SYNCHRONICITY

GA no:	732240
Action full title:	SynchroniCity: Delivering an IoT enabled Digital Single Market for Europe and Beyond
Call/Topic:	Large Scale Pilots
Type of action:	Innovation Action (IA)
Starting date of action:	01.01.2017
Project duration:	36 months
Project end date:	31.12.2019
Deliverable number:	D2.2
Deliverable title:	Guidelines for the definition of OASC Shared Data Models
Document version:	V1.0
WP number:	WP2
Lead beneficiary:	38 – FF
Main author(s):	José M. Cantera (FF)
Internal reviewers:	Francesco Arigiliano (ENG), Eunah Kim (UDG Alliance), Martin Brynskov (AU)
Type of deliverable:	Report
Dissemination level:	PU
Delivery date from Annex 1:	M14
Actual delivery date:	09.03.2018 (M15)

Co-funded by the European Union

This deliverable is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732240.

Executive Summary

The availability of shared, well-adopted information models is a key interoperability mechanism for enabling a global market for IoT-enabled urban services. Such models provide an essential element in the common technical ground needed for standards-based innovation and procurement, by making replicability and portability of smart city solutions practical. Global convergence is needed to attract investments. This document describes a simple framework for defining new information models in SynchroniCity, taking a multiple baseline approach based on and extending the OASC Minimal Interoperability Mechanisms (MIMs). The report includes an information meta-model through which concrete, domain-specific data models can be expressed in a coherent way across domains. This allows for sector-specific focus in a procurement or development process, while maintaining cross-domain consistency. In addition to the information meta-model, the report provides recommended design guidelines, documentation guidelines and compliance rules. Finally, an annex provides concrete examples of existing data models in different domains. The document represents work in progress in the convergence of several streams of activities, including ETSI ISG CIM, SF-SSCC, ITU-T FG-DPM and others, and is part of an ongoing effort to establish a practical and converging instantiation of the OASC MIMs. The SynchroniCity data model guidelines are complemented by the SynchroniCity Architecture Guidelines, and they are an integral part of the SynchroniCity Architecture.

Abbreviations

CIM	Context Information Management
D	Deliverable
EC	European Commission
EIP-SCC	European Innovation Partnership on Smart Cities & Communities
ETSI	European Telecommunications Standards Institute
FG-DPM	Focus Group on Data Processing and Management for IoT and Smart Cities & Communities
GSMA	GSM Industry Association
HTTP	Hypertext Transfer Protocol
ISG	Industry Specification Group
ITU-T	United Nations International Telecommunications Union
JSON	Javascript Object Notation
MIM	Minimal Interoperability Mechanism
NGSI	Next Generation Service Interface
OASC	Open & Agile Smart Cities
OMA	Open Mobile Alliance
SF-SSCC	Sector Forum for Smart and Sustainable Cities and Communities
WP	Work Package
WT	Work Task

Contents

1 Introduction

1.1 Introduction

The following document is intended to specify guidelines and compliance for SynchroniCity Data Models, which is a concrete extension of the OASC Minimal Interoperability Mechanisms (MIMs). The main audiences are open data publishers, developers, smart city administrators or other stakeholders pursuing standards-based innovation and procurement.

The document starts by explaining and helping readers to understand the necessity of a harmonised set of data models in the context of the SynchroniCity Architecture (described by Deliverable D2.1).

Then, a justification of the selected OASC reference information meta-model is provided. The selected information meta-model is based on the OMA NGSI meta-model (described by Deliverable D2.1, section 4.2), largely used by smart city projects based on FIWARE, and especially by at least five of the core cities in the SynchroniCity project. One of the key advantages of using NGSI is that instantiations of such information meta-model can be encoded using JSON, so that application developers or other parties can consume data easily.

Such information meta-model shall be used as the basis when instantiating concrete, domain-specific models. In fact, these guidelines are intended to define a series of recommendations that must be followed when designing new domain-specific models for SynchroniCity, based on the OASC MIMs.

Then, a description of the main design guidelines is provided, followed by an enumeration of the documentation guidelines.

The document continues with the compliance rules that shall be met by all SynchroniCity domain specific data models.

The last chapter is devoted to future work related to this deliverable, and particularly an overview of the on-going work made by the ETSI ISG CIM and ITU-T FG-DPM standardization initiatives.

Finally, an annex is included enumerating the baseline data models for SynchroniCity coming from FIWARE and GSMA.

1.2 Architectural considerations

Section 2.2 of the SynchroniCity Architecture deliverable introduces the system, data management, service, security and privacy requirements of the SynchroniCity platform that will drive the design of its architecture and support the development of the digital single market.

In particular with regards to data management, the requirement states that the system has to support and reuse open and standard data models and metadata by providing pre-built taxonomies to describe different assets. The final aim is to facilitate replicability of solutions across cities while avoiding vendor lock-in.

Thus, following the OASC principles, in the SynchroniCity architecture, data models play a crucial role because they define the harmonised representation formats and semantics that will be used both by city applications to consume data (through the northbound interfaces) and by data sources at the southbound (sensors, existing information systems or city databases) to publish data. Furthermore, data models are one of the key “interoperability points” (Deliverable 2.1, Section 3.4.2), enabling the participation in a digital single market. The implementation of the SynchroniCity data models, together with other interoperability points will ensure that each of the 8 heterogeneous cities can deploy interoperable IoT-enabled urban services.

