Grant Agreement No.: 723174
Call: H2020-ICT-2016-2017
Topic: ICT-38-2016 - MEXICO: Collaboration on ICT
Type of action: RIA

# D.2.1 Reference data models for data intensive and IoT based Smart City, Smart Health and Smart Security Applications

Revision: v.1.0

| | |
|---|---|
| Work package | WP2 |
| Task | Task 2.1 |
| Due date | 31/05/2017 |
| Submission date | 30/05/2017 |
| Deliverable lead | INAOE |
| Version | 1.0 |
| Authors | Miguel Palacios (INAOE), Netzahualcóyotl Hernández (CICESE), Blanca Vázquez (INFOTEC), Enrique Sucar (INAOE), Ricardo Cuevas (ITESM), Leon Alberne (CENIDET), Hugo Estrada (INFOTEC), Alicia Martinez Rebollar (CENIDET), Fernando Ramirez (CNEIDET), Miguel González (ITESM), German Molina (HOPU), Francisco Cardoso (UBIWHERE), Francisco Monsanto (UBIWHERE),Luis Valentin (INAOE), |
| Reviewers | Federico M. Facca (MARTEL) and Daniele Pizzolli(FBK) |

| | |
|---|---|
| Abstract | This deliverable documents the data models extended/developed in the SmartSDK project for Smart applications based on FIWARE. |
| Keywords | Data Models, FIWARE, Data Management, Smart Health, Smart City, Smart Security. |

**Document Revision History**

| Version | Date | Description of change | List of contributor(s) |
|---|---|---|---|
| V0.1 | 20/03/2017 | Table of Content. Introduction. | Miguel Palacios (INAOE) |
| V0.4 | 28/04/2017 | Integration of data models | Miguel Palacios (INAOE), Enrique Sucar (INAOE), Netzahualcóyotl Hernández (CICESE), Blanca Vázquez (INFOTEC), Ricardo Cuevas (ITESM), Hugo Estrada (INFOTEC), Leon Alberne (CENIDET), Alicia Martinez Rebollar (CENIDET), Fernando Ramirez (CNEIDET), Miguel González (ITESM), Luis Valentin (INAOE), Reinier Oves (INAOE) |
| V0.7 | 19/05/2017 | Implementation of reviewers' comments | Miguel Palacios (INAOE), Netzahualcóyotl Hernández (CICESE), Blanca Vázquez (INFOTEC), Ricardo Cuevas (ITESM) |
| V1.0 | 30/05/2017 | Final version | Miguel Palacios (INAOE), Enrique Sucar (INAOE), Hugo Estrada (INFOTEC), Blanca Vázquez (INFOTEC), German Molina (HOPU), Francisco Cardoso (UBIWHERE), Francisco Monsanto (UBIWHERE). |

## Disclaimer

The information, documentation and figures available in this deliverable, is written by the SmartSDK (A FIWARE-based Software Development Kit for Smart Applications for the needs of Europe and Mexico) – project consortium under EC grant agreement 723174 and does not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein.

## Copyright notice



| Project co-funded by the European Commission in the H2020 Programme | | |
|---|---|---|
| **Nature of the deliverable:** | R | |
| **Dissemination Level** | | |
| PU | Public, fully open, e.g. web | ✓ |
| CI | Classified, information as referred to in Commission Decision 2001/844/EC | |
| CO | Confidential to SmartSDK project and Commission Services | |

** R: Document, report (excluding the periodic and final reports)*

*DEM: Demonstrator, pilot, prototype, plan designs*

*DEC: Websites, patents filing, press & media actions, videos, etc.*

*OTHER: Software, technical diagram, etc.*

# EXECUTIVE SUMMARY

SmartSDK, to facilitate the development of Smart applications, provides a set of recipes that combine a number of reusable elements, namely, Generic Enablers, Reference architectures and Data Models. As regards Data Models, SmartSDK, following the FIWARE design principles, formalizes dynamic and context-related data models using NGSI, thus providing standard means to exchange data among the different services composing a Smart application.

