Ref. Ares(2024)381753 - 17/01/2024

InCUBE
sustainable building innovations

## WP3 – D3.1

# Interoperable specifications of APIs and data models


## Main authors:

KENT



**HORIZON-CL5-2021-D4-01**
**EUROPEAN COMMISSION**

**European Climate, Infrastructure and Environment Executive Agency**
**Grant agreement no. 101069610**

This project is funded by the European Union's 'Horizon Europe Research & Innovation programme' under grant agreement No 101069610. This publication reflects the authors' view only and the European Commission is not responsible for any use that may be made of the information it contains.

# Project contractual details

| | |
|---|---|
| Project title | An Inclusive toolBox for accElerating and smartening deep renovation |
| Project acronym | In CUBE |
| Grant agreement no. | 101069610 |
| Project duration | 48 months (01/07/2022 – 30/06/2026) |

# Document details

| | |
|---|---|
| Deliverable no. | 3.1 |
| Dissemination level | PU |
| Work package | WP3 |
| Task | T3.1 |
| Due date | M17 (30/11/2023) |
| Actual submission date | M19 (17/01/2024) |
| Lead beneficiary | 2 (KENT) |
| Contributing beneficiary/is | 1 (CERTH), 7 (FBK), 8 (RINA-C), 11 (TERA), 12 (EVOLVERE), 15 (CIRCE), 16 (METRO7), 16.1 (ESTUDIO), 20 (VW) |

# Authors


| Full Name | Beneficiary | Contact Information |
|---|---|---|
| Sofia Kleisarchaki | KENT | sofia.kleisarchaki@kentyou.com |
| Cristina Verde Ramis | CIRCE | mcverde@fcirce.es |
| Andrea Cavallaro | TERA | andrea.cavallaro@terasrl.it |
| Rita De Stefano | RINA-C | rita.destefano@rina.org |
| Fabio Bolletta | RINA-C | fabio.bolleta@rina.org |
| Evangelos Bellos | CERTH | e.bellos@certh.gr |
| Eleni Chatzigeorgiou | CERTH | e.chatzigeo@certh.gr |
| Petros Iliadis | CERTH | iliadis@certh.gr |
| Ioannis Lampropoulos | CERTH | i.labropoulos@certh.gr |
| Stefanos Efstratios Petridis | CERTH | s.petridis@certh.gr |
| Andreas Sitarists | CERTH | a.seitaridis@certh.gr |
| Antonios Zacharis | CERTH | a.zacharis@certh.gr |
| Stylianos Zikos | CERTH | szikos@iti.gr |

| | | |
|---|---|---|
| Nikolaos Tarsounas | CERTH | tarsounasnikos@iti.gr |
| Giorgos Giannopoulos | CERTH | ggiannopoulos@iti.gr |
| Alessandro Burgio | EVOLVERE | alessandro.burgio@evolvere.io |
| Andrea Gobbi | FBK | agobbi@fbk.eu |

## Reviewers

| Full Name | Beneficiary | Contact Information |
|---|---|---|
| Stylianos Zikos | CERTH | szikos@iti.gr |
| Helen Chatzigeorgiou | CERTH | e.chatzigeo@certh.gr |

## History of changes

| Version | Date | Beneficiary | Changes |
|---|---|---|---|
| 0.1 | 18/10/23 | KENT | Initial version |
| 0.2 | 13/11/23 | KENT | Integrating CIRCE, TERA, RINA-C contributions |
| 0.3 | 15/11/23 | KENT | Integrating CERTH contributions |
| 0.4 | 17/11/23 | KENT | Integrating CERTH, EVOLVERE and RINA-C contributions |
| 0.5 | 22/11/23 | KENT | Integrating RINA-C contributions and updating Section 4 and 5. |
| 0.6 | 24/11/23 | KENT | Updates from all partners |
| 0.9 | 12/12/23 | KENT | Final version ready for internal review |
| 1.0 | 16/01/24 | KENT | Final version ready for submission |

# Executive Summary

This document expounds upon the architecture, data model, available protocols, and designed APIs for each InCUBE component. Building upon the foundation laid out in D1.5, it extends the initial work by delving into the details of each InCUBE component and their-between interactions.

Upon comprehending each InCUBE component, their interconnected relationships and maturity level, this deliverable goes on to define the InCUBE interoperability framework. This framework is composed of specifications, including features of the digital twin of buildings and ontologies. These specifications play a pivotal role in facilitating data exchange across a spectrum of entities, encompassing devices, building systems, data platforms, products, applications, services, and operations. Importantly, these specifications build upon the achievements of T3.1, ensuring an efficient and cohesive ecosystem.

To demonstrate the InCUBE's interoperability layer, a use case is presented. This use case serves to showcase the practical applications and capabilities of the InCUBE's interoperability layer. Through this illustrative example, stakeholders can gain insights into how the InCUBE's integrates diverse components, fostering an environment of interoperability and efficiency.

# Table of contents

# List of figures

# List of tables



# List of Acronyms and Abbreviations

| Term | Description |
|---|---|
| AI | Artificial Intelligence |
| API | Application Programming Interface |
| BIM | Building Information Modelling |
| CIM | Common Information Model |
| CSV | Comma-Separated Values |
| DBL | Digital Building Logbook |
| DT | Digital Twin |
| EMS | Energy Management System |
| GUI | Graphical User Interface |
| HTTP | Hypertext Transfer Protocol |
| HVAC | Heating, Ventilation, and Air Conditioning |
| IFC | International Foundation Class |
| IoT | Internet of Things |
| IRL | Integration Readiness Level |
| JSO | Job Scheduling Optimizer |

| | |
|---|---|
| JSON | JavaScript Object Notation |
| KPI | Key Performance Indicator |
| LCA | Life Cycle Assessment |
| LCC | Life Cycle Cost |
| MQTT | Message Queuing Telemetry Transport |
| PDF | Portable Document Format |
| PPE | Personal Protective Equipment |
| PV | Photovoltaic |
| RD | Resilience Dashboard |
| REST | REpresentational State Transfer |
| SAREF | Smart Applications REFerence (ontology) |
| SRI | Smart Readiness Indicator |
| SRL | System Readiness Level |
| TLS | Terrestrial Laser Scanning |
| TRL | Technology Readiness Level |
| VR | Virtual Reality |
| WebRTC | Web Real-Time Communication |
| WP | Work Package |

# 1 Introduction

## 1.1 Aim of the deliverable

This deliverable aims to provide the architecture, data model, available protocols and designed APIs for each InCUBE component, which will enable the seamless interaction among these components. Moreover, it is detailing common specifications, such as features of the digital twin of buildings and ontologies. These specifications will be applied to ensure the efficient data exchange among devices, building systems, data platforms, products, applications, services and operations, building upon outcomes of T3.1. Last but not least, this deliverable is describing a use case that will demonstrate the capabilities of the InCUBE's interoperability layer.

## 1.2 Dependencies with other tasks

This deliverable relies on the outcomes of task T1.5 which defined the InCUBE architecture, system requirements and technical integration. The work of T1.5 is reported in D1.5 and enriched the content of this deliverable.

Moreover, the deliverable describes the outcomes of task T3.1 – Interoperability Framework for building systems, products, and operations. The task T3.1 specified the APIs and data models to ensure data exchange and seamless connection among InCUBE components.

The work reported in this deliverable on the interoperable specifications of APIs and data models will facilitate the creation of the InCUBE interoperability framework and will feed the upcoming deliverable D3.2 – InCUBE Interoperability Framework.

Last but not least, the common specifications of the Digital Twin reported in this deliverable will be implemented within the activities of T3.4 and will be reported in the upcoming deliverable D3.5 – DTs of Buildings.

## 1.3 Structure of the deliverable

Section 2 provides an overview of the InCUBE software components and their between interactions. Moreover, it assesses the designed specifications based on the integration guidelines defined in D1.5 and quantifies the maturity of the different components.

Section 3 presents the designed specifications for each InCUBE component, by discussing its architecture, data model, protocols and APIs. Moreover, it presents the common InCUBE specifications; features of Digital Twin and standard ontologies.

Section 4 defines a use case for demonstrating the benefits of the InCUBE interoperability layer.

Section 5 summarizes the deliverable.

# 2 Interoperability Framework

## 2.1 InCUBE Software components

Figure 1 gives a quick overview of the different InCUBE software components, the groups that they formulate as well as their between interactions. An arrow indicates that the tail (source) component interacts with the head (target) component in read (green), write (blue) or read&write (red) mode. The arrow represents the interaction source and the colour the data direction.

A detailed explanation of Figure 1 can be found in D1.5. Here, we use the figure as a reminder of all the existing components that are available to the pilot demonstrations.

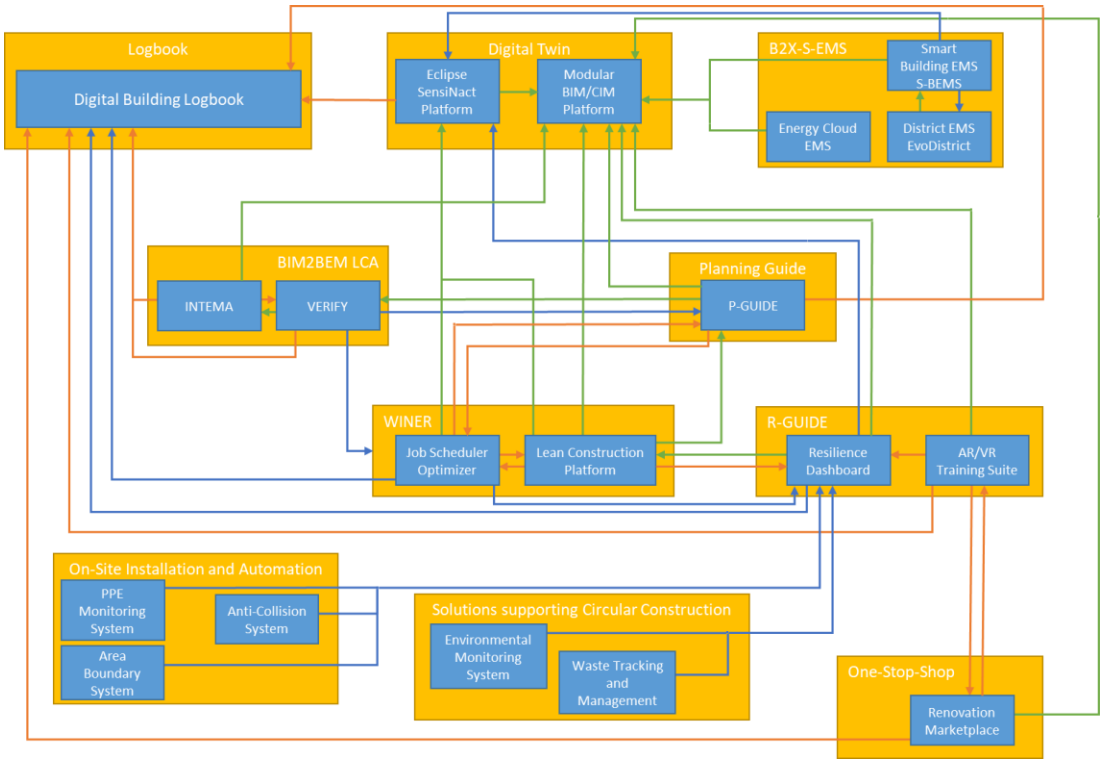



*Figure 1 Software Components Interactions*

## 2.2 Components Integration Guidelines

Given the software components and the interactions between them, presented in Section 2.1, this section will provide some advice on the implementation of these interactions.

Table 1 provides general guidelines that need to be respected before the integration of two components. These guidelines were initially defined in D1.5 and they should be considered as good practices instead of absolute rules. The third column of Table 1 attempts to evaluate each guideline based on the current status of the integration work.

*Table 1 Integration Guidelines*

| Ref | Integration Guideline Description | Status (M17) |
|---|---|---|
| **IG1** | Use similar network protocols over the project. Most of the components are intended to be accessed via:<br>• HTTP to access static files,<br>• HTTP REST to access APIs,<br>• WebSocket for API subscriptions,<br>• MQTT for sensor updates.<br>This list is not exhaustive, but adding non-standard or proprietary protocols should be discussed. | According to Section 3, all network protocols to be used lie within the proposed list of IG1. |
| **IG2** | Data, payloads, and files should be transferred using a protocol and a format that doesn't require a specific consumer technology (e.g., use a JSON representation instead of Java Serializable Format). | According to Section 3, IG2 is followed as data and payloads are exchanged using JSON format. |
| **IG3** | All public APIs should be documented.<br>Components providing REST APIs should be accompanied by an OpenAPI2 description. Components providing WebSocket and MQTT endpoints should be described using the AsyncAPI3 specification.<br>Other protocols should be described either following the OpenAPI or AsyncAPI specification, or by providing a readable Markdown file describing accesses and payloads. | All currently available API specifications respect IG3 and provide proper documentation (e.g., see Section 3.1.3.4). |
| **IG4** | Components exchanging files must agree on supported file formats. Open standards should be preferred, but proprietary file formats are allowed if supported by both components. | File formats have been specified in Section 3. |
| **IG5** | Operations requiring a date, time, duration, or time interval information should follow the ISO 8601 standard. All time-related strings must either contain an explicit time offset from UTC or be considered UTC. Local time should be avoided in APIs. | All time-related strings follow IG5. |
| **IG6** | Each component should be associated to a set of tests validating its own access points. Those tests can be used as examples for IG3. | To be evaluated |

| | | |
|---|---|---|
| **IG7** | Each component should be associated to a set of integration tests validating its accesses to the components it works with. | To be evaluated |
| **IG8** | When applicable, components should interact using similar ontologies. Components requiring a transformation should document it explicitly. | To be evaluated |
| **IG9** | Components should share security credentials and rights managements to ease user interactions. This can be done either by configuration (secrets sharing) or by centralized authentication (token-based). | To be evaluated |

## 2.3 System Maturity

Figure 2 shows the different InCUBE components as nodes of the graph and their-between interconnections as edges. Each InCUBE component is annotated with a value showing the Technology Readiness Level (TRL) of that component. Similarly, each connection is represented with a different line thickness indicating the Integration Readiness Level (IRL) between two components.

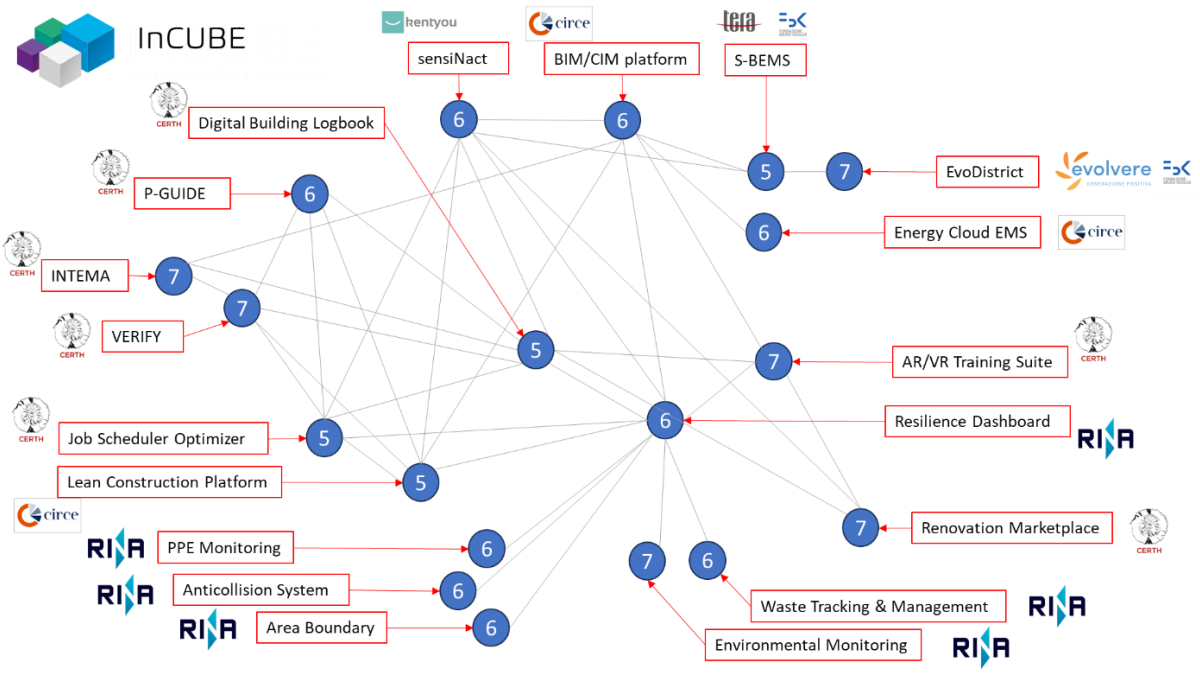



*Figure 2 InCube Framework Maturity*

The evaluation of the TRL and IRL, as well as of the entire system's readiness, will be performed and reported periodically within task T1.5. This will allow us to track the evolution of the components maturity.

# 3 Specifications for Interoperability

## 3.1 Specifications of InCUBE Components

### 3.1.1 Digital Building Logbook

#### 3.1.1.1 Data Models

The definition of the common data models to describe main entities will allow supporting activities and functionalities of InCUBE software solutions applied at various renovation phases: Design, planning, renovation execution, monitoring, and assessment. Initial versions of data models in JSON format have been developed in order to describe building topology, building systems, products, as well as operations related to renovation activities.

The main entities that were identified are the following: Building, Device, Sensor, Product, Activity, RenovationScenario, User, and Event. Data of these entities will be stored in the Digital Building Logbook component, which is the central database of InCUBE. Each entity is described by a set of attributes. The proper type of JSON data is utilised for each attribute, which can be numeric, string, Boolean, Array, or Object. In addition, there are specific attributes for declaring any existing associations with other entities. There shall also be the possibility to add new attributes (extensibility) to each data model in a dedicated section.

Relevant existing ontologies have been considered during the development of the InCUBE data models (refer to the corresponding section for more information about the ontologies). In particular, the selected attributes, naming conventions, data types, and logical structure were defined in accordance with existing ontologies depending on each entity defined. A short description of each entity is provided next.

Building

The Building entity is used to describe a building by including related attributes such as a unique ID, name, location, type, and others. It adopts properties and the organization structure from the Building Topology Ontology. In particular, a building is located at a site and can be composed of storeys, spaces, and zones.

Device

The Device entity is used to describe equipment designed to accomplish one or more functions in the building. Indicative instances of devices are HVAC, Heat Pump, PV, and Lights. The attributes that have been utilised to describe devices are in accordance with the SAREF ontology. It is worth noting that a device can be controllable through specific commands that are listed in the device definition. The Type attribute characterizes the device type, while the Category attribute describes

whether the device is physical or virtual. Lastly, the states and various functions of the device are described.

Sensor

The Sensor entity can be regarded as a specific type of device, which detects and responds to events or changes in the physical environment such as light, motion, or temperature changes. Therefore, the same base attributes with the device are used (e.g. ID, type, model, location), according to the SAREF ontology. Indicative sensor types are Environmental (ambient temperature, humidity, luminance, $CO_2$), Occupancy, and others. The sensing functions are included in the description of a sensor, defined by the property measured and the unit of measure.

Product

The Product entity is used to describe any complete product of interest that is already installed in a building, or planned to be installed in a building under renovation, or is simply available in the market. The representation of a building product that has been adopted is based on the Building Product Ontology. In particular, a product can be composed of several other components or assemblies, and the properties of its structure and elements can be described. Moreover, the relevant elements from the Schema product[1] have been considered.

Activity

The Activity entity can describe any kind of activity related to building renovation work, such as for example an installation activity necessary for installing a product at the building, a demolition activity, a waste collection activity, and other. Attributes such as ID, expected duration, priority, execution status, start and completion date times are included. Moreover, a *constraints* section exists in order to include any constraints that affect the execution of the activity. As far as the needed resources is concerned, requirements for the personnel, equipment and materials for executing the activity can be declared. This information can be used when making decisions for planning and scheduling the activity. Lastly, it is possible to include tasks within an activity that have their own characteristics. In such a case, an activity is considered completed when all its tasks have been finished.

RenovationScenario

The RenovationScenario entity is a defined representation of a renovation scenario that is planned to be evaluated during the renovation design phase. The InCUBE P-GUIDE is the software component that is directly connected with this entity. A renovation scenario includes a unique ID, a short description, and information on the involved building where the renovation scenario can be applied.

[1] https://schema.org/Product

In addition, it is composed of a list of renovation activities belonging to the *Activity* entity, that are included in the scenario.

User

The User entity includes properties (both static and dynamic) that are needed in order to describe an actor. Apart from the unique ID, the name of the user and the associated building(s), personal preferences can be included. Users of different roles are supported through the *role* property, with indicative values such as Owner, Occupant, Worker, and Manager. A user can have access to one or more InCUBE software components; therefore, in this case, information related to user authentication exists (username, email address, password). Lastly, the user representation allows for the assignment of an experience level and individual skills to users, which is particularly useful for the automated assignment of the appropriate workforce to an activity.

Event

The Event entity is defined as a message with specific properties, content, and timestamp. Events can be generated by a sensor, device, or software component, and it is the main method for inserting and keeping historical information in the InCUBE Digital Building Logbook. In relation to the kind of information that is included, various types of events have been defined, such as Measurement, ForecastPowerGeneration, ForecastEnergy, Notification, RenovationStatus, RenovationSchedule. It shall be noted that the retrieval of multiple events from the DBL is possible by providing a date time range as input. Common required attributes of all events, regardless of their type, are the unique ID, type, dateTime, and sourceId (which denotes the sender of the event). The payload of each event is placed within the "content" property. The exact format and included properties depend on the event type.

Preliminary data models of the DBL, defined in JSON format, are presented in the Annex.

#### 3.1.1.2 Protocols

**Input**

The components that use the Logbook to store documents, reports, or other output data will access it using a HTTP REST interface. In addition, the components notifying the Logbook to store new events, such as measurements from sensors, will be able to use either the provided HTTP REST interface or alternatively the Message Queuing Telemetry Transport (MQTT) protocol.

A list of components that will store data to the Digital Building Logbook is provided in the table below:

| Client component | Access description | Access protocol | Data Format | Frequency of Collection | Data Sensitivity |
|---|---|---|---|---|---|
| Eclipse SensiNact Platform | Propagates events from IoT sensors | HTTP REST / MQTT | JSON and/or XML | Real-time | Internal |

| | | | | | |
|---|---|---|---|---|---|
| | and smart devices | | | | |
| Report generators | Store (push) generated documents by InCUBE components | HTTP REST (for sending and storing files) / MQTT (for event only) | various file extensions (e.g., PDF) | Periodic or on-demand | Internal |
| Other data generators | Store (push) generated output data by InCUBE components | HTTP REST / MQTT | JSON and/or XML | Periodic or on-demand | Internal |

**Access by other components**

A list of components that retrieve data from the Digital Building Logbook is provided in the table below:

| Client component | Access description | Access protocol | Data Format | Frequency of Collection | Data Sensitivity |
|---|---|---|---|---|---|
| P-GUIDE | Retrieves properties of products, pre-defined set of activities (renovation scenarios) | HTTP REST | JSON | On-demand | Internal |
| Eclipse SensiNact Platform | Update on live events | HTTP REST / MQTT | JSON | Periodic | Internal |
| VERIFY | Retrieves performance data and data from EMS sensors | HTTP REST | JSON | Periodic | Internal |
| INTEMA | Retrieves sensors/performance information | HTTP REST | JSON | Periodic | Internal |
| Job Scheduling Optimizer | | HTTP REST | JSON | Periodic | Internal |
| AR/VR Training Suite | Retrieves information about devices, products, activities, training material of products' installation | HTTP REST | JSON | Periodic | Internal |
| Resilience Dashboard | | HTTP REST | JSON | Periodic | Internal |
| Renovation Marketplace | Retrieves stored data (e.g. products, devices data) | HTTP REST | JSON | Periodic / On-demand | Internal |
| Future client components | Client-specific needs | HTTP REST | JSON | Periodic / On-demand | Internal |

#### 3.1.1.3 APIs

The Digital Building Logbook (DBL) provides an HTTP REST web service Application Programming Interface (API) for data insertion and retrieval using HTTP calls. The API supports the four main persistent storage functions, which are *Create*, *Read*, *Update*, and *Delete*.

In particular, GET and POST HTTP methods have been developed to support the four different types of requests (insertion, deletion, update, retrieval). The responses of the requests are in JSON format and contain the "status" property, which can be either "success" or "failed" depending on the outcome of the query.

The endpoints of the API are used for storing and retrieving data to/from the database. Several examples are presented next in the form of tables. More details about the endpoints will be provided in the InCUBE deliverable D3.6 - InCUBE Digital Logbook (v1).

The base URL of all API calls that are handled by the DBL is of the following structure:

http://<*ip_address*>:8080/api/dbl/api

DBL supports authentication via the use of access tokens. A unique token can be generated for each InCUBE component. Subsequently, each GET/POST request made must include the generated token of the caller component within the header of the request as "Authorization: Bearer <token>". The DBL authentication function validates the access token in order to ensure that the caller is a recognised component. In case of unsuccessful validation, the handling of the request is terminated and a proper notification is sent back.

***User***

| | |
|---|---|
| Description | Add a new user |
| Method | POST |
| Active URL | <baseURL>/addUser |
| Request Parameters/Body | Content-type: application/json<br>{user json} |
| Response | {"status":"failed"} on failure or {"status":"success"} on success |

| | |
|---|---|
| Description | Authenticate user by username. The body of the request contains the username and the user's password in JSON format |
| Method | POST |
| Active URL | <baseURL>/authenticateUserByUsername |
| Request Parameters/Body | Content-type: application/json<br>{<br>"site": "site_name",<br>"username": "name",<br>"passwd": "userpassword"<br>} |
| Response | {"status":"failed"} on failure or {"status":"success" , "userId": <"id">} on success |

| Description | Delete a user |
|---|---|
| Method | POST |
| Active URL | <baseURL>/deleteUser |
| Request Parameters/Body | Content-type: application/json<br>{<br>“site”: “site_name”,<br>“userId”: “user1”<br>} |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| Description | Set password of user |
|---|---|
| Method | POST |
| Active URL | <baseURL>/setUserPassword |
| Request Parameters/Body | Content-type: application/json<br>{<br>“site”: “site_name”,<br>“userId”: “user1”,<br>“currentPasswd”: “userpassword”,<br>“newPasswd”: “newuserpassword”<br>} |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| Description | Update property of user |
|---|---|
| Method | POST |
| Active URL | <baseURL>/updateUserProperty |
| Request Parameters/Body | Content-type: application/json<br>{<br>“site”: “site_name”,<br>“userId”: “user1”,<br>“propertyName”:”name”,<br>“propertyValue”: value<br>} |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| Description | Get information about user |
|---|---|
| Method | GET |
| Active URL | <baseURL>/getUser?site=sitename&userId=user1 |
| Request Parameters/Body | site<br>userId |
| Response | JSON which contains information of the user with the given id |

| Description | Get information of all users |
|---|---|
| Method | GET |
| Active URL | <baseURL>/getUsers?site=sitename |
| Request Parameters/Body | site |
| Response | JSON which contains information of all users found |

***Building***

| Description | Add new building |
|---|---|
| Method | POST |
| Active URL | <baseURL>/addBuilding |
| Request Parameters/Body | Content-type: application/json<br>{building json} |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| Description | Delete a building |
|---|---|
| Method | POST |
| Active URL | <baseURL>/deleteBuilding |
| Request Parameters/Body | Content-type: application/json<br>{<br>“site”: “site_name”,<br>“buildingId”: “building1”<br>} |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| Description | Retrieve a building by the id |
|---|---|
| Method | GET |
| Active URL | <baseURL>/getBuilding?site=sitename&buildingId=building1 |
| Request Parameters/Body | site<br>buildingId |
| Response | JSON of the building |

| Description | Retrieve buildings of user |
|---|---|
| Method | GET |
| Active URL | <baseURL>/getBuildingsByUser?site=sitename&userId=user1 |
| Request Parameters/Body | site<br>userId |
| Response | JSON of the buildings found |

| Description | Retrieve all buildings in site |
|---|---|
| Method | GET |
| Active URL | <baseURL>/getBuildings?site=sitename |
| Request Parameters/Body | site |
| Response | JSON of all buildings found |

***Product***

| Description | Add new product |
|---|---|
| Method | POST |
| Active URL | <baseURL>/addProduct |
| Request Parameters/Body | Content-type: application/json<br>{product json} |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| Description | Delete a product |
|---|---|
| Method | POST |
| Active URL | <baseURL>/deleteProduct |
| Request Parameters/Body | Content-type: application/json<br>{<br>“site”: “site_name”,<br>“productId”: “product1”<br>} |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| Description | Update property of existing product |
|---|---|
| Method | POST |
| Active URL | <baseURL>/updateProductProperty |
| Request Parameters/Body | Content-type: application/json<br>{<br>“site”: “site_name”,<br>“productId”: “product1”,<br>“propertyName”:”product property”,<br>“propertyValue”: value<br>} |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| Description | Get product |
|---|---|
| Method | GET |
| Active URL | <baseURL>/getProduct?site=sitename&productId=product1 |
| Request Parameters/Body | site<br>productId |
| Response | JSON with information of the product with the given id |

***Device***

| Description | Add new device |
|---|---|
| Method | POST |
| Active URL | <baseURL>/addDevice |
| Request Parameters/Body | Content-type: application/json<br>{device json} |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| Description | Delete a device |
|---|---|
| Method | POST |
| Active URL | <baseURL>/deleteDevice |
| Request Parameters/Body | Content-type: application/json<br>{<br>“site”: “site_name”,<br>“deviceId”: “device1”<br>} |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| Description | Update property of existing device |
|---|---|

| Method | POST |
|---|---|
| Active URL | <baseURL>/updateDeviceProperty |
| Request Parameters/Body | Content-type: application/json<br>{<br>“site”: “site_name”,<br>“deviceId”: “device1”,<br>“propertyName”:”device property”,<br>“propertyValue”: value<br>} |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| Description | Get device |
|---|---|
| Method | GET |
| Active URL | <baseURL>/getDevice?site=sitename&deviceId=device1 |
| Request Parameters/Body | site<br>deviceId |
| Response | JSON with information of the device with the given id |

| Description | Get devices of given type |
|---|---|
| Method | GET |
| Active URL | <baseURL>/getDevicesByType?site=sitename&deviceType=HVAC |
| Request Parameters/Body | site<br>deviceType |
| Response | JSON with information of the devices found |

| Description | Get devices of specific type and user |
|---|---|
| Method | GET |
| Active URL | <baseURL>/<br>getDevicesByTypeUser?site=sitename&userId=user1&deviceType=HVAC |
| Request Parameters/Body | site<br>userId<br>deviceType |
| Response | JSON with information of the devices found |

***Event***

| Description | Set a new event |
|---|---|
| Method | POST |
| Active URL | <baseURL>/setEvent |
| Request Parameters/Body | Content-type: application/json<br>{event json} |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| Description | Delete an event |
|---|---|
| Method | POST |
| Active URL | <baseURL>/deleteEvent |
| Request Parameters/Body | Content-type: application/json<br>{<br>“site”: “site_name”,<br>“eventId”: “24d5d1eb-8974-4a40-b34e-b9eda80524fc” |

| | |
|---|---|
| | } |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| | |
|---|---|
| Description | Retrieve existing event by the id |
| Method | GET |
| Active URL | <baseURL>/getEvent?site=sitename&eventId=id |
| Request Parameters/Body | site<br>eventId |
| Response | JSON of the event |

| | |
|---|---|
| Description | Retrieve all events by type within specified time period |
| Method | GET |
| Active URL | <baseURL>/getEventsByType?site=sitename<br>&eventType=typeName<br>&startDateTime=2023-05-10T00:00:00<br>&endDateTime=2023-06-10T23:00:00 |
| Request Parameters/Body | site<br>eventType<br>startDateTime<br>endDateTime |
| Response | JSON list of events found |

| | |
|---|---|
| Description | Retrieve all events by type and source id within specified time period |
| Method | GET |
| Active URL | <baseURL>/getEventsBySourceType?site=sitename<br>&sourceId=toolName&eventType=typeName<br>&startDateTime=2023-05-10T00:00:00<br>&endDateTime=2023-06-10T23:00:00 |
| Request Parameters/Body | site<br>sourceId<br>eventType<br>startDateTime<br>endDateTime |
| Response | JSON list of events found |

***File transfers***

| | |
|---|---|
| Description | Add new file |
| Method | POST |
| Active URL | <baseURL>/addFileVideo<br><baseURL>/addFileImage<br><baseURL>/addFileDocument |
| Request Parameters/Body | Content-type: application/ x-www-form-urlencoded<br>Body: form-data file:<filename> |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| | |
|---|---|
| Description | Delete existing file |
| Method | POST |
| Active URL | <baseURL>/deleteFileVideo |

| | |
|---|---|
| | <baseURL>/deleteFileImage<br><baseURL>/deleteFileDocument |
| Request Parameters/Body | Content-type: application/json<br>{ “name”: “video1.mp4” } |
| Response | {“status”:”failed”} on failure or {“status”:”success”} on success |

| | |
|---|---|
| Description | Retrieve files information<br><br>(Note: A file can be retrieved via HTTP directly by accessing the URL: e.g. http://ip_address:port/files/videos/<video_filename>) |
| Method | GET |
| Active URL | <baseURL>/getFilesVideos |
| Request Parameters/Body | - |
| Response | {“status”:”failed”} on failure or JSON with files information |

MQTT interface

A sub-component of the DBL is able to connect to a deployed MQTT broker, subscribe to MQTT topics for events, and receive incoming events from other InCUBE components or devices via MQTT. Then, conversion of the event payload to the proper JSON format is performed if needed, and insertion of the event in the database is made.

### 3.1.2 Modular BIM/CIM Platform

This solution is a modular (able to swap BIM object categories) and interoperable web-based repository of BIM objects (incl. conventional and novel building solutions). In addition, the platform stores the different BIM and CIM files developed in the project, which contain information related to the demonstrator buildings and the cities in which they are located.

#### 3.1.2.1 Architecture

The structure of the platform is composed of different folders (extensible and modular):

- Four folders for each demo site: BIM modelling, where .rvt files are uploaded; BIM Objects Data, containing information about Revit families; IFC exports, which contains .ifc 3D models; and Point Clouds, where linked .rcp files are stored.
- BIM Objects Repository: This library contains conventional products and innovative InCUBE products organized into manufacturer-specific folders. To facilitate the seamless information exchange with developers working on the demo site BIM models, these BIM objects have been developed in Revit software, utilizing both .rfa and .rvt formats.
- CIM repository: includes the CIM models of the three demo cities: Zaragoza (ES), Trento (IT), and Groningen (NL). These City Information Models (CIM) for each demo city, serves as repositories of urban-level building information. Additionally, there is a simplified 2D version of the BIM models available on .json file formats.

Figure 3 shows the organisational structure of the platform and the information contained within each folder. Access rights are allocated to partners based on their specific needs to safeguard the confidentiality, integrity and prevention of modification of the platform's information.

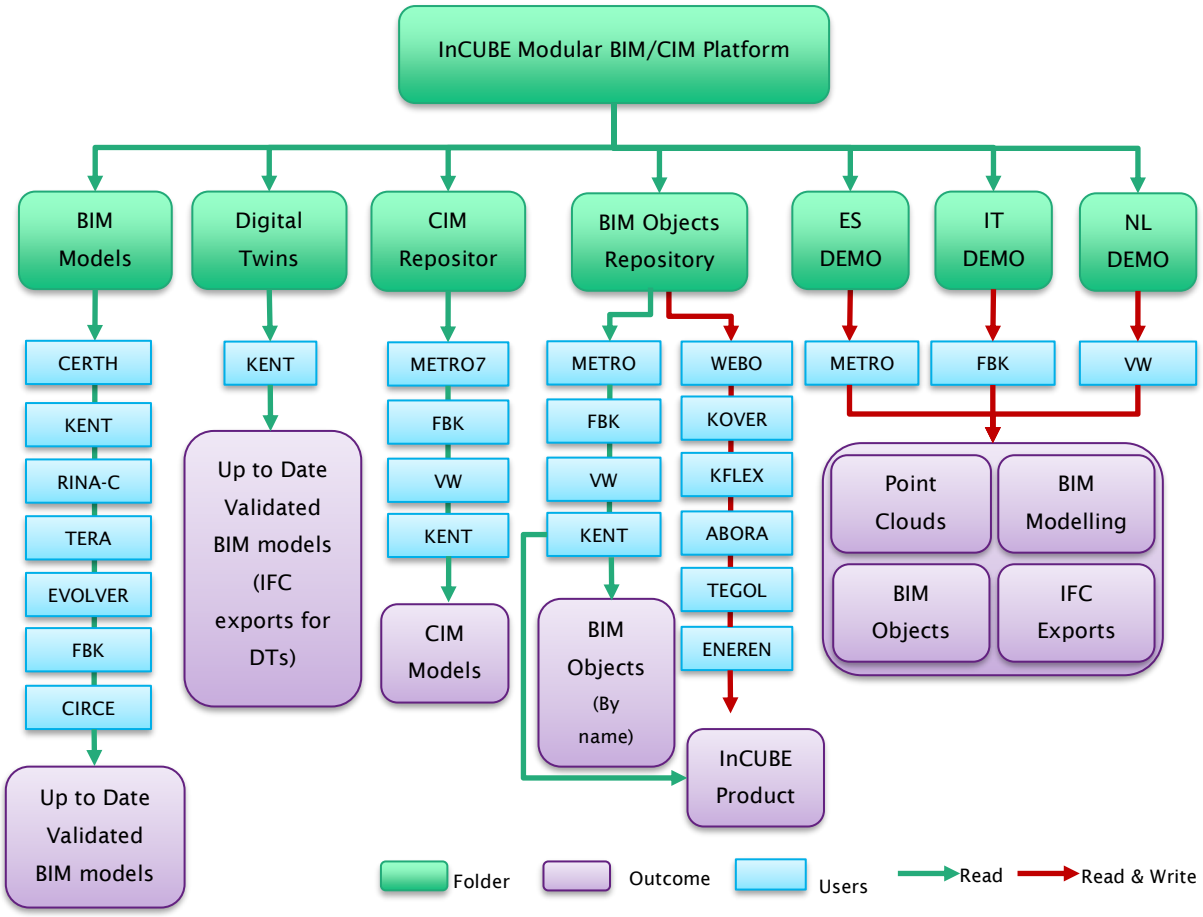



*Figure 3 Organizational Structure of BIM/CIM Platform*

### 3.1.2.2 Protocols

**Read access to components**

The Modular BIM/CIM platform does not get data from other components as it is expected to be updated directly and manually.

**Access by other components**

The information and 3D models must be easily accessible in formats that enhance interoperability. The platform is accessible through a web interface. It will be necessary for it to be usable programmatically by other components using a REST interface over HTTP(S) and returning files in an interoperable format e.g., IFC for BIM details and PLY for 3D models. As of now, the files for the other components are readily available on the BIM/CIM repository. To facilitate easier access, the option of implementing a REST API has been explored, specifically considering the OneDrive API.

The OneDrive API offers a seamless and secure way to interact with files and data. By integrating this API, users can enjoy enhanced flexibility and convenience in retrieving, uploading, and managing files from the repository. While we are considering the OneDrive API, we are also open to exploring alternative options that might better suit the specific needs of the components.

Here is the list of components retrieving data from this component:

| Client component | Access description | Access protocol | Data Format | Frequency of Collection | Data Sensitivity |
|---|---|---|---|---|---|
| AR/VR Training Suite | 3D models of buildings and items | HTTP | Ifc, .rvt files | On-demand | Internal |
| Renovation Marketplace | Description of BIM items<br>BIM assets list | HTTP | To be defined | On-demand | Internal |
| INTEMA | Building BIM (with 3D model) | HTTP | .Ifc, .rvt files | Scarce (infrequent occurrences) | Internal |
| Lean Construction Platform | 3D models of the building | HTTP | To be defined | Non-periodic (irregular intervals) | Internal |
| P-GUIDE | Retrieve properties of products, characteristics of the buildings | HTTP | Ifc, .rvt files | On-demand | Internal |
| Eclipse SensiNact Platform | Access BIM models | HTTP | Ifc, files | Periodic | Internal |
| Smart Building EMS (S-BEMS) | Building information and 3D structure | HTTP | Ifc, .rvt files | Once | Internal |
| Energy Cloud EMS | Building information and 3D structure | HTTP | Ifc, .rvt files | Once | Internal |

### 3.1.3 Eclipse sensiNact

Eclipse sensiNact is an Open Source framework that aims at creating a common environment in which heterogeneous devices can exchange information and interact among each other in the IoT world.

It is a horizontal platform dedicated to IoT and in particularly used in various smart city and smart home applications. Eclipse sensiNact aims at managing IoT protocols and devices heterogeneity and provides synchronous (on demand) and asynchronous (periodic or event based) access to data/actions of IoT devices, as well as access to historic data with generic and easy-to-use API.

#### 3.1.3.1 Architecture

Figure 4 shows the architecture of Eclipse sensiNact. Eclipse sensiNact is composed of two main parts; the southbound and the northbound bridges.

Southbound bridges are specialized into the interaction with devices, which can be sensors, actuators, other systems, or applications. Each bridge oversees the communicating with a specific kind of device, using a given protocol. Thanks to an OSGi based architecture, it is possible to add bridges on the fly, while the gateway is running, to allow the communication with new kind of devices. The creation of bridges relies on an API which delegates most of the integration work to the gateway, letting the programmer focus on the communication protocol and the data model of the device to be integrated. A southbound bridge, based on the MQTT protocol, was used to communicate with the IoT edge gateway.

Symmetrically to the southbound bridges, northbound bridges are in charge of publishing information to remote systems. The set of northbound bridges is also extensible, for tailoring special needs or singular systems. The REST API, which is a northbound bridge, is a key part in our architecture. It is designed for the administration of the gateway, thanks to a well-documented API. Such a REST API is also available to facilitate the access to data from the IoT edge gateway.

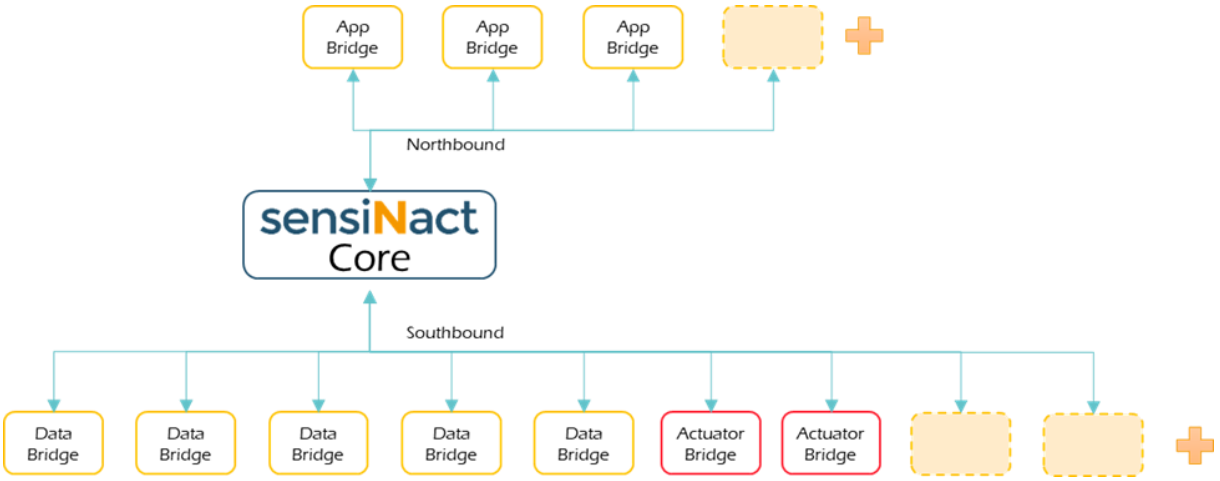



*Figure 4 Architecture of Eclipse sensiNact*

The architecture of sensiNact has the following three key characteristics.

- Extensible: sensiNact provides a framework that can adapt and grow as requirements evolve, without requiring a complete overhaul.
- Modular: sensiNact's modules are designed to work independently but can also interact with each other through well-defined interfaces or APIs (Application Programming Interfaces). This modularity makes it easier to understand, develop, maintain, and scale a system.
- Interoperable: sensiNact's architecture is designed to ensure that its components can interact with other systems or components, regardless of their origins or technologies

### 3.1.3.2 Data Models

Eclipse sensiNact represents its data using the following model, shown in Figure 5:

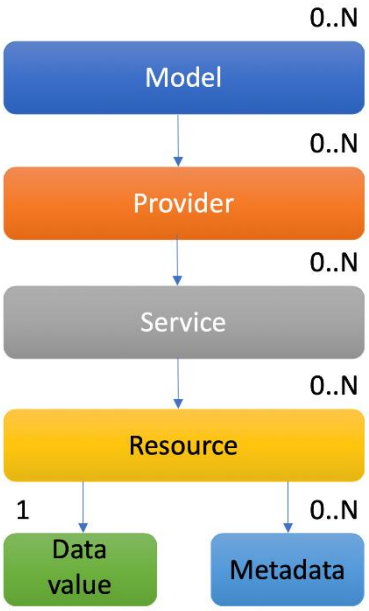



*Figure 5 sensiNact Data Model*

- **Model**: a static or dynamic representation of a kind of provider. This allows to describe the structure of a provider, i.e. its services and the type, default value, description, … of their resources
- **Provider**: a provider of services. It usually represents a physical object, but the granularity depends on the use case. For example, it can be an IoT device with multiple sensors, a car, etc.
- **Service**: the name of a set of resources. All providers have the reserved admin service which holds sensiNact details about them, including their location resource.
- **Resource**: a resource can either represent a property of the provider (like the temperature read from a sensor) or an action (for example to send a text message).

### 3.1.3.3 Protocols

**Read access to components**

The Eclipse SensiNact platform will access the Digital Building Logbook and the Modular BIM/CIM platform to obtain the history of the building and its BIM description. It will not proactively retrieve data from other components.

| Target component | Access description | Access Protocol | Data Format | Frequency of Collection | Data Sensitivity |
|---|---|---|---|---|---|
| Digital Building Logbook | Access history of events | HTTP | JSON | Periodic | Internal |
| Modular BIM/CIM Platform | Access BIM models | HTTP | JSON, GeoJSON, .rvt | Periodic (e.g., 2 times - before/after renovation) | Internal |

**Access by other components**

InCUBE software components are expected to push the events they receive to the Eclipse SensiNact platform, which will update its digital twin model and push those events to the Digital Building Logbook. It is preferred that components that need to subscribe to live events should subscribe to the Eclipse SensiNact platform instead of listening to the sensors directly. This will increase the possibilities of sensor abstraction and the reusability of the overall platform.

| Client component | Access description | Access protocol | Data Format | Frequency of Collection | Data Sensitivity |
|---|---|---|---|---|---|
| Smart Building EMS (S-BEMS) | Push sensors events | MQTT / MQTTS | JSON | Every 15 mins | Internal |
| Resilience Dashboard | Push safety events | HTTP REST / MQTT | JSON | Real-time | Internal |
| Job Scheduler Optimizer | Real time sensor data | HTTP REST / MQTT | JSON | Real-time | Internal |
| Components requiring real-time events | Subscription to real-time events | HTTP REST / MQTT | JSON | Real-time | Internal |

**Write access to components**

The Eclipse SensiNact platform will propagate the events it receives to the Digital Building Logbook. It can also push BIM updates to the logbook.

| Target component | Access description | Access protocol | Data Format | Frequency of Collection | Data Sensitivity |
|---|---|---|---|---|---|
| Digital Building Logbook | Propagates events from IoT sensors and smart devices | MQTT | JSON | Real-time | Internal |
| Digital Building Logbook | Pushes the description of buildings read from Modular BIM/CIM platform | HTTP | JSON, GeoJSON, .rvt | Periodic (e.g., 2 times - before/after renovation) | Internal |

#### 3.1.3.4 APIs

The Eclipse sensiNact gateway provides a northbound API which interfaces with users and/or machines to provide access to the digital twin. This is typically using a specific protocol over a more generic transport (e.g. REST/HTTP or JSON-RPC/Websocket). It is possible to have multiple northbound providers deployed to a single Eclipse sensiNact gateway instance, allowing access via different interfaces, protocols and transports.

In the context of the InCUBE project, eclipse sensiNact will provide a RESTful API to allow access to its data model. For this purpose, the API specifications described below will be supported and provided to third parties. A more detailed documentation of the above API can be found here: https://eclipse-sensinact.readthedocs.io/en/latest/northbound/RestDataAccess.html.

**GET resources**

The following GET resources are available:

- /sensinact - The root of the REST interface returns rich details of all available providers in the system
- /sensinact/providers/ - Returns an array containing the names of all providers in the system
- /sensinact/providers/{id} - Returns rich details of the provider with id id

- /sensinact/providers/{id}/services - Returns an array containing the names of all services for the provider with id id
- /sensinact/providers/{provider}/services/{id} - Returns rich details of the service with id id in provider provider
- /sensinact/providers/{provider}/services/{id}/resources - Returns an array containing the names of all resources for the service with id id in provider provider
- /sensinact/providers/{provider}/services/{service}/resources/{id} - Returns rich details of the resource with id id in service service with provider provider
- /sensinact/providers/{provider}/services/{id}/resources/{id}/GET - Gets the value of a resource
- /sensinact/providers/{provider}/services/{id}/resources/{id}/SUBSCRIBE - Gets a Server Sent Event Stream which is notified for each change to the value of a resource

**POST resources**

- /sensinact/providers/{provider}/services/{id}/resources/{id}/SET - Sets the value of a resource
- /sensinact/providers/{provider}/services/{id}/resources/{id}/ACT - Acts on an action resource

**Responses**

All of the defined REST resources return a response body. The response body is used to return information to the client.

**Response URI**

The uri of the response corresponds to the URI of the incoming request, and identifies the provider, service, or resource being returned.

### 3.1.4 INTEMA

#### 3.1.4.1 Architecture

INTEMA.building is a web-based application for conducting accurate transient building energy simulations, able to model multiple energy systems (production, consumption and storage). The tool is able to estimate with high accuracy the electrical and heating needs of the building. Its architecture is schematically depicted In Figure 6. The application receives BIM data from the user, retrieves weather data from an external service2, conducts a building energy performance simulation and produces timeseries for all major variables of the physical system. Besides timeseries, INTEMA also calculates a set of KPIs that summarize the overall energy efficiency.

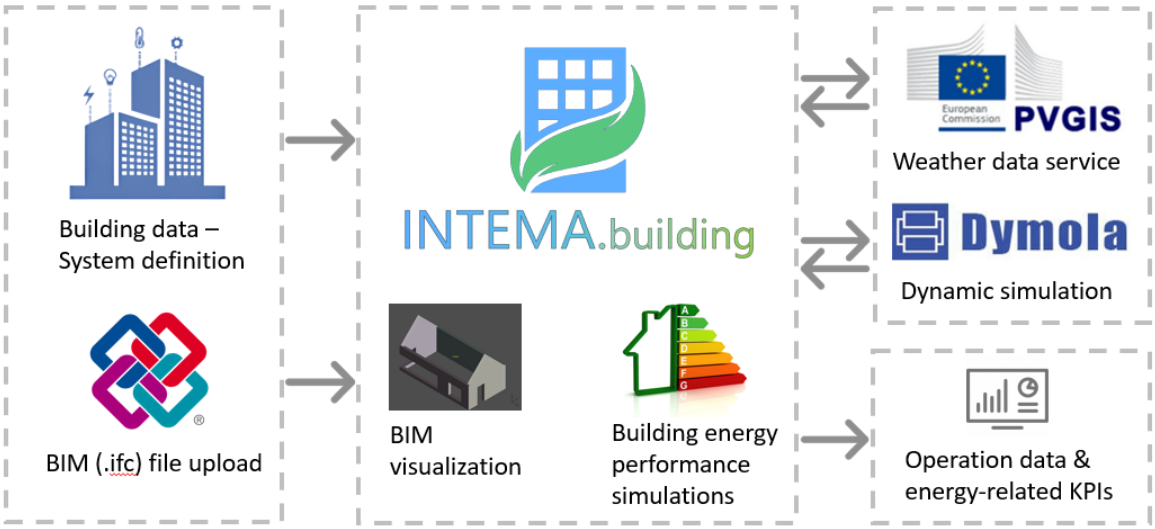



Figure 6 INTEMA.building Architecture

To forecast the thermal behavior of the building in all the important heating-zones / rooms, a significant number of variables are considered, such as:

- gains/losses from areas inside the building,
- gains/losses from surroundings areas (outside) the building,
- solar gains,
- human presence and internal activities,
- psychrometric chart and operation of HVAC
- equipment and BEMS, and
- local energy production (e.g. PV)

INTEMA.building has streamlined the system definition process into distinct steps presented in Figure 7. This eliminates common errors during data input. The user successively defines the relative data starting from the location and weather data, then the thermal zones and the various

[2] Photovoltaic Geographical Information System (PVGIS) (europa.eu)

building elements (e.g., wall, slabs, windows etc), and finally the HVAC and electrical systems. This concludes the system definition process, and the simulation can be performed.

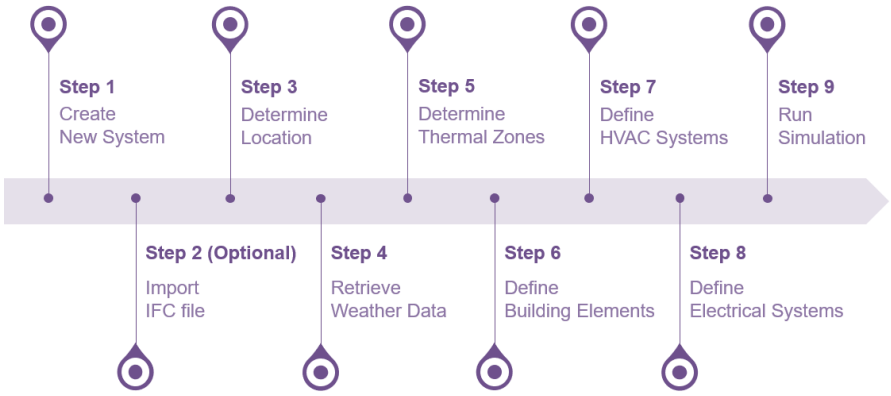



*Figure 7 INTEMA.building System Definition Process*

### 3.1.4.2 Data Models

INTEMA's internal building representation is implemented with an extensive building data model that covers all relevant physical quantities. The data model follows the general structure presented in Figure 8. The main node is the System model, which refers to the building system in the context of INTEMA.building. Systems are connected with the several building elements classes i.e., walls, slabs, windows etc. Systems also include one or more Runs objects which save the simulation results in the local database. A useful property is that systems can be shared among multiple users enabling collaboration.

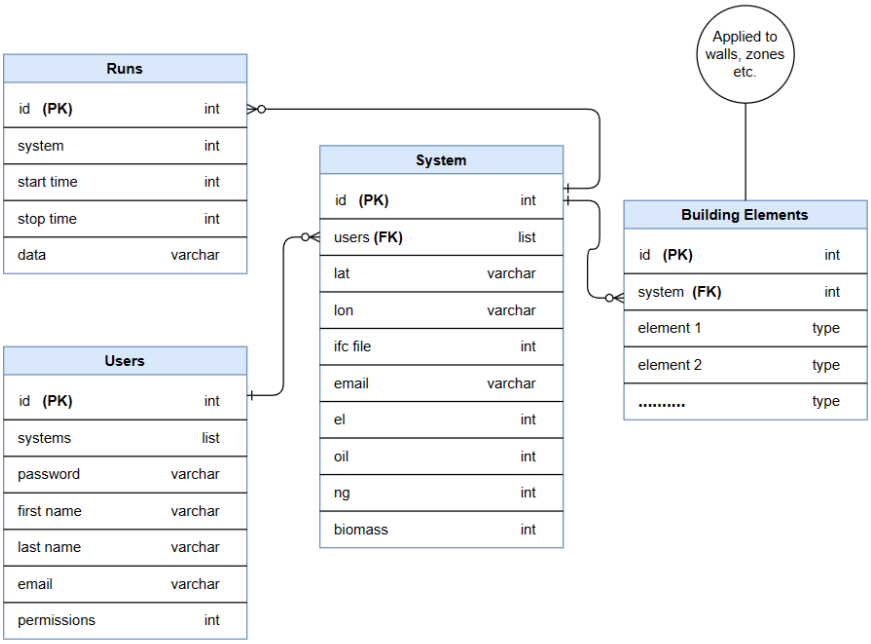



*Figure 8 INTEMA.building Data Model*

### 3.1.4.3 Protocols

**Read access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| sensiNact | Retrieve BIM data | HTTP REST | JSON | Scarce (manual calls) | Internal |
| BIM/CIM platform | Retrieve BIM data | HTTP REST | .Ifc, .rvt files | Scarce (manual calls) | Internal |

**Write access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| VERIFY | Push simulation results/KPIs | HTTP REST | JSON | Scarce (manual calls) | Confidential |
| Digital Building Logbook | Pushes the simulation results/KPIs for each pilot site/scenario | HTTP | JSON | Scarce (manual calls) | Confidential |

### 3.1.4.4 APIs

In the context of the InCUBE project, CERTH provides a RESTful API to allow access to its data model. For this purpose, the API specifications described below are supported and provided to third parties.

**GET resources**

The following GET resources are available:

- /api/building/runs/- Returns a json object that contains information about the past runs

**POST resources**

- /api/building/zones/ – posts a form with values of zones properties
- /api/building/constructions/ – posts a form with values of constructions properties
- /api/building/layers/ – posts a form with values of layers properties
- /api/building/walls/ – posts a form with values of wall properties
- /api/building/slabs/ – posts a form with values of slabs properties
- /api/building/titleds/ – posts a form with values of titleds properties
- /api/building/windows/ – posts a form with values of windows properties
- /api/building/doors/ – posts a form with values of doors properties
- /api/building/hvac/ – posts a form with values of HVAC systems properties
- /api/building/elec/ – posts a form with values of electrical systems properties
- /api/building/runs/ – posts a form with run values

**Responses**

All of the defined REST resources return a response body. The response body is used to return information to the client.

**Response URI**

The URI of the response corresponds to the URI of the incoming request, and identifies the provider, service, or resource being returned.

### 3.1.5 VERIFY

#### 3.1.5.1 Architecture

VERIFY is an online web-based platform, performing environmental and cost oriented analysis and computations to evaluate building renovations according to the ISO 14040/14044, ISO 15686 and LEVEL(S) directives. The core of VERIFY is a web application which can be utilized by several entities such as physical users and external tools. The users can easily set up the current and the planned state of their building, perform the environmental analysis and review the results through the front-end of the application. A PostgreSQL database is responsible for keeping all the important information secure and updated.

For the analysis to be successfully performed timeseries data considering the consumption and production of energy are required. These data can originate from three different sources: i) CSV files, ii) external dynamic simulation tools, such as INTEMA, and iii) IoT infrastructure. The CSV files can be manually uploaded by the user through the user interface of the application. In case such data are not available, the user can request timeseries from an external simulation tool. The process is automatic and the communication with the simulation tool is achieved using REST APIs. Finally, VERIFY can be connected to physical IoT infrastructure and continuously ingest real time data through the MQTT protocol. This data is usually stored in a private big data repository and retrieved for the purposes of the analysis upon request. Moreover, such data can be even retrieved from external big data repositories through REST APIs. The overall architecture of VERIFY is presented in Figure 9.

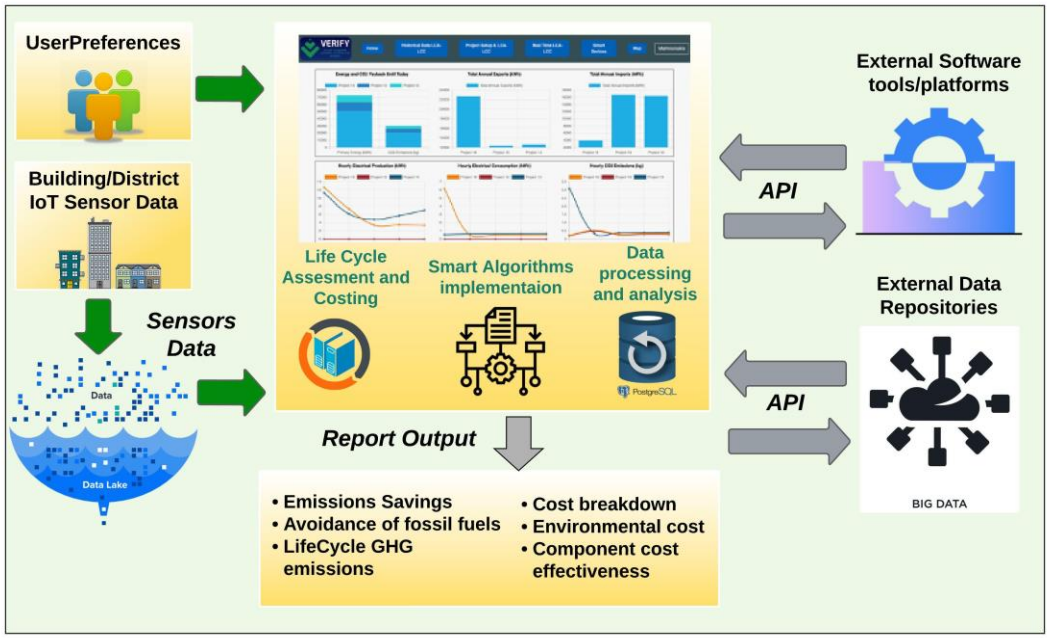



*Figure 9 VERIFY's Architecture*

### 3.1.5.2 Data Models

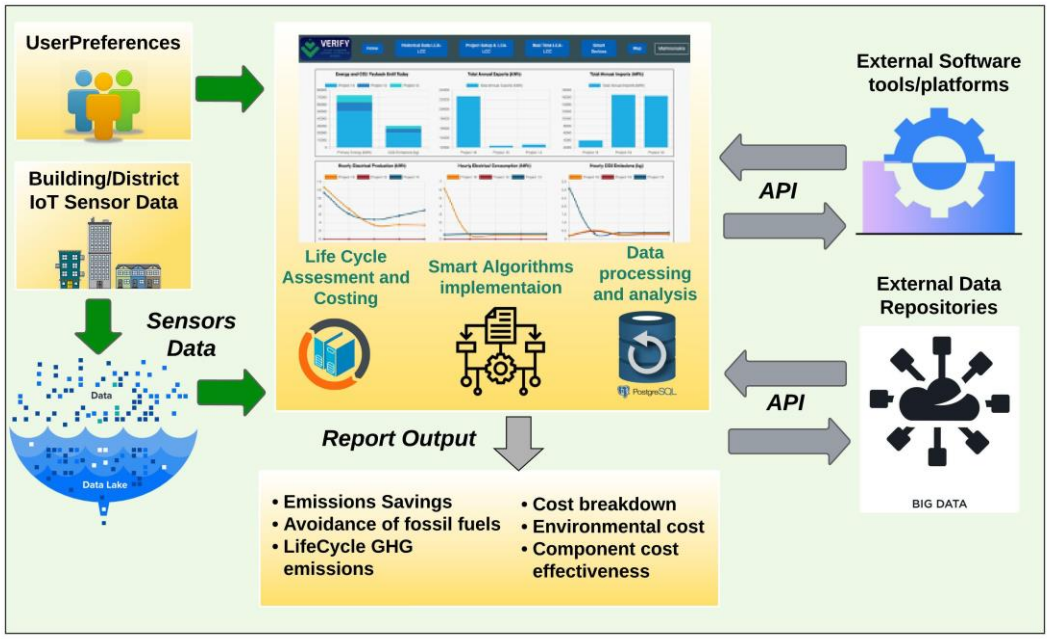



*Figure 10 VERIFY's Data Model*

VERIFY represents its data using the following model, shown in Figure 10:

- **User**: Represents a user of VERIFY. Each user may own/manage 0 to multiple buildings.
- **Building**: Represents a building and stores its basic information, such as an identifier name, its location etc.
- **Baseline/Upgrade scenario**: It represents the current state of the building and the planned scenarios. The state of the building is a collection of the envelope parameters, the thermal components that are installed on the building and the pricing and investment details. All of those details are used to perform the lifecycle environmental and costing analysis. A building has a single baseline scenario and may have 1 or multiple renovation plans.
    - **Consumption components**: Represents a collection of the components that are closely related to the consumption of energy. This might include active or even passive components. Active components (**HVAC**) are those that consume electricity or other fuels in order to produce heat or cool, such as boilers and heat-pumps. Passive components (**INS**) are those that prevent thermal losses, such as wall insulation or glazing. Moreover, the infrastructure considering hot water supply (**DHW**) are included.
    - **Production components**: Represents a collection of the components that are closely related to the production of electricity through renewable resources, such as photovoltaics (**PV**) and wind turbines (**WT**). Additionally, electricity storage components (**BAT**), i.e. batteries, are included.
    - **Electrical plan**: It includes details about the electrical consumption of the building, such as the electricity consumption and the pricing scheme, considering the energy consumption and production.
    - **Investment plan**: It includes several economic factors that affect the assessment of the success of the investment, i.e. the renovation of the building. These factors include economic parameters (e.g. VAT, tax rate), financing parameters (e.g. loans), cost parameters (e.g. CAPEX of the technologies to be installed) and real estate parameters (e.g. land acquisition cost).

### 3.1.5.3 Protocols

**Read access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| INTEMA | Requests INTEMA to check for status updates | HTTP | JSON | Daily | Internal |
| Digital Building Logbook | Access to EMS sensors/performance data | HTTP | JSON | Hourly | Confidential |

**Access by other components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| INTEMA | Push building parameters and renovation scenarios definition | HTTP | JSON | On demand | Internal |

**Write access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Digital Building Logbook | Store LCC/LCA reports/information | HTTP | JSON | On demand | Internal |

### 3.1.5.4 APIs

In the context of the InCUBE project, VERIFY will provide a minimal RESTful API to allow access to its data model. For this purpose, the API specifications described below will be supported and provided to third parties.

**GET requests**

The following GET resources will be available:

- /api/buildings – Returns the list of buildings that are stored in VERIFY of an authenticated user. The list will contain basic, internal and non-confidential information, e.g. the name of the project and a global identifier.
- /api/buildings/{building_id} – Returns details about the building with building _id. This endpoint may include confidential information thus only authenticated entities will be able to access it.
- /api/buildings/{building_id}/analysis_results – Returns the results of the environmental and costing analysis KPIs for the buidling with building_id for authenticated users.

**POST requests**

The following POST resources will be available:

/api/buildings – Receives the definition of a project and creates the required resources in the local database in order to store and represent the input building.

### 3.1.6 InCUBE Planning Guide

The InCUBE Planning Guide (P-GUIDE) software solution is able to evaluate various renovation scenarios and propose the optimal scenario during the renovation design phase based on different Key Performance Indicators. These indicators are related to aspects such as cost, retrofitting time, smart readiness, environmental and energy impact, and others. One of the advantages of the solution is its high configurability, which allows it to adapt to different cases and users' preferences.

#### 3.1.6.1 Architecture

The P-GUIDE consists of two parts: a back-end engine that handles input data and performs the calculations, and a web-based user interface. The former makes use of external HTTP REST APIs and also provides HTTP REST API for data exchange with other components and the user interface.

#### 3.1.6.2 Protocols

**Read access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Digital Building Logbook | Retrieve properties of products, and dynamic information (current situation). Store and re-use output results. Retrieve outputs of modelling and evaluation tools of InCUBE (INTEMA) | HTTP | JSON | On-demand | Internal |
| Modular BIM/CIM Platform | Retrieve properties of products, characteristics of the buildings | HTTP | IFC | On-demand | Internal |
| Eclipse SensiNact Platform (DiTi) | Sensory data provider | HTTP | JSON | Periodic | Internal |
| VERIFY | Environmental and cost indicators values | HTTP | JSON | On-demand | Internal |
| INTEMA | Energy indicators | HTTP | JSON | On-demand | Internal |

| | | | | | |
|---|---|---|---|---|---|
| Job Scheduling Optimizer | Optimal sequence of interventions and retrofitting time | HTTP | JSON | On-demand | Internal |
| Lean Construction Platform | Environmental indicator | HTTP | JSON | On-demand | Internal |

**Write access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Digital Building Logbook | Store outputs | HTTP | JSON | Periodic | Internal |

### 3.1.6.3 APIs

P-GUIDE HTTP APIs have a structure similar to the one that is followed in the next indicative example.

Description: Posts a JSON containing essential information about a KPI's indicator for a specific renovation scenario of a demo site.

Method: POST http://<server_address>/dssinputscenarios/postValuePerIndicatorPerScenario

Request message body example:

```
{
"Indicator": 3,
"Unit": "%",
"Criteria": 1,
"Value": 36.74,
"Scenario": 1,
"SiteId": 1
}
```

Where:

- "Indicator" : Id of the indicator
- "Unit" : Measurement unit of the indicator
- "Criteria" : Boolean criteria of the indicator, 0 if the bigger the value the better, 1 for the opposite. For example the "CO2 emission savings" indicator would take a Criteria value of 0, whereas the "Overall project waste" indicator would take a Criteria value of 1
- "Value" : Value of the indicator
- "Scenario" : Id of the Scenario in which the indicator is metered [1,2,3]
- "SiteId" : Id of the pilot of the Scenario. 1 for Spanish Demo Site, 2 for Italian Demo Site, 3 for Dutch Demo site

### 3.1.7 Resilience Dashboard

The purpose of the dashboard will be to visualise the technological elements (described in the various paragraphs), in terms of their positioning on the map, operating status and current activity. Specifically, it returns possible alarms on the map in order to be able to proceed as quickly as possible to deal with an emergency due to the state of health of a worker, for example. The dashboard is not only a dynamic visualisation of the technological elements of the site, but also a network for querying daily reports in terms of verified alarms, near misses, etc. These will be displayed in PowerBI as a useful result for other tools.


*Figure 11 Resilience Dashboard*

#### 3.1.7.1 Architecture

It will be structured as PowerBI.

### 3.1.7.2 Protocols

**Read access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Lean Construction Platform | Renovation status | HTTP | JSON | Periodic | Internal |
| Modular BIM/CIM Platform | Possible updates of buildings description | HTTP | .ifc | Periodic | Internal |

**Access by other components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| AR/VR Training Suite | Access to reports and simulated situations | HTTP | .pdf / JSON | Periodic | Internal |
| Lean Construction Platform | Renovation status | HTTP | JSON | Periodic | Internal |
| Job Scheduler Optimizer | | | | | Internal |
| PPE Monitoring System | Push alerts/reports from safety components | HTTP / MQTT | JSON | Real-time | Internal |
| Anti-Collision System | Push alerts/reports from safety components | HTTP / MQTT | JSON | Real-time | Internal |
| Area Boundary System | Push alerts/reports from safety components | HTTP / MQTT | JSON | Real-time | Internal |
| Environmental Monitoring System | Periodic reports of statistical analysis | HTTP | JSON | Periodic | Internal |
| Waste Tracking and Management | Push reports | HTTP | JSON | Periodic | Internal |

**Write access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Digital Building Logbook | Store reports (human readable) | HTTP | JSON | Real-time | Internal |
| Eclipse SensiNact Platform (DiTi) | Notification of alerts<br>Store reports (values) | HTTP / MQTT | JSON | Real-time | Internal |

#### 3.1.7.3 APIs

It will draw on data available via REST API on the Smart Track platform and some excel or csv files containing simulated data for sensors and waste.

### 3.1.8 AR/VR Training Suite

The AR/VR Training Suite, part of the InCUBE Retrofitting Guide (R-Guide), utilises Augmented Reality technology for providing "off-line" and "on-the-job" training to workers. The training topics cover different aspects of renovation work, such as the use of new technologies and the installation of novel equipment, as well as the application of construction techniques. Furthermore, the AR/VR Training Suite can be used by workers in order to receive online assistance from more experienced workers or managers, upon request. The integration of the AR/VR Training Suite with the Resilience Dashboard allows for the display of information about installed devices and sensors, and the forwarding of notifications or alerts to be visible within the AR environment. The main objectives are to increase workers' safety and satisfaction, and accelerate the learning process compared to traditional techniques.

#### 3.1.8.1 Architecture

The suite comprises a set of applications that are either web-based or for mobile devices (i.e., smartphones, tablets, and HoloLens devices).

- A web application allows the user to author tutorials for procedures such as assemblies and installations of novel products related to retrofitting.
- A web server is responsible for establishing real-time communication between participants in the renovation works.
- A mobile application can display tutorials in AR on site, display information about monitored attributes in the building and planned activities, and allows a worker to initiate real-time communication with a colleague in order to get assistance.

#### 3.1.8.2 Protocols

**Read access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Resilience Dashboard | Location and status of installed sensors/devices per space | HTTP | JSON | On-demand | Internal |
| Renovation Marketplace | Online assistance communication | HTTP / WebRTC | JSON | On-demand / Real-time | Internal |
| Digital Building Logbook | Training status and history per user | HTTP | JSON | On-demand | Internal |
| Modular BIM/CIM Platform | 3D representation of buildings and assets | HTTP | JSON | Periodic | Internal |

**Access by other components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Renovation Marketplace | Online assistance communication, Training status | HTTP / WebRTC | JSON | Real-time | Internal |

**Write access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Digital Building Logbook | Store training reports | HTTP | JSON | Periodic | Internal |
| Renovation Marketplace | Online assistance communication | HTTP | JSON | Periodic | Internal |

#### 3.1.8.3 APIs

HTTP REST API is provided for internal data exchange among the applications of the AR/VR Training Suite (e.g. for managing tutorials and storing them to the DBL in the appropriate data format).

e.g. HTTP POST /addTrainingTutorial

HTTP POST /deleteTrainingTutorial

HTTP GET /getTrainingTutorial

Furthermore, a WebRTC server is included for the implementation of the WebRTC protocol by maintaining websocket connections and keeping the peers synchronised.

The AR/VR Training suite makes use of HTTP APIs of other InCUBE components in order to retrieve or store information, e.g.:

- Digital Building Logbook for retrieving information about users, products, and training material and storing training report events.
- Modular BIM/CIM Platform for retrieving BIM models of buildings and assets.
- Resilience Dashboard for retrieving data about sensors/devices configurations and their values.

### 3.1.9 Lean Construction Platform

#### 3.1.9.1 Architecture

Lean Construction Platform is part of WINER, along with the Job Scheduling Optimizer (JSO). Both tools complement each other to provide the supervision and optimization of the entire lifecycle of the building renovation process, generating the optimal construction/renovation actions sequence in terms of time, cost, and safety. Since these tools, along with P-GUIDE, share several input parameters to generate the actions sequence, there has been discussion about establishing a single

form. This would allow the user to fill it out only once, and then it would be shared following the guidelines indicated in Section 2.2. However, the specifics of how this exchange will be carried out, as well as the format or syntax, have not yet been defined.

Additionally, there is an intention that Lean Construction Platform retrieves information from the BIM/CIM platform and R-GUIDE, which is also pending further specification. From the BIM/CIM platform, the plan is to obtain information related to the renovation process: generated waste, transportation, etc. On the other hand, from the R-GUIDE, the aim is to gather information from historical events that may be related to safety on the construction site.

#### 3.1.9.2 Protocols

**Read access to components**

The Lean Construction Platform will access the BIM\CIM platform and the R-GUIDE when required to compute or update the recommended sequence of actions or some parameter to be displayed in the interface. It will not proactively retrieve data from any component.

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Resilience Dashboard | Access history of events | HTTP REST\ DB Queries | To be defined | Non-periodic | Internal |
| Modular BIM/CIM Platform | Access BIM models | HTTP | To be defined | Non-periodic | Internal |

**Access by other components**

As previously mentioned, the Lean Construction Platform (LCP) will share information with the Job Scheduling Optimizer (JSO) and the P-GUIDE, as they share numerous user-input parameters. They also may share results related to the performance of the recommended action sequence.

| Client component | Access description | Access protocol | Data Format / Syntax | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Job Scheduler Optimizer | Information provided by the user | HTTP REST / MQTT | To be defined | Non-periodic | |
| *P-GUIDE* | Information provided by the user | HTTP REST / MQTT | To be defined | Non-periodic | |

**Write access to components**

Similarly, the LCP will be able to share information with the JSO and P-GUIDE regarding user-entered parameters or sequence of actions.

| Target component | Access description | Access protocol | Data Format / Syntax | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Job Scheduler Optimizer | Information provided by the user | HTTP REST / MQTT | To be defined | Non-periodic | |
| *P-GUIDE* | Information provided by the user | HTTP REST / MQTT | To be defined | Non-periodic | |

#### 3.1.9.3 APIs

As mentioned above, the communication between the LCP, the JSO and the P-GUIDE has yet to be defined. However, the LCP will probably be provided with a RESTful API to allow access to this information. The definition and implementation are in process at the time of writing this deliverable.

### 3.1.10 Job Scheduling Optimizer

#### 3.1.10.1 Architecture

The Job Scheduling Optimizer (JSO) is an online tool, generating the optimal construction/renovation actions sequence, in terms of time and cost efficiency, according to the selected constraints and preferences (e.g., tenant's disruption, available workers, precedence relations between activities). JSO has a fairly simple architecture which is presented in Figure 12. It consists of a web application and a database where its local data are stored. The users can interact with the application through its graphical user interface in order to set up renovation plans and perform scheduling. The task of scheduling is assigned to another in-house web service which utilizes scheduling algorithms on a set of jobs to be scheduled and returns the results to the web application. The communication is realized using the REST API that the scheduling service provides. Finally, JSO has API capabilities on its own, thus, external tools can request data considering the renovation plans and their scheduling.

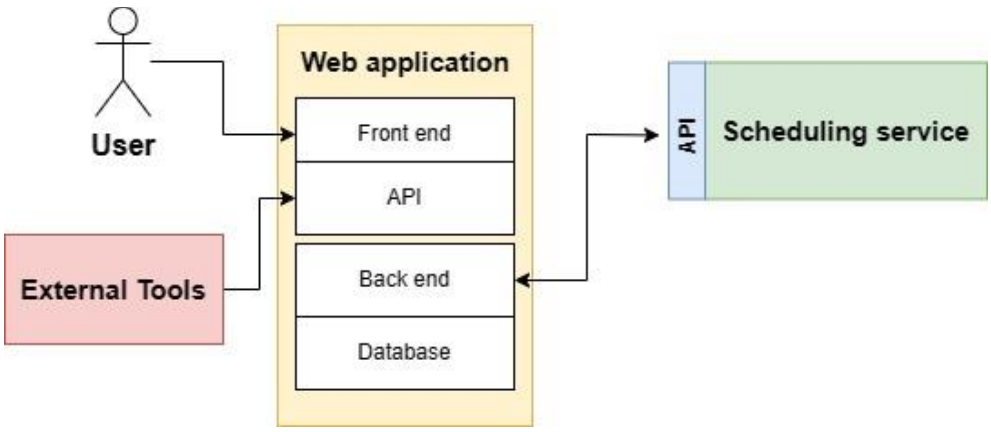



*Figure 12 JSO Architecture*

### 3.1.10.2 Data Model

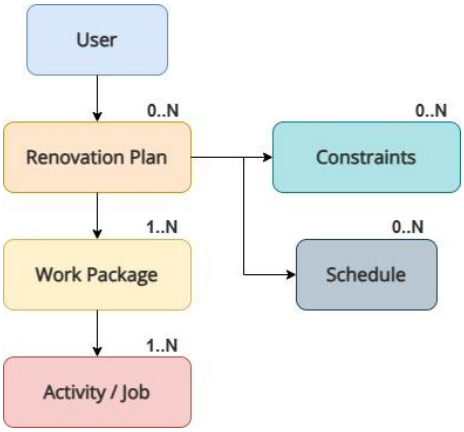



*Figure 13 JSO Data Model*

JSO represents its data using the following model, shown in Figure 13:

- **User**: The representation of a user of JSO. Each user may own/manage 0 to multiple renovation plans.
- **Renovation plan**: Represents the renovation plan that will be applied to a building. A renovation plan may include one or more Work Packages.
- **Work package**: Represents a set of jobs that will be performed during the renovation plan. Each work package is an isolated set of jobs that are closely related to one another. The work packages are used for better organization of jobs in a renovation plan. Each work package may have one or more activities/jobs.
- **Activity / Job**: Represents a physical activity that will be performed on the building during the renovation plan. Each activity/job has several characteristics that are useful during the planning, e.g. its duration and the number or workers required.
- **Constraints**: Represents constraints that are applied during the scheduling and consider the available resources of the building, owners or construction companies. A renovation plan may have none or many constraints. A constraint may be the limited available budget per period, which might prevent some jobs from taking place.
- **Schedule**: Represents the calculated schedule for a renovation plan. The schedule indicates the time point an activity/job should start being processed and how many days it will last. Moreover, it contains general statistics about the schedule, e.g. the total duration. A renovation plan may have none or many schedules, since different schedules may be calculated to test different constraints, methods or optimization factor.

### 3.1.10.3 Protocols

**Read access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Lean Construction platform | Internal information about scheduled actions | HTTP | JSON | On demand | Internal |
| P-GUIDE (Dynamic Decision-Making Tool) | Retrieve the optimal renovation scenario | HTTP | JSON | On demand | Internal |

| | | | | | |
|---|---|---|---|---|---|
| Resilience Dashboard | Value chain processes to be scheduled | HTTP | JSON | On demand | Internal |

**Access by other components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| P-GUIDE | Retrieve the optimal renovation scenario | HTTP | JSON | On demand | Internal |
| Lean Construction Platform | Internal information about scheduled actions | HTTP | JSON | On demand | Internal |
| Resilience Dashboard | Value chain processes to be scheduled | HTTP | JSON | On demand | Internal |

**Write access to components**

Besides the Digital Building Logbook, JSO will only communicate with the Lean Construction platform in the context of the WINER tool. In more detail, JSO will provide LCP with jobs data and several KPIs considering the finalized schedule.

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Lean Construction platform | Internal information about actions to be scheduled | HTTP | JSON | On demand | Internal |
| Digital Building Logbook | Optimal sequence of actions of various scenarios | HTTP | JSON | On demand | Internal |

### 3.1.10.4 APIs

In the context of the InCUBE project, JSO will provide a minimal RESTful API to allow access to its data model. For this purpose, the API specifications described below will be supported and provided to third parties.

**GET requests**

The following GET resources will be available:

- /api/renovation_plans – Returns the list of renovation_plans that are stored in JSO for the authenticated user. The list will contain basic, internal and non-confidential information, e.g. the name of the project and a global identifier.
- /api/renovation_plans/{renovation_plan_id} – Returns details about the renovation_plan with *renovation_plan_id*, including the list of jobs and their details. This endpoint may include confidential information thus only authenticated entities will be able to access it.

- /api/renovation/{ renovation_plan_id}/schedule – Returns the results of the scheduling, including the detailed schedule and the calculated KPIs for the project with renovation_plan_*id*.

**POST requests**
The following POST resources will be available:

- /api/renovation_plans – Receives the definition of a renovation_plan and creates the required resources in the local database in order to store and represent the input project.

### 3.1.11 Renovation Marketplace

The InCUBE Renovation Marketplace makes digitally available the innovative products, processes and business models related to deep renovation of buildings. In particular, information on novel envelope and construction materials solutions, energy systems, advanced HVAC systems, retrofitting techniques, and training material will be included. The Renovation Marketplace is a web-based tool openly accessible. Various stakeholders such as engineers, workers, technology developers, and building owners, will be able to communicate and react among each other. In addition, users will be able to monitor the overall renovation activities flow as well as buildings operations. Users who take part in the renovation works will be able to enhance their capacity on deep-renovation related processes and also provide assistance to workers on site. One of the main characteristics of the tool is that it can be used by third parties, allowing them to both access InCUBE results and present their own technologies.

#### 3.1.11.1 Architecture

The Renovation Marketplace is a web application that is consisted of the back-end part and the user interface part (front-end). The former exposes HTTP web services that are utilised by the user interface in order to retrieve and send the data. Moreover, it accesses HTTP APIs that are provided by other InCUBE tools.

#### 3.1.11.2 Protocols

**Read access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Digital Building Logbook | Information on renovation works. Information on materials, products, and processes | HTTP | JSON | On demand | Internal |
| Module BIM/CIM Platform | Retrieve the list of available BIM assets | HTTP | JSON | On demand | Internal |

| | | | | | |
|---|---|---|---|---|---|
| AR/VR Training Suite | Training status data | HTTP | JSON | On demand | Internal |

**Access by other components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| AR/VR Training Suite | Online assistance communication | HTTP / WebRTC | JSON | On demand / Real-time | Internal |

**Write access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Digital building Logbook | Addition of descriptions of new products/technologies | HTTP | JSON | On demand | Internal |
| AR/VR Training Suite | Online assistance communication | HTTP REST / WebRTC | JSON | On demand / Real-time | Internal |

#### 3.1.11.3 APIs

Indicative examples part of the HTTP REST API provided by the back-end of the Renovation Marketplace are the following:

HTTP GET /marketplace/getProductInformation

HTTP POST /marketplace/addProduct

HTTP POST /marketplace/addProcess

HTTP POST /marketplace/addTechnologyDescription

HTTP POST /marketplace/addNewsArticle

HTTP POST /marketplace/addConnection

HTTP GET /marketplace/getConnections

### 3.1.12 Energy Cloud EMS

This solution applies the performance of an industrial SCADA system to the Building Energy Management, being able to collect Energy Data from many different hardware devices and enabling to visualize the Energy consumption the building in an intuitive and simple way via a 3D BIM model. CIRCE ENERGY CLOUD will show in a multiplatform and comprehensive way the energy consumption of the shareholders, providing in a 3D environment energy information of every user. Through InCUBE an innovative approach to the BEMs will become available, empowering the non-technically trained citizens via 3D BIM intuitive platforms, and providing real energy estimations of their day-by-day appliances, just with one electricity meter.

#### 3.1.12.1 Architecture

Figure 14 shows the architecture of the EMS (Energy Cloud System) which is composed of three main parts: sensor, gateway and broker.

The AM307(L) series is a compact indoor environment monitor that measures temperature, humidity, light, $CO_2$ concentration, TVOC, barometric pressure and motion. The data is displayed on the screen in real time, helping to measure indoor environment and comfort. The sensor data is transmitted via LoRaWAN technology. By combining Milesight's LoRaWAN gateway and Milesight IoT Cloud, all sensor data can be managed remotely and visually.

UG65 is a robust indoor LoRaWAN gateway, has a line-of-sight of up to 15km and can cover about 2km in an urban environment, which is ideal for certain indoor applications. The plurality of connections of this equipment makes it interesting for the development of the project. In our case, the communication protocol to be used is MQTT.

An MQTT (Message Queuing Telemetry Transport) broker is a server that facilitates communication between devices using the MQTT protocol. MQTT is a lightweight and efficient messaging protocol designed for resource-constrained devices such as sensors and actuators on the Internet of Things (IoT).

The MQTT broker acts as an intermediary between devices that send messages (called clients) and devices that receive those messages. Its main function is to receive messages from the clients and route them correctly to the target clients.
Some key features of an MQTT broker include:
**Publish/Subscribe**: MQTT uses a publish/subscribe model, where clients can publish messages to specific "topics" and subscribe to topics of interest. The broker ensures that messages are delivered to customers subscribed to those topics.
**Message Retention:** MQTT brokers can retain the last message posted in a topic, which means that a new subscriber can receive the most recent message as soon as they subscribe to that topic.
**Quality of Service (QoS):** MQTT supports QoS levels that determine the level of message delivery guarantee. Levels range from 0 (unconfirmed delivery) to 2 (guaranteed delivery).
**Persistence:** Some MQTT brokers can store messages persistently, which means that messages are not lost even if a client or the broker itself is rebooted.
**Security:** MQTT brokers often include security mechanisms, such as authentication and encryption, to protect communication between clients and the broker.

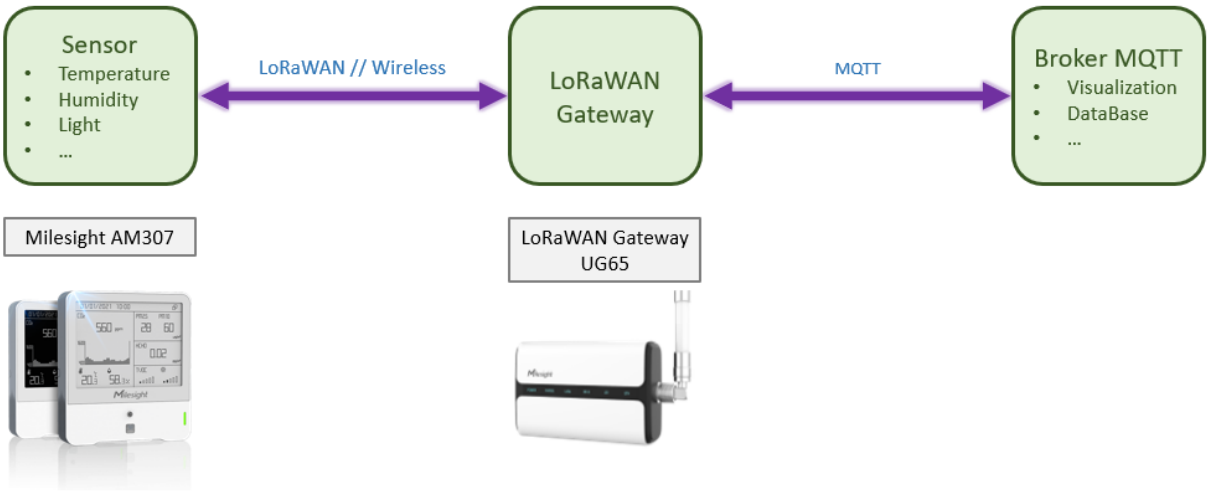



*Figure 14 Energy Cloud System Architecture*

### 3.1.12.2 Protocols

**Inputs**

| Input Origin | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity | Read format |
|---|---|---|---|---|---|---|
| Electric consumption | Electric consumption via Datadis | MQTT | Json | Daily | Confidential | MQTT |
| Gas consumption | Gas consumption in bills | Manually | Json | Monthly | Confidential | MQTT |
| Comfort devices | Comfort data | MQTT | Json | 15 min | Confidential | MQTT |
| Modular BIM/CIM Platform | Building information and 3D structure | HTTP | IFC /.rvt (to be defined) | Once | Internal | MQTT |

**Read access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Modular BIM/CIM Platform | Access BIM models | HTTP | BIM/CIM files | Periodic | Internal |

### 3.1.13 Smart Building EMS (S-BEMS)

The Smart Building EMS distributes its intelligence following an edge-to-cloud architecture where the edge element equipped with intelligence is provided by TERA, while the cloud element equipped with intelligence corresponds to the AI algorithms developed by FBK.

The main edge element is the Beeta™ Box (hereinafter BeetaBox) that is an edge computer - multiprotocol gateway - designed for indoor IoT ecosystems. It is based on LINUX Embedded platform, which allows implementing software solutions which can run in a standalone mode or interfaced to remote web services. The use of standardized protocols and communication interfaces makes this electronic control unit an unprecedented multi-protocol gateway and allows full configurability, modularity, and scalability of BeetaBox, whose embedded SW can be upgraded

remotely (OTA). This feature is of great value for the maintenance and upgrading of the BeetaBox to ensure that the number of devices and protocols supported are compatible and aligned with the market evolutions.

It can be used in combination with third party software platforms/tools/frameworks for the implementation of an integrated management and control systems, in applications like Smart Home (Behind the meter), Smart Metering, asset management, (Building/Energy Management System), Smart Grid services, security, automation.

BeetaBox is an edge computer characterised by a range of factures, performances and communication interfaces that is one of a kind, being however able to be customise for different applications, configuring its equipment from the top of the range up to ad-hoc versions (outfitting). BeetaBox is based on ARM Cortex A7 processor, with several embedded IoT wireless modules (Wifi, 802.15.4, Bluetooth, Z-wave, WM-Bus 169MHz, NB-IoT or 868MHz LoRaWAN) and wired connectivity like RS485 (e.g., Modbus, Sunspec and others for photovoltaic/battery management inverters etc.), Gigabit Ethernet (Bacnet, Modbus, KNX, Daikin, etc.), USB, S0 and dedicated I/O (Dry Contact, Open Collector).

Moreover additional 3 USB ports can be used to easily expand the BeetaBox to include modules like GPRS/UMTS/4G, etc. The available I/Os, make it possible also to get data from smart meter directly or to drive load or boiler through relay modules.

BeetaBox has also internal devices such as sensors for Temperature, relative humidity and air pressure, microphones, speakers and an optional Trusted Platform Module (TPM 1.2) soldered chip; sensors can be used in combination to visual and acoustic embedded actuators for smart feedback to the users.

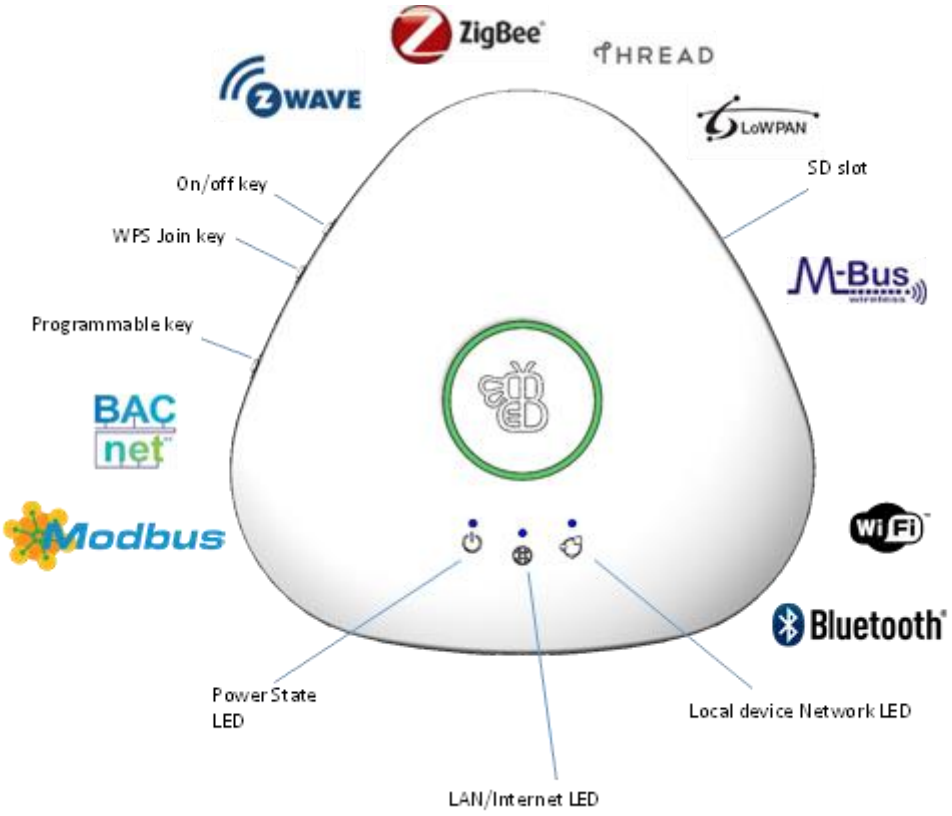



*Figure 15 BeetaBox IoT Edge Computing Gateway Front Face*

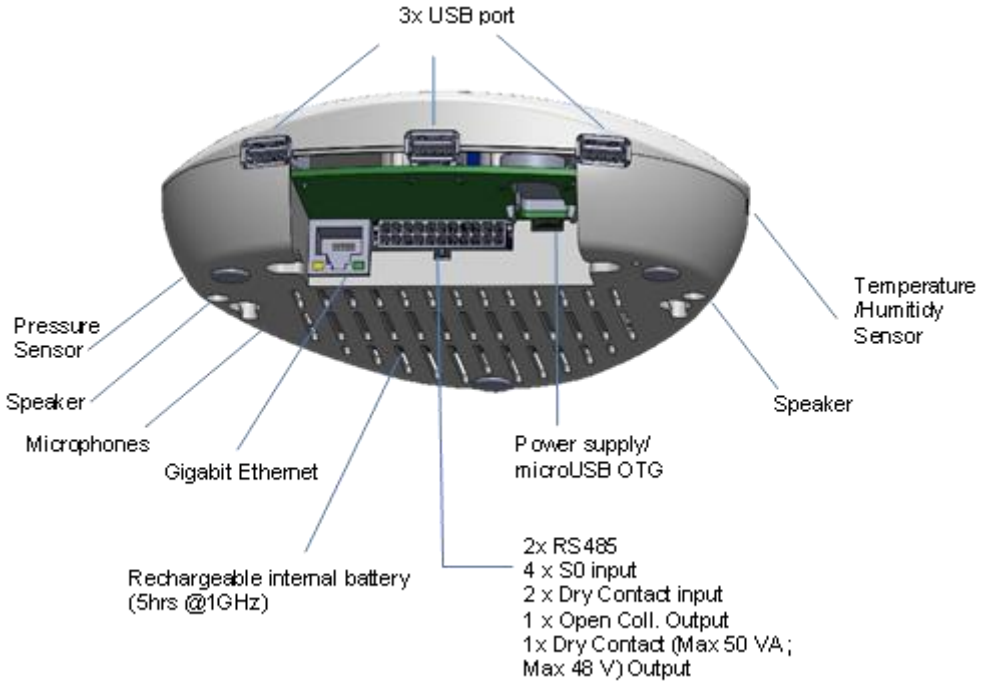



*Figure 16 BeetaBox IoT Edge Computing Gateway Rear Face*

The BeetaBox is able to receive and store the data provided by the smart meters “1G” and “2G” installed in by e-Distribuzione (the main Distributor in Italian electricity grid and in some other countries) via Power Line, connected via USB port:

- in "A" band, when connected to devices such as the e-distribution "Smart Info" (or other devices equipped with an e-distribution "MOME" card, such as the Beeta Power in the MOME version), communicating with the first-generation counters (also called "1G", for example GEM, GEMIS) or even with 2G generation meters (2.1), enabled to send data on band "A";
- in band "C", when connected with devices such as the Beeta Power "2G" version, in this case communicates only with the "second generation" meters, or "2.0 electronic meters" or "smart meter 2.0", or simply "2G" (e.g., models CEG2, CE2G, "gen2"); it should be noted that, for those of the second generation, new features will be activated which are currently being tested.

As a powerful edge computer, there are several options in using BeetaBox:

a. without SW: customer/partner installs Linux and other SW tools;
b. equipped with pre-installed Linux arranged and tuned by us (ArmBian); several open-source software tools and frameworks can be used, like NodeRed, Home Assistant etc.
c. equipped with pre-installed Linux arranged and tuned by us (ArmBian) plus a middleware (OSGI compliant) customized by TERA and some "drivers" developed by Tera;
d. equipped with pre-installed Linux arranged and optimized by us (ArmBian) plus some middleware (OSGI compliant) customized by TERA and some "drivers" developed by Tera, including MQTT configured to send data to an MQTT broker (a sort of "IoT" platform " that some potential partners have);
e. equipped with pre-installed Linux arranged and optimized by us (ArmBian) **plus “FIN FRAMEWORK” (by J2 Innovation- A Siemens Company);** thanks to FIN framework is compliant with the open-source initiative Project Haystack to streamline working with data from the Internet of Things.

In this initial pre-monitoring phase, the gateway will only act as a data logger – a dedicated feature designed for InCUBE project purposes since no direct communication to the destination cloud platform will be made available for the pre-monitoring phase. Retrieved data will be stored inside its 128 GB microSDHC internal memory.

Only in 2025 collected data will be directly sent to the cloud-based software platform developed by EVOLVERE.

The acquired data from the IoT gateway acting as a data logger could be exported at the next file extensions: *.json. To facilitate project partners data manipulation, TERA will develop a software module for parsing *.json data file to *.csv or *.xlsx file.

#### 3.1.13.1 Data Model

The data recorded will be used for training the predictive models (forecasting of energy consumption and RES production, with special emphasis given on the maximization of self-consumption). Once the EVOLVERE platform will be available, the results of the model can return to the BeetaBox using standard MQTT protocol.

### 3.1.13.2 Protocols

**Read access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Digital Building Logbook | Access to EMS sensors/indoor comfort data | MQTT | JSON | To be defined | internal |
| P-GUIDE | Information provided by the user | MQTT | JSON | To be defined | internal |
| Multimeter | Energy data from Cuminetti Teather | Modbus | JSON | 15 minutes | internal |
| Smart plug | Appliances energy data | Zwave | JSON | 15 minutes | internal |
| Multiparamter sensors | Indoor comfort data | Zwave | JSON | 15 minutes | internal |

**Write access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Digital Building Logbook | Propagates events from IoT sensors and smart devices | MQTT | JSON | To be defined | internal |
| P-GUIDE | Information provided by the user | MQTT | JSON | To be defined | internal |
| EvoDistrict | Energy and comfort data from field sensors | MQTTS | JSON | 15 minutes | internal |

### 3.1.13.3 APIs

TERA will provide a RESTful API to allow third parties read/write data from IoT edge gateway via its internal Device Configuration Platform software. The current version API v1 (/api/v1) containing initial references to the following features:

- Auth /auth
- Super /super
- System /system
- MQTT /mqtt
- Logs /logs

Furthermore, it is expected that some data (to be identified) will be updated in real time over web socket.

### 3.1.14 District EMS (EvoDistrict)

EvoDistrict is a Software-as-as-a-service cloud platform that manages the distributed energy resources that fall within the demo site #1 of the INCUBE project, i.e. the Santa Chiara District, located in the city of Trento, Italy. At present, these distributed resources are renewable sources generators (e.g. PV plants) and electrical loads (offices, meeting rooms, gyms, etc.); storage systems (e.g. Li-ion batteries) and heat pumps may be included later in the INCUBE project. EvoDistrict aims to transform and manage demo site #1 as an energy community, in accordance with the Italian regulatory framework currently in force.

#### 3.1.14.1 Architecture

The EvoDistrict architecture consists of three parts: edge, gateway and cloud platform as shown in Figure 17 a). This architecture applies to sensors and meters in general, but also to the BeetaBox IoT Edge Computing Gateway produced by partner Tera. The sending of telemetry data (e.g. energy and power measurements) from the edge to the EvoDistrict platform occurs securely and reliably:

1. Edge to cloud: sensors, smart meters and photovoltaic plants communicate locally with Eugenio gateway (Figure 18) via proprietary application protocols or via MQTT protocol on the local Wi-Fi network. Eugenio gateway transmits data to EvoDistrict using MQTT protocol.
2. Cloud to cloud: edge devices transmit data to their manufacturers' cloud platforms and the latter transfers the data to EvoDistrict via third-party APIs or MQTT protocol.

Security in the scenarios described above is met by the application of the three key principles of cybersecurity (CIA triad):

- Confidentiality: the confidentiality (privacy) of the data is protected by mean of authentication and double encryption techniques which inhibit access to unauthorized users. Contextualizing in EvoDistrict, edge-to-cloud communication adopts the MQTTs protocol (secure version of the MQTT protocol based on TLS), while cloud-to-cloud communication adopts the MQTTs or HTTPs protocol. In both cases, authentication is mandatory.
- Integrity: data protection from unauthorized external tampering in EvoDistrict is guaranteed by authentication policies that monitor access attempts, by authorization mechanisms according to the "Least Privilege" (PoLP) principle, and by anti-intrusion systems.
- Availability: this principle states that data must be provided when a resource put the request, and no service interruption can compromise the data availability. In EvoDistrict, concepts of hardware, network and software redundancy as well as disaster recovery and backup plans are applied as countermeasures to protect data availability.

The architecture in Figure 17a) changes when photovoltaic plants and smart meters are considered. In fact, the measurements on the photovoltaic plants are shared via a cloud-to-cloud communication, between the cloud infrastructure of the photovoltaic inverter manufacturer and the EvoDistrict platform, as shown in the Figure 17b). This new architecture is possible on condition

that the photovoltaic inverter manufacturer is among those manufacturers that have signed a specific confidentiality agreement with Evolvere.

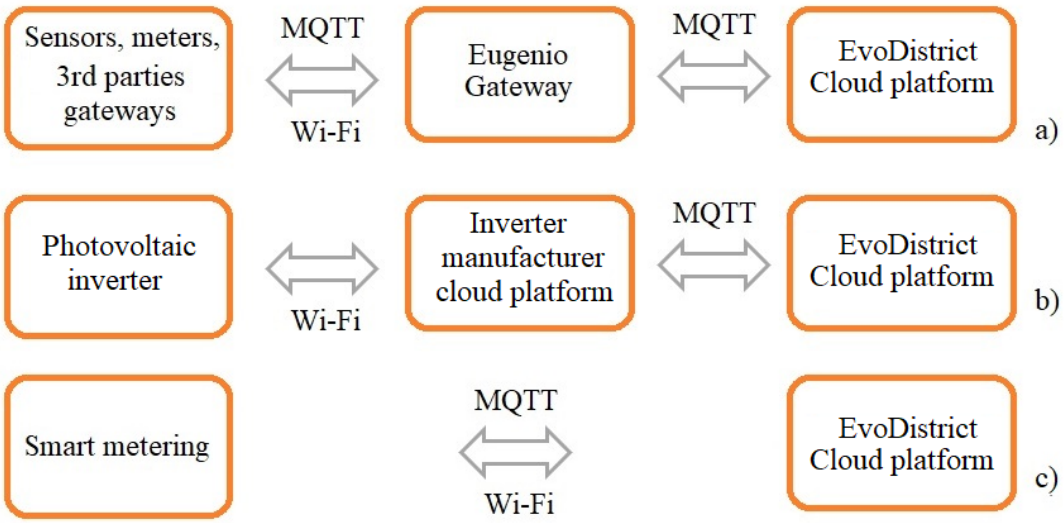



Figure 17 EvoDistrict Architecture

Figure 18 Eugenio gateway

On the contrary, smart meters send the local measurements directly to EvoDistrict platform, as shown in Figure 17c). More precisely, the smart meter shown in Figure 19 communicates via power line communication with the local DSO meter, using the so-called Chain 2 protocol, and receives the measurements of the energy withdrawn/injected at the point of delivery. Then, the smart meter sends directly these measurements to the EvoDistrict platform, via local Wi-Fi and MQTT protocol.

Figure 19 Smart Meter

### 3.1.14.2 Protocols

**Write access to components via Eugenio Gateway**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Data from Smart meters (at the point of delivery) | energy withdrawn and injected | MQTTs | JSON | 15 minutes | Internal |
| Data from Sensor, meters, BeetaBox | Indoor/outdoor parameters | MQTTs | JSON | To be defined | Internal |
| Data from PV plants | Energy produced | MQTTs | JSON | To be defined | Internal |

**Write access to components via cloud-to-cloud integration**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Data from Smart meters (at the point of delivery) | energy withdrawn and injected | HTTPS | JSON | 15 minutes | Internal |
| Data from Sensor, meters, BeetaBox | Indoor/outdoor parameters | HTTPS | JSON | To be defined | Internal |
| Data from PV plants | Energy produced | HTTPS | JSON | To be defined | Internal |

### 3.1.14.3 APIs

EvoDistrict will provide a RESTful API to allow third parties read/write access to the database. At the moment, it is not known which data will need to be shared therefore the specifications of the RESTful API have not yet been defined.

The predictive models implemented (forecasting of energy Consumption/self-consumption and RES production) will use all the available data collected by the EvoDistrict. More in detail, data will be

extracted using the implemented API's and used for training the models in a separate ad-hoc ecosystem (GPU-cluster based). Once validated, the models will be uploaded to the cloud and a dedicated API will serve the results of such models writing the forecasted values as a standard component. Such data can be, for example, retrieved by the BeetaBox and consumed locally.

### 3.1.15 PPE Monitoring System

The helmets will be equipped with accelerometer sensors to understand the position of the worker and possibly transmit an alarm. These sensors are not only a warning system but also a way of checking that the helmet is used correctly according to the regulations in force. All the sensors are connected to the dashboard (see description) so both the position of the worker and the alarms detected will be recorded and collected in an event register.

#### 3.1.15.1 Architecture

The system is composed of the following elements:

- N Anchor / Beacon device with periodic BLE signal transmission functionality to be installed in indoor areas. Their purpose is to function as an indoor satellite for the localization of the emergency event generated by the WeTAG.
- N small DPI Tag devices to be installed on the DPI and with periodic BLE signal transmission functionality
- A WeTAG device wearable by the operator on the ground

The WeTAG is equipped with:

- BLE receiver capable of picking up the signal from the indoor localization Anchors and from the BLE Tags
- microcontroller that implements event localization, detection and forwarding algorithms
- GPS receiver for outdoor localization
- WiFi transceiver to communicate events to the server
- 2G/2.5G transceiver to communicate events to the server

The WeTAG allows you to alert the operator when:

- An involuntary or voluntary man down condition is detected
- a prolonged lack of at least one of the PPE Tags worn is detected

The WeTAG stops alerting the operator when:

- the radio signal of the missing DPI Tag returns to being perceived according to the set parameters
- the man down condition is manually restored

The WeTAG works as a signal concentrator and allows the forwarding of events to the server via 2G / 2.5G or WiFi connectivity and through specific APIs. Figure 20 shows the block diagram of the system:

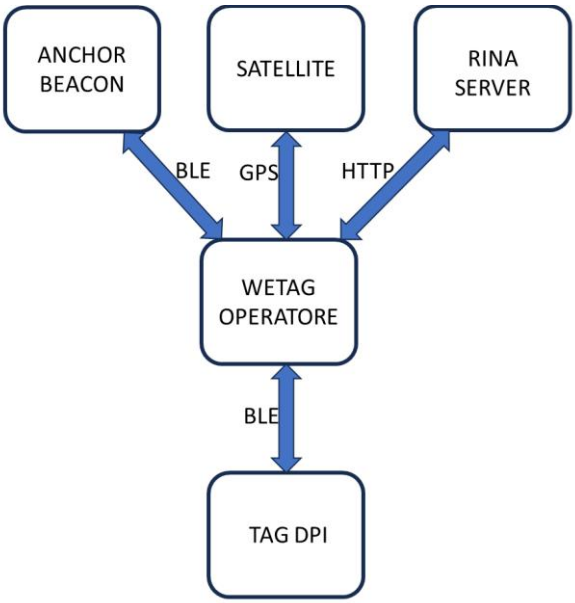



*Figure 20 Block Diagram of PPE Monitoring*

### 3.1.15.2 Protocols

**Write access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Resilience Dashboard | Reports on safety rules compliance | HTTP | .pdf / JSON | Periodic | Internal |

### 3.1.15.3 APIs

The following API has been defined for downloading data from the SmartTrack platform:

GET(POST) /events

From: date
To: date
Source: source_type
Type: event_type
Location: string (it may be a concatenation of location-floor-area)
Floor: string (it may be a concatenation of location-floor-area)
Area: string (it may be a concatenation of location-floor-area)
Response:
[events] (with all admissible values)

### 3.1.16 Anti-Collision System


*Figure 21 Anti-Collision System*

The anti-collision system (Figure 21) will be adapted to the only operating machine on the site, namely the tower crane. This is because after the inspection we considered monitoring the loading area of the crane by sensing the boom and the loading system for the danger of suspended loads, so when manoeuvring it will detect the presence of workers in areas at risk of falling loads.

#### 3.1.16.1 Architecture

The system is composed of the following elements:

- A tag / beacon device to be applied to the CRANE and which allows you to detect the movement of the load in two directions, in particular and equipped with:
  - accelerometer to detect if the moving part is moving
  - BLE transmitter (ptx + 6.2 dBm) ON/OFF controlled by the accelerometer
  - BLE transmitter (ptx + 6.2 dBm) ON connectable for its reprogramming
  - CR2450 lithium battery which allows it to work for years
  - industrial 3M double-sided tape or buttonholes to be applied using cable ties
- An architecture of at least 4 UWB Anchors in the external loading/unloading area with related support and power poles
- WiFi network with network output to a defined endpoint near the loading/unloading area. Request the Municipality for this necessary aspect.
- A WeTAG device wearable by the operator on the ground and equipped with:
  - BLE receiver capable of picking up the signal transmitted by the tag / beacon
  - microcontroller that implements event localization, detection and forwarding algorithms
  - UWB transceiver to communicate with the Anchor UWB architecture
  - WiFi transceiver to communicate events to the server

The proposed system allows the operator to be alerted in the field when he is inside the prohibited polygon and only if, at the same time, the crane is moving.

To detect when the crane is moving, a tag is applied to the moving part of the crane which, as it moves, will produce some vibration / movement on the tag, activating it. The sensitivity of tag vibration/motion detection is configurable via BLE.

Below are the functional logical steps:

1. Creation of the logical geo-fencing polygon and insertion within the WeTAG
2. When the tag detects continuous vibrations / movements for n seconds (parametrizable) it transmits a periodic BLE signal for a time window of m seconds (parametrizable).
3. The WeTAG calculates its position periodically using the Anchor UWB architecture (the rate can be parameterised and is between 10s and 100ms) and continuously checks whether:
    a. is located within the geo-fencing polygon
    b. if it senses the BLE signal transmitted by the tag.

The WeTAG alerts the operator (sonic / vibrating / visual feedback) when:

- both conditions a and b are verified

The WeTAG stops alerting the operator when:

- condition a or b or both are not met

Each change of state produces the recording of an event which is forwarded by the WeTAG via a specific API and WiFi network.

Figure 22 shows the block diagram of the system:

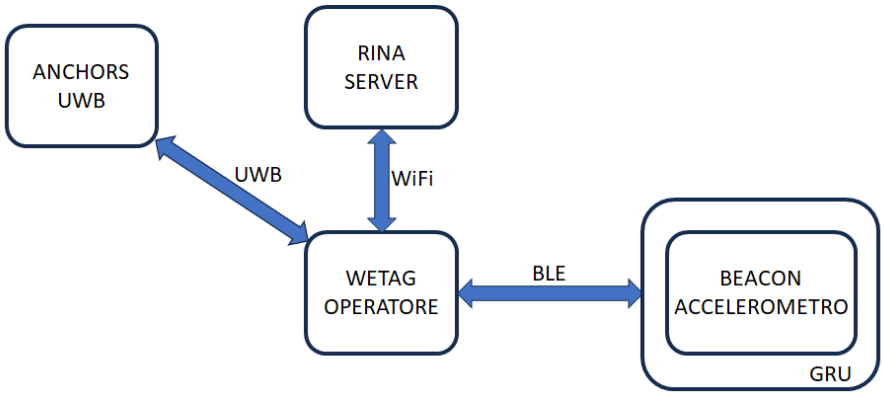



*Figure 22 Block Diagram of Anti-Collision System*

### 3.1.16.2 Protocols

**Write access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Resilience Dashboard | Reports on detections of risk of collisions | HTTP / MQTT | .pdf / JSON | Periodic | Internal |

### 3.1.16.3 APIs

The following API has been defined for downloading data from the SmartTrack platform:

GET(POST) /events
From : date

To: date
Source: source_type
Type: event_type
Location: string (it may be a concatenation of location-floor-area)
Floor: string (it may be a concatenation of location-floor-area)
Area: string (it may be a concatenation of location-floor-area)
Response:
[events] (with all ammissible values)

### 3.1.17 Area Boundary System

The area boundary system prevents the presence of workers and machine in not allowed or dangerous areas. In case of workers, this is possible areas through alarms that will ring when the operator is in the proximity of selected areas. When it refers to machines it is possible to implement alarm that can notice the danger to the operators as well as directly implementing remote control features. This feature will be used for workers by using the hardware installed on the smart helmet. Similarly, the sensors can be installed on machines to verify that their operation remains within a certain area.

#### 3.1.17.1 Architecture

The system is composed of the following elements:

- N Anchor / Beacon device with periodic BLE signal transmission functionality to be installed in no-access areas. Their purpose is to function as an indoor satellite for localizing the event of access to a prohibited area
- A WeTAG device wearable by the operator

The WeTAG is equipped with:

- BLE receiver capable of picking up the signal transmitted by the indoor localization Anchors
- microcontroller that implements event localization, detection and forwarding algorithms
- GPS receiver for outdoor localization
- WiFi transceiver to communicate events to the server
- 2G/2.5G transceiver to communicate events to the server

The WeTAG allows you to alert the operator when:

- the radio signal transmitted by one of the prohibited areas signalling Anchors is detected and above threshold (indicates proximity to the prohibited area). The transmission power set on the prohibited area signalling anchors is low (a few meters)

The WeTAG stops alerting the operator when:

- the radio signal transmitted by one of the prohibited areas signalling Anchors returns below threshold or is no longer perceived continuously until a certain timeout (indicates distance from the prohibited area)

Each change of state produces the recording of an event which is forwarded to RINA via a specific API.

The WeTAG works as a signal concentrator and allows the forwarding of events to the server via 2G / 2.5G or WiFi connectivity with specific APIs. Figure 23 shows the block diagram of the system:

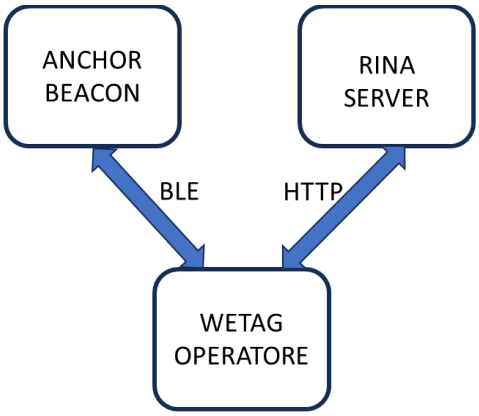



*Figure 23 Block Diagram of Area Boundary System*

#### 3.1.17.2 APIs

The following API has been defined for downloading data from the SmartTrack platform:

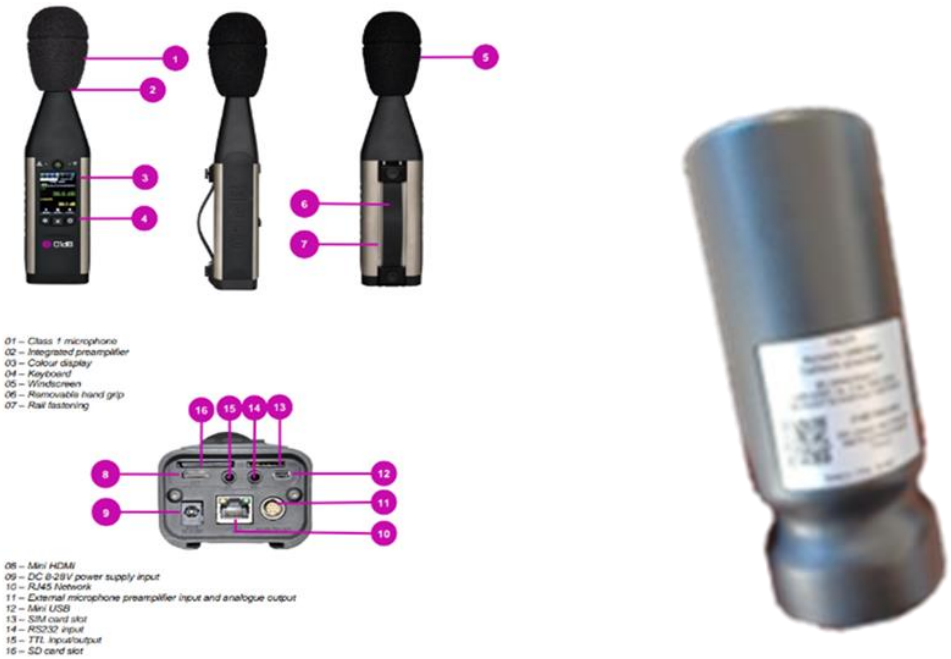
GET(POST) /events
From : date
To: date
Source: source_type
Type: event_type
Location: string (it may be a concatenation of location-floor-area)
Floor: string (it may be a concatenation of location-floor-area)
Area: string (it may be a concatenation of location-floor-area)
Response:
[events] (with all ammissible values)

### 3.1.18 Environmental Monitoring System

The Environmental monitoring system is composed of two main instruments:

- The sound level meter 01dB Smart FUSION 4G provided by ACOEM Group: the function of this instrument is to assess noise on the worksite. During a measurement with the instrument in hand, the sound level meter must be pointed at the source according to standard IEC 60651. The instrument is shown in Figure 24 with a general hardware overview and its calibrator.

The thermohygrometer DMA875 provided by LSI LASTEM (Figure 25): this is an instrument able to




measure environmental temperature, relative humidity and dew point. The determination of dew

*Figure 24 Sound Level Meter*

point takes place according to the formulations specified by ISO 7726. The humidity sensor works thanks to the hygro-capacity principle, works in a range of 0÷100% RH, has an accuracy of ±1,5 % RH @ 23°C and has a resolution of 0,1% RH.

*Figure 25 Thermohygrometer*

This system is supposed to detect severe conditions for workers during renovation work on the construction site.

The alert system is made possible thanks to the implementation of this system in the Resilience Dashboard (RD) which is a single management platform that enables the communication and orchestrated use of different technological tools that lead to a "Resilient Construction Site".

The Resilience Dashboard will be capitalized by the R-GUIDE: it will guarantee optimal operational continuity and efficiency in the construction incorporating the process innovations developed within WP2, whilst preserving and enhancing safety of workers, reducing wastes, and ensuring the surrounding environment is not affected by the site.

At date, the data logger selected and available to record data provided by the thermohygrometer is the IMC 5008. Connectors ACC/DSUBM-I2 have been used to test connection between data logger and thermohygrometer (Figure 26).

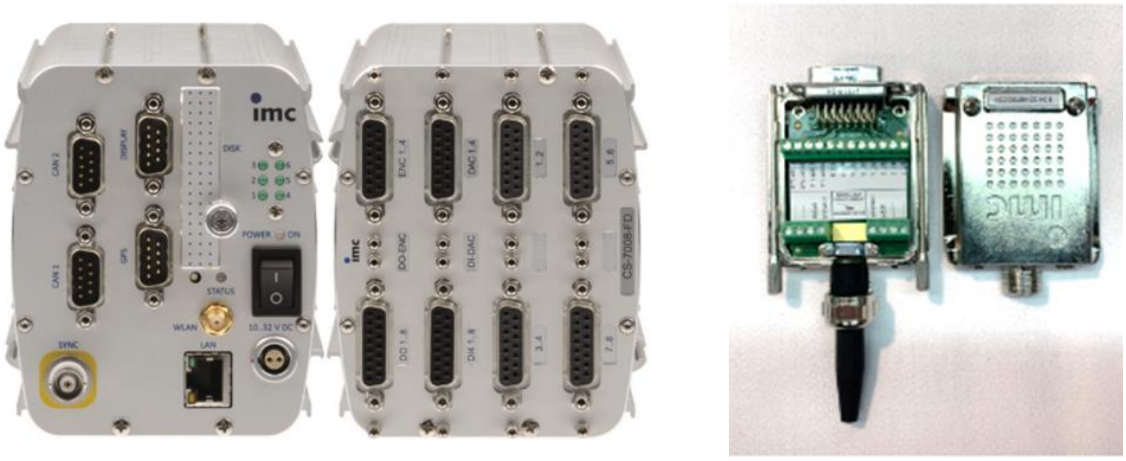


*Figure 26 On the left: IMC 5008 Data Logger. On the right: connectors ACC/DSUBM.*

IMC C-SERIES is a family of real-time acquisition systems with very compact dimensions, robust construction and fanless type: it is useful for a wide variety of measurement, analysis and control situations.

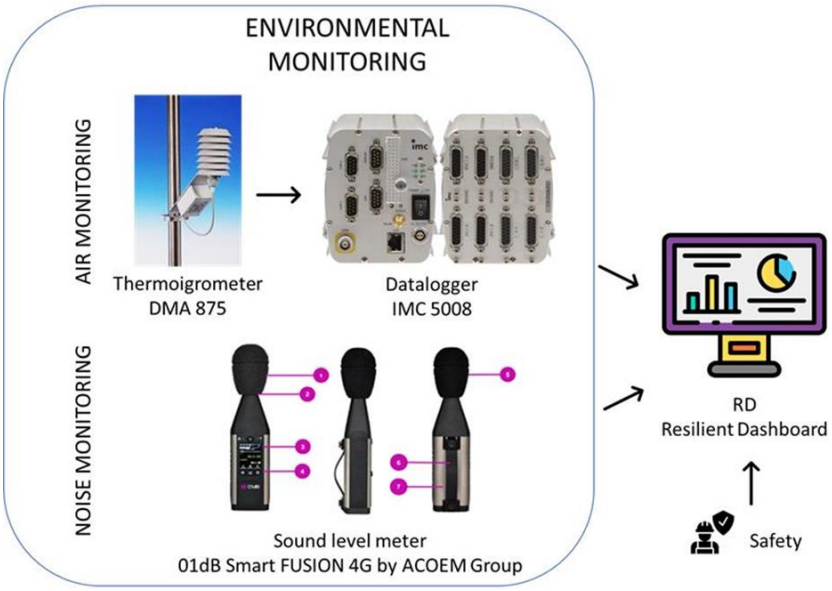



*Figure 27 Environmental Monitoring System*

The abovementioned sensors and data logger could be changed to match the onsite requirements.

### 3.1.18.1 Data Model

The data model of this architecture is based on the data processing of the sensors composing the system:

- Sound level meter FUSION 4G has no data logger; data will be managed by SW provided by ACOEM Group and exported in .csv format in order to be saved in the Resilience Dashboard and processed.

- Thermohygrometer DMA 875 will be managed by data logger IMC 5008 that allows to save data in .raw format: this data can be converted to .csv format to be saved in the Resilience Dashboard and processed.

The post-processing will include the storage of the data, comparing them with thresholds preliminary set according to the legislative framework (e.g. INAIL directives).

#### 3.1.18.2 Protocols

**Read access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Resilience Dashboard | Data transfer from Phonometer | HTTP / MQTT | .csv | 1 noise data every minute | Internal |
| Resilience Dashboard | Data transfer from IMC 5008 control unit | HTTP / MQTT | .raw / .csv | 1 humidity data every 15 min/ 1 temperature data every 15 minutes | Internal |

**Write access to components**

| Target component | Access description | Access Protocol | Data Format | Frequency of collection | Data sensitivity |
|---|---|---|---|---|---|
| Resilience Dashboard | Periodic reports of statistical analysis | HTTP | .pdf / JSON | Periodic | Internal |

## 3.2 Common Specifications

### 3.2.1 Common Building Features

Table 2 summarizes the common features describing all and every building of the demo sites. For each demo site (first column), we present the common features (second column) of all buildings and building elements, as well as their properties (third column). The common features of Table 2 will be used to create the data model of the digital twin of buildings.

*Table 2 Common Features of Buildings and Building Elements*

| Demo Site | Feature | Property | Comments |
|---|---|---|---|
| Zaragoza / Trento / Groningen | Building | building_id | Static information derived by the IFC files. It provides general information (e.g., name, location, floors, spaces) about each building of each pilot site. It will be updated once. |
| | | building_name | |
| | | building_location | |
| | | Latitude, Longitude | |
| | | building_type | |
| | building_id | zone_ids | It provides information related to the building zones (portal, stair, main door of the building), floors and spaces (dwellings, spaces common, etc) A zone_id can contain several floors. A space_id belongs to a single floor_id. A room_id belongs to a single space_id. |
| | | zone_names | |
| | zone_id | floor_ids | |
| | | floor_names | |
| | floor_id | space_ids | |
| | | space_names | |
| | space_id | room_ids | |
| | | room_names | |
| | Qto_Ifc (e.g., window_dimensions) | Width | Static information derived by the IFC files. It describes the dimensions (e.g., width, length) of each building element. It will be updated twice, once for the current status and once during the renovation. |
| | | Length | |
| | | Height | |
| | | Net Lateral Area | |
| | | Net volume | |
| | Pset_Classifications | IfcExportAs | Static information derived by the IFC files. There are unified classifications systems in the construction industry such us Uniclass. It provides information about the classification of each element of the building (e.g. beam, pilar, window, electrical device, etc). |
| | | IfcExportType | |
| | | Uniclass2023Code | |
| | | Uniclass2023Description | |

| | | | |
|---|---|---|---|
| | | | It will be updated twice, once for the current status and once during the renovation. |
| | Pset_Definition (e.g., wall_definition) | Description<br>IsExterior<br>System<br>Zone<br>Space<br>Level<br>Status | Static information derived by the IFC files.<br>It provides a definition (e.g., short description) of each building element.<br>It will be updated twice, once for the current status and once during the renovation. |
| | Pset_TypeCommon (By type of element, technical characteristics) | type<br>voltage<br>power factor<br>acoustic rating<br>fire rating<br>thermal transmittance | Static information derived by the IFC files.<br>It provides specific information of each element of the building, such us technical information (type, voltage, power factor, acoustic rating, fire rating, thermal transmittance, etc)<br>This information will be updated during the renovation phase. |
| | Pset_Tendering (e.g., window_tendering) | Reference<br>Manufacturer<br>ProductURL<br>Technical Information<br>Series<br>Model Label<br>Production Year<br>Country Available in<br>Measurement Unit | Static information derived by the IFC files.<br>It provides information about the tendering (e.g., manufacturer, production year) of each building element.<br>This information will be updated during the renovation phase. Some of the features (i.e., Country Available In, Price Database 1) will be periodically updated. |
| | Pset_Logistics (e.g., door_logistics) | Assembly Place<br>Wrapping material<br>Container material<br>Installation Time | Static information derived by the IFC files.<br>It summarizes logistics information (e.g., installation time) about each building element.<br>This information will be updated during the renovation phase. |
| | Pset_Management | Contact<br>Service Life Duration<br>Maintenance<br>Is Extended Warranty<br>Warranty Period | Static information derived by the IFC files.<br>It includes management-oriented information (e.g., contact, maintenance) for each building element.<br>This information will be updated during the renovation phase. |

| | | | |
|---|---|---|---|
| | Pset_Health & Safety | Free Space for Access | Static information derived by the IFC files. It contains information related to health and safety aspects of building elements. This information will be updated during the renovation phase. |
| | | Minimum Required Height | |
| | | Dependent of | |
| | | Accessible | |
| | | Risk Level | |
| | Pset_Sustainability & LCA | Unit | Static information derived by the IFC files. It contains information evaluating the environmental impact of building elements. This information will be updated during the renovation phase. |
| | | Life Cycle Phase | |
| | | Expected Service Life | |
| | | Climate Change Per Unit | |
| | | Water Consumption Per Unit | |
| | | Total Primary Energy Consumption Per Unit | |

### 3.2.2 Standard Ontologies

Main relevant Standard Ontologies were reviewed in order to select the most appropriate entities and attributes to be adopted within InCUBE. They are briefly presented in this section.

**Industry Foundation Classes (IFC)**

IFC[3] is a data schema designed to describe data related to the building, architecture, and construction industries. It is an open data schema definition that is platform-neutral and not governed by a single vendor or consortium of suppliers. It is an object-based data schema with a data model developed by buildingSMART to facilitate interoperability in the architecture, engineering and construction industry. This format is frequently used for cooperation in projects that involve building information modeling (BIM).

The construction, operation, and usage of a facility or installation may all be explained by the schema definition. Building materials, manufactured goods, mechanical and electrical systems, cost breakdowns, work schedules, and more may all be defined using IFC, in addition to more abstract structural and energy analysis models. The required IFC data can be handled in centralized or linked databases, imported/exported in files, or encoded in a variety of formats[4], including XML, JSON, and STEP, and sent over web services. Lastly, IFC data may be sent and received by hundreds of software programs used by many parties.

[3] https://www.buildingsmart.org/standards/bsi-standards/industry-foundation-classes/
[4] https://technical.buildingsmart.org/standards/ifc/ifc-formats/

**Building Topology Ontology**

Building design, planning, construction, and maintenance are multifaceted endeavors involving several stakeholders, each with distinct needs and interpretations of the shared information. Additionally, during the course of the building's entire life cycle, each stakeholder consumes, processes, and modifies information about it. A minimal, extendable ontology that describes anything in its context of a building is needed because it is frequently necessary to describe a sensor, product, or device in the context of the building in which it resides and because the building itself is also an interesting feature in the context of a smart city. A simple ontology for summarizing the fundamental topological concepts of a building is the Building Topology Ontology (BOT). BOT[5] is a minimal OWL DL ontology for specifying relationships between a building's sub-components. It was proposed as an expandable baseline for usage in conjunction with more domain-specific ontologies that adhere to the basic W3C guidelines of promoting reuse and limiting the complexity of the schema to what is absolutely essential.

A list of principles considered by the BOT ontology is the following:

1. Zones are areas with spatial 3D volumes, and include Buildings, Storeys, and Spaces.
2. Zones may contain other zones, Buildings may contain storeys, Storeys may contain spaces.
3. Zones may intersect or be adjacent to other zones.
4. There are building elements, which may have sub elements.
5. Zones may have elements, either contained, adjacent, or intersecting it.
6. Adjacent zones and/or elements share some interface.
7. Zones and Elements have a 3D Model.

Zones are described as a segment of the real world or a virtual environment that is intrinsically situated within it and has a three-dimensional spatial extent. Here are the definitions of the four subtypes of Zones:

Site: An area with one or more structures;

Building: A stand-alone constructed unit having a distinctive spatial structure;

Storey: A level part of a building;

Space: A constrained three-dimensional area that can be conceptually or physically delimited.

[5] https://w3c-lbd-cg.github.io/bot/

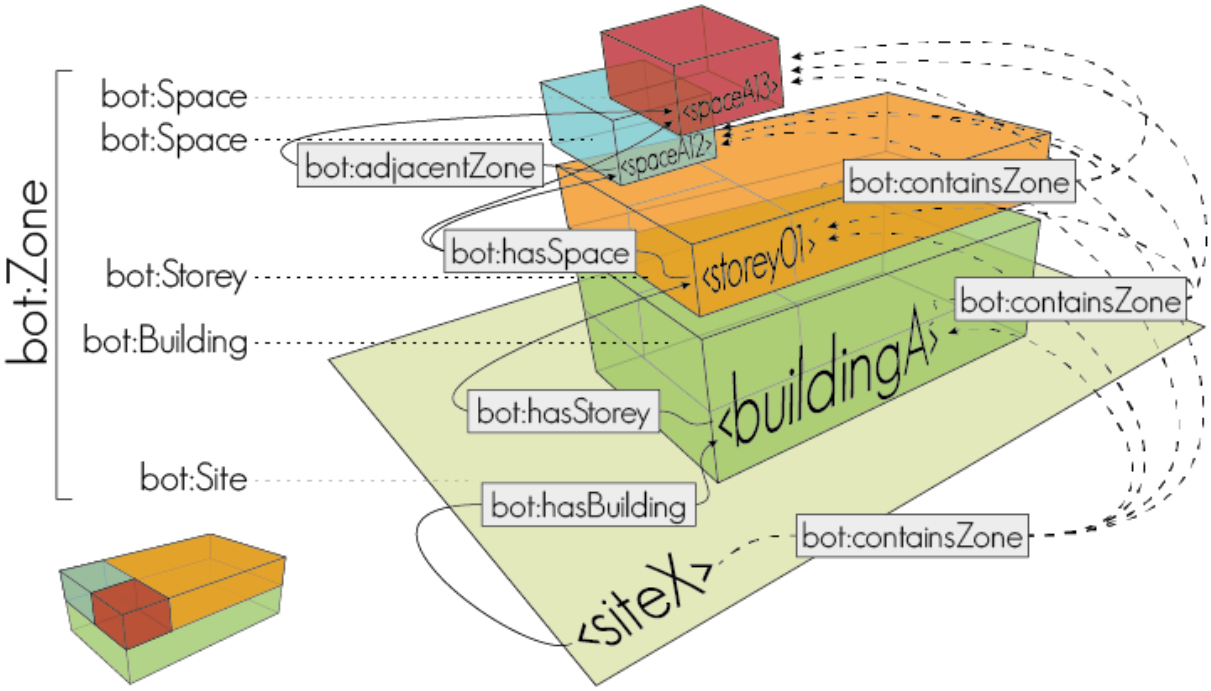



*Figure 28 Classes and relationships involved in Zones*

The Turtle version of the BOT ontology is available at http://www.w3id.org/bot/bot.ttl

**BPO: Building Product Ontology**

The Building Product Ontology defines concepts to describe building products in a schematic way. As is frequently the case with template-driven product descriptions, it offers ways to describe assembly structures and component interconnections as well as attribute properties to any component without limiting their types. It also includes terminology for unordered, two-dimensional lists to enable the expression of complex attributes.

The BPO's purview is limited to the product's schematic description; geometry and material compositions are not included. As a result, it may be used to describe both theoretical product components having geometric representations and ones without them, without any limitations. In order to fully leverage the benefits of the Semantic Web, particularly for online searches, the BPO is tightly matched with popular upper-level ontologies like SEAS[6] and schema.org. For classification purposes, BPO uses the buildingSMART Data Dictionary by referencing to the terms' bSDD GUID.

The description of assembly structures with BPO relies on three classes and two properties, presented in the table below.

[6] https://w3id.org/seas

*Table 3 Building Product Ontology*

| ***Classes*** | |
|---|---|
| **Components** | They serve as an abstract superclass of the following two classes, for ease of querying and enhanced reasoning |
| **Elements** | They are components that cannot be decomposed into further subcomponents and thereby pose as the smallest and most elemental components in this structure |
| **Assemblies** | They are components that consist of at least two sub-components (elements or other assemblies) |
| ***Properties*** | |
| **Consists of** | A property to indicate that an assembly consists of the related component(s). This property is defined to be transitive |
| **Is part of** | It defines of which assembly a certain component is a part of (inverse property of *consists of* ) |

In addition, a class for products is provided that inherits from the abstract component class to specify which components can be offered as a product of the manufacturer. This class is a subclass of the schema product[7], according to the alignment to well-known upper-level ontologies. The example below shows how these classes may be used to describe a product structure.

[7] https://schema.org/Product

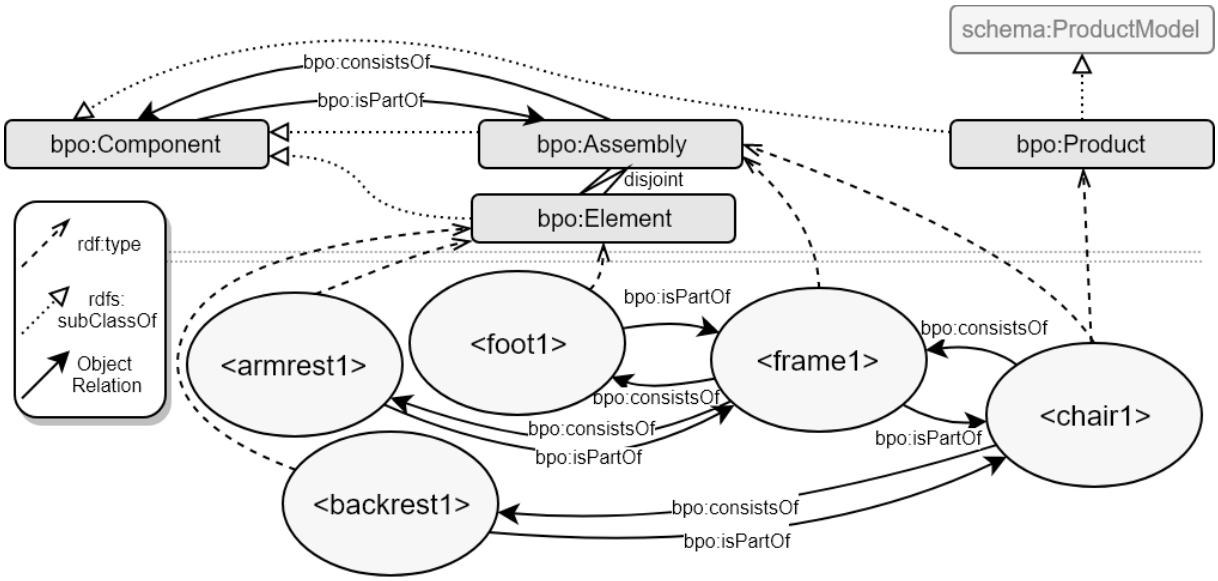



*Figure 29 Modelling product structures example using the BPO*

The component connection class is introduced to define the interconnections of a product's components. The component connection class is supplemented with five relations that allow direct interactions between linked components and reified relations.

· **has outgoing connection**, defining the relationship between an entity's output and its component connection

· **connects input of**, defining the relationship between a component connection and an input of an entity it connects

· **is connected to**, a chain property connecting two entities directed from the domain entity's output to the range entity's input. This property is reinforced by a chain axiom and may be derived by modeling the reified relation using the preceding two relations and a component connection

· **is connected from**, the inverse property of *is connected to* that connects two interconnected entities directed from the domain entity's input to the range entity's output

· **is connected with**, defining an undirected interconnection between two components and is thus defined as a symmetric property

Any component, and therefore any assembly, element, or product, can be specified by attributes with a single value or attributes with value ranges, lists, or intervals according to the BPO schema. Both types of attributes are linked to the schema.org property value and the SEAS Feature of Interest property to align this schema with current upper-level ontologies. Furthermore, datatype properties of schema.org, and unit definitions of the QUDT catalogue[8] are re-used. When no existing property met the exact interest of this ontology, new properties were defined and, if possible, aligned. The

[8] http://www.qudt.org/release2/qudt-catalog.html

*has attribute* property connects a component to its attributes and should be compatible with both the schema.org and SEAS schemes. Figure 29 provides an overview of attribute modeling options.

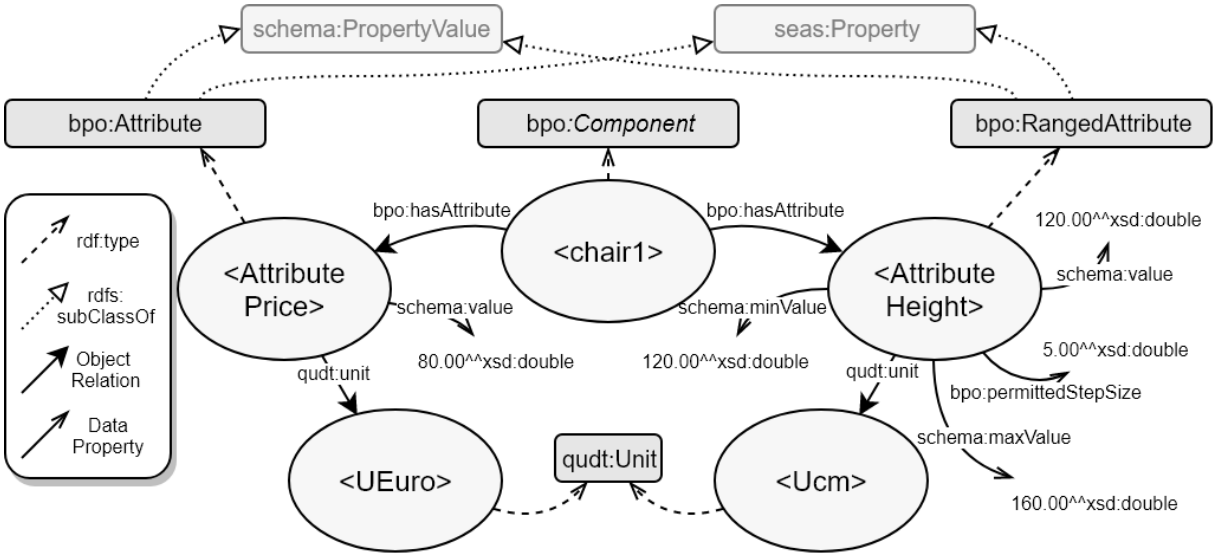



*Figure 30 Component attributes example using BPO*

The BPO can be combined with other ontologies. For example, BPO can be combined with an RDF-based geometry description and a non-RDF taxonomy. It shall be noted that the BPO is not complete but aims to serve as a compact ontology that can be re-used in various use cases and scenarios.

**Smart Appliances REFerence (SAREF) ontology**

The Smart Appliances REFerence (SAREF) ontology is a shared model of consensus that facilitates the matching of existing assets (standards/protocols/datamodels/etc.) in the smart appliances domain. An overview of the SAREF ontology is provided in Figure 31, where the main classes and their relationships are illustrated.

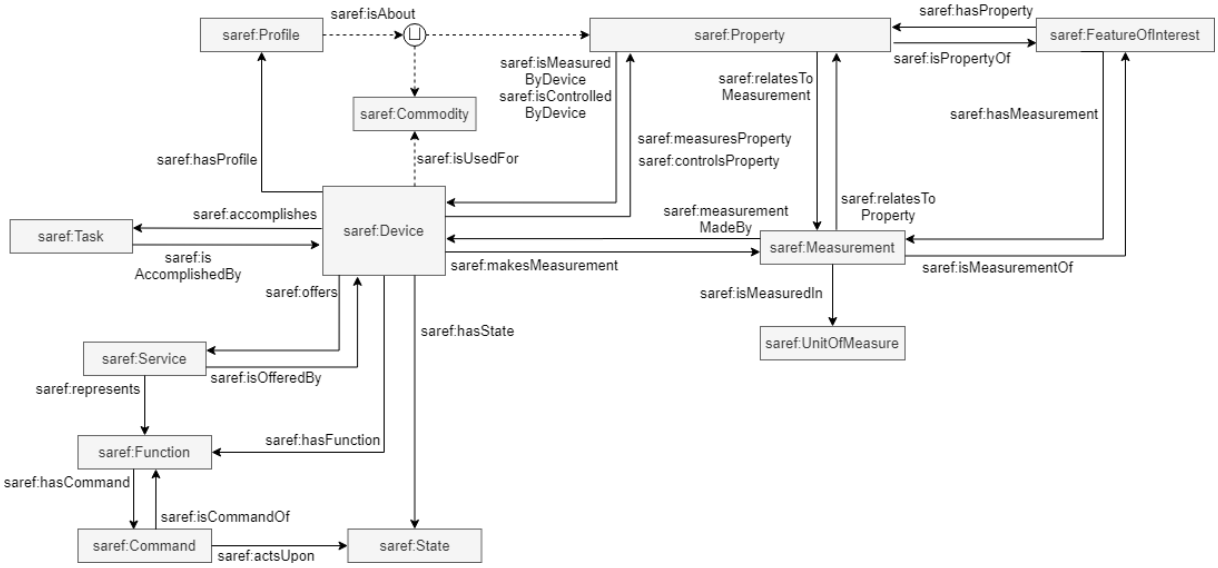



*Figure 31 SAREF ontology overview [https://saref.etsi.org/core/v3.1.1/]*

SAREF is based on the concept of Device (for example, a switch). Devices are physical objects meant to perform one or more functions in homes, public places, or offices. The SAREF ontology provides a set of basic functions that can be combined to create more complex functions in a single device. A switch, for example, provides an actuating function of type "switching on/off". Each function contains a set of instructions that may be selected as building blocks from a list. For instance, "switching on/off" is related with the commands "switch on", "switch off", and "toggle".

A Device provides a Service to a network, which is a representation of a Function that makes the function discoverable, registerable, and remotely controlled by other networked devices. One or more functions can be represented by a Service. A Service is provided by a device that needs for its function(s) to be discoverable, registerable, and remotely controlled by other devices.

A Device may have certain distinguishing characteristics, such as its model and manufacturer. SAREF is designed in a modular fashion so that any device may be constructed using pre-defined building components depending on the functions that the device performs. The diagram below depicts common types of devices. Appliances, actuators, sensors, and meters are examples of devices that can be represented, as illustrated in the image. It is worth mentioning that there are more types of devices, sensors, and actuators that can be defined to extend SAREF.

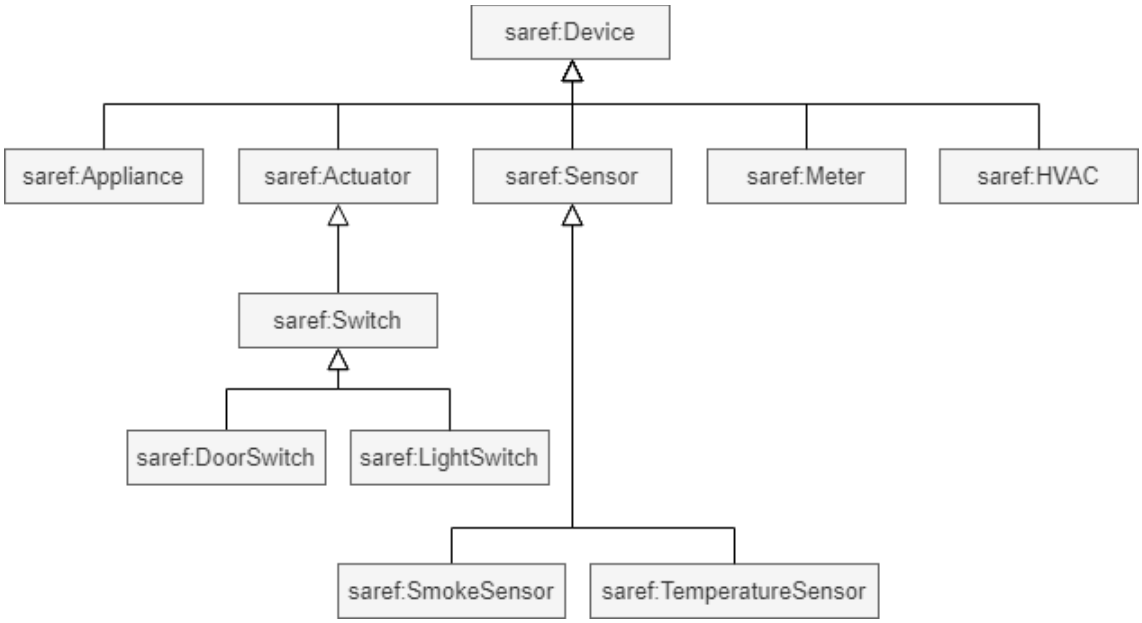



*Figure 32 SAREF Types of devices [https://saref.etsi.org/core/v3.1.1/]*

A function is represented in SAREF with the Function class and is defined as the functionality necessary to accomplish the task for which a device is designed. Examples of functions are *ActuatingFunction*, *SensingFunction*, *MeteringFunction* and *EventFunction*. The Function class and its properties are shown in Figure 33.

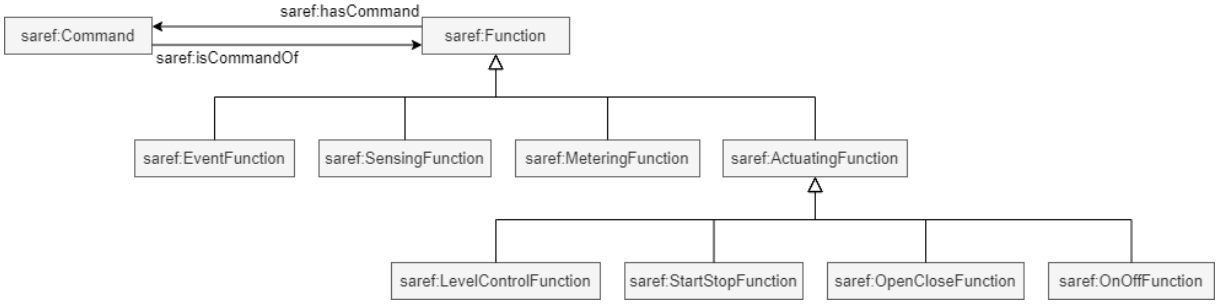



*Figure 33 SAREF Function class [https://saref.etsi.org/core/v3.1.1/]*

- *ActuatingFunction*: It allows to transmit data to actuators, such as level settings or binary switching (on/off).
- *SensingFunction* allows to transmit data from sensors, such as measurement values (e.g. humidity) or sensing data.
- *MeteringFunction* allows to get data from a meter, such as current meter reading or instantaneous demand.
- *EventFunction* allows to notify another device that a certain threshold value has been exceeded.

More details about the SAREF ontology can be found at the Official ETSI portal for SAREF[9].

**Brick schema**

Brick[10] is an open-source ontology-based metadata schema that captures the entities and relationships necessary for effective representations of buildings and their subsystems. It describes buildings in a machine-readable format to enable programmatic exploration of different operational, structural and functional facets of a building. Brick adheres to the following design principles: Completeness, Expressivity, Usability, Consistency and Extensibility. The four essential concepts of Brick are presented below.

**Entity**: an entity is an abstraction of any physical, logical or virtual item; the actual "things" in a building. There are physical, virtual, and logical entities. Physical entities are anything that has a physical presence in the world. Typical examples are spaces and floors, mechanical equipment such as lighting systems, and devices such as energy meters and EV chargers. Virtual entities are anything whose representation is based in software, such as the values of sensors, the energy consumption of an appliance, and actuation points that allow software to modify values (e.g. HVAC temperature setpoints). Logical entities are entities or sets of entities that are defined by a set of rules. Zones, and concepts such as class names and tags are included in this category.

**Tag**: a tag is an atomic fact or attribute of an entity. Indicative examples of tags are setpoints and sensors. Brick utilizes the concept of tags from Project Haystack to ensure ease of use and flexibility. Nevertheless, Brick uses more ways than just tags to identify the type of an entity.

**Class**: a class is a named category used for grouping entities. Classes are organised hierarchically, while entities are instances of one or multiple classes. Associated tags can be set for a class to provide useful annotations for discovery.

**Relationship**: a relationship defines the connection between two related entities. 'Encapsulation' (one entity is contained within another entity), 'Sequence' (one entity is applied before another entity in a process) and 'Instantiation' (one entity's type is given by another entity) are examples of relationships.

[9] https://saref.etsi.org/core/v3.1.1/
[10] https://brickschema.org/

# 4 Use Case on Cross-Domain Data Interoperability

| **ID:** AnyDemoSite, RenovationManagerUC<br>**Title: Management of resources, waste, assets and monitor of working conditions** |
|---|
| **Use Case short description and rationale** |
| The renovation project manager should have access to the digital twin of each renovated building. The Digital Twin will provide information related to the renovation site (e.g., material, waste, assets), building energy simulations, environmental and costing analysis KPIs (e.g., electricity consumption and the pricing scheme).<br><br>The goal of this use case is to demonstrate not only the benefits for the renovation management, but also to show the interoperability among the different InCUBE components, from pilot sites, monitoring tools to the logbook and dashboard. |
| **Main Actors (initiators of the Use Case), their benefits and roles** |
| **Renovation Project Manager.** She/He is responsible for the management of the renovation including site management, site safety and project completion by performing quality control and trade actions. |
| **Secondary Actors (participating in the Use Case), their roles** |
| **System Administrator.** She/He ensures the efficient and secure operation of the dedicated user interface (Kentyou UI). The system administrator will be responsible for setting and configuring the system, providing user support, monitoring system's performance and health, ensuring security standards, maintaining backup and creating disaster recovery plans. |
| **Business Goal** |
| Achieving a better coordination of all renovation aspects from planning and design to construction and completion can lead to the following business benefits: 1) project delivery on time, 2) staying on budget, 3) assuring quality and safety, 4) improving client and stakeholders' satisfaction, 5) better risk management and resource allocation, 6) complying with laws and regulations and 7) ensuring sustainability and minimizing environmental impact. |
| **Preconditions** |
| Any necessary privacy consents should be obtained. Moreover, the necessary infrastructure should be set in place, such as IoT devices, edge devices, gateways, cloud servers, Kubernetes instances, etc. |
| **Detailed Scenario and main supporting blocks** |
| **Detailed Scenario** |

The renovation project manager will be able to login to the user interface of the Digital Twin provided by Kentyou. After being logged in, s/he will be able to:

1. Explore on a map the different renovation site.
2. For each site, s/he will be able to check its renovation status (e.g., number of materials, status of renovation of different components, etc), the building energy simulations and the environmental and costing analysis KPIs (e.g., electricity consumption and the pricing scheme)
3. For each site, s/he will be able to check statistical data and key operational characteristics (e.g., energy performance).

**Supporting blocks**

The main supporting technological blocks of this Use Case are:

- Digital Building Logbook: Collects and stores information for each building. This information can be static (e.g., building ID, building characteristics) or real-time (e.g., environmental conditions per site).
- Digital Twin: Aggregates data from cross-border pilot sites, analyses it and displays data and their insights using Kentyou UI.
- BIM2BEM LCA: Estimates key operational characteristics (e.g., energy performance), performs environmental and cost analysis and shares results with the Digital Twin.
- R-GUIDE: Receives IoT data from the on-site installation and automation systems and shares it with the Digital Twin.

**Postconditions**

After the realization of this use case, we expect access to the digital twin of each building from the different renovation sites via a dedicated user interface (Kentyou UI).

**User Requirements**

Below we summarize the user requirements (UR) of this use case.

UR_01: Allow the user to access the renovation information per building.
UR_02: Allow the user to see the available material, waste and assets per building.
UR_03: Allow the user to check the results on the environmental and cost analysis, as well as on the building energy simulations.

**Derived System Requirements**

Below we summarize the function (FR) and non-functional (NFR) user requirements of this use case.

FR_01: The system shall provide statistical information regarding the number of materials, wastes, assets per building.
FR_02: The system shall provide access to real-time IoT measurements from the pilot sites.
FR_03: The system shall provide building energy simulated results and environmental and costing analysis KPIs.

NFR_01: The system shall ensure robustness from electric supply and internet connectivity.
NFR_02: The system shall ensure data privacy and data storage security.
NFR_03: The system shall ensure easy access to the user interface.

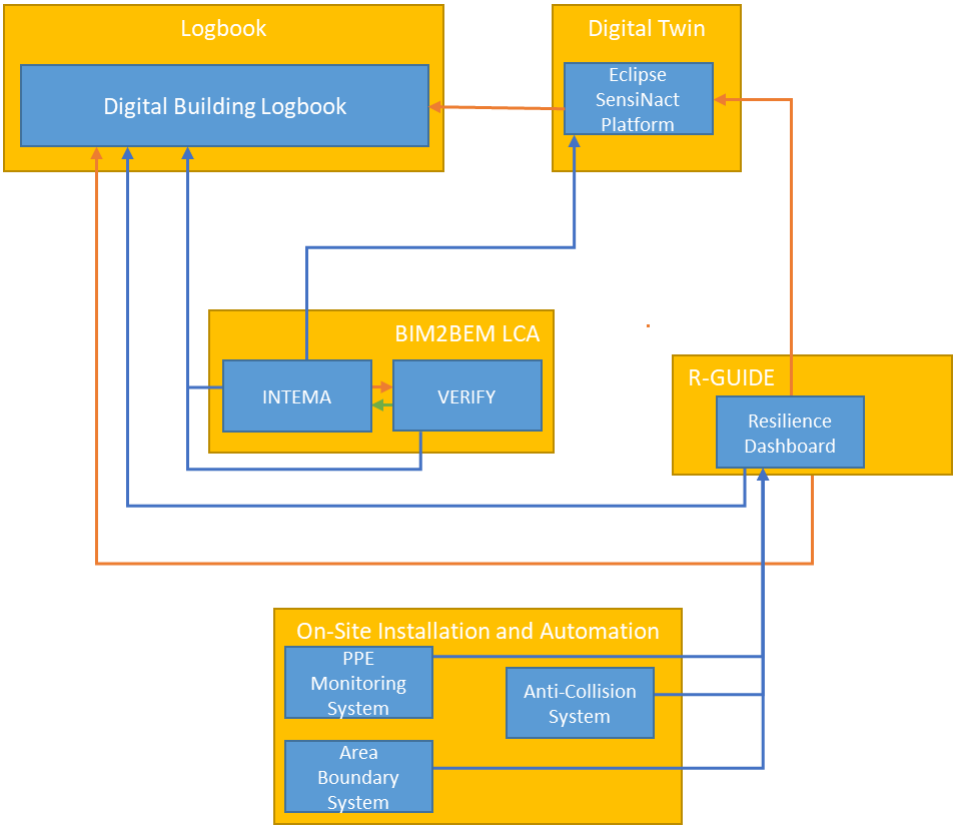
**Constraints and Barriers (technical, regulatory, …)**

The following constraints and barriers should be considered for the successful realization of this use case:

- Privacy and security concerns - threats to privacy or security should be addressed.
- Hardware requirements - necessary resources (e.g., IoT devices) should be obtained and deployed.
- Unstable power or internet supply on the renovation site.

**Relevant InCUBE Components involved**

The Figure below shows the minimum desired InCUBE components to be put in place for the realization of this UC.




**How will this use case be demonstrated?**

This use case will be demonstrated using the Digital Twin of Kentyou. The Digital Twin will be visualized with the help of Kentyou UI.

Kentyou UI will provide a digital twin of the different buildings showing the cross-border integration of all relevant information, ensuring access only to authenticated users.

# 5 Conclusion

This deliverable has presented an initial perspective on the architecture, data model, available protocols and designed APIs for each InCUBE component. Moreover, it is detailing common specifications, such as features of the digital twin of buildings and ontologies and describes a use case that will demonstrate the capabilities of the InCUBE's interoperability layer.

The document emphasizes the importance of fostering collaboration among different assets, to formulate and adopt interoperable specifications. By doing so, the industry can overcome the challenges associated with siloed ecosystems, promoting seamless integration and data interchangeability.

The exploration of interoperable specifications for APIs and data models underscores the critical need for standardized approaches in the realm of data exchange. Standardized APIs and data models facilitate a more agile and adaptive environment, enabling the focus on creating value-added features rather than grappling with compatibility issues. This, in turn, accelerates time-to-market and reduces overall development costs.

This work will be taken-up and continued through the work of WP3 (as a basis for interoperability evaluation), but also through the work of WP9 on the exploitation strategy. The outcome of this future work will be reported in the upcoming deliverable of D3.2.

Overall, the pursuit of interoperable specifications for APIs and data models is a cornerstone in building the InCUBE's interoperability layer.

# 6 Annex: DBL Data Models

Below we present the JSON data models (v0.1) of the Digital Building Logbook. The final version will be reported in the next deliverable (D3.2).


```
{
  "building": {
    "id": "b662cf63-e5d6-410d-a015-cf4702390d5a",
    "name": "main building",
    "siteName": "Trento",
    "location": {
      "address": "street address",
      "latitude": 41.648,
      "longitude": -0.89
    },
    "owner": "user1",
    "type": "Dwelling",
    "storeys": [
      {
        "name": "",
        "spaces": [
          {
            "name": "",
            "zones": [
              {
                "name": ""
              }
            ]
          }
        ]
      }
    ]
  }
}
```


Building

```
{
  "device": {
    "id": "32b09070-77eb-4970-8a0a-c9b3ee2024ef",
    "description": "",
    "type": "HVAC",
    "manufacturer": "manufacturer name",
    "model": "model number",
    "owner": "user1",
    "isOperational": true,
    "isUsedFor": "Comfort",
    "category": "<Physical/Virtual>",
    "attachedSensors": [
      "sensor_id"
    ],
    "location": {
      "buildingId": "building1",
      "spaceId": "space1",
      "zoneId": "zone1"
    },
    "hasFunction": [
      {
        "type": "ActuatingFunction",
        "hasCommand": [
          {
            "type": "OnCommand"
          },
          {
            "type": "OffCommand"
          },
          {
            "type": "SetLevelCommand"
          }
        ]
      },
      {
        "type": "SensingFunction",
        "measuresProperty": [
          {
            "type": "Temperature",
            "unitOfMeasure": "DegreeCelsius"
```


Device

```
{
  "sensor": {
    "id": "05d1d4a0-ed83-46ab-b460-39eb4293197f",
    "description": "",
    "type": "Environmental",
    "manufacturer": "manufacturer name",
    "model": "model number",
    "owner": "user1",
    "isOperational": true,
    "isUsedFor": "Metering fuction",
    "category": "<Physical/Virtual>",
    "location": {
      "buildingId": "building1",
      "spaceId": "space1",
      "zoneId": "zone1"
    },
    "sensingFunction": [
      {
        "property": "Temperature",
        "type": "numeric",
        "unitOfMeasure": "Celsius"
      },
      {
        "property": "Humidity",
        "type": "numeric"
```


Sensor

```
{
  "product": {
    "id": "d76221aa-3d99-4cc2-a3b2-e762d1ccf8bf",
    "description": "",
    "manufacturer": "manufacturer name",
    "model": "model number",
    "category": "<Physical/Virtual>",
    "url": "URL of the product",
    "releaseDate": "year-month-date",
    "countryOfOrigin": "country name",
    "countryOfAssembly": "country name",
    "dimensions": {
      "width": 1.5,
      "height": 2,
      "depth": 2.5,
      "weight": 35
    },
    "hasEnergyConsumptionDetails": {
      "hasEnergyEfficiencyCategory": "<yes/no>",
      "energyEfficiencyScaleMin": "A++",
      "energyEfficiencyScaleMax": "A+++"
    },
    "hasMeasurement": {},
    "hasComponent": [
      {
        "name": "",
        "property": [
          {
            "name": "",
            "value": ""
          },
          {
            "name": "",
            "value": "",
            "unitOfMeasure": ""
          }
```


Product

```
{
  "activity": {
    "id": "c0a638dc-515d-441d-bbbf-368cb0955fa9",
    "type": "Installation",
    "description": "",
    "location": {
      "buildingId": "building1"
    },
    "durationHours": 24,
    "priorityLevel": 1,
    "linkedProducts": [
      "product_id"
    ],
    "constraints": [],
    "personnelRequirement": [
      {}
    ],
    "equipmentRequirement": [
      {}
    ],
    "materialRequirement": [
```


Activity


```
{
  "renovationScenario": {
    "id": "",
    "description": "",
    "buildingId": "",
    "renovationActivities": [
      {
        "id": "",
        "description": "",
        "type": ""
      },
      {
        "id": "",
        "description": ""
```


RenovationScenario

```
{
  "user": {
    "id": "a791942a-38e1-4d39-9b83-5723e04adc8c",
    "account": {
      "mail": "example@incube.com",
      "username": "johndoe",
      "passwd": "123456"
    },
    "generic": {
      "fullname": "John Doe",
      "role": "Worker",
      "trade": "Electrical Technician"
    },
    "associatedBuildings": [
      "building_id1",
      "building_id2"
    ],
    "preferences": {
      "propertyName": "Value"
    },
    "skills": {
      "experienceLevel": {
        "id": "1",
        "value": "Novice"
      },
      "targetExperienceLevel": {
        "id": "2",
        "value": "Experienced"
      },
      "typical": [
        {
          "skillID": "1",
          "description": "Equipment installer",
          "level": 2
        },
        {
          "skillID": "2",
          "description": "Malfunctions handling",
          "level": 1
        }
      ]
    }
  }
}
```

User


```
{
  "event": {
    "id": "4fc93a0a-fe55-4d10-a21f-facf54561e49",
    "type": "Measurement",
    "description": "A measurement event",
    "dateTime": "2023-10-10T12:00:10.000",
    "sourceId": "device_id",
    "buildingName": "",
    "spaceName": "",
    "content": [
      {
        "description": "Temperature measurement",
        "property": "Temperature",
        "value": 23.1,
        "unitOfMeasure": "DegreeCelsius",
        "refDateTime": "2023-10-10T12:00:00.000",
        "metadata": {}
      },
      {
        "description": "Relative humidity measurement",
        "property": "Humidity",
        "value": 35,
        "unitOfMeasure": "Percent",
        "refDateTime": "2023-10-10T12:00:00.000",
        "metadata": {}
      }
    ]
  }
}
```


Event (Measurement)

```
{
  "event": {
    "id": "4fc93a0a-fe55-4d10-a21f-facf54561e49",
    "type": "ForecastPowerGeneration",
    "description": "Generation forecasting event",
    "dateTime": "2023-10-10T14:00:00.000",
    "sourceId": "pv1",
    "buildingName": "",
    "spaceName": "",
    "content": [
      {
        "description": "Generation forecasting",
        "property": "Power",
        "value": 1050,
        "unitOfMeasure": "Watt",
        "refDateTime": "2023-10-10T14:15:00.000",
        "metadata": {}
      },
      {
        "description": "Generation forecasting",
        "property": "Power",
        "value": 1120,
        "unitOfMeasure": "Watt",
        "refDateTime": "2023-10-10T14:30:00.000",
        "metadata": {}
      },
      {
        "description": "Generation forecasting",
        "property": "Power",
        "value": 900,
        "unitOfMeasure": "Watt",
        "refDateTime": "2023-10-10T14:45:00.000",
        "metadata": {}
      },
      {
        "description": "Generation forecasting",
        "property": "Power",
        "value": 860,
        "unitOfMeasure": "Watt",
        "refDateTime": "2023-10-10T15:00:00.000",
        "metadata": {}
      }
    ]
  }
}
```

Event (Forecast)

```
{
  "event": {
    "id": "4fc93a0a-fe55-4d10-a21f-facf54561e49",
    "type": "Notification",
    "description": "A contact message event",
    "dateTime": "2023-10-10T12:00:00.000",
    "sourceId": "Component name",
    "buildingName": "",
    "spaceName": "",
    "content": [
      {
        "category": "Notification",
        "priority": "<Normal/High>",
        "topic": "Message title",
        "text": "Message body",
        "metadata": {
          "destination": "Everyone"
        }
      }
    ]
  }
}
```

Event (Notification)