1.3 Information Meta-Model Selection Rationale

The rationale for selecting OMA NGSI as a reference meta-model for instantiating data models in SynchroniCity can be summarised in the following points:

- It is the baseline reference implementation supported by the Open & Agile Smart Cities initiative, with more than 100 member cities in 24 countries globally, who support the idea of Minimal Interoperability Mechanisms (MIMs) that enable agile and standards-based procurement and innovation.
- At least 5 of the Reference Zones are already using it, Furthermore, many of these RZs are currently making use of the harmonised FIWARE / GSMA Data Models, which are actually based on NGSI.
- The NGSI meta-model provides a powerful but simple, well-known approach for modelling data in terms of attributes and extra metadata. Experience demonstrates that instantiations of this meta-model can be easily mapped / implemented using a wide variety of data stores including NoSQL, SQL or Graph Databases.
- Instantiations of the NGSI information meta-model can be encoded using simple JSON structures (key-value structures), so that application developers or other parties can consume data easily.
- The NGSI information meta-model is quite close to other meta-models currently used in the industry, namely schema.org, thus enabling interoperability and reusability.
- Future-wise, it ensures a seamless transition to the future ETSI ISG CIM standard, NGSI-LD, as ETSI ISG CIM is defining an information model with direct mappings to OMA NGSI. This work is further linked to global efforts in the UN ITU-T FG-DPM.

2 Common Information Meta-Model

2.1 Introduction

It is *recommended* to use the OASC reference information model, in this case the OMA NGSI meta-model, for the reference implementation of the SynchroniCity data models. The rationale is its simplicity and the fact that it is supported as a baseline by OASC, including early adoption by more than half of the cities involved in the SynchroniCity project.

In addition, instantiations of the NGSI information model can be easily encoded using JSON, which is a very popular data interchange format among application developers.

2.2 OMA NGSI meta-model

The figure below describes the NGSI meta-model. There are three main elements in this meta-model, as shown in the figure below: *Entities, Attributes and Metadata*.

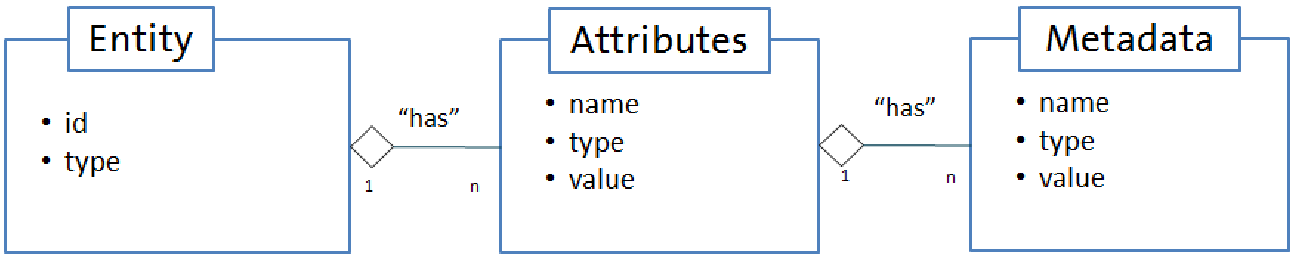

Figure 1.- The OMA NGSI meta-model

Entities are the center of gravity in the NGSI information model. An entity represents a thing, i.e., any physical or logical object (e.g., a sensor, a person, a room, an issue in a ticketing system, etc.). Each entity has an entity id.

Furthermore, the type system of NGSI enables entities to have an entity type. Entity types are semantic types; they are intended to describe the type of thing represented by the entity. For example, a context entity with id "sensor-365" could have the type "*TemperatureSensor*". Each entity is uniquely identified by the combination of its **id** and **type.** It is noteworthy that these elements are always mandatory when the meta-model is instantiated.

Attributes are properties of context entities. For example, the current speed of a car could be modelled as attribute "speed" of entity "car-104". In the NGSI information model, attributes have an **attribute name, an attribute type, an attribute value and metadata**.

The *attribute name* describes what kind of property the attribute value represents of the entity, for example "*speed*". The *attribute type* represents the NGSI value type of the attribute value, which is usually similar or equivalent to the JSON data type. The *attribute value,* finally, contains the actual data and optional metadata describing properties of the attribute value like e.g. *accuracy, provider, or a timestamp*.

Metadata is used as an optional part of the attribute value as described above. Similar to attributes, each piece of metadata has a *metadata name*, describing the role of the metadata in the place where it occurs; for example, the metadata name "accuracy" indicates that the metadata value describes how accurate a given attribute value is; a *metadata type*, describing the NGSI value type of the metadata value; a *metadata value* containing the actual metadata

2.3 JSON Representation

An entity is represented by a JSON object with the following structure:

- The entity id is encoded by the object's *id* property, which value is a string containing the entity id.
- The entity type is encoded by the object's *type* property, which value is a string containing the entity's type name.
- Entity attributes are specified by additional properties, whose names are the name of the attribute and whose representation is described below. Obviously, “id” and “type” are not allowed to be used as attribute names.

An attribute is represented by a JSON object with the following syntax:

- The attribute value is specified by the *value* property, which value might be any JSON value.
- The attribute NGSI type is specified by the *type* property, which value is a string containing the NGSI type, which usually is just the type of the JSON value and can be omitted.

The attribute metadata is specified by the *metadata* property. Its value is another JSON object which contains a property per metadata element defined (the name of the property is the name of the metadata element). Each metadata element, in turn, is represented by a JSON object containing the following properties:

- *value*: Its value contains the metadata value, which may correspond to any JSON value.
- *type*: Its value contains a string representation of the metadata NGSI type, which usually is just the type of the JSON value and can be omitted.

The FIWARE reference implementation defines certain further syntactic restrictions of NGSI API on entity ids, types or attribute/metadata names, which SynchroniCity supports. For a detailed description please check http://fiware.github.io/specifications/ngsiv2/latest/.

2.3 The key-value (“keyValues”) simplified format

The representation described above is usually referred as the NGSI *normalized format*. Its main drawback is that it is very verbose and more difficult to be parsed or validated. That is the main reason behind the introduction of simplified formats in NGSI, namely the key-value format (which can be activated by the *options=keyValues* query URL parameter in the REST API).

The key-value simplified format allows representing entity attributes by their values only, leaving out the information about type and metadata. This is a very convenient format particularly for client applications, as it simplifies considerably the parsing and data extraction processes.

2.4 Representing Geospatial properties of entities

The geospatial properties of an entity can be represented by means of regular attributes. The provision of geospatial properties enables the resolution of geographical queries. The format that must be used by the SynchroniCity data models is **GeoJSON**. GeoJSON is a geospatial data interchange format based on JSON. GeoJSON provides greater flexibility allowing the representation of point altitudes or even more complex geospatial shapes, for instance multi geometries.

2.4 Using JSON-Schema

JSON Schema is a JSON media type for defining the structure of JSON data. JSON Schema is intended to define validation, documentation, hyperlink navigation, and interaction control of JSON data.

The specification of domain-specific SynchroniCity data models **must** use JSON Schema. At least a JSON Schema must be provided, so that “key-value” representations of entities can be properly validated. Optionally it can be provided a more complex schema to validate representations conformant with the NGSI normalized format (the one that explicitly represents metadata and attribute values).

Another advantage of adopting JSON-Schema is that other tools, such as Swagger (https://swagger.io/), will be enabled, allowing developers to create specific REST APIs on top of the existing NGSI APIs.

2.5 Examples of Domain Specific Data Models

Below it can be found an excerpt of a Domain Specific Data Model for Weather, specified using JSON-Schema. The vocabulary of attributes used is defined by the FIWARE Data Models for Weather, which ultimately are based on the work done by the GSMA under the IoT Big Data Project.

```
{
    "properties": {
      "type": {
        "type": "string",
        "enum": [
          "WeatherObserved"
        ],
        "description": "NGSI Entity type"
      },
      "dateObserved": {
        "type": "string",
        "format": "date-time"
      },
      "precipitation": {
        "type": "number",
        "minimum": 0
```

```json
    },
    "barometricPressure": {
      "type": "number",
      "minimum": 0
    },
    "solarRadiation": {
      "type": "number",
      "minimum": 0
    },
    "pressureTendency": {
      "oneOf": [
        {
          "type": "string",
          "enum": [
            "raising",
            "falling",
            "steady"
          ]
        },
        {
          "type": "number"
        }
      ]
    },
    "dewPoint": {
      "type": "number"
    }
  }
}
```

```
  ],
  "required": [
    "id",
    "type",
    "dateObserved",
    "location"
  ]
}
```

Figure 2.- JSON-Schema excerpt

The example above is describing an excerpt of the schema for a weather observation entity. It is noteworthy that such schema is only modelling the key-value representation of the data model. Similarly, a schema for the NGSIv2 normalized model could be devised.

First of all, the different NGSI attributes are defined as properties of the JSON object. In addition, there are four elements required in this data model:

- id: corresponding to the entity id (automatically generated by data providers)
- type: corresponding to the entity type which shall be equal to “WeatherObserved”
- dateObserved: which is the date and time at which the observation was made. Please note how this attribute was defined (format=date-time)
- location: which is a GeoJSON geometry which defines the location associated to the weather observation

2.5 Instantiation Examples

Below there are some examples of instantiations of the common information meta-model using the key-values representation format.

In the first example an entity of type ‘WeatherObserved’ is represented.

In the second example an entity of type “OffStreetParking” is represented.

```
{
  "id": "Spain-WeatherObserved-Valladolid-2016-11-30T07:00:00.00Z",
  "type": "WeatherObserved",
  "barometricPressure": 938.9,
  "dataProvider": "Fiware",
  "dateObserved": "2016-11-30T07:00:00.00Z",
  "location": {
    "type": "Point",
    "coordinates": [
      -4.754444444,
      41.640833333
    ]
  },
  "precipitation": 0,
  "pressureTendency": 0.5,
  "relativeHumidity": 1,
  "source": "http://www.aemet.es",
  "temperature": 3.3,
  "windDirection": -45,
  "windSpeed": 2
 }
```

Figure 3.- WeatherObserved entity instance

Apart from the mandatory entity type and id, there are different attributes and values in accordance with the JSON Schema described previously. The format shown is very compact and can be consumed very easily by different applications or services.

```
{
  "id": "porto-ParkingLot-23889",
  "type": "OffStreetParking",
  "name": "Parque de estacionamento Trindade",
  "category": [
    "underground",
    "public",
    "feeCharged",
    "mediumTerm",
    "barrierAccess"
],
"chargeType": [
  "temporaryPrice"
],
"layout": [
  "multiLevel"
],
"maximumParkingDuration": "PT8H",
"location": {
  "coordinates": [
    -8.60961198807,
    41.150691773
  ],
  "type": "Point"
},
"allowedVehicleType": [
  "car"
],
"totalSpotNumber": 414,
"availableSpotNumber": 132,
"address": {
  "streetAddress": "Rua de Fernandes Tomás",
  "addressLocality": "Porto",
  "addressCountry": "Portugal"
},
"description": "Municipal car park located near the Trindade",
}
```

Figure 4.- OffStreetParking entity instance

The example above shows an entity of type ‘OffStreetParking’ which includes different attributes. In this case there is one attribute, named category, which it is multivalued and represented as an array. Another point to be noted is that there is a property which denotes a duration ‘maximumParkingDuration” and is represented using a text string following the ISO8601 standard for representing durations.

3 Design guidelines

3.1 Introduction

This chapter is intended to describe different design guidelines that have to be taken into account when devising new data models for SynchroniCity in general. Each design guideline is numbered and includes an example when needed.

3.2 Syntax Guidelines

S1.1 Use **English terms**, preferably in American English.

Example: “speed”, “Alert”

S1.2 Use **camel case** syntax for vocabulary terms (attributes, entity types or metadata).

Example: “brandName”, “OffStreetParking”

S1.3 Use **camel case** syntax for **enumerated** values which are strings.

Example: “outOfOrder”

S1.4 Entity **type names** must start with a **capital** letter.

Example: “WasteContainer”

S1.5 Attribute names or metadata names must start with a **lowercase** letter.

Example: “name”

S1.6 When **enumerated values** are strings they must start with a **lowercase** letter.

Example: “open”

S1.7 Use names and not verbs for attribute or metadata names, qualifying them when necessary.

Example: “location” , “totalSpotNumber”, “dateCreated”

S1.8 Avoid plurals in attribute or metadata names, but state clearly in the documentation or in the associated JSON Schema when a list of items fits.

Example: “category” may contain more than one category name.

S1.9 Use the **“*date*” prefix** for naming entity attributes representing dates (or complete timestamps).

Example: “dateLastEmptying”.

3.2 General Guidelines

G2.1 Avoid the over generalization trap. Design **concrete** entity types and attributes. Try to find the right initial level of generality.

Example: Escape from very generic terms like ‘Agent’.

G2.2 Complexity should be added **incrementally**, so it is recommended to start **simple**. Then, the work can be revisited and the blanks filled in. Applications can ask for sophistication later, after mass adoption.

G2.3 Be **agile**. Move **fast**. Iterate fast. Accept mistakes. Publish schemas early, even though it is known that improvements will be needed.

G2.4 Before defining a new attribute, check whether is **already defined** on any of the existing data models and try to reuse it. Some of the most used attributes are listed at the end of http://fiware-datamodels.readthedocs.io/en/latest/guidelines/index.html.

Example: "temperature" can apply to multiple entity types, so there is no need to duplicate, but just to add a new entity type to such attribute's domain.

G2.5 It is always recommended to have a look at http://schema.org trying to **find a similar term** with the **same semantics**.

Example: "http://schema.org/name" for "name"

G2.6 Try to find **common used ontologies** or existing standards well accepted by the Community, or by governments, agencies, etc.

Example: *Open311* for civic issue tracking, Datex II for intelligent transport systems, or the SAREF for smart appliances in the smart home.

G2.7 Whenever possible, **reuse** *schema.org* data types

Example: "Text", "Number", "DateTime", "StructuredValue", etc.

G2.8 Use http://schema.org/DateTime as the NGSI attribute type for attributes representing dates or timestamps (which name should start with "date").

G2.9 Use the "*address*" attribute for civic locations as per schema.org i.e. the NGSI type of the corresponding attribute must be http://schema.org/PostalAddress.

G2.10 Use the "*location*" attribute for geographical coordination encoded using **GeoJSON**.

Example: See instantiation examples in the previous chapter

G2.11 Unit codes must be encoded using the *UN/CEFACT Common Code* (max. 3 characters). See http://wiki.goodrelations-vocabulary.org/Documentation/UN/CEFACT_Common_Codes

Example: "GP" represents milligrams per cubic meter.

3.3 Attribute design Guidelines

A3.1 State clearly what attributes are **mandatory** and what are **optional**. If needed, state clearly what is the meaning of a "*null*" value.

A3.2 Enumerate the allowed values for each attribute. Generally speaking it is a good idea to leave it open for applications to **extend** enumerations, provided new values are not semantically covered by the existing ones.

Example: possible values for "status" can be ("*open*", "*closed*", "*outOfOrder*")

A3.3 Define a **default unit** for magnitudes. It is recommended to use the unit as stated by the International System of Units.

Example: "meter" for length

A3.4 If a quantity is expressed in a different unit than the default one, use the "***unitCode***" metadata attribute to convey the unit used.

A3.6 Use values between **0 and 1** for relative quantities (parts per one)

Example: "relativeHumidity" or "precipitationProbability" should be expressed with a value between 0 and 1, instead of a percentage.

A3.7 Use a metadata attribute named "***timestamp***" for capturing the last update timestamp of a dynamic attribute.

It is noteworthy that this is the actual date at which the measured value was obtained (from a sensor, by visual observation, etc.), and that date might be different than the date (metadata attribute named "dateModified" as per NGSIv2) at which the attribute of the digital entity was updated, as typically there might be a delay, specially on IoT networks which deliver data only at specific time slots.

A3.8 If an attribute is used for **linking** an entity to another, because there is a relationship between them, the name of the attribute should start with the **prefix "ref"**. Then, the name of the target (linked) entity type should follows.

Example: an attribute named "*refWasteContainerModel*" can be used to convey a link between a waste container and an entity of type "WasteContainerModel" which includes all the static data corresponding to the characteristics of a particular model of container.

A3.9 An attribute named "dateCreated" must be used to denote the (digital) entity's creation date.

"dateCreated" is a special entity attribute provided off-the-shelf by NGSIv2 API implementations. It is noteworthy that it can be different than the actual creation date of the real world entity represented by its corresponding digital entity.

A3.10 An attribute named "dateModified" must be used to denote the (digital) entity's last update date.
"dateModified" is a special entity attribute provided off-the-shelf by NGSIv2 API implementations. It is noteworthy that it can be different than the actual update date of the real world entity represented by its corresponding digital entity.

A3.11 When necessary define additional attributes to capture precisely all the details about dates.

Example: to denote the date at which a weather forecast was delivered an attribute named "dateIssued" can be used. In that particular case just reusing "dateCreated" would be incorrect because the latter would be the creation date of the (digital) entity representing the weather forecast which typically might have a delay.

3.4 Compliance

A SynchroniCity compliant data model **must** follow all the design guidelines described by the present chapter.

4 Documentation guidelines

4.1 Introduction

This chapter is intended to describe the documentation guidelines that have to be followed when devising new data models for SynchroniCity. Each guideline is numbered and includes an example when needed.

4.2 Guidelines

D1.1 It is highly recommended to use **markdown** format to document data models. It allows easy publication of the data models in a readthedocs portal.

Example http://github.com/fiware/dataModels is published to

http://fiware-datamodels.readthedocs.org

The recommended documentation template can be found at

https://github.com/Fiware/dataModels/blob/master/datamodel_template.md

D1.2 It is highly recommended to upload new data model specifications to **Github** and to contribute them to broader industry initiatives such as FIWARE Data Models, GSMA, TMForum or ultimately schema.org.

D1.3 Check that for each data model the following elements are included in the documentation:

- A general description of the data model and the main entities modelled.
- A description of the attributes of the data model that are formalised using JSON Schema (see below).
- An example of data model instance encoded as a JSON code fragment.
- A description of reference / example endpoints that offer data following the specified data model (if there are any available).
- A list of activities yet to be completed on the data model or open issues.

D1.4 A **JSON Schema** for the data model **must** be included, at least the one that is intended to represent the data using the NGSIv2 "keyValues" mode.

D1.5 Reuse existing base **JSON Schemas** already provided by the community, such as

https://github.com/Fiware/dataModels/blob/master/common-schema.json

which includes common general properties (name, id, etc.), and

https://github.com/Fiware/dataModels/blob/master/geometry-schema.json

which includes all the JSON Schema definitions needed for using GeoJSON.

D1.6 A recommended structure for the **folders** in a Github repository containing data model specifications can be seen below:

`EntityTypeName/`

- `doc/`
  - `spec.md`: Data model description based on the data model template
- `README.md`: A summary file

- `schema.json`: The JSON Schema definition
- `example.json`: Example file including an instance encoded in JSON format

D1.7 Use Github **pull requests** and code reviews when creating new data models.

D1.8 Define **owners** and peers of each data model. Owners or peers should review all proposed changes to a data model.

4.3 Compliance

A SynchroniCity compliant data model must follow all the documentation guidelines described by the present chapter.

5 Future work

5.1 Introduction

The OMA NGSI information model (to be used by SynchroniCity) is currently being evolved to better support linked data (entity relationships), property graphs and semantics (exploiting the capabilities offered by JSON-LD). This work is being conducted under the ETSI ISG CIM initiative and it is going to be branded as NGSI-LD.

It is noteworthy that the ETSI ISG CIM information model is a generalization of the existing OMA NGSI information model. As a result, it is expected a good level of compatibility and a clear migration path between both information models.

Although at the time of writing it is unknown when implementations will start supporting this new information model, it is worth devoting some words to it, so that the SynchroniCity stakeholders know the new possibilities offered and have a clearer view of the roadmap.

5.2 ETSI ISG CIM information model

The figure below shows a UML diagram which describes the ETSI ISG CIM information model. The main constructs are NGSI-LD Entity, NGSI-LD Property and NGSI-LD Relationship. NGSI-LD Entities (instances) can be the subject of NGSI-LD Properties or NGSI-LD Relationships. In terms of the traditional NGSI data model, NGSI-LD Properties can be seen as the combination of an attribute (property) and its value. NGSI-LD Relationships allow to establish relationships between instances using linked data. In practice, they are an NGSI attribute, but with a special value which happens to be a URI which points to another entity. They are similar to the “ref” attributes recommended by these guidelines.

NGSI-LD Properties and NGSI-LD Relationships can be the subject of other NGSI-LD Properties or NGSI-LD Relationships. Thus, in the ETSI ISG CIM information model there are no attribute’s metadata but just “properties of properties”. It is not expected to have infinite graphs and in practice only one or two levels of property or relationship “chaining” will happen. Typically, there will be one, equivalent to the NGSI metadata abstraction.

NGSI-LD Entities are represented using JSON-LD a JSON-based serialization format for Linked Data. The main advantage of JSON-LD is that it offers the capability of expanding JSON terms to URIs so that vocabularies can define terms unambiguously.

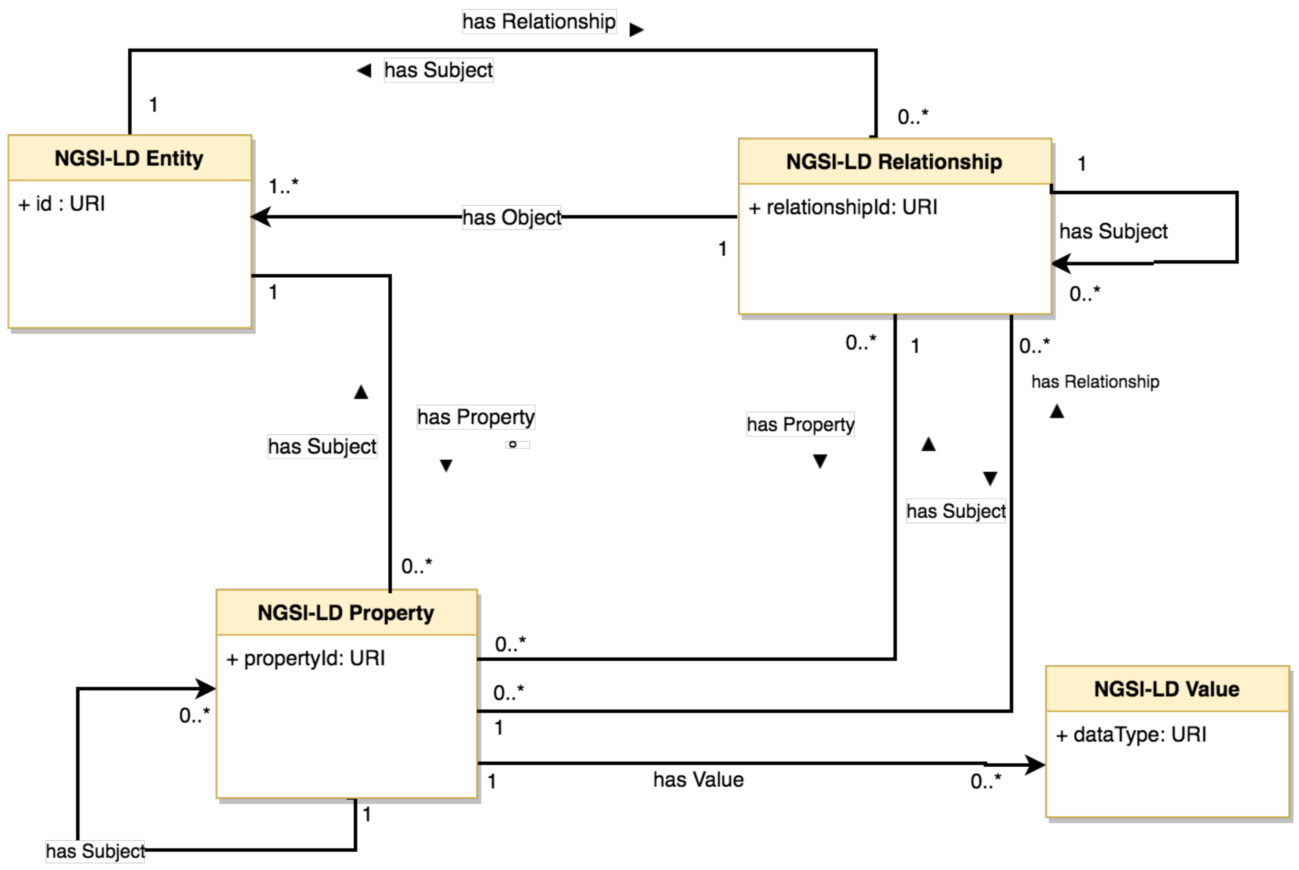

Figure 5 .- ETSI ISG CIM (NGSI-LD) Information model represented using UML

The following figure shows an instantiation example of the NGSI-LD information model. It conveys that there is an instance of an entity of type *Vehicle* which is parked at a certain parking house (entity of type *OffStreetParking*). Different properties about those entities are provided and additional properties of properties (for instance, a timestamp) or properties of relationships (*parkingDate*) are described.

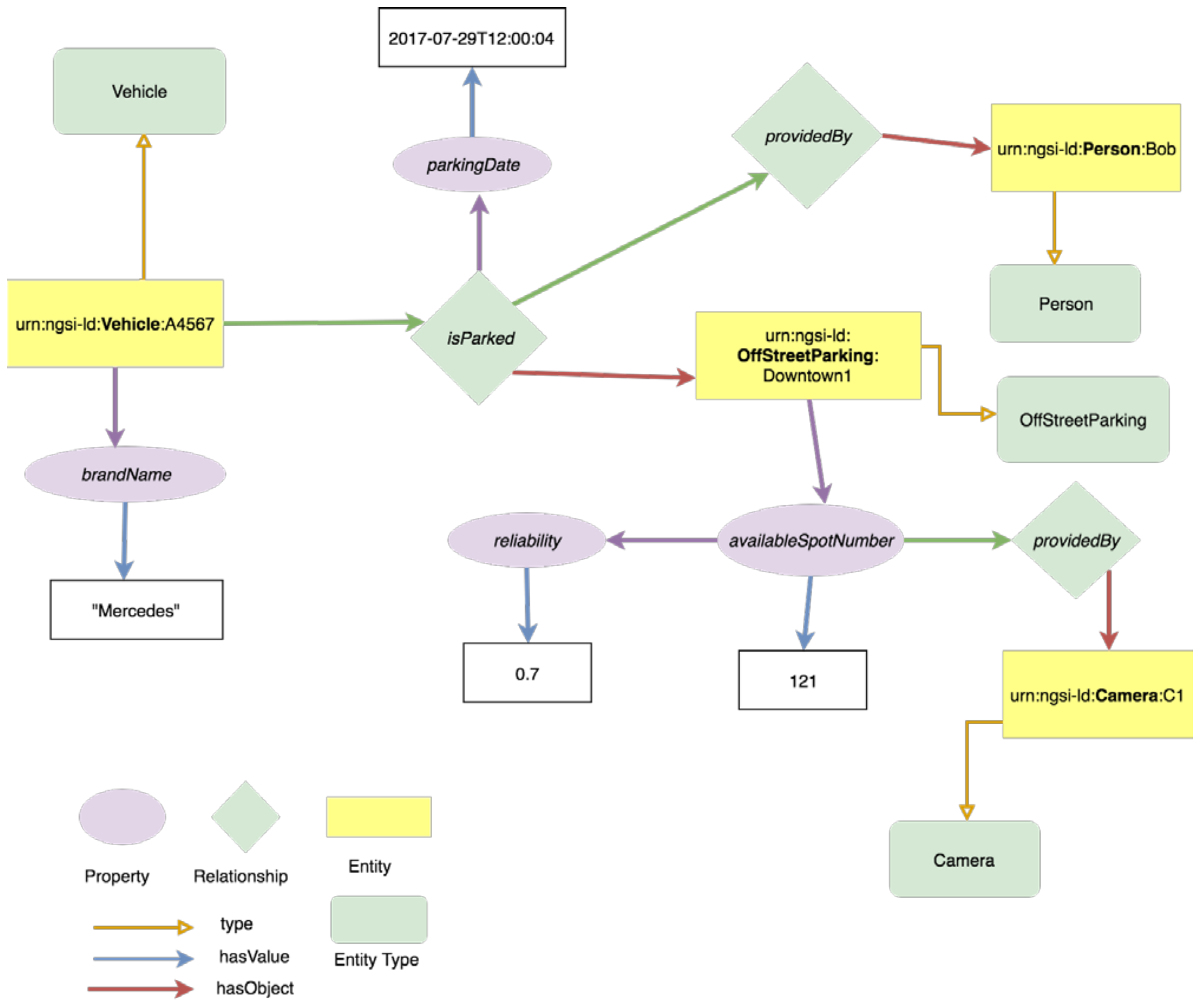

Figure 6.- Instantiation Example of the NGSI-LD Information Model

Below there is an example of the JSON-LD serialization corresponding to an entity of type *Vehicle* .

```
{
        "id": "urn:ngsi-ld:Vehicle:A4567",
        "type": "Vehicle",
        "brandName": {
                "type": "Property",
                "value": "Mercedes"
        },
        "isParked": {
                "type": "Relationship",
                "object": "urn:ngsi-ld:OffStreetParking:Downtown1",
                "observedAt": "2017-07-29T12:00:04",
                "providedBy": {
                        "type": "Relationship",
                        "object": "urn:ngsi-ld:Person:Bob"
                }
        },
        "@context": [
                "http://uri.etsi.org/ngsi-ld/coreContext.jsonld",
                "http://example.org/cim/commonTerms.jsonld",
                "http://example.org/cim/vehicle.jsonld",
                "http://example.org/cim/parking.jsonld"
        ]
}
```

Figure 7.- JSON-LD Representation of an entity of type *Vehicle*

Similarly, an entity of type *OffStreetParking* can be represented as follows:

```
{
    "id": "urn:ngsi-ld:OffStreetParking:Downtown1",
    "type": "OffStreetParking",
    "name": {
            "type": "Property",
            "value": "Downtown One"
    },
    "availableSpotNumber": {
            "type": "Property",
            "value": 121,
            "observedAt": "2017-07-29T12:05:02",
            "reliability": {
                    "type": "Property",
                    "value": 0.7
            },
        "providedBy": {
                "type": "Relationship",
                "object": "urn:ngsi-ld:Camera:C1"
        }
    },
    "totalSpotNumber": {
        "type": "Property",
        "value": 200
    },
    "location": {
        "type": "GeoProperty",
        "value": {
                "type": "Point",
                "coordinates": [-8.5, 41.2]
        }
    },
    "@context": [
        "http://uri.etsi.org/ngsi-ld/coreContext.jsonld",
        "http://example.org/cim/parking.jsonld"
    ]
}
```

Figure 8.- JSON-LD representation of an entity of type *OffStreetParking*

A Catalogue of baseline Data Models

Introduction

This annex is intended to describe the baseline for the SynchroniCity Data Models. There are several starting points to be considered:

- The IoT Big Data Harmonised Data Models developed by **GSMA**.
- The FIWARE Data Models developed by the **FIWARE Community**.
- The **schema.org** Data Models developed by the Browser and search engine communities
- Ontologies promoted by the European standards community, particularly **SAREF**.

The first two initiatives are closely related and in general well synchronized. However, there are some Data Models developed by FIWARE not yet adopted by GSMA and vice versa. And there are some GSMA data models that were extended by FIWARE.

To be taken as reference by partners and other stakeholders, the following table describes the available Data Models, grouped by verticals, that should be used as a baseline by SynchroniCity.

Baseline Data Models

Vertical	**Data Model**	**Source**
Building	Building	GSMA
Building	BuildingType	GSMA
Building	BuildingOperation	GSMA
Civic Issue Tracking	Open311:ServiceType	FIWARE / Open311
Civic Issue Tracking	Open311:ServiceRequest	FIWARE / Open311
Devices	Device	GSMA - SAREF Extended by FIWARE
Devices	DeviceModel	GSMA - Extended by FIWARE
Devices	DeviceOperation	GSMA
Environment	AirQualityObserved	GSMA
Environment	WaterQualityObserved	GSMA
Environment	NoiseLevelObserved	FIWARE

Indicators	KeyPerformanceIndicator	FIWARE
Parking	OffStreetParking	FIWARE
Parking	OnStreetParking	FIWARE
Parking	ParkingGroup	FIWARE
Parking	ParkingAccess	FIWARE
Parking	ParkingSpot	FIWARE
Parks & Gardens	Garden	FIWARE
Parks & Gardens	GreenspaceRecord	FIWARE
Parks & Gardens	Flowerbed	FIWARE
PointOfInterest	PointOfInterest	GSMA
PointOfInterest	Museum	FIWARE
PointOfInterest	Beach	FIWARE
PointOfInterest	TouristInformationCenter	schema.org
PointOfInteraction	SmartPointOfInteraction	FIWARE
PointOfInteraction	SmartSpot	FIWARE
Street Lighting	Streetlight	FIWARE
Street Lighting	StreetlightModel	FIWARE
Street Lighting	StreetlightGroup	FIWARE
Street Lighting	StreetlightControlCabinet	FIWARE
Transportation	BikeHireDockingStation	FIWARE
Transportation	Road	GSMA

Transportation	RoadSegment	GSMA
Transportation	TrafficFlowObserved	FIWARE
Transportation	Vehicle	GSMA - Extended by FIWARE
Transportation	VehicleModel	FIWARE
Transportation	VehicleFault	GSMA
Weather	WeatherObserved	GSMA - Extended by FIWARE
Weather	WeatherForecast	GSMA - Extended by FIWARE
Waste Management	WasteContainerIsle	FIWARE
Waste Management	WasteContainerModel	FIWARE
Waste Management	WasteContainer	FIWARE
Events	Event	schema.org

B References

ETSI Industry Specification Group – Context Information Management (ETSI ISG CIM)

https://docbox.etsi.org/ISG/CIM/Open/ETSI_ISG_CIM_Brochure_20170223.pdf

ITU-T Focus Group on Data Processing and Management to support IoT and Smart Cities & Communities

https://www.itu.int/en/ITU-T/focusgroups/dpm/Pages/default.aspx

GSMA IoT Big Data Harmonised Data Model (Version 4.0 December 2017)

https://www.gsma.com/iot/wp-content/uploads/2016/06/CLP.26-v4.0.pdf

FIWARE Data Models

http://fiware-datamodels.readthedocs.org or https://github.com/Fiware/dataModels

W3C, "JSON-LD 1.0 - A JSON-based serialization for Linked Data"

http://www.w3.org/TR/2014/REC-json-ld-20140116/

NGSI-LD Preliminary API Draft for public comment (As of March 2018)

https://docbox.etsi.org/ISG/CIM/Open/ISG_CIM_NGSI-LD_API_Draft_for_public_review.pdf

IETF RFC 7946: "The GeoJSON format"

https://tools.ietf.org/html/rfc7946

JSON Schema

http://json-schema.org/

Schema.org

http://schema.org

ETSI SAREF

http://ontology.tno.nl/saref/