FISMEP API SPECIFICATION DESCRIPTION

June 17, 2019

Contents

INTRODUCTION

The FISMEP project provides, via the FISMEP API, open access to live and historical data from real grids and buildings accessed by the FISMEP Smart Energy use cases. Physical infrastructures, the FISMEP Trial Sites, in different European countries are used to implement the use cases. These use cases were chosen in areas expected to gain the most from the use of ICT.

The aim of the present work is to present a Smart Energy Model (SEM) for the FISMEP Smart Energy Platform (SEP). This model is meant to integrate the existing different data models developed for the trial sites and also to be able to cope with other data models, being proprietary or standardized models, in order to make them compatible with the Platform.

In the following, we will provide an introduction to the main concepts behind the design of data models and data ontologies, introducing the main concepts and the specific terminology. we will present the modelling approach used for the design of the SEM, from the identification of the semantics and the conceptual model to the definition of the meta-model from which the proposed unified data model has been derived. This clause describes technical principle behind the context information management framework supported by NGSI-LD.

The work presented here applies the SEM concept to the data accessible via the FISMEP API,mapping the heterogeneous data models used by the different trials to the unified SEM data model.

0.0.1 Ontology Engineering and Data Modelling

A computer ontology is said to be an "agreement about a shared, formal, explicit and partial account of a conceptualisation". In addition an ontology contains the vocabulary (terms or labels) and the definition of the concepts and their relationships for a given domain. In many cases, the instances of the application (domain) are included in the ontology as well as domain rules that are implied by the intended meanings of the concepts. Domain rules restrict the semantics of concepts and conceptual relationships in a specific conceptualization of a particular application domain. These rules must be satisfied by all applications that want to use an ontology.

A data model represents the structure and integrity of the data elements of the specific applications by which it will be used. Therefore, the conceptualization and the vocabulary of a data model are not intended a priory to be shared by other applications.

A meta-model is an explicit model of the constructs and rules needed to build specific models within a domain of interest. This characterizes a valid meta-model as an ontology,

since such constructs and rules represent entities in a domain and their relationships, i.e., a set of building blocks used to build domain models. In other words, a meta-model is an ontology used by modellers. For example, when software developers use UML to construct models of software systems, they actually use an ontology implemented in it. This ontology defines concepts such as objects, classes, and relations. However, not all ontologies are modelled explicitly as meta-models.

Modelling ontologies for a wide usage in an open environment, such as the Semantic Web, obviously is a challenging task. Providing more ontology rules, which are important for effective and meaningful interpretation between applications, may limit the generality of an ontology. However, light ontologies, i.e. holding none or few domain rules, are not very effective for communication between autonomous software agents.

Since an ontology is a model of a domain describing objects that inhabit it, all three types of data models can be thought of as ontologies. They range from the most expressive one that describes business concepts and processes (the conceptual model) to less expressive and progressively moving from describing business semantics to describing physical structures of the data as it is stored in the databases (the logical and physical data model). The physical model can be thought of as an ontology of a particular database.

The main problem to be addressed is interoperability between different models and current solutions mainly rely on the specification of a common data exchange format.The modelling approach adopted for our work is based SAREF (Smart Appliance REFerence ontology), it is a shared model of consensus that facilitates the matching of existing assets (standards/protocols/datamodels/etc.) in the smart appliances domain. By using SAREF we could provide building blocks that allow separation and recombination of different parts of the ontology depending on specific needs.

0.0.2 NGSI-LD Information Model

New applications in Smart Energy domain collates and interprets data from fast-changing and geographically dispersed sources. It also actuates devices and effects many real-world results. To produce,interpret and exchange data,these applications need to define unambiguously the data used, and to share those definitions with other applications.The data relevant to a service, and the definitions that describe its format and meaning,can be called the context of the service.For example,location,time,temperature,and application specific information must have common definitions and be understood by all the applications which manipulate it.

The ETSI Industry Specification Group for cross-cutting Context Information Management is addressing this problem. This section explains the main concepts behind a new data exchange protocol called NGSI-LD which aims to make it easier to find and exchange information with open databases, mobile Apps and IoT platforms. It fills the gap between brief press releases and detailed specification documents for NGSI-LD API and related use cases.

Context information exchange using NGSI-LD has major advantages. Firstly, within the NGSI-LD framework, applications can flexibly discover and query relevant information. The data discovery is dynamic,and the built-in query patterns support the most common questions that are practical in unbounded federated information systems. Secondly,NGSI-LD helps to precisely communicate the nature of the context information for a given service, such as its period of validity, its geographic constraints, and other semantically important information, by enabling direct inclusion of pointers to the relevant parameters and definitions. To ensure interoperability, the NGSI-LD API defines the meaning of the most commonly needed terms and provides the tools to create domain-specific extensions to model any other type of information.Thirdly, NGSI-LD provides a scalable solution to connect,publish and federate diverse data sources using a developer-friendly interface for data sharing and usage.

More specifically, the data in NGSI-LD are structured like a data graph model and linked with each other.The relationships between entities are easily handled in NGSI-LD in a similar way to graph databases.It also provides the update, query and subscription support needed to allow applications to automatically access data from various data sources.

NGSI-LD builds upon previous work, including OMA and FIWARE. It is intended to complement and work alongside IoT system specifications, while providing a path to integrate data from open linked data and other information sources.In NGSI-LD, information not relevant to the application layer of that particular service, for example details of the IoT or network technologies used to connect entities, or to manage IoT devices and gateways, is not explicitly considered. Such aspects are covered elsewhere, for example in oneM2M, where ETSI is a founding member along side other major international standards bodies. OneM2M has defined a common platform for IoT, which can interoperate with a wide range of networks and systems

The NGSI-LD information model makes it easy to create models of real-world entities,relationships and properties; moreover,the information model is expressive enough to connect and federate other existing information models, using JSON-LD.It is also compatible with RDF so that triplestores and application logic found e.g. in SPARQL or DataCube software can be applied.

The NGSI-LD Information Model is defined at two levels, consisting of the Core Meta Model, the Cross-domain Ontology, and the open-ended domain-specific models see Figure 1.

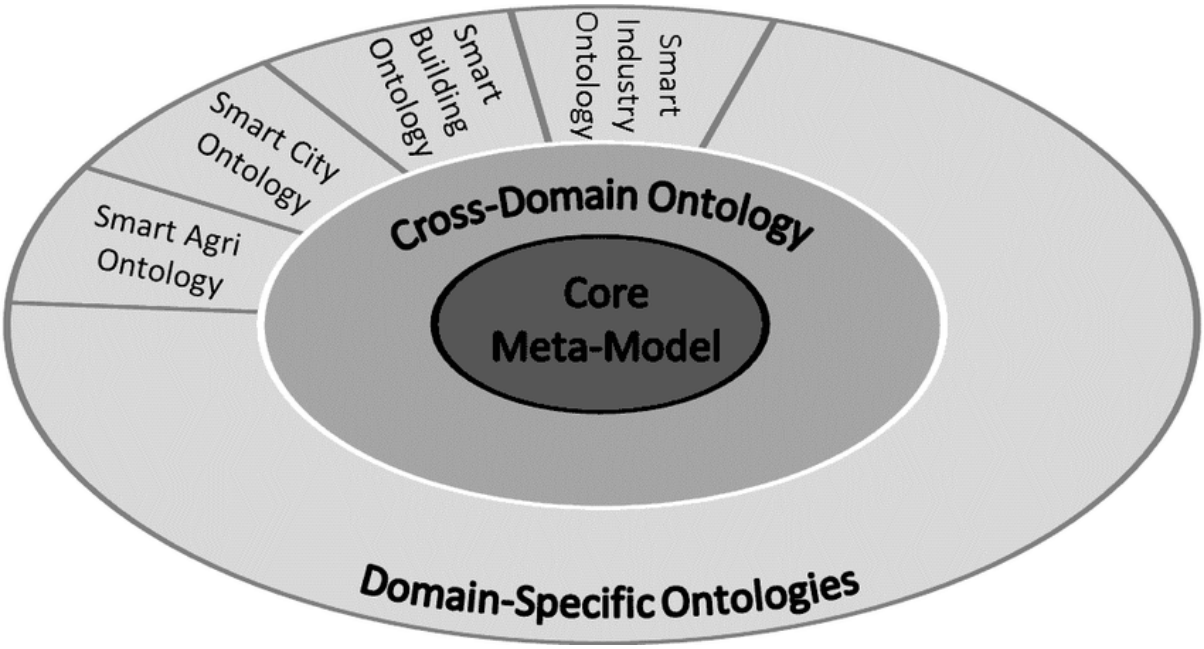

Figure 1: NGSI-LD Information Model

In this model, the NGSI-LD Core Meta Model, in the centre of the figure, represents Entities, their Relationships, and their Properties with values. It contains the core terms needed to uniquely represent the key concepts of the NGSI-LD Information model as well as the terms that define the API-related Data Types. These are encoded using JSON-LD, which provides the advantage of being familiar and accessible to developers.For example, an Entity, representing a device or other data source, is encoded using a JSON-LD object.

The NGSI-LD Cross Domain models provide commonly used constructs such as time and geographical location. These are generally applicable to many domains so standardizing their representation provides valuable cross-domain interoperability.The main features of the NGSI-LD Cross Domain model have been added to distinguish between: (a) the location of an entity; (b) the entities observation space which is the geographic location that is being observed by the entity;and (c) the operation space which is the geographic location in which an entity is active.

Domain-Specific Ontologies for entities (real world devices, databases, or other information sources) can be created by extending the Cross Domain Ontologies and the Core MetaModel, with specialized terms drawn from other ontologies. For instance, the SAREF On-

tology can be mapped to the NGSI-LD Information Model, so that smart home applications will benefit from this Context Information Management API specification.

0.0.3 NGSI-LD Architectural Model

In the ISG CIM framework, context consumers use the API to access information provided by context producers, often through Context Brokers of several types.A Context Broker mediates exchanges between context consumers and producers.The architectures shown in the following are prototypical ones; real systems may instantiate a combination of these architectures or form a more complex one, e.g. consisting of a mesh of Context Brokers.