Within SmartSDK, the focus relates to three specific domains: Smart Health, Smart City and Smart security. These domains are covered in the project by means of three scenarios and related applications, for which SmartSDK either reuses existing harmonised FIWARE Data Models, or develops new ones according to the needs of the scenarios. The first scenario, deals with the collection and analysis of health data; the second scenario deals with the provisioning of multi-modal traffic routes that take into account pollutants; the third scenario deals with the analysis of parking and build video camera streamings to detect potential security issues.

The deliverable documents, based on the guidelines included in Deliverable 3.1 "SmartSDK Reference Models and Recipes", the data models developed to cover the main requirements elicited by considered scenario without loss of generalization. Thus, novel data models formalised in NGSI by SmartSDK, are anyhow based on existing best practises, data models and ontologies used in relevant state-of-the art services and/or standard bodies.

For each model, the deliverable provides a description of the entities that compose it, the relations among the entities, and the attributes that define an entity. The rationale for the design choices is also briefly discussed in relation to existing relevant state-of-the-art data models.

Novel date models include:

- Alert
- Alert
- Questionnaire
- Questionnaire/Question
- Questionnaire/Answer
- Transport Schedule
- Agency
- Route
- Stop
- VideoObject
- VisualObject

The deliverable discuss as well how existing FIWARE data models are used in the context of the three application scenarios covered in SmartSDK.

Reused data models include:

- Device/DeviceModel
- Vehicle/VehicleModel
- AirQualityObserved
- PublicVehicleModel
- OffStreetParking
- Building
- UserContext

The data models will be refined following the deployment of the applications and their trialling. Their

updated version will be document in D2.4: “Reference data models for data intensive and IoT based Smart City, Smart Healthcare and Smart Security applications v2”.

# TABLE OF CONTENTS

# ABBREVIATIONS

| | |
|---|---|
| **GTFS** | General Transit Feed Specification |
| **OCB** | Orion Context Broker |
| **NGSI** | Next Generation Service Interface |
| **SDK** | Standard Development Kit |
| **SAREF** | Smart Appliances REference Ontology |
| **MNC** | Mobile Network Code |
| **MCC** | Mobile Country Code |
| **GSM** | Global System for Mobile Communications |
| **CDMA** | Code Division Multiple Access |
| **iDEN** | Integrated Digital Enhanced Network |
| **TETRA** | Terrestrial Trunked Radio |
| **3G** | Third generation of wireless mobile telecommunications |
| **4G** | Third generation of wireless mobile telecommunications |
| **HVAC** | Heating, Ventilation and Air Conditioning |
| **LAN** | Local Area Network |
| **RFC** | Request For Comments |
| **UN/CEFACT** | United Nations Centre for Trade Facilitation and Electronic Business |
| **GPS** | Global Positioning System |
| **ABS** | Anti-lock Braking System |
| **LPG** | Liquefied Petroleum Gas |
| **CNG** | Compressed Natural Gas |
| **URL** | Uniform Resource Locator |
| **POI** | Point Of Interaction |
| **ViSOR** | Video Surveillance Online Repository |

# 1. INTRODUCTION

To ensure that models and architectures defined in FIWARE [1] are enough general to cover data requirements and regulations in different domains, a standardisation process is required. Thus, one of the main objectives of the SmartSDK [2] project is to design abstract models that organize information and standardize the relationship existing in different scenarios, ensuring interoperability across different applications and facilitating the deployment of applications based on FIWARE.

SmartSDK provides three application scenarios in the following domains:

- Smart Health: the scenario deals with the collection and analysis of health data; the second scenario;
- Smart City: the scenario deals with the provisioning of multi-modal traffic routes that take into account pollutants;
- Smart Security: the scenario deals with the analysis of parking and build video camera streamings to detect potential security issues

This document describes the structure of the data models proposed / extended in the context of this application scenarios. In the design process, FIWARE data model guidelines were followed.

## 1.1 Structure of the deliverable

- Section 2 overviews the SmartSDK scenarios that are developed in the context of the Smart Health, Smart City and Smart Security domains.
- Section 3 describes the shared data models and data models specific to a given application scenario.
- Section 4 presents conclusions of the data models design process.

## 1.2 Audience

This deliverable is mainly intended for:

- Developers interested into reusing SmartSDK Data Models in their applications.
- Developers and Knowledge modellers interested into contributing to SmartSDK Data Models.

# 2 OVERVIEW OF SMARTSDK APPLICATION SCENARIOS

## 2.1 Smart Health

Advances in mobile and wearable sensing are allowing the inference of activities and behaviors associated with health by facilitating the collection of daily-life data.

The healthcare application developed in SmartSDK aims to facilitate the harmonisation and sharing of mobile sensing datasets for healthcare. This application focuses on mobile devices that collects sensor data from physical tests conducted by following clinical protocols to assess the risk of falls in older adults.

The developed application has been designed for research purposes, thus, parameter of interest (associated to the risk of falling) are analyzed *a posteriori* and raw sensor data is kept for further inspection.

In this context, the SmartSDK project focuses on creating a cloud-based infrastructure that facilitates the development of mobile healthcare applications based on analyzing data gathered from sensors to allow different stakeholders, such as patients and physicians to better understand how their activities and behavior influence a healthier lifestyle.

## 2.2 Smart City

The application developed in the Smart City domain (called Green Route) will focus on supporting the citizen mobility in high polluted cities, like Mexico City, with the aim of improving the life quality of citizens and fostering environmental friendly behaviors by citizens. The end-user perspective is shortly summarized below.

The objective of Green Route is to help the final user to determine the best route to follow to reach a destination, considering the user profile (such as health conditions), and the user preferences, such as transport type. Green Route proposes the ideal route for the user, avoiding routes with high levels of pollution, traffic jam or pollen, etc., allowing for instance, to obtain the preferred routes for people with respiratory diseases.

## 2.3 Smart Security

Taking care of events happening in a video surveillance area is a complex task. The security system cameras are sending visual information to the main system online and the end-user (commonly a security guard) could not be able to pay attention to all the visual information.

The Smart Security application aims to support the security guard to prevent risk situations and consequently improve the quality of life of the people who live in the surveillance area.

The Smart Security application will focus on detecting and analyzing security risk such as, theft, access controls, people detection, fights, crowd analysis, etc., through the combination of video cameras and mobile sensors, in both, indoor and outdoor scenarios, for instance, parking lots and buildings.

# 3 DATA MODELS FOR SMART APPLICATION SCENARIOS

Re-usable and harmonized data models are an important objective of the FIWARE initiative. This section describes the data models (new or reused) identified during the development of the smart applications in SmartSDK.

## 3.1 Data models shared across scenarios

The next sections describe the data models that are common to at least two smart scenarios.

### 3.1.1 Reused data models

This section describes the models that are used as defined by the current data model version in FIWARE. Thus, as the data model design is an incremental process we include the current definition (as a Snapshot) of the data model reused. The *Example of use* subsection shows how a data model is used by one application. If an extension is proposed to these data models, the one is identified as a requirement (in bold) in the application's *Example of use* subsection.

#### 3.1.1.1 Device/DeviceModel

This model is used by the three SmartSDK project's application: Smart City, Smart Security, and Smart Health. An extension is required by Smart Security and SmartHealth applications.

➔ Device: represents an apparatus (hardware + software + firmware) intended to accomplish a particular task. It is a tangible object which contains some logic and is producer and/or consumer of data [3][4].

**Data Model:**

- id: Unique identifier.
- type: Entity type. It must be equal to Device.
- category: See attribute category from DeviceModel. Optional but recommended to optimize queries.
- controlledProperty: See attribute controlledProperty from DeviceModel. Optional but recommended to optimize queries.
- controlledAsset: The asset(s) (building, object, etc.) controlled by the device.
    - Attribute type: List of Text or Reference(s) to another entity.
    - Optional
- mnc: This property identifies the Mobile Network Code (MNC) of the network the device is attached to. The MNC is used in combination with a Mobile Country Code (MCC) (also known as a "MCC / MNC tuple") to uniquely identify a mobile phone operator/carrier using the GSM, CDMA, iDEN, TETRA and 3G / 4G public land mobile networks and some satellite mobile networks.
    - Attribute type: Text
    - Optional
- mcc: Mobile Country Code - This property identifies univoquely the country of the mobile network the device is attached to.
    - Attribute type: Text
    - Optional
- macAddress: The MAC address of the device.

    - Attribute type: Text
    - Optional
- ipAddress: The IP address of the device. It can be a comma separated list of values if the device has more than one IP address.
    - Attribute type: Text
    - Optional
- supportedProtocol: See attribute supportedProtocol from DeviceModel. Needed if due to a software update new protocols are supported. Otherwise it is better to convey it at DeviceModel level.
- configuration: Device's technical configuration. This attribute is intended to be a dictionary of properties which capture parameters which have to do with the configuration of a device (timeouts, reporting periods, etc.) and which are not currently covered by the standard attributes defined by this model.
    - Attribute type: StructuredValue
    - Attribute metadata:
        - dateModified: It captures the last modification timestamp of this attribute.
        - Type: DateTime
    - Optional
- location: Location of this device represented by a GeoJSON geometry of type point.
    - Attribute type: geo:json.
    - Normative References: https://tools.ietf.org/html/rfc7946
    - Optional.
- name: A mnemonic name given to the device.
    - Normative References: name
    - Optional
- description: Device's description.
    - Normative References: description
    - Optional
- dateInstalled: A timestamp which denotes when the device was installed (if it requires installation).
    - Attribute type: DateTime
    - Optional
- dateFirstUsed: A timestamp which denotes when the device was first used.
    - Attribute type: DateTime
    - Optional
- dateManufactured: A timestamp which denotes when the device was manufactured.
    - Attribute type: DateTime
    - Optional
- hardwareVersion: The hardware version of this device.
    - Attribute type: Text
    - Optional
- softwareVersion: The software version of this device.
    - Attribute type: Text
    - Optional
- firmwareVersion: The firmware version of this device.
    - Attribute type: Text
    - Optional
- osVersion: The version of the host operating system device.
    - Attribute type: Text
    - Optional

- dateLastCalibration: A timestamp which denotes when the last calibration of the device happened.
  - Attribute type: DateTime
  - Optional
- serialNumber: The serial number assigned by the manufacturer.
  - Normative References: https://schema.org/serialNumber
  - Mandatory
- provider: The provider of the device.
  - Normative References: https://schema.org/provider
  - Optional
- refDeviceModel: The device's model.
  - Attribute type: Reference to an entity of type DeviceModel.
  - Optional
- batteryLevel: Device's battery level. It must be equal to 1.0 when battery is full. 0.0 when battery is empty. null when cannot be determined.
  - Type: Number
  - Allowed values: Interval [0,1]
  - Attribute metadata:
    - timestamp: Timestamp when the last update of the attribute happened. This value can also appear as a FIWARE TimeInstant
    - Type: DateTime
  - Optional
- deviceState: State of this device from an operational point of view. Its value can be vendor dependent.
  - Type: Text
  - Attribute metadata:
    - timestamp: Timestamp when the last update of the attribute happened. This value can also appear as a FIWARE TimeInstant
    - Type: DateTime
  - Optional
- dateLastValueReported: A timestamp which denotes the last time when the device successfully reported data to the cloud.
  - Attribute type: DateTime
  - Optional
- value: A observed or reported value. For actuator devices, it is an attribute that allows a controlling application to change the actuation setting. For instance, a switch device which is currently *on* can report a value “on”of type Text. Obviously, in order to toggle the referred switch, this attribute value will have to be changed to “off”.
  - Attribute type: Any type, depending on the device. Usually Text or QuantitativeValue.
  - Attribute metadata:
    - timestamp: Timestamp when the last update of the attribute happened. This value can also appear as a FIWARE TimeInstant
    - Type: DateTime
  - Optional
- dateModified: Last update timestamp of this entity.
  - Attribute type: DateTime
  - Optional
- dateCreated: Entity's creation timestamp.
  - Attribute type: DateTime
  - Optional

➔ DeviceModel: represents a model of device, capturing its static properties [2. 3].

- id: Unique identifier.
- type: Entity type. It must be equal to DeviceModel.
- category: Device's category(ies).
    - Attribute type: List of Text
    - Allowed values, one of the following or any other meaningful to the application:
        - sensor: A device that detects and responds to events or changes in the physical environment such as light, motion, or temperature changes. https://w3id.org/saref#Sensor.
        - actuator: A device responsible for moving or controlling a mechanism or system. https://w3id.org/saref#Actuator.
        - meter: A device built to accurately detect and display a quantity in a form readable by a human being. Partially defined by SAREF.
        - HVAC: Heating, Ventilation and Air Conditioning (HVAC) device that provides indoor environmental comfort. https://w3id.org/saref#HVAC.
        - network: A device used to connect other devices in a network, such as hub, switch or router in a LAN or Sensor network. (https://w3id.org/saref#Network.
        - multimedia: A device designed to display, store, record or play multimedia content such as audio, images, animation, video. https://w3id.org/saref#Multimedia
    - Mandatory
- deviceClass: Class of constrained device as specified by RFC 7228. If the device is not a constrained device this property can be left as null or undefined.
    - Attribute type: Text
    - Normative References: RFC7228
    - Allowed values: (C0, C1, C2)
    - Optional
- controlledProperty: Anything that can be sensed, measured or controlled by.
    - Attribute type: List of Text
    - Allowed values: (some of this values are defined as instances of the class Property in SAREF)
        - (temperature, humidity, light, motion, fillingLevel, occupancy, power, pressure, smoke, energy, airPollution, noiseLevel, weatherConditions, precipitation, windSpeed, windDirection, barometricPressure, solarRadiation, depth, pH, pressure, conductivity, conductance, tss, tds, turbidity, salinity, orp, cdom, waterPollution, location, speed, heading, weight, waterConsumption, gasComsumption, electricityConsumption)
    - Mandatory
- function: The functionality necessary to accomplish the task for which a Device is designed. A device can be designed to perform more than one function. Defined by SAREF.
    - Attribute type: List of Text
    - Allowed values: (levelControl, sensing, onOff, openClose, metering, eventNotification), from SAREF.
    - Optional
- supportedProtocol: Supported protocol(s) or networks.
    - Attribute type: List of Text.
    - Allowed values: (ul20, mqtt, lwm2m, http, websocket, onem2m, sigfox, lora, nb-iot,

ec-gsm-iot, lte-m, cat-m, 3g, grps) or any other value meaningful for an application.
    - Optional
- supportedUnits: Units of measurement supported by the device.
    - Attribute type: List of Text.
    - Allowed values: The unit code (text) of measurement given using the UN/CEFACT Common Code (max. 3 characters).
    - Optional
- energyLimitationClass: Device's class of energy limitation as per RFC 7228.
    - Attribute type: Text
    - Normative References: RFC7228
    - Allowed values: (E0, E1, E2, E9)
    - Optional
- brandName: Device's brand name.
    - Attribute type: Text
    - See also: https://schema.org/brand
    - Mandatory
- modelName: Device's model name.
    - Attribute type: Text
    - See also: https://schema.org/model
    - Mandatory
- manufacturