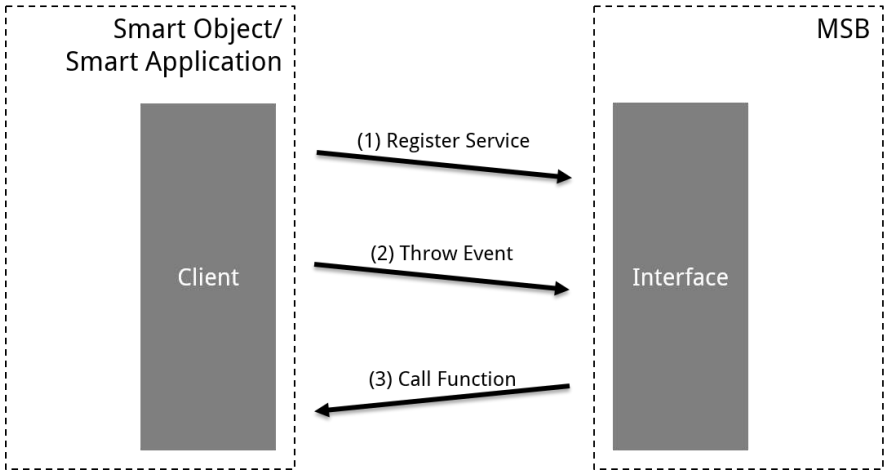


Figure 73: Communication Pattern of the Manufacturing Services Bus

2.3.3.1 Self-description of services

Each service has a self-description describing its characteristics. The structure of the self-description is shown in Figure 74. A service is classified as an Application or as a Smart Object and can be identified by its unique UUID. Data that is send by a service is described as events. Data that is send to the service can be received as function. Functions can be used to trigger capabilities of the service by internally mapping the incoming function to a callback function in the service-specific code. Such a callback function can trigger return events as well.

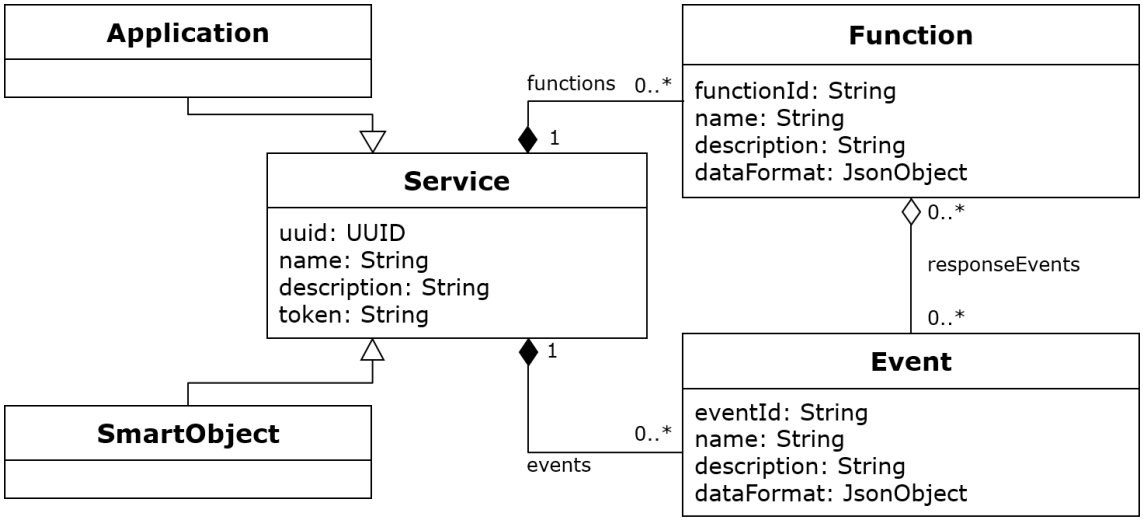


Figure 74: Self-description of services

Once a component (smart object or application) is registered at the MSB, the MSB can be configured to transfer information to and from the component automatically by manual configuration via a graphical user interface (GUI). A simple example for the communication pattern is shown in Figure 75. To achieve the shown information exchange, three main configuration steps must be completed: selection of the two components, selection of the corresponding event and function and finally mapping of output data of the smart object to the input parameter of the function of the application. The data emitted by the smart object is attached to the event as a JSON string. The data is then mapped to the corresponding input parameters of the function of the smart application and wrapped in a JSON string again.

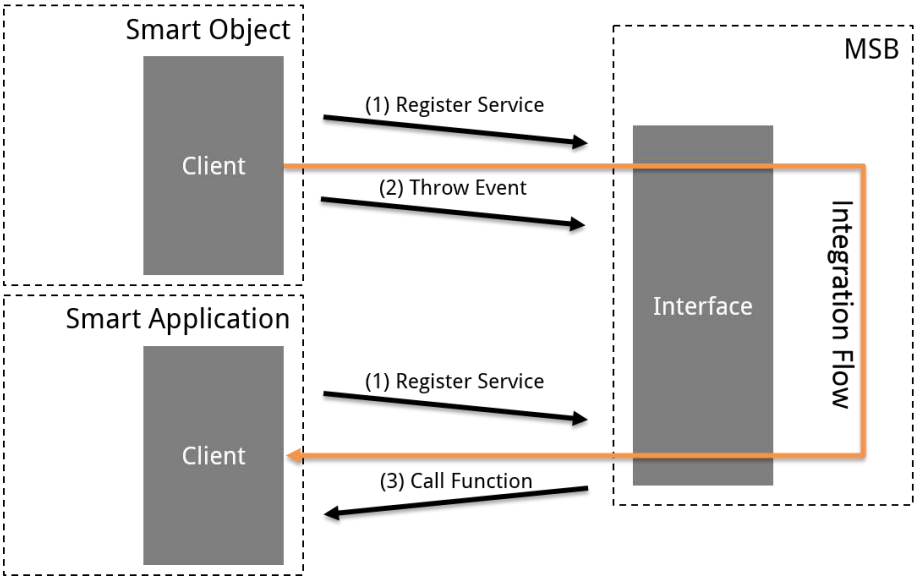


Figure 75: Exemplary pattern for data transfer of smart object to an application

For WebSocket communication ready-to-use client libraries in several programming languages are available, which developers of smart objects and applications can use to connect their own product to the MSB.

2.3.3.2 Meta data format

The data format of events and functions is shown in next tables. It is based on the OpenAPI Specification 2.0 (fka Swagger Specification) derived from the JSON Schema for programming language independent definitions of data format. The complete OpenAPI specification that is used for the

Swagger-UI as well as for the applications JSON definition can be found under https://github.com/OAI/OpenAPI-Specification.

Table 5

Name	Type	Format	Comments
Integer	integer	int32	signed 32 bits
Long	integer	int64	signed 64 bits
Float	number	Float	
Double	number	double	
String, Short	string		
Byte	string	Byte	
Boolean	boolean		
Date	string	date-time	As defined by date-time - RFC3339

Table 6

Name	Type	Format	Comments
Array, List, Set	array	<items>	

Table 7

Name	Type	Format	Comments
Model	object	<properties>	

Table 8

Name	Type	Format	Comments
Reference	#/definitions/<Model>		

2.3.4 RESTful API

2.3.4.1 Registration

The MSB supports plain old REST to communicate with applications and smart objects. While the WebSocket interfaces allow to infer the connection state of connected smart objects and applications, the REST interface does not allow this. The reason for this is the stateless nature of REST interfaces.

The MSB's REST interface can be reached at port 8083, regardless of the cell the MSB resides in. The OpenAPI specification can be found at the same port under the path /swagger-ui.html. That means, that for MSB reachable under the URL msb.vfk.de, the REST API would be reachable under the URL msb.vfk.de:8083, with the API documentation available under msb.vfk.de:8083/swagger-ui.html.

The Swagger-UI is a documentation tool for APIs that provide an OpenAPI specification. The MSB provides such a specification. While the OpenAPI specification should provide enough information on the interface for basic use cases, it does not provide enough information for complex applications. This documentation serves to supplement the Swagger-UI specification.

As shown in Figure 76, the self-description described in Figure 74 has been extended to support the integration of the endpoints of a RESTful application. A RESTful application can be registered to the MSB in two different ways. One possibility is to use the MSB GUI and the other one the REST API of the MSB.

Listing 1: An example for the specification of a complex object

```
{
	"dataObject": {
		"$ref": "#/definitions/alarm"
	},
	"alarm": {
```

```
                "type": "object",
                "properties": {
                        "reason": {
                                "type": "string"
                        },
                        "errorCode": {
                                "type": "integer",
                                "format": "int32"
                        },
                        "machine": {
                                "$ref": "#/definitions/machine"
                        }
                }
        },
        "machine": {
                "type": "object",
                "properties": {
                        "id": {
                                "type": "integer",
                                "format": "int64"
                        },
                        "name": {
                                "type": "string"
                        }
                }
        }
}
```

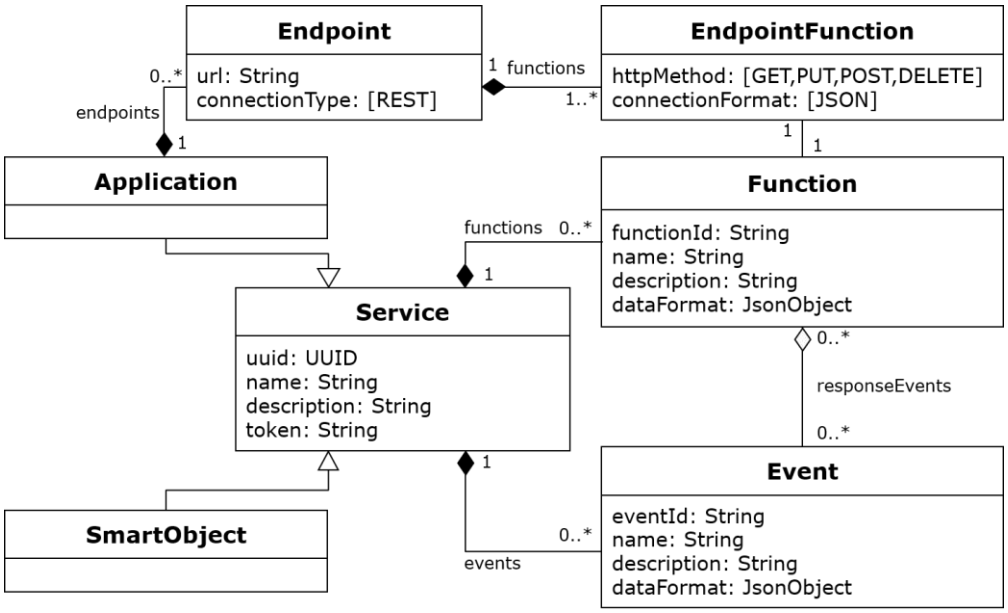


Figure 76: Extended self-description for RESTful API

2.3.4.2 Registration with REST API

The REST API two endpoints: One for the registration of an application (Figure 77) and the other one for the registration of Smart Objects (Figure 78). The self-description of the application that should be registered must be described as a JSON object. Listing 2 shows such a JSON description of a simple application and is meant to serve as an example. The fields contained in the JSON definition are described in Table 9

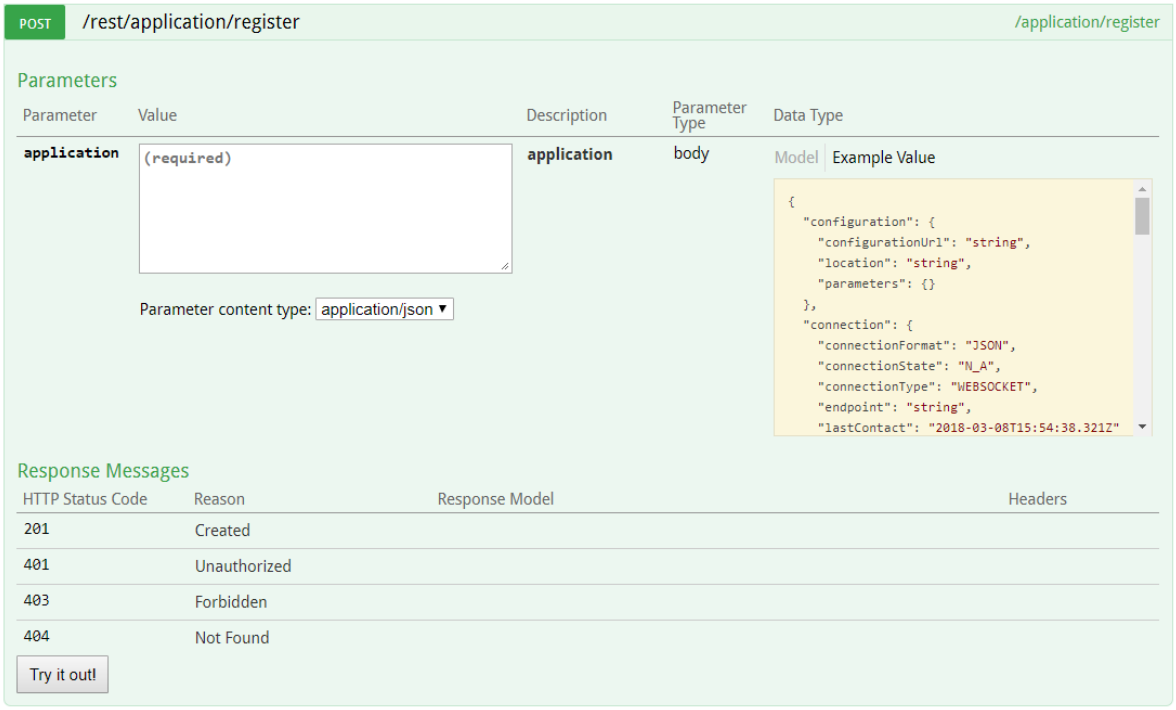


Figure 77: REST endpoint to register application

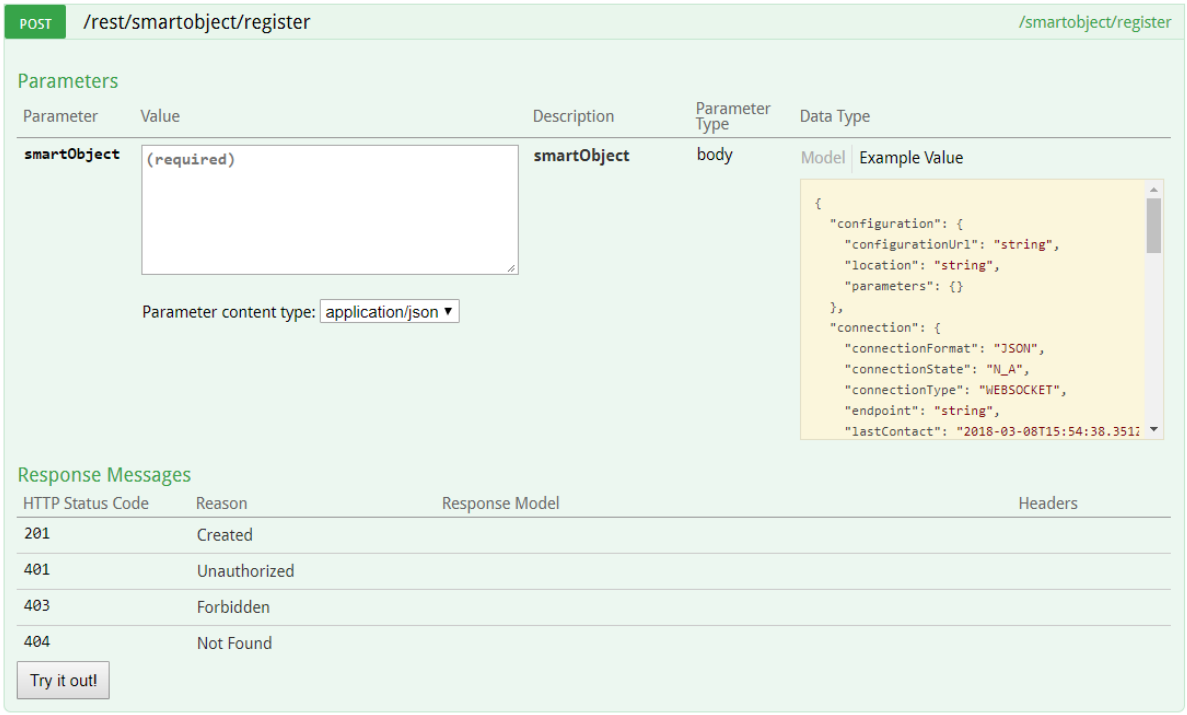


Figure 78: REST endpoint to register Smart Object

Listing 2: Sample self-description of REST application as JSON object

```
{
    "@class": "Application",
"uuid": "71f747a8-b12e-476a-bdb7-85c68c59c282",
"name": "System Information",
    "description": "Provides information about a remote system",
    "token": "auniquestring",
    "events": [
        {
```

```
        "@id": 1,
        "dataFormat": {
          "dataObject": {
            "type": "object",
            "properties": {
              "system": {
                "type": "string"
              },
              "name": {
                "type": "string"
              }
            }
          }
        },
        "description": "Displays live information about a remote system.",
        "eventId": "live-information",
        "name": "Live System Information"
      }
    ],
    "functions": [
      {
        "@id": 1,
        "functionId": "/system-information",
        "name": "System Information",
        "description": "Provides System information",
        "dataFormat": {},
        "responseEvents": [
          1
        ]
      }
    ],
    "endpoints": [
      {
        "url": "http://www.example.com",
        "connectionType": "REST",
        "functions": [
          {
            "httpMethod": "POST",
            "connectionFormat": "JSON",
            "function": 1
          }
        ]
      }
    ]
}
```


Table 9: Description of fields contained in JSON object of a self-description

@class	Can either be Application or Smart Object, depending on which type of object is described.
uuid	A unique identifier for the application. A valid identifier can be generated under https://www.uuidgenerator.net/version4.
name	The name of the application that will be displayed on the MSB GUI.
description	A textual description of the application
token	A token that will be entered in the MSB GUI to complete the registration of the application
events	A JSON description of the events that the application provides to the MSB. The defined events must provide in incremental numerical id, as well as a unique textual event id. The numerical id is used to refer to the event as a response event in a function definition, while the textual id is used to route the events information within integration flows. The data format follows the OpenAPI specification 2.0. The outer object of the data format must be called dataObject and must be of type object. Everything within the dataObject is optional and can be defined by the developer.

functions	A JSON description of the functions that the application provides to the MSB. The defined functions must provide in incremental numerical id, as well as a unique textual function id. The numerical id is used to refer to the event as a response event in a function definition, while the textual id is used to route the function information within integration flows. Additionally, the functionId is the path that is attached to an endpoint. If a function can be reached under the URL www.example.com/someFunction, the functionId must be /someFunction. The numerical ID is used under endpoints to connect endpoint definitions with functions. The data format follows the OpenAPI specification 2.0 and is completely developer defined.
endpoints	A description of the endpoints under which the application can be accessed by the MSB. A URL and a connection type must be provided. The functions section further describes how the functions can be accessed by the MSB. The function attribute in the specification refers to the @id attribute of a function defined in the outer application description scope.

2.3.4.3 Registration with MSB GUI

A REST Application can be also added using the MSB GUI. Therefore, you must select the APPLICATIONS tab and press the “+” Button in the left corner. In the pop-up you must press “Create App” and the “Manual app creation wizard” as shown in Figure 79 will appear. In Step 1 an UUID is automatically generated and you can enter other basic information like the name and the description of the application.

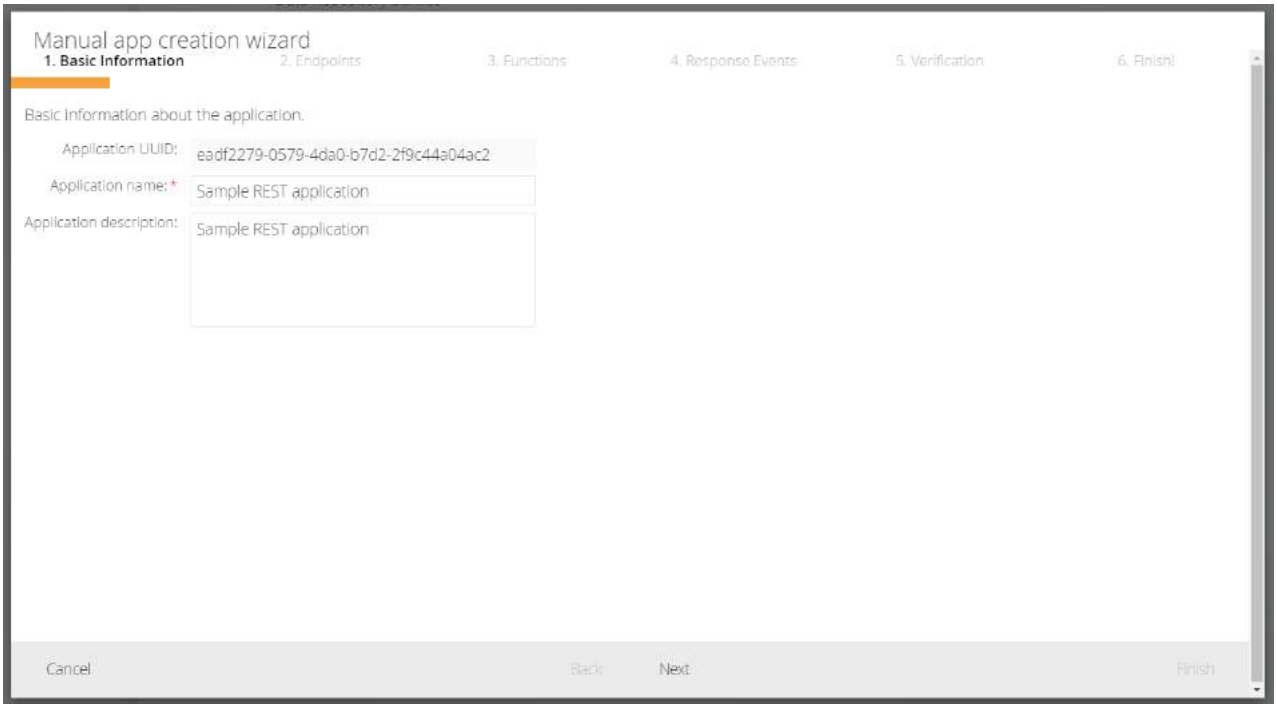


Figure 79: Manual app creation wizard – Step 1: Basic Information

In Step 2 (see Figure 80) the URL of the REST endpoints must be defined.

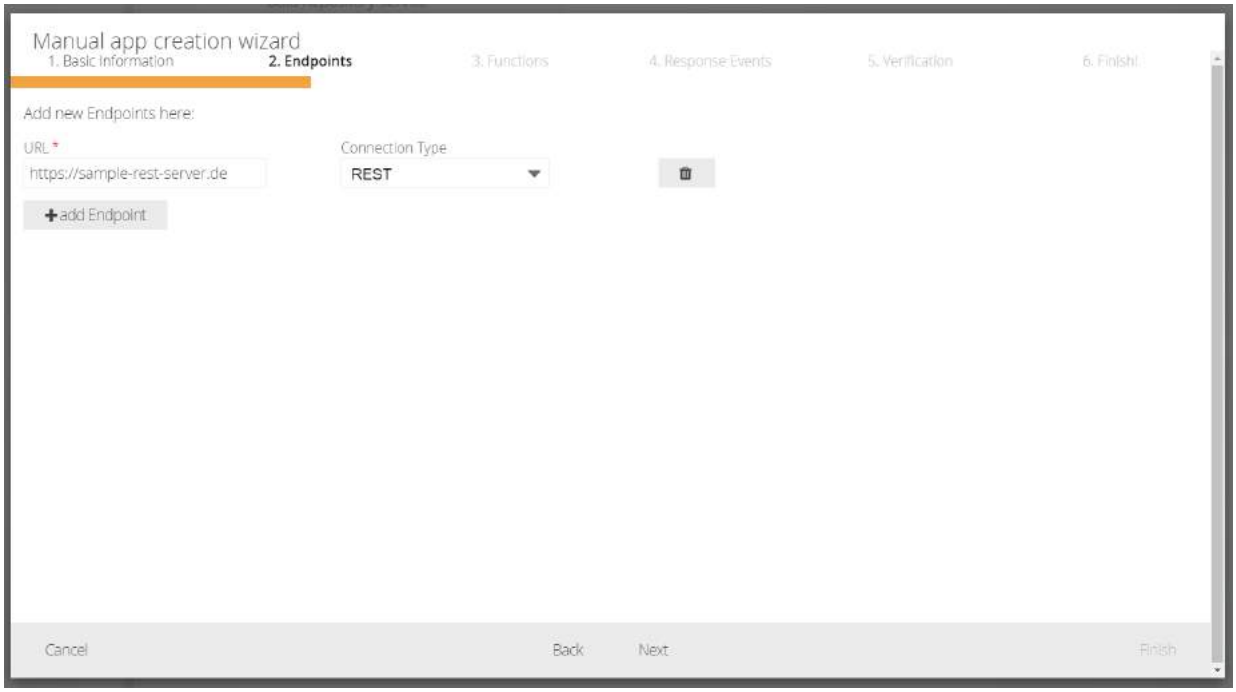


Figure 80: Manual app creation wizard – Step 2: Endpoints

In Step 3 (see Figure 81) for each defined REST endpoint functions can be defined. The path can contain parameters in the form of {parameter1}. This data format of the parameters must be defined in the "Request Schema" as described in section 2.3.3.2. If the function is called via an integration flow the parameter can be mapped from the triggering event. The parameter will be replaced with the value of provided by the event and the REST call will be executed.

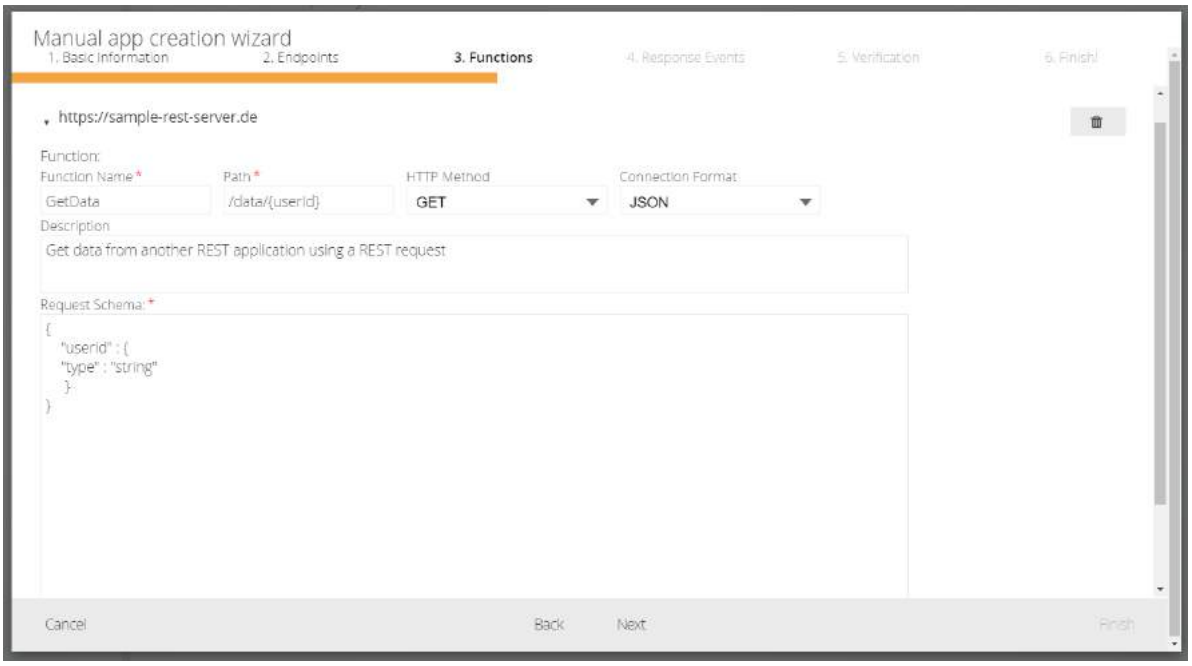


Figure 81: Manual app creation wizard – Step 3: Functions

The response of the executed REST call will be send as a response event. The response events for the functions can be defined in Step 4. In the "Response Event Schema" you must describe the data format of the data that will be responded by the REST application (see section 2.3.3.2).

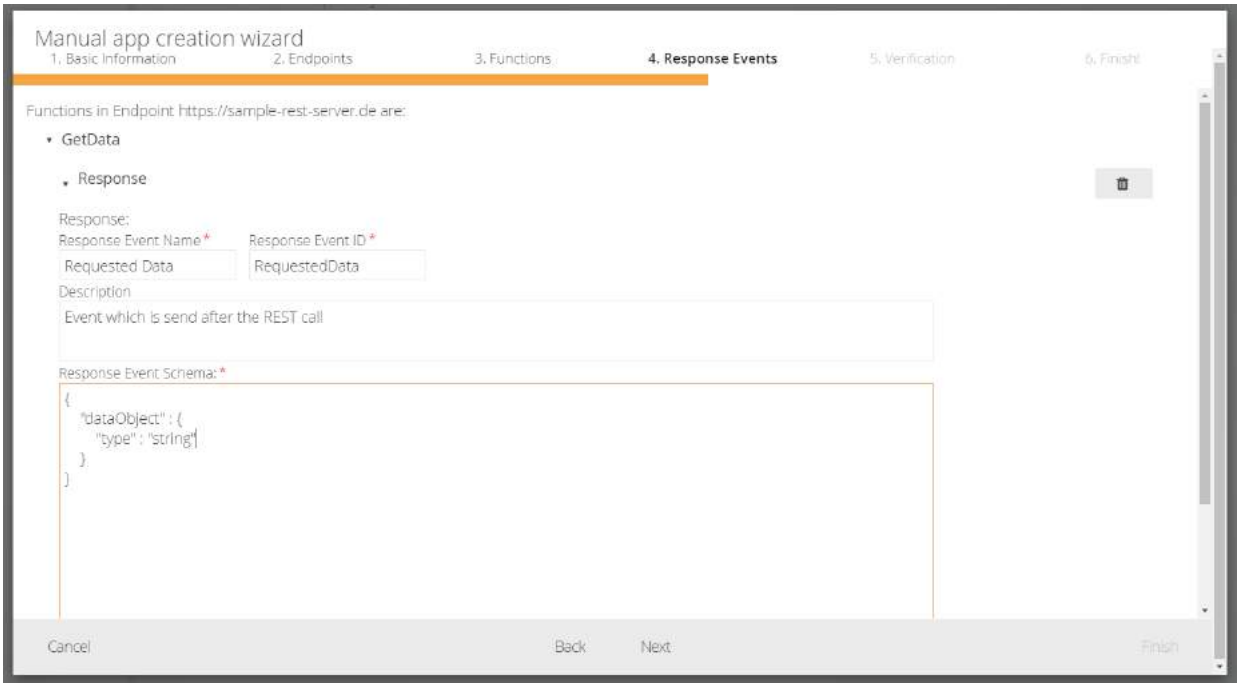


Figure 82: Manual app creation wizard – Step 4: Response Events

In the last two steps you can verify your input and finish the wizard. After that the application will automatically appear in the applications list.

2.3.4.4 Send data

Once your application has been registered and verified to the MSB it is ready to receive information via a function call and to send information to the MSB via an event. An event is send as JSON object with the fields described in Table 10.

Table 10: Description of fields contained in JSON object of an event

uuid	UUID of the Service that sends the event.
eventId	Id of the event as defined in the self-description of the Service.
priority	Priority with which the event is to be processed by the MSB
dataObject	JSON object that contains the data of the event.

The event is then sent to the REST endpoint (/rest/data) shown in Figure 83.

POST /rest/data /data

Parameters

Parameter	Value	Description	Parameter Type	Data Type
incomingData	(required)	incomingData	body	Model | Example Value

Parameter content type: application/json

```
{
  "dataObject": {},
  "eventId": "string",
  "postDate": "2018-03-08T15:54:38.341Z",
  "priority": "0",
  "uuid": "string"
}
```

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	OK		
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out!

Figure 83: REST endpoint to send data

2.3.4.5 Registration and verification of a component

Upon first boot up of a service, the service connects to the MSB and provides a self-description (more on how to achieve this and details on the self-description in section 2.3.3.1). At this point the component is not yet visible to the user, who needs to activate it first. To do so the user has to navigate to the *SMART OBJECTS* or *APPLICATION* tab, depending on the self-description of the component and click the button for new components ("+"), as can be seen in Figure 84. The classification into smart objects or application is based on the self-description, set by the developer. Its purpose is to allow easier distinction for human users and has no further implication beyond that. As a rule of thumb, smart object should contain at least one sensor or actuator.

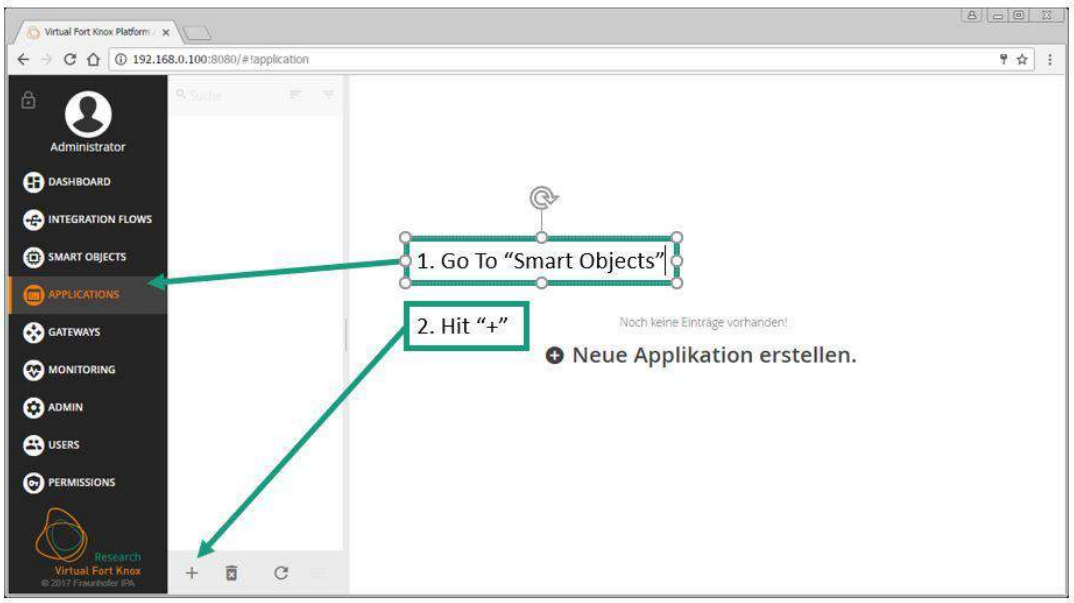


Figure 84: Activate new Component in GUI of MSB

The user is then asked to verify the component by inserting the security token which is part of the self-description and should be provided by the developer. At this stage the user can also decide if the component should only be visible to him or the entire organization (see Figure 85 below). Visibility decides if other user can see the component when they connect to the MSB with their user credentials. If they do not see the component, they cannot set up new information flows including the component. However, they still might be affected, as information flows set up by one user by trigger actions in organization wide visible components. An example might be a component which takes a considerable amount of time to process data and blocks any other request in this state. One user may find the component permanently blocked when setting up his information flow without any capability to identify who is blocking the component, since he cannot see the integration flow of another component which is invisible to him.

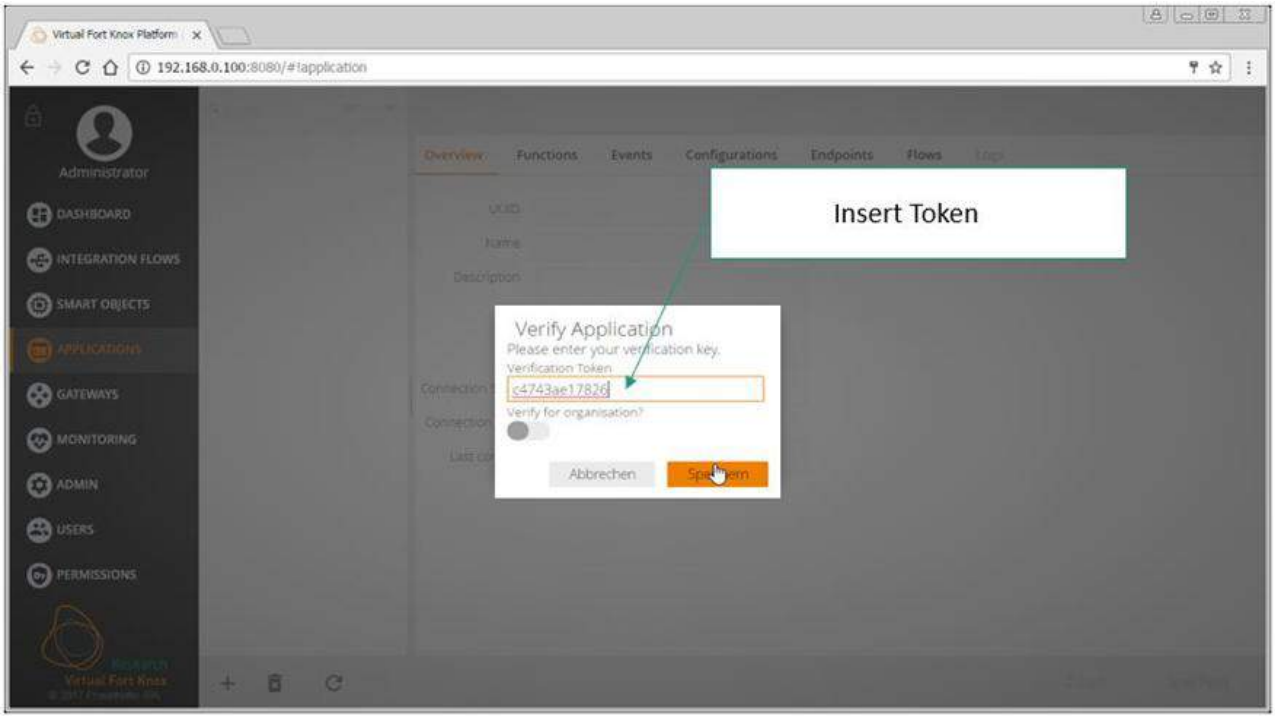


Figure 85: Security Token and Visibility Setting (MSB GUI)

2.3.4.6 Reviewing of registered components

After completion of the registration process, the new component will be listed in the appropriate tab (*SMART OBJECTS* or *APPLICATIONS)* with its self-description. This description includes general information like name and prose description of its general purpose, as well as specific information regarding the outgoing events the component can throw and functions which can be linked to incoming events, as seen in Figure 86. The component is now ready for the modelling of information flows. If the user wishes to delete the component, he can do so at any time by selecting the component in the respective tab, clicking the small garbage bin icon in the bottom left and confirming his decision.

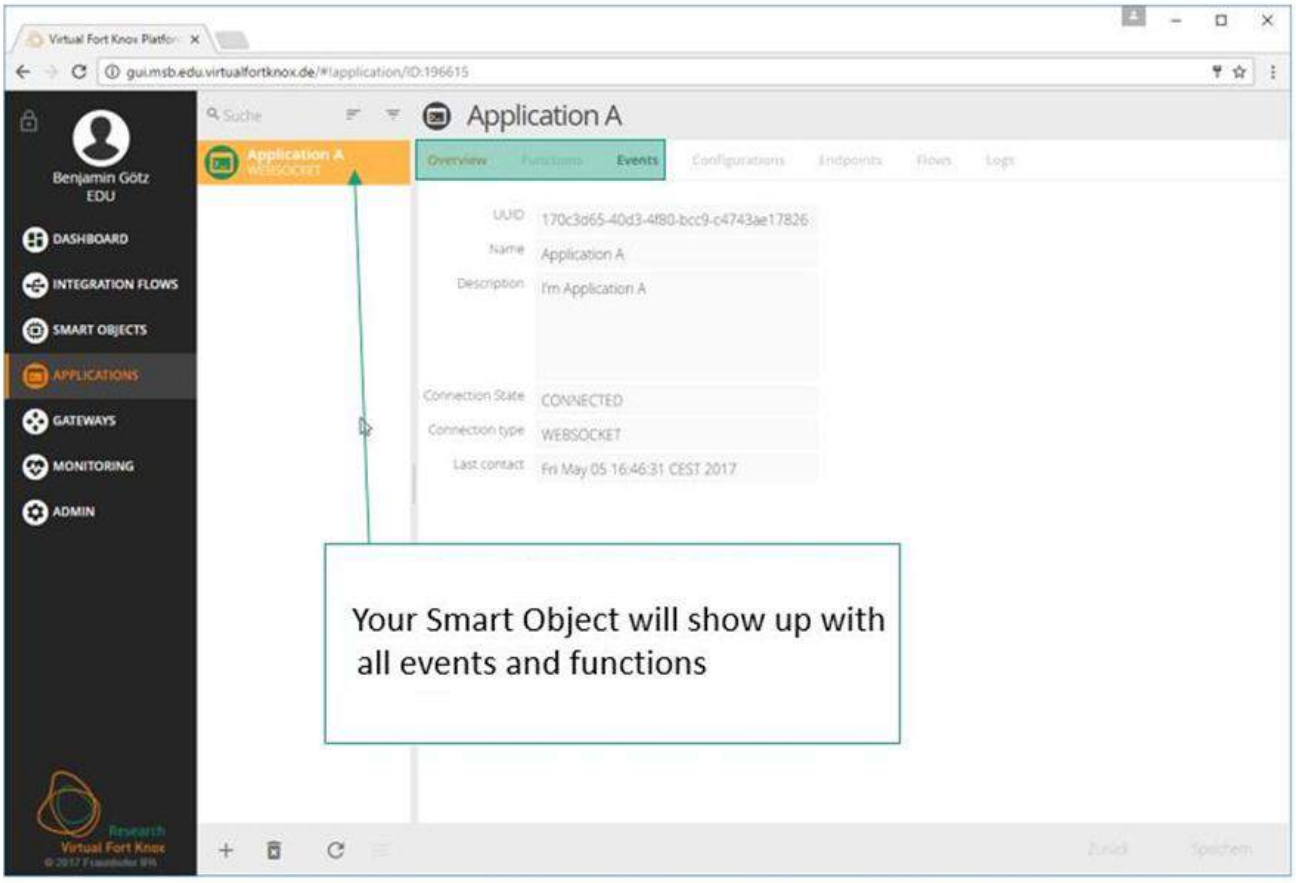


Figure 86: Detailed Information about Component (MSB GUI)

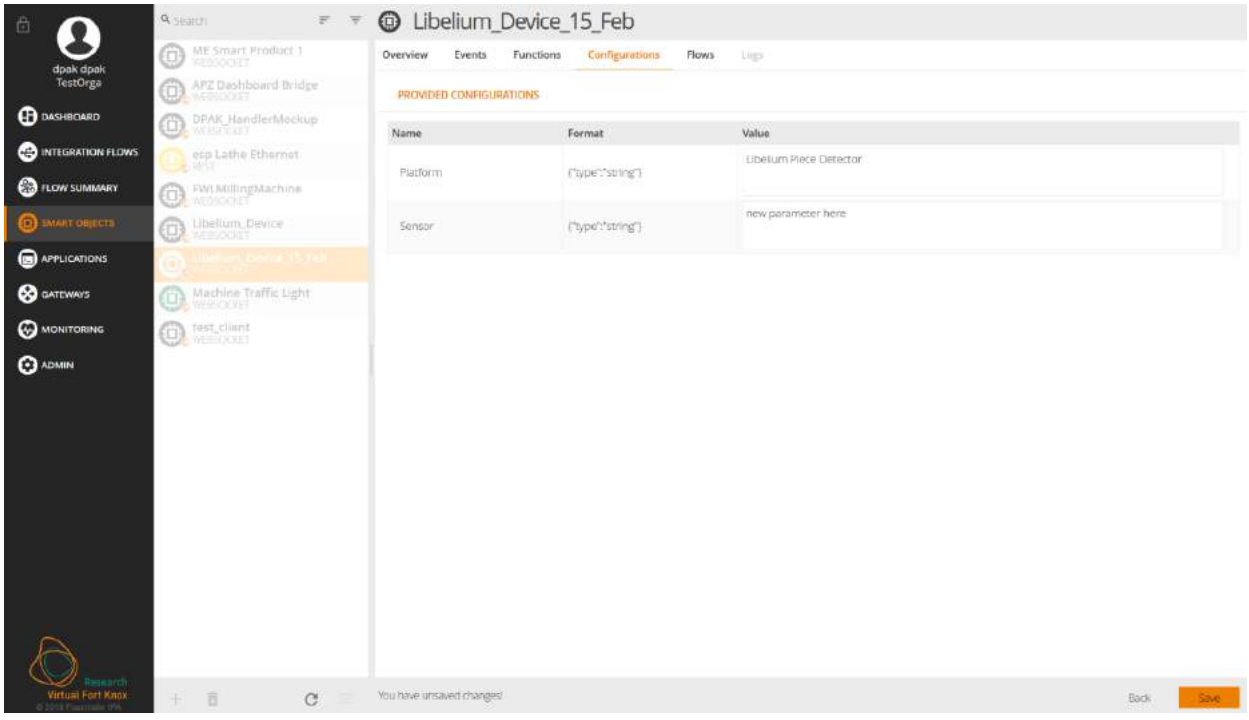


Figure 87: Detailed View on Component - Configurations Tab

2.3.4.7 Settings configuration for components via MSB

Optionally, the configurations tab, shown in Figure 87, allows the user to configure internal values of a component remotely via the MSB. This feature is optional and has to be set up by the developer in the program code of the component. If no remote configuration is allowed, the tab name is greyed out and inactive. New parameters can be set by change the value on the right and pushing the orange save button on bottom right. If the new value does not correspond to the required format indicated in the middle row, the changes will not take effect. If a component is modified which is currently offline, the changes will not take effect.

The *Flows* tab show all integration flows which the component is currently a member of (see Figure 88). When activating a new component, this tab is empty. Clicking on the arrow on the right lets the user jump directly to the selected flow.

Figure 88: Detailed View on Component - List of all associated Flows (MSB GUI)

2.3.4.8 Data routing with integration flows

2.3.4.8.1 General information about integration flows

On the one hand, the middleware approach replaces the otherwise required point-to-point connections between the components and reduce the maintenance effort for the IT-personnel. On the other hand, it takes over the function of the event listener for all components and allows the configuration of this function at run time. This simplifies the adaption process in case changes to existing solutions are required (e.g. replacement of an old component). It also simplifies the implementation of novel solutions, which rely on data or processing capabilities of existing components and can be configured at run-time without shutdown of the entire system.

To understand how to design components for the use with the MSB, it is useful to first understand, how the user configures his solution based on the available components. Figure 89 shows a simple example for such a solution from view of a user who configures two components to communicate. Once the integration process is complete, the left component can send information to the second one for further processing. The user who implements the solution used a building block concept in a graphical user interface to setup this connection. To achieve the desired information flow, three conditions need to be met, which are represented in Figure 90:

- The components are registered with the MSB (self-description is provided) and activated.
- The information flow is modelled in the MSB by the user.
- The information flow is triggered by the first component in the chain at run-time.

Figure 89: Simple Information Flow modelled in the MSB GUI (only part of GUI is shown)

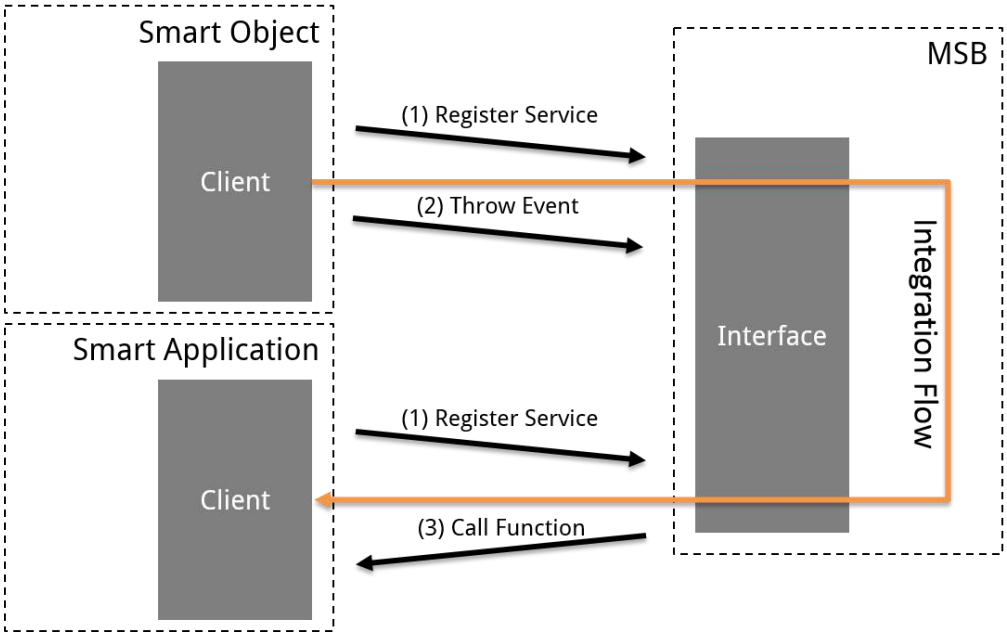


Figure 90: Steps to compete the Exchange of Information

It is important to understand that each component solely communicates with the MSB and does not per se know preceding or subsequent component in the chain of information. This has implications on the design as intrinsic knowhow between components cannot be expected and should therefore be avoided and all contextually required information needs to be available in the self-description or needs to be exchanged in the events.

2.3.4.8.2 Modelling of an Information / Integration Flow

The MSB is technically able to map events from a component onto functions of the same component. However, this capability should not be used in general to keep the load on the MSB low. Besides, the latency of the MSB typically exceeds component-internal communication by a large margin due to the underlying IP-based communication.

2.3.4.8.3 Initial Creation of an Integration Flow

To build a meaningful information flow, at least two separate components are required. In context of the MSB a model for an information flow are called *integration flow*. Figure 91 shows the first step in creation of such a flow in the *INTEGRATION FLOWS* tab which is initiated similarly to the activation of a

component. Once the blank flow is created, it needs to be named, while a description by the user is optional. The modelling can then be initiated by clicking the *Flow Designer* tab (3. in Figure 91).

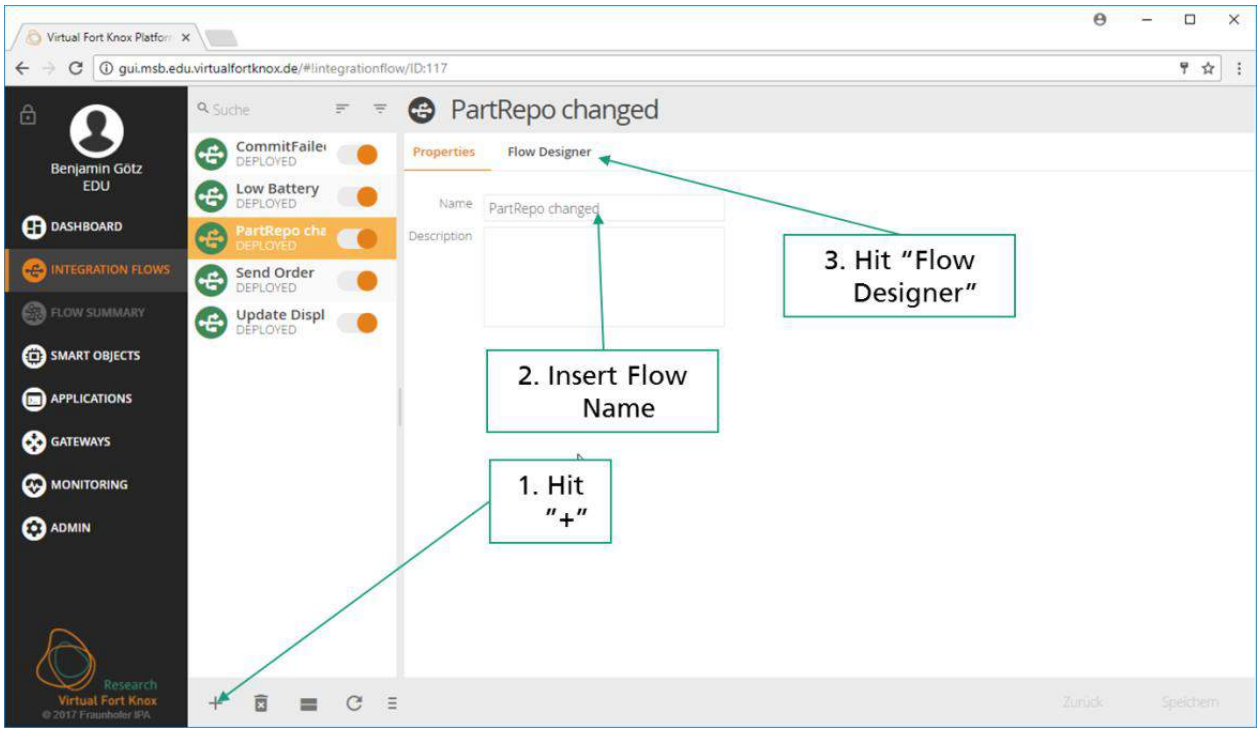


Figure 91: First Step in Creating a new Integration Flow (MSB GUI)

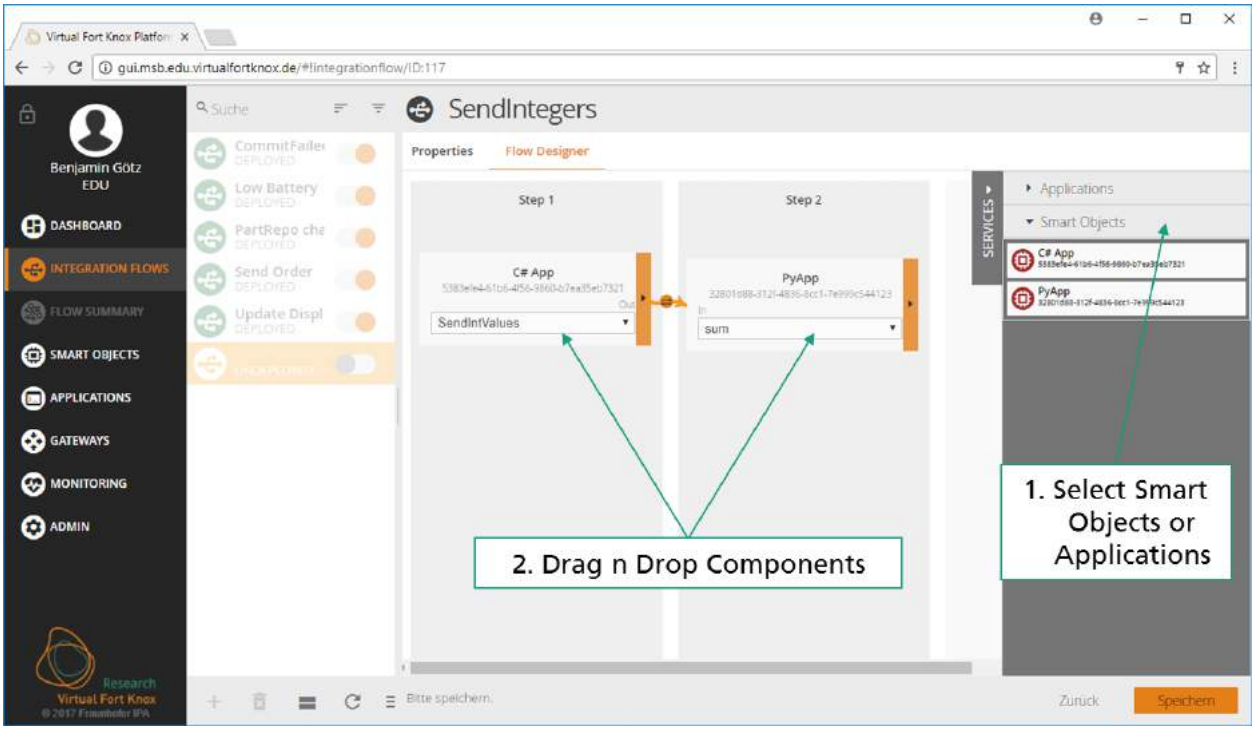


Figure 92: Drag and Drop of Components to initiate the Creation of a new Integration Flow

2.3.4.8.4 Selection of Components for an Integration Flow

Within the flow designer view, all available components are show on the right side. If the desired component is not shown, the list can be extended to show all smart objects or all applications by clicking on the respective fields. If the component cannot be found, it has not been activated and the steps of section 2.3.4.5need to be competed again. The components, required for the integration flow need to be dragged and dropped onto the main area, as indicated in Figure 92.

After dragging all components into the main area of the GUI, the user should check if all components are positioned in the correct order according to the desired information flow from left (first component in chain) to right (last component in the chain). This is not necessary but advised, as it improves readability.

2.3.4.8.5 Selection of required Events and Functions for each Component

Before connecting the components, the user needs to select the appropriate event and function for components he wishes to connect next. This is done by clicking on the drop-down menu for the component and then clicking the desired event or function as indicated in Figure 93. By default, the first event from the list in the self-description is selected for all components. If a component does not supply events, the first function from the list in the self-description is set as default. Selecting a function manually may result in a second drop down menu to appear next to the previous one as can be seen in Figure 94.

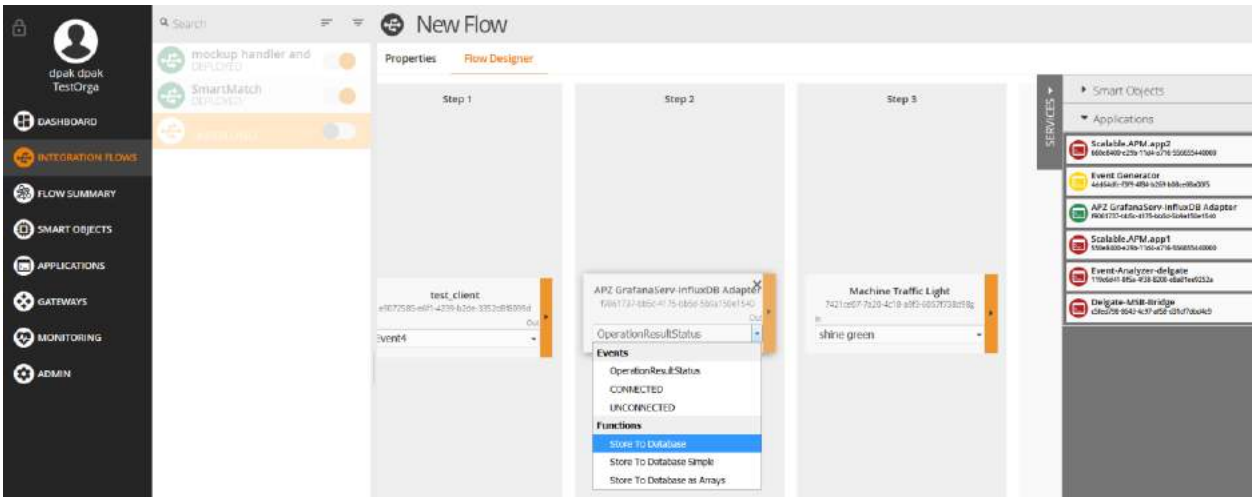


Figure 93: Selection of Event or Function from Drop Down Menu

This implies that the selected function may trigger one of the listed events. The user has to select the desired event from this new list, unless the component is the last one in the chain, where the output event is irrelevant. Due to this behaviour of the GUI it is strongly advised to begin the selection of events and functions at the last component in the desired integration flow.

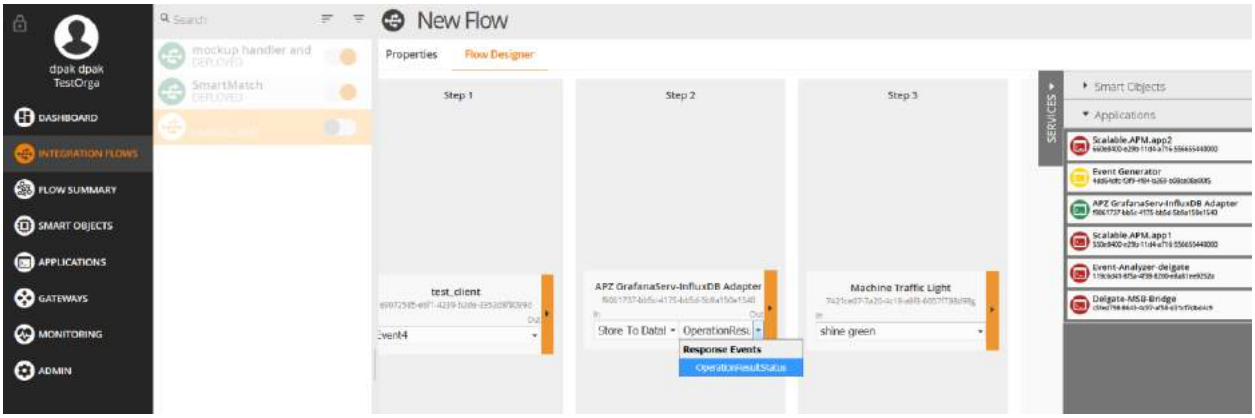


Figure 94: Selection of Return Event based on preselected Function

2.3.4.8.6 Linking of Events to Functions

After the selection process the links between the components have to be set up. Links are always initiated from an event towards a function. The user achieves this by clicking on the orange area of a component with the event and dragging the mouse to the component with the function which he wishes to link to. A successful link is indicated by an orange arrow between the two components, where the arrow is directed towards the component which's function should be executed. An example for successful links can be seen in Figure 95.

Figure 95: Successfully linked Components

2.3.4.8.7 Mapping of Event Data to Function Parameter Inputs

If the selected functions require no input parameters, specific mapping of parameters can be forgone and the flow is ready to be saved. However, in most cases a mapping of data from the event to the input parameters of the corresponding function is required. The user does this by clicking the small orange dot in the middle of each arrow which will result in a similar GUI to Figure 96. On the left, all available data from the event is displayed and on the right all input parameters are shown. The type is displayed for every parameter as well.

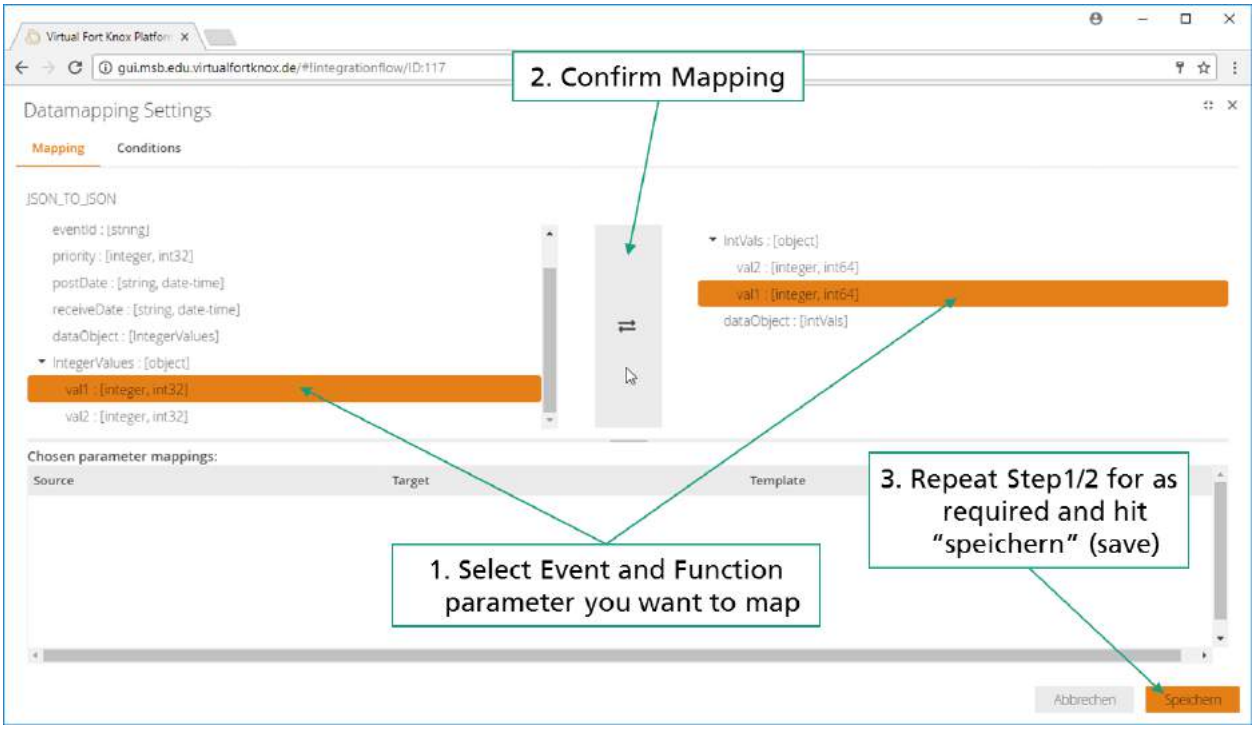


Figure 96: Detail View for Mapping Event Data to Input Parameters for a Function

The MSB is capable of simple type casts which require no further specification such as integer to float, integer to string. Most input and all output parameters are specific to the component developers. However, the parameters mentioned in Table 11 are always available for mapping on the side of the event. The mapping is conducted by selecting the input and output parameters which are supposed to be mapped, indicated by an orange background and clicking the button with the two arrows in the middle. Once the mapping is completed it will show up in the bottom half of the screen, were it can also be deleted by clicking the icon with the garbage bin. Once the user has mapped all function input parameters to the respective event output parameters, he can click the orange save button in the bottom right to save his changes. In general double mappings should be avoided, as they can lead to confusion. If a double mapping occurs, the latest mapping, indicated by being lower in the list, takes precedence. The mapping process has to be repeated for all links (all orange arrows in Figure 95), where mapping is required.

Table 11: List of standard event properties

uuid	UUID of the component which sends the event.
eventId	ID given to the event by the developer
priority	Priority set by component developer for transfer by the MSB which might be relevant in case of high load. Possible values are: 0-low / 1(default)-medium / 2-high
postDate	Time when event was thrown by the component.
recieveDate	Time when the event was published to the MSB. The distinction is relevant when a component is set up in such a way that it can function autonomously without MSB connection (e.g. in remote regions without WIFI connection. Optionally events can be buffered in this case and published to the MSB once reconnected.

2.3.4.8.8 Setting Conditions for Data Transfer

In some cases, the information of an event should only be forwarded to a function when specific conditions are met, which were not foreseen by the developers who designed the components. In this

case the *Conditions* tab in the mapping view can be used to set conditions. Conditions can only be set for the data associated with the current event. Other conditions, for example including information from the previous event, are not configurable. In some cases, smart design of the integration flow using the branching (section 2.3.4.8.9) and merging (section 2.3.4.8.10) can be used to achieve the desired results in combination with conditions. A new condition is set by clicking the relevant input parameter form the associated event and selecting the parameter on the left side and clicking the large button in the middle. This yields a view with a drop-down menu like Figure 97 from which the desired comparator can be chosen. Once the user does so, he can set the compare value in a newly appeared field.

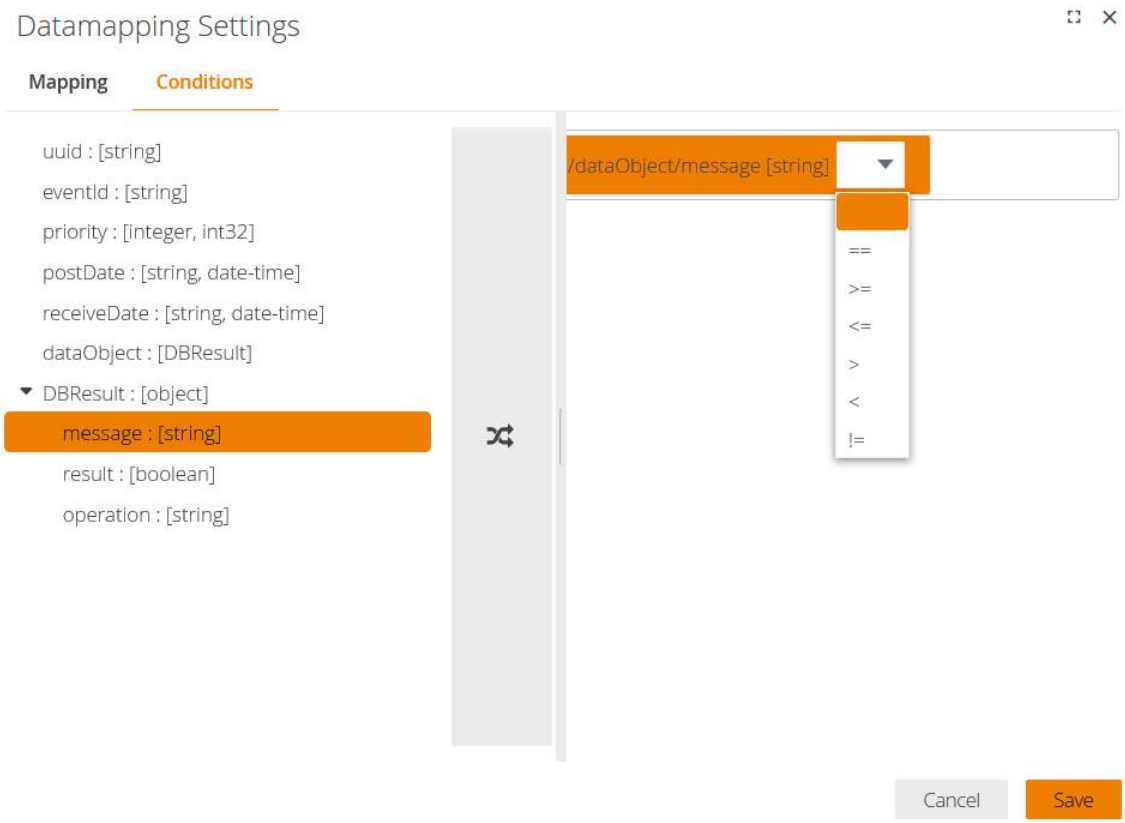


Figure 97: View to set Condition which incoming Events are forwarded to the next Component in the Flow.

2.3.4.8.9 Branching of Integration Flows

The length of integration flows, as in the number of event to function links, is largely unlimited. The MSB allows more complex designs as well, beside strictly linear integration flows. It is possible to map one event to several functions of one or more components, effectively creating branches in the flow, as shown in Figure 98. In case a subset of these functions is from the same component, the component needs to be dragged and dropped once for each individual mapping. If the branches do not merge again, the user should decide if two separate integration flows (with two separate names) would improve the overall overview.

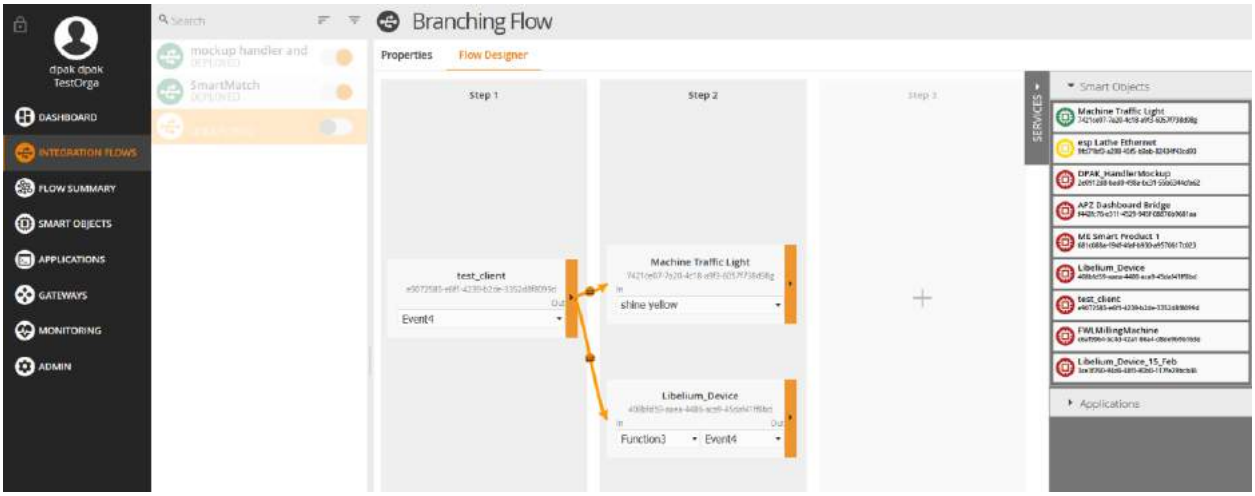


Figure 98: Branching Integration Flow where one Event is forwarded to Two Components

2.3.4.8.10 Merging of Branches in Integration Flows

The merging branches works in a similar fashion as linking and mapping (see Figure 99 for the final view after linking). Once the data of the first event is mapped to the input parameters and saved by clicking the button, the mapping of the data of the second input event can be conducted accordingly. The mappings of the other event will appear in the list of existing mappings (lower half of Figure 96) and vice versa.

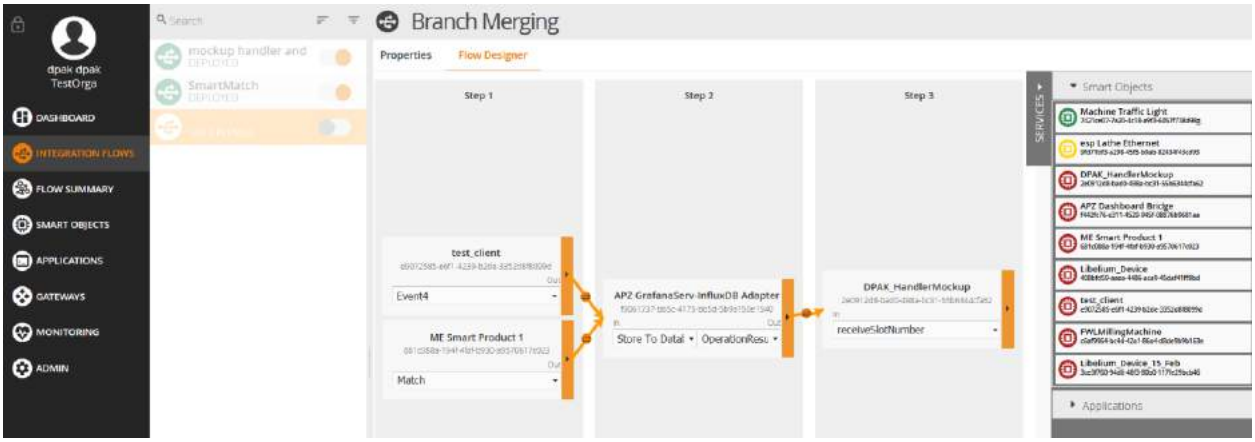


Figure 99: Merging of Branches in an Integration Flow

2.3.4.8.11 Wrap up and Saving the Integration Flow

The final steps before the integration flow can be activated, is saving the current setting and activating the flow. Before doing so, the user should be sure that the steps described before are completed. For better overview they are mentioned here once again:

- Name and description represent the purpose of the integration flow sufficiently.
- All components are in the main area of flow manager (at least one square for each component).
- For each component the correct function / event is selected (lower half of the graphical representation of the component).
- All required links are in place (orange arrows).
- All mappings are set as required for each link.
- All conditions (if required) are set (can only be checked in the respective detail views by clicking the orange nobs in the middle of the arrows and switching to the *Conditions* tab).

If this is the case, the user can click the orange save button on the bottom right of the browser window. As a result, the name of the integration flow in the list of flows on the left half of the screen will change from light gray to dark grey, as seen in Figure 100.

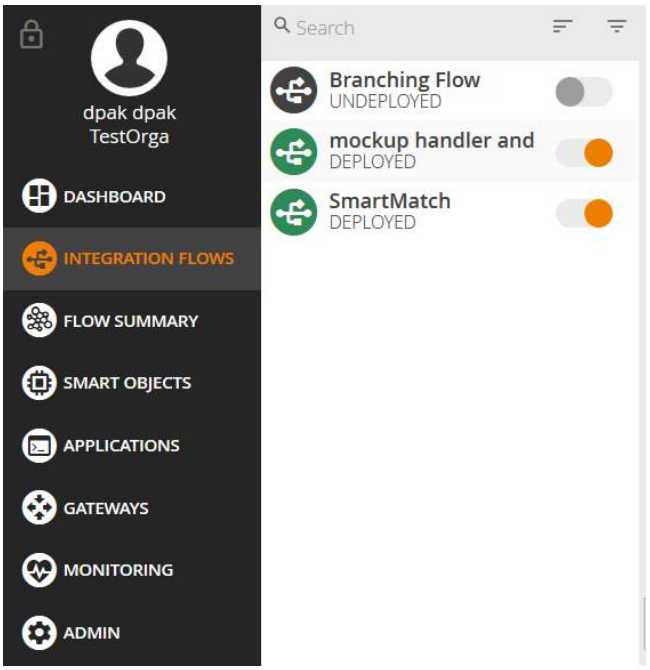


Figure 100: Integration Flow after Saving

By default, the integration flow is still deactivated. To activate it, the user has to hit the toggle button to the right of the name of the integration flow. The status of the flow can also be seen by a quick glance on the colour of the icon to the left of the name. Activated flows are indicated by a green icon, while deactivated ones are indicated by a black icon. The activation of the integration flow in the backend of the MSB takes between 1 s – 10 s.

3 Step by step implementation of test scenario for Application Experiment: Agrifood

This section presents a detailed step-by-step guide for integration with the system, which implements the test scenario for Application Experiment: Agrifood, as described in section 1.2.

3.1 A general overview

The integration flow among the modules that are described in section 2 is depicted in Figure 101. The section below provides further information for each step.

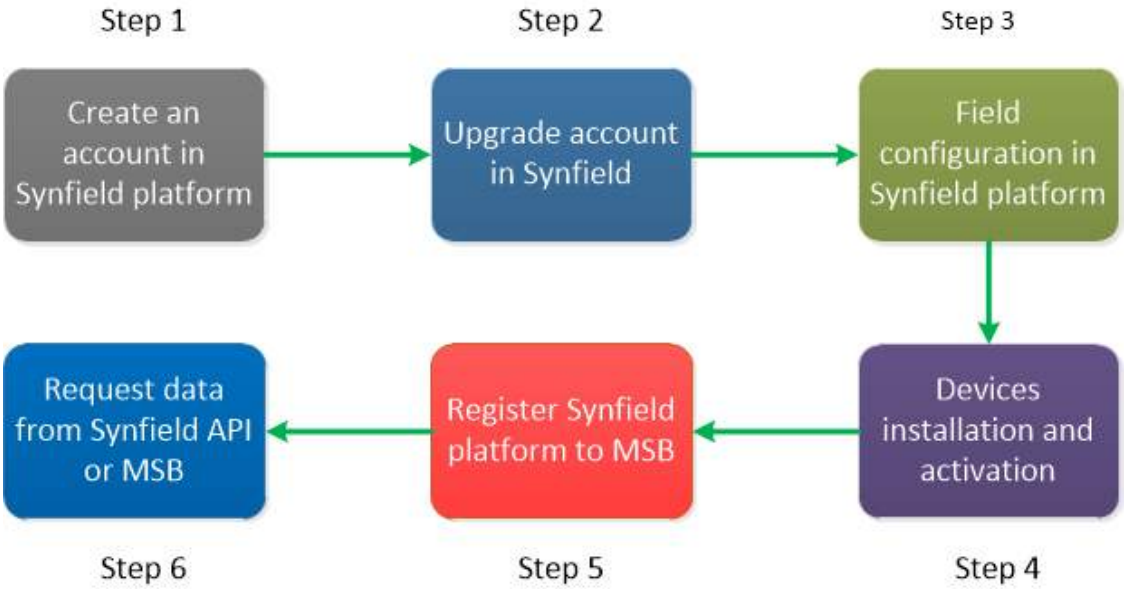


Figure 101: Integration steps

3.1.1 Step 1: Create account in Synfield platform

The creation of the account in Synfield platform is necessary. The procedure is described in section 2.2.1.

3.1.2 Step 2: Upgrade the account

The procedure is described in section 2.2.2. After that the user can register Synfield nodes in the platform and use the Synfield API.

3.1.3 Step 3: Configure the fields in Synfield platform

Follow the guidelines in sections 2.2.3 - 2.2.7 to complete this step. This step is very important since it associates the user with his/her fields and the network of devices.

3.1.4 Step 4: Enable devices to send data

Install the Synfield and the Plant-O-Meter devices in the fields and activate them to upload the measurements in the Synfield platform through the Synfield API. This step is also significant since the sensor measures are stored in the Synfield database. The user is able to compose the plots of his/her

interest as shown in section 2.2.8, create/enable rules as described in the section 2.2.10 or detect the spread of diseases as described in section 2.2.9.

3.1.5 Step 5: Register Synfield Platform as application to Manufacturing Service Bus

To integrate the Synfield Platform into VFK the REST API must be mapped to the MSB. The Synfield Platform is represented in the MSB as an Application. Each REST endpoint of Synfield Platform is mapped to an endpoint and function of the MSB Application. The response of the REST call is represented by a response event. The available endpoints of the Synfield REST API are depicted in Figure 44. The representation in the MSB Application is shown in Figure 102. For instance, the endpoint "*/en/api/v1/nodes/{sn}/sensing_services*" is mapped to the function "*Get Sensing Services*". The path parameter "sn" is passed as a parameter in the function call. How to define such a MSB Application is described in detail in section 2.3.4.1. The data format of the response event must be defined as described in section 2.3.3.2.

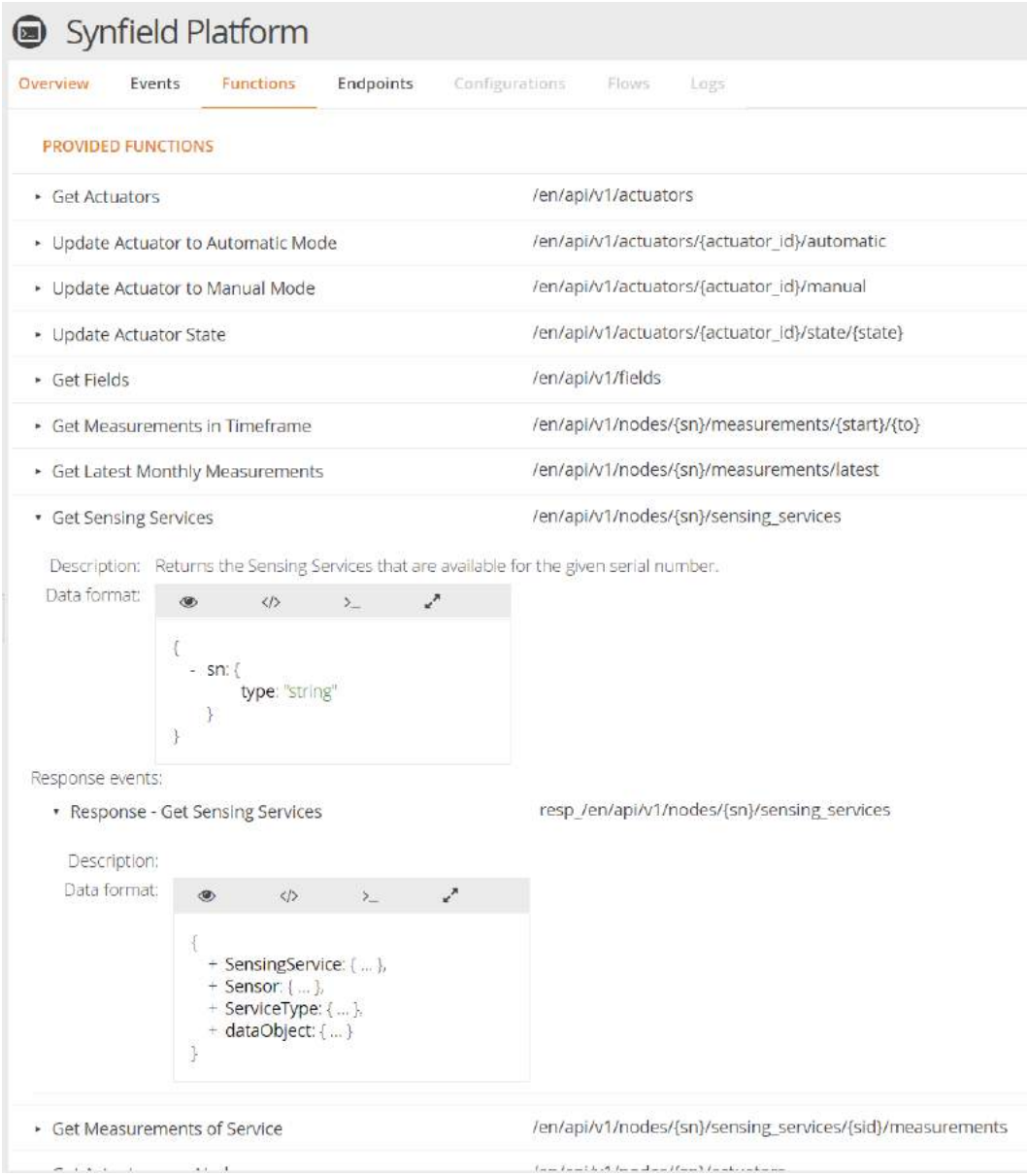


Figure 102: Synfield Platform as Application in MSB

After the Application is defined it can be registered to the MSB with a POST request sent to https://msb.vfk.de:8083/rest/smartobject/register. The registration process is described in detail in section2.3.4.1. If the registration has been successful, the HTTP Status Code 201 will be returned and the Application can be verified in the MSB GUI with the token defined in the self-description (as described in section 2.3.3.1. If the POST request returns an HTTP ERROR code or the verification is not possible you should check if the self-description is defined correctly.

POST	https://msb.vfk.de:8083/rest/application/register
Parameters	
Key	*Value*
smartObject	Self description as JSON.

Figure 103: POST request to register Synfield Platform as application to MSB

3.1.6 Step 6: Request data from Synfield Platform

A small sample application has been developed to show how the integrated Synfield Platform can be used within the MSB. The connection between these two applications is done via an Integration Flow (see Figure 104). The sample application sends the event "*Trigger – Get Sensing Services*" to trigger the function "*Get Sensing Services*" of the Synfield API. If this function is triggered, the proper web service of the Synfield REST API will be invoked and the response of this call will be sent as response event to the MSB. The response event can then be forwarded to the requesting sample application which receives the data through the call of the function "*Receive Sensing Services*". The creation of the integration flow is described in detail in section 2.3.4.8. This is just a very simple scenario to illustrate the integration of Synfield API into VFK. But it can be easily extended to include any additional services (e.g. databases, UIs, Machine Learning, …).

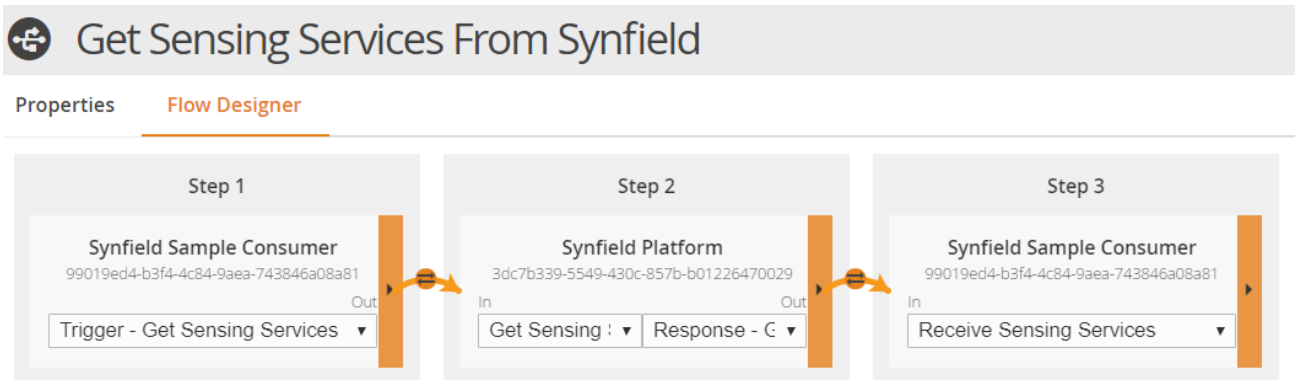


Figure 104: Integration flow to request data from Synfield Platform

4 Annex

4.1 Nitrogen rate experiment in maize

4.1.1 Field trial set up

A commercial field was selected for setting up the N rate experiment. The criterion for the field selection was the lack of fertilizer application within 2018. Three maize varieties were selected belonging to different maturity groups (FAO 300, FAO 400 and FAO 500). The N treatments were installed directly after sowing incorporating Urea in the soil using a wheat planter. The experimental deign contains five treatments (different N rates starting from 0 kg of N ha^{-1} until 200 kg of N ha^{-1}) replicated three times. Each experimental plot contains 4 rows (2.8 m wide) and is 12 m long.

4.1.2 Distribution of Head and peripheral nodes and soil moisture sensors

The head node of the SynField system will be installed at the borders between Var.1 and Var.2 (Figure 1) approximately in the centre of width of the setup in a location that will not interfere with the farming applications. Similarly, the peripheral node will be installed between Var.2 and Var.3. A moisture sensor will be installed in each replication of each variety (9 sensors in total) in the centre of the width of the setup at the plots receiving fertilization of 200, 0 and 100 kg of N ha^{-1}.

4.1.3 Description of measurement procedure with Plant-O-Meter

The measurements will start during the mid-growing stages of maize (around V8) and will continue until maturity. It is expected that measurements will be done every two weeks to monitor the plant status during different growth and reproductive stages of maize. Only the two middle rows in each plot will be measured and the rest will serve as guard-rows. Plant-O-Meter will be set to automatically measure and save results every second. It will be held at a constant distance from plant canopy and moving along the row to cover the whole row length moving up and down along the two middle rows. In addition to that, the second mode for taking stationary measurements in two concentric circles centred on the nodes will be tested.

4.1.4 Expected results in terms of the influence of nitrogen rate on vegetation and soil moisture content

Detailed mapping of plant vigour and biomass will be performed at different growth and reproductive stages of maize. The data from the two extreme and the median fertilizing plans (0, 100 and 200 kg of N ha-1) will be combined with the soil moisture content data derived from the soil moisture sensors. The plants in the non-fertilized plots are expected to appear less vigorous, paler and smaller compared to the other treatments with increasing intensity towards the high rate. The water uptake should be correlated to the biomass and plant vigour since the largest plants need more water. Therefore, it is expected to extract the relationship between crop vigour and biomass with the soil water availability. The system will be tested under real field conditions providing feedback for possible upgrade.

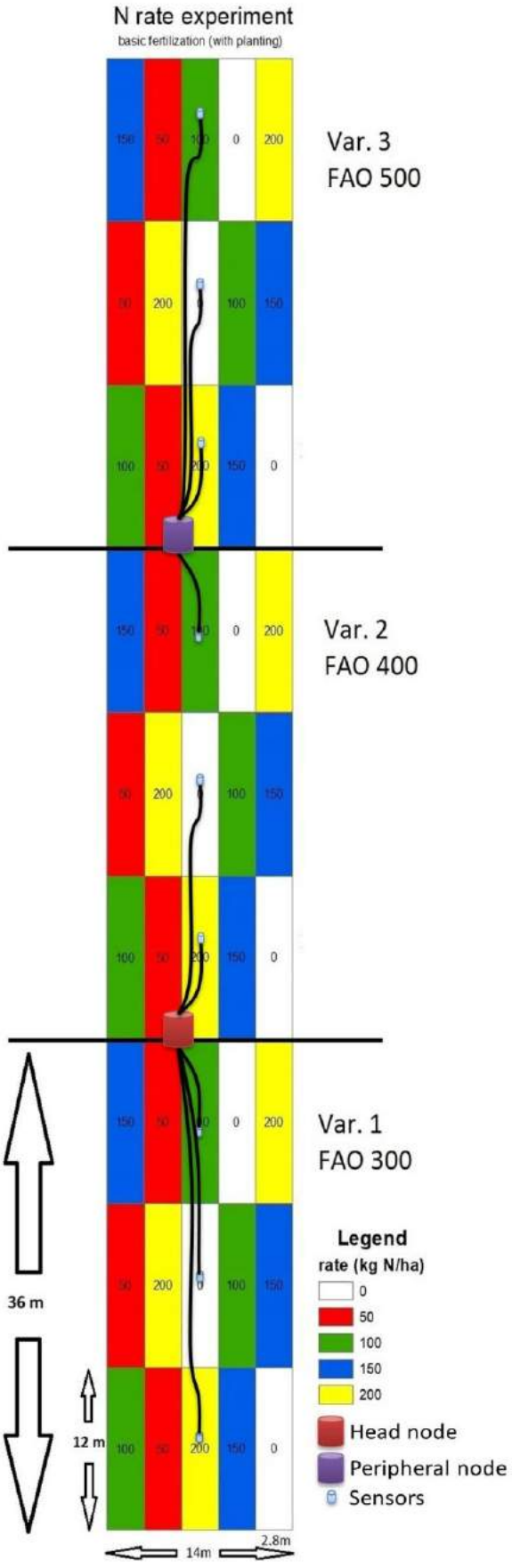


Figure 105: Nitrogen trial setup

4.2 Smart irrigation experiment

4.2.1 Field trial set up

The experiment will be conducted on a 6 ha maize farm, irrigated using solar pumping system. Field is of dimensions 200 x 300 m will be divided on three equal experimental plots. On each plot, different irrigation strategy will be applied. The first treatment will be “farmer’s practice” (FP) which is solely based on farmer’s experience. The second treatment will be based on soil moisture sensors readings “sensor-based irrigation” (SBI), and the third one will use SBI plus 20% more water, “sensor-based irrigation with excess water” (SBIE). Adjacent field will be used as a non-irrigated treatment and will receive the same management practices as the other three treatments.

All four plots will be treated with sufficient N rate that enables proper growth and development of maize. Only one maize variety, named Exxupery (Agrimatco) belonging to FAO 560 group, has been planted on whole field.

4.2.2 Head node placement and soil moisture sensor distribution

In the initial phase, the electromagnetic probe EMC-38 was used for soil scanning and detection of various properties, such as mechanical soil composition, compression and mineral composition. Based on the scanned results and the elevation model, the field is split to 4 "management zones" with homogenous characteristics (see Figure 106). Since it is expected that different water treatments and soil properties will generate different biomass and plant vigour that in turn will need different amount of water, a minimum of one soil moisture sensor will be placed per one experimental plot plus management zone. Based on this fact a total of 7 sensors will be used. Two sensors are going to be placed on each irrigated treatment and one on the non-irrigated.

The sensors will be integrated in SynField system and their readings will be delivered via head node to the SynCloud and be visible through SynField web app. The head node will be positioned not to interfere with the farming applications.

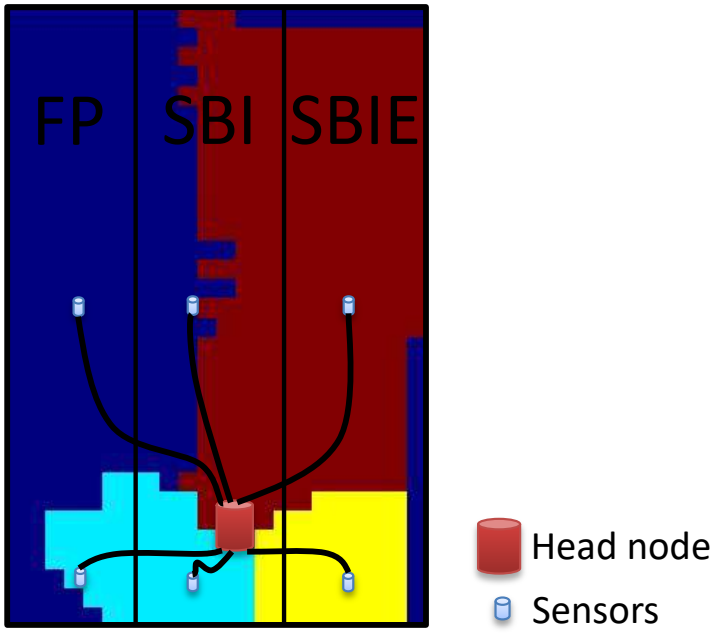


Figure 106: Sensors topology

4.2.3 Description of measurement procedure with Plant-O-Meter

With the help of Plant-O-Meter phenological stages of the maize will be monitored including growth stages and reproductive maize growth stages. Obtained results will be in the form of vegetation indices, e.g. NDVI, and will give a clear picture regarding crop condition.

It is expected that measurements will be done every two weeks. Two types of measurements will be conducted. The first one is continuous mode which implies automatic measurement and saving data every second. On this occasion, Plant-O-Meter will be positioned at the constant height from the plant canopy and moved along the row. The second one is concentric circles mode centred at the middle of the field. In this mode point measurements will be done that are positioned on the circles and shifted for 45°.

4.2.4 Expected results in terms of the influence of the irrigation strategies

During the experiment a detailed map of vegetation indices will be created for different growth and reproductive stages of maize. It is expected that different irrigation strategies will result in different vegetation indices for the same growing or reproductive stage of the maize. The performance of each strategy will be validated and assessed against specific indicators, such as water consumption per unit and economic efficiency of irrigation.