A Context Producer(e.g.a sensor gateway, which sends streams of updates or timed batches of updates) simply delivers context information to context brokers, which arrange its storage and(later) delivery to context consumers.

A Context Source is the name given to more complex systems which can collect and store a wide range of information, e.g. from devices or databases, register the availability of its information with a broker's Context Registry, and deliver the requested context information in response to a redirected query from a Broker or directly to context consumers.

Figure 2 shows a distributed architecture. The underlying idea here is that all information is stored by the Context Sources. Context Sources implement the query and subscription part of the NGSI-LD API as a Context Broker does. They register themselves with the Context Registry, providing information about what context information they can provide, but not the context information itself, e.g. a certain Context Source registers that it can provide the indoor temperature for Building A and Building B or that it can provide the speed of cars in a geographic region covering the centre of a city.

Context Consumers can query or subscribe to the Distribution Broker. On each request, the Distribution Broker discovers or does a discovery subscription to the Registry for relevant Context Sources, i.e. those that may provide context information relevant to the respective request from the Context Consumer. The Distribution Broker then queries or subscribes to each relevant Context Source, if possible it aggregates the context information retrieved from the Context Sources and provides them to the Context Consumer. In this mode of operation, it is not visible to the Context Consumer, whether the Broker is a Central Broker or a Distribution Broker. Alternatively, the architecture allows that Context Consumers can discover Context Sources through the Registry themselves and then directly request from Context Sources.

We can depict a "smart" service as an interconnection of context providing services and

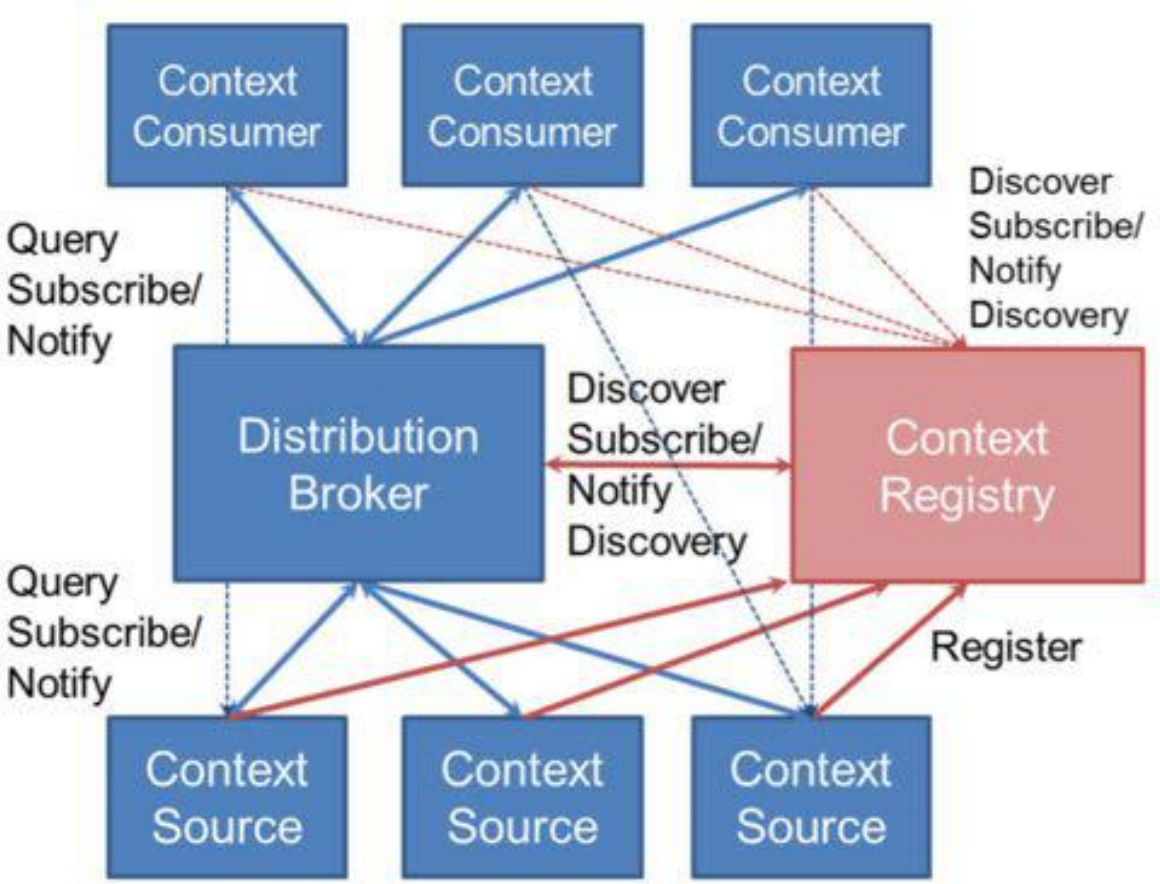

Figure 2: Distributed Architecture

context consuming applications. These work together to ensure that each application has the information it requires to deliver knowledge and insight, and to exercise control. We can regard the context of an application as being all the relevant aspects of its operating environment that are required for it to work as intended. Each application needs a different mix of data (context) from one or more sources. A context producer may be a sensor, a gauge, a database an open data repository, etc.

In figure 3, context producers and consumers are connected by a cross-connecting Context Information Management System. The NGSI-LD API is used by data consumers to query for and receive updates on context information.

In the ETSI ISG CIM framework, context information is considered to be any relevant information about entities, their properties (temperature, location, or any other such parameter), and their relationships with other entities. Entities may be representations of real-word objects but may also be more abstract notions such as a legal entity, corporation, nation state,

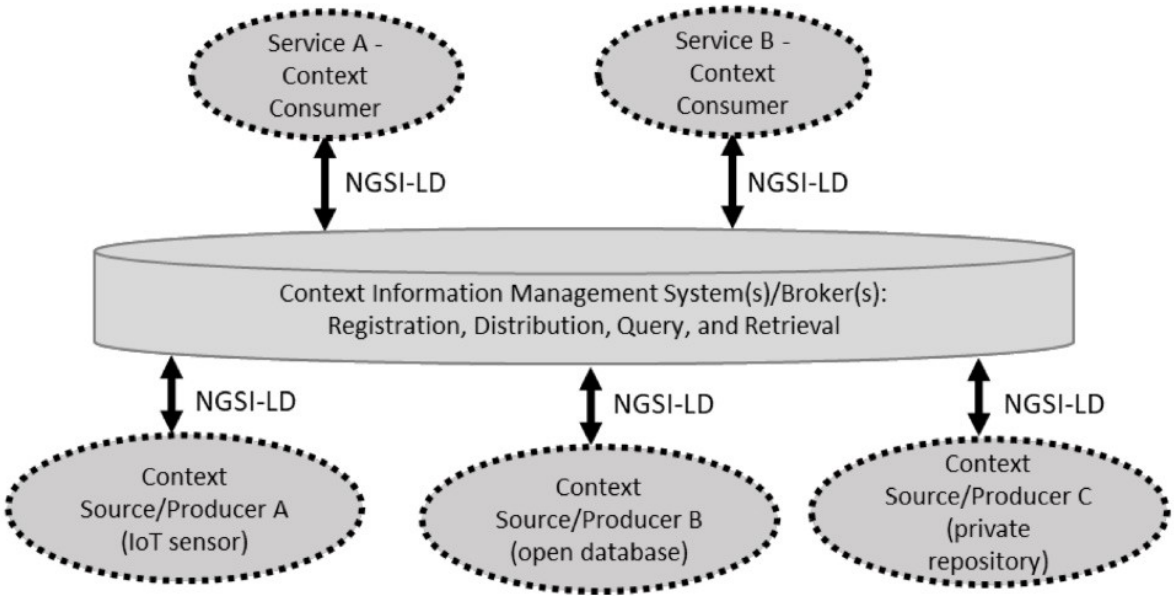

Figure 3: Context Information Connection

or groups of entities.Having mind the above discussion, in the next sections the FISMEP API is documented.

API SPECIFICATION DESCRIPTION

FISMEP API follows a common convention with regards to responses, using a standard, lightweight envelop to wrap the FISMEP-API compatible responses. Specifically, this standard wrapper includes a Metadata object documented in Table 1 and Table 2 and is depicted as a UML Class diagram in Fugure 4.

Figure 4: FISMEP API Interface

Name	Type	Description
metaData	MetaData	The metadata of API see Table 2

Table 1: Attribute of API interface

Name	Type	Description
API-Version	String	The version of the API specification is written
Field-Test	String	name of response Field test

Table 2: Attribute of API metadata object

Every class implemented as a response of a FISMEP API resource inherits this standards interface.

Authentication

Authenticate a User

Building Information

This set of APIs provide access to information related to smart buildings presentation.

Get List of Buildings

Endpoint URL: [/version/Field-Test/Buildings]

List the buildings known to be selected a trial infrastructure. 'Sweden' is the only trial offering this service.

Name	Type	Example	Description
Version	string	V0-.1	verson of API to use
Field-Test	String	Sweden	The responsible field test to use for invoking the respective service

Table 3: Parameters

The class diagram of the response (instance of Buildings) of the service follows in Figure 5.

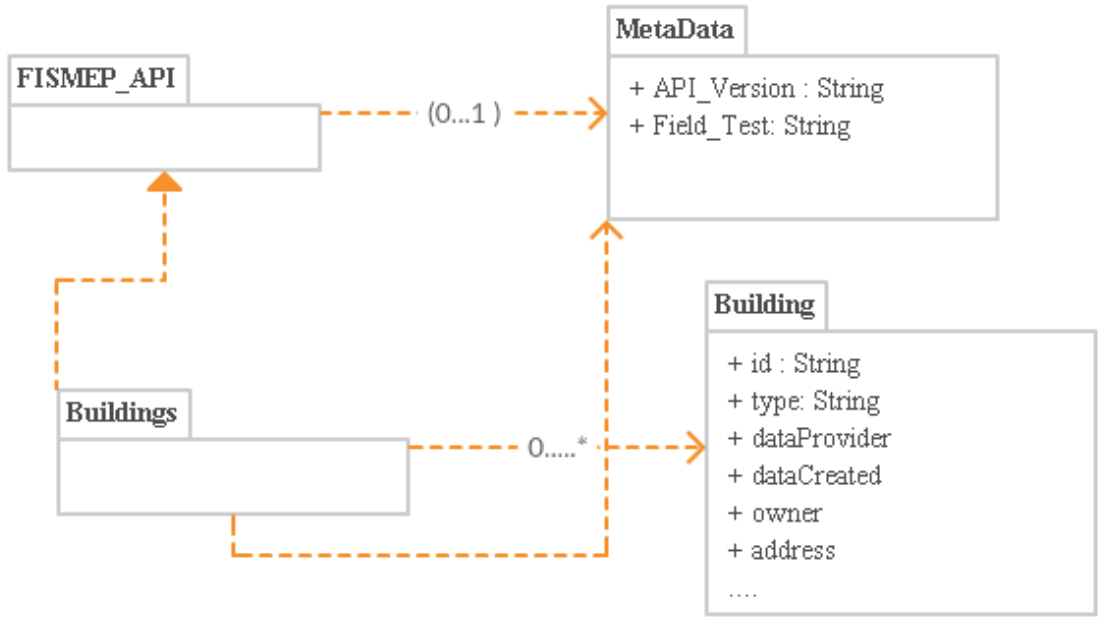

Figure 5: Class Diagram of Related to a List of Buildings Request

A list attributes which can be retrieved from context registry of the buildings, is like:

Name	Type	Description
id	String	The id of the Building
type	String	Building
dataProvider	relationship	URL to information of data provider
owner	String	optional
location	Geolocation	Geo location of the Building
description	String	more information about the Building
refRelatedDeviceOperation	relationship	A list of references to an entity of type Device

Table 4: Attributes of Building


```
[
 {
 "id": "57b912ab-eb47-4cd5-bc9d-73abece1f1b3",
 "type": "Building",
 "properties": ["owner","location", "description"],
 "relationships": ["dataProvider", refRelatedDeviceOperation]
},
{
 "id": "57b912ab-eb47-4cd5-bc9d-73abece1f1b4",
 "type": "Building",
 "properties": ["owner","location", "description"],
 "relationships": ["dataProvider", refRelatedDeviceOperation]
   }
]
```


Get Building Description

Endpoint URL: [/version/Field-Test/Buildings/Description/id]

It provides all list of information for specific building based on identified id. 'Sweden' is the only field test offering this service.

The data model of the answer is the same as the one documented in Figure 5 and by giving id of specific building to can retrieve all information about it.

Name	Type	Example	Description
Version	string	V0-.1	verson of API to use
Field-Test	String	Sweden	The responsible field test
id	String	"57b912ab-eb47-4cd5-bc9d-73abece1f1b3"	id of building

Table 5: Parameters


```
{
  "id": "57b912ab-eb47-4cd5-bc9d-73abece1f1b3",
  "type": "Building",
  "dataProvider": {
    "value": "OperatorA"
     },
  "location": {
    "type": "geo:json",
    "value": {
            "type": "Polygon",
            "coordinates": [[[100, 0], [101, 0], [101, 1], [100, 1]]]
        }
    },
  "owner": {
    "type": "Relationship",
    "value": [
            "cdfd9cb8-ae2b-47cb-a43a-b9767ffd5c84",
            "1be9cd61-ef59-421f-a326-4b6c84411ad4"
        ]
    },
  "description": {
    "value": "Office block"
    },
  "refRelatedDeviceOperation": {
    "type": "Relationship",
    "value": [
            "36744245-6716-4a28-84c7-0e3d7520f143",
            "33b2b713-9223-40a5-87a0-3f80a1264a6c"
        ]
    },
]
```

Get Devices in Specific Building

Endpoint URL: [/version/Field-Test/Buildings/devices/id]

Name	Type	Example	Description
Version	string	V0-.1	verson of API to use
Field-Test	String	Sweden	The responsible field test
id	String	"57b912ab-eb47-4cd5-bc9d-73abece1f1b3"	id of building

Table 6: Parameters

Provides information regarding the available measurements types for the specific building. A valid id can be obtained by the /version/trial/buildings service. "Sweden" is the only trial offering this service in the building domain.


```
{
 "deviceType":[
    {
    "id": "57b912ab",
    "type": "Sensor",
    "controllProperty": {
    "value": "tempreture Sensor"
    }
       {
    "id": "57b912cb",
    "type": "Actuator",
    "controllProperty": {
    "value": "Humidity"
    }
]
```


Get Specific Devices in Buildings

Endpoint URL: [/version/Field-Test/Buildings/device-types/type]

Provides information regarding the specific type of devices from available buildings. A valid type can be obtained by the /version/trial/buildings service. "Sweden" is the only trial offering this service in the building domain.

Name	Type	Example	Description
Version	string	V0-.1	verson of API to use
Field-Test	String	Sweden	The responsible field test
type	String	Sensor	The type of the measurement type to query for

Table 7: Parameters


```
{
    "id": "57b912ab",
    "type": "Sensor",
    "controllProperty": {
    "value": "tempreture Sensor"
    }
       {
    "id": "57b912cb",
    "type": "Sensor",
    "controllProperty": {
    "value": "Humidity"
    }
]
```


Get Specific Device Description in the Building

Endpoint URL: [/version/Field-Test/Buildings/device-description/id]

Name	Type	Example	Description
Version	string	V0-.1	verson of API to use
Field-Test	String	Sweden	The responsible field test
id	String	57b912ab	The id of the device

Table 8: Parameters

Provides information regarding the specific device from a building. A valid id can be obtained by the /version/trial/buildings service. "Sweden" is the only trial offering this service in the building domain.


```
{
```

```
    "id": "57b912ab",
    "type": "Sensor",
    "controllProperty": {
    "value": "tempreture Sensor"
    }
}
```


Get Building in Specific location

Endpoint URL: [/version/Field-Test/Buildings/Zone/location]

Provides a list buildings in details which are located in specific zone. "Sweden" is the only field test offering this service.

Name	Type	Example	Description
Version	string	V0-.1	verson of API to use
Field-Test	String	Sweden	The responsible field test
location	GeoLocation	[[[100, 0], [101, 0], [101, 1], [100, 1]]	specify location

Table 9: Parameters


```
{
    "id":  "57b912ab-eb47-4cd5-bc9d-73abece1f1b3" ,
    "type": "Building",
    "location": {
       "type": "geo:json",
       "value": {
            "type": "Polygon",
            "coordinates": [[[100, 0], [101, 0], [101, 1], [100, 1]]]
        }
    }
]
```

Smart Grid Information

This set of APIs provide access to information related to smart grids presentation

Get List of Devices in Field Test

Endpoint URL: [/version/Field-Test/SmartGrid]

Provide a detailed list of devices in the field test, 'Aachen' is the only trial offering this service.

Name	Type	Example	Description
Version	string	V0-.1	verson of API to use
Field-Test	String	Aachen	The responsible field test to use for invoking the respective service

Table 10: Parameters

The class diagram of the response (instance of Smart Grids) of the service follows in Figure 7.

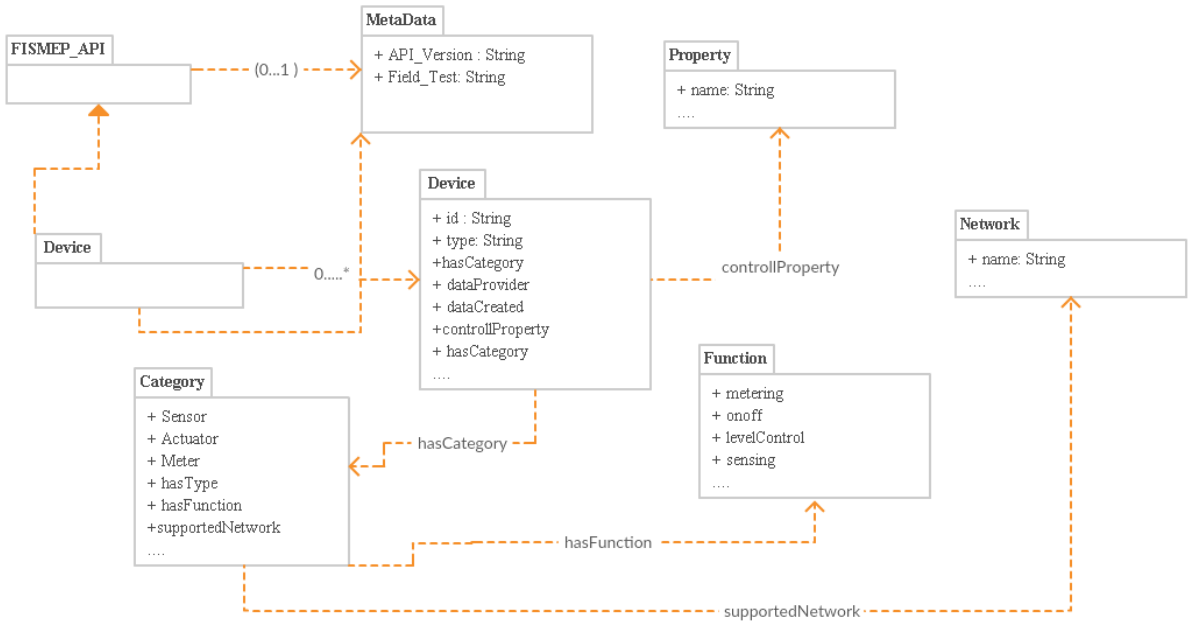

Figure 6: Class Diagram of Related to a List of Smart Grid device request

A list attributes which can be retrieved from context registry of the smart grids, is like:

```
[
 {
 "id": "57b912ab-3",
 "type": "Actuator",
 "properties": ["name","primaryV", "primaryC","secondaryV",...],
 "relationships": ["controlProperty","hasFunction","supportedNetwork" ]
},
{
 "id": "57b912ab-4",
 "type": "meter",
 "properties": ["name","primaryV", "primaryC","secondaryV",..],
 "relationships": ["controlProperty","hasFunction","SupportedNetwork"]
   }
]
```


Name	Type	Description
id	String	The id of the Building
type	String	Device category(Actuator, Sensor, Meter)
controllProperty	relationship	URL to information of Property
name	String	optional (converter , CT-VT)
location	Geolocation	Geo location of the Building
description	String	more information about the Building
hasFunction	relationship	a type of supported function
supportedNetwork	relationship	a link to information of network
primaryV	property	a float value voltage
primaryC	property	a float value of current
secondaryV	property	a float value of voltage
secondaryC	property	a float value of current

Table 11: Attributes of Smart Grid

Get Device Description

Endpoint URL: [/version/Field-Test/SmartGrid/Description/id]

Provide description of specific device in the field test, 'Aachen' is the only trial offering this service. Device in this field could be an actuator or a meter which are named converter and CT-VT respectively.

Name	Type	Example	Description
Version	string	V0-.1	verson of API to use
Field-Test	String	Aachen	The responsible field test
id	String	57b912ab-3	The specific device

Table 12: Parameters

A list attributes which can be retrieved from context registry of the smart grids for specific device, is like:

```
[
 {
 "id": "57b912ab-3",
 "type": "Actuator",
 "name" : {
        type: Property
        value: "convertor 1"
     },
 "primaryV": {
        type: Property
        value: "0.122"
     },
 "primaryC":{
        type: Property
        value: "5.122"
     },
 "secondaryV":{
        type: Property
        value: "622"
     },
 "secondaryC":{
        type: Property
        value: "1,52"
     },
 "location": {
      "type": "geo:json",
```

```
        "value": {
            "type": "Polygon",
            "coordinates": [[[100, 0], [101, 0], [101, 1], [100, 1]]]
        }
    }
  "supportedNetwork":{
          type: Relationship
          value: "MMS"
      },
  "controllProperty": {
        type: Relationship
        value: "Energy"
      },
  "description": {
        value: "a fault message "
      }
},

]
```

Get Devices in Specific Location

Endpoint URL: [/version/Field-Test/SmartGrid/Zone/location]

Provide list of all devices in the field test according to the given location, 'Aachen' is the only trial offering this service. Device in this field could be an actuator or a meter which are named converter and CT-VT respectively.

Name	Type	Example	Description
Version	string	V0-.1	verson of API to use
Field-Test	String	Aachen	The responsible field test
location	GeoLocation	[[[100, 0], [101, 0], [101, 1], [100, 1]]]	location of the specific device

Table 13: Parameters

A list attributes which can be retrieved from context registry of the smart grids for specific device, is like:

```
[
 {
 "id": "57b912ab-3",
 "type": "Actuator",
 "name" : {
        type: Property
        value: "convertor 1"
     },
 "primaryV": {
        type: Property
        value: "0.122"
     },
 "primaryC":{
        type: Property
        value: "5.122"
     },
 "secondaryV":{
        type: Property
        value: "622"
     },
 "secondaryC":{
        type: Property
        value: "1,52"
     },
 "location": {
      "type": "geo:json",
      "value": {
           "type": "Polygon",
           "coordinates": [[[100, 0], [101, 0], [101, 1], [100, 1]]]
       }
   }
 "supportedNetwork":{
          type: Relationship
          value: "MMS"
     },
```

```
  "controllProperty": {
        type: Relationship
        value: "Energy"
      },
  "description": {
        value: "a fault message "
      }
},
]
```

Metering Information

This set of APIs provide access to information related to PMU presentation.

Get List of All PMUs

Endpoint URL: [/version/Field-Test/PMU]

Provide a detailed list of devices in the field test, 'Bucharest' is the only trial offering this service.

Name	Type	Example	Description
Version	string	V0-.1	verson of API to use
Field-Test	String	Bucharest	The responsible field test to use for invoking the respective service

Table 14: Parameters

The class diagram of the response of the service follows in Figure 7.

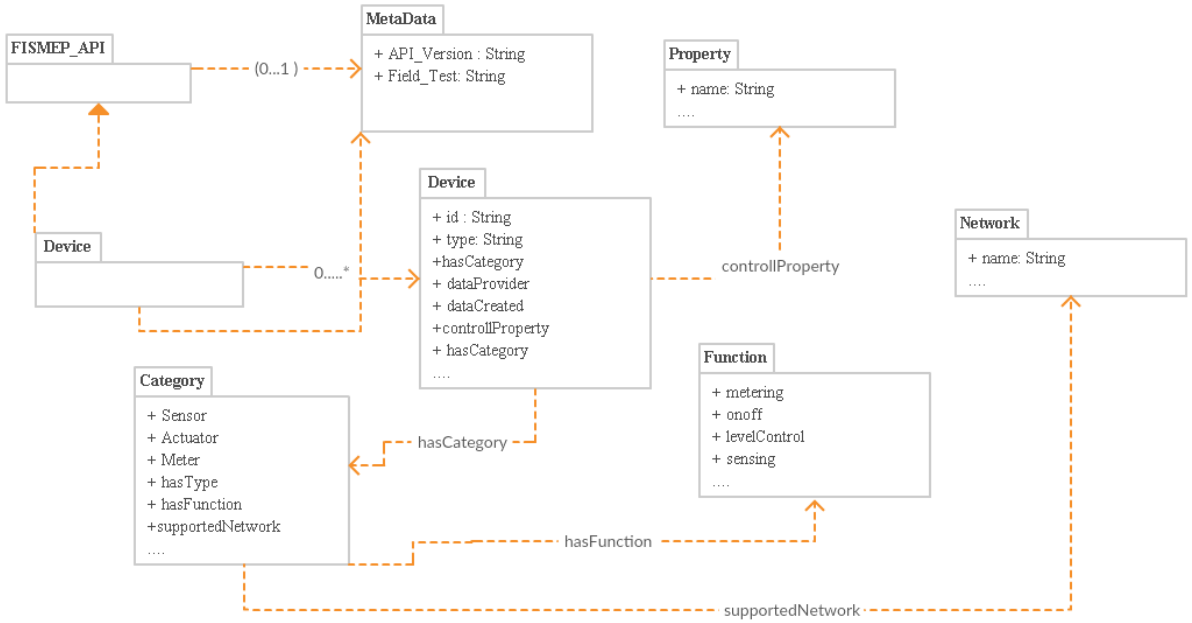

Figure 7: Class Diagram of Related to a List of PMUs

A list attributes which can be retrieved from context registry of the PMUs, is like:

Name	Type	Description
id	String	The id of the Building
type	String	Device category(Actuator, Sensor, Meter)
pmuType	String	naming of PMU
controllProperty	relationship	URL to information of Property
name	String	PMU as a type meter
location	Geolocation	Geo location of the Building
description	String	more information about the Building
hasFunction	relationship	a type of supported function
supportedNetwork	relationship	a link to information of network
voltageLevel	property	a float value voltage level
voltageTransformer	property	a float value of voltage transformer
installationTime	DateTime	a date and time of installation
reportingRate	property	a float value of reporting rate

Table 15: Attributes of PMUs


```
[
 {
 "id": "57b912ab-3",
 "type": "meter",
 "properties": ["name","location", "voltageLevel","voltageTransformer",...],
 "relationships": ["controlProperty","hasFunction","SupportedNetwork" ]
},
{
 "id": "57b912ab-4",
 "type": "meter",
 "properties": ["name","location", "voltageLevel","voltageTransformer",...],
 "relationships": ["controlProperty","hasFunction","SupportedNetwork"]
   }
]
```


Get Specific Meter Description

Endpoint URL: [/version/Field-Test/PMU/Description/id]

Provide description of specific PMU in the field test, 'Bucharest' is the only trial offering this service.

Name	Type	Example	Description
Version	string	V0-.1	verson of API to use
Field-Test	String	Bucharest	The responsible field test
id	String	57b912ab-3	The specific meter/PMU

Table 16: Parameters

A list attributes which can be retrieved from context registry of the smart grids for specific device, is like:

```
[
 {
 "id": "57b912ab-3 ",
 "type": "Meter",
 "name" : {
        type: Property
        value: "PMU"
     },
 "pmuType": {
        type: Property
        value: "arbiter-SEL"
     },
 "installationTime":{
        type: DateTime
        value: "5:122"
     },
 "reportingRate":{
        type: Property
        value: "10"
     },
 "voltageLevel":{
        type: Property
        value: "1,52"
     },
 "currentTransformer": {
        type: Property
```

```
        value: "0.1 "
      }
  "voltageTransformer": {
        type: Property
        value: "2.5 "
      }
  "location": {
       "type": "geo:json",
       "value": {
            "type": "Polygon",
            "coordinates": [[[100, 0], [101, 0], [101, 1], [100, 1]]]
       }
    }
  "supportedNetwork":{
          type: Relationship
          value: "http"
      },
  "controllProperty": {
        type: Relationship
        value: "Energy"
      },
  "description": {
        value: "a fault message "
      }
},
]
```


Get List of PMUs in Specific Location

Endpoint URL: [/version/Field-Test/PMU/Zone/location]

Provide list of all devices in the field test according to the given location, 'Bucharest' is the only trial offering this service. Device in this field are a kind of meter which are named PMU.

A list attributes which can be retrieved from context registry of the meters for specific device, is like Table 17

Name	Type	Example	Description
Version	string	V0-.1	verson of API to use
Field-Test	String	Bucharest	The responsible field test
location	String	[[[100, 0], [101, 0], [101, 1], [100, 1]]]	location of the specific PMU

Table 17: Parameters


```
[
{
  "id": "57b912ab-3",
  "type": "Meter",
  "name" : {
        type: Property
        value: "PMU"
      },
 "pmuType": {
        type: Property
        value: "arbiter-SEL"
      },
 "installationTime":{
        type: DateTime
        value: "5:122"
      },
 "reportingRate":{
        type: Property
        value: "10"
      },
 "voltageLevel":{
        type: Property
        value: "1,52"
      },
 "currentTransformer": {
        type: Property
        value: "0.1 "
      }
```

```
  "voltageTransformer": {
        type: Property
        value: "2.5 "
      }
  "location": {
       "type": "geo:json",
       "value": {
            "type": "Polygon",
            "coordinates": [[[100, 0], [101, 0], [101, 1], [100, 1]]]
       }
    }
  "supportedNetwork":{
          type: Relationship
          value: "http"
      },
  "controllProperty": {
        type: Relationship
        value: "Energy"
      },
  "description": {
        value: "a fault message "
      }
},
]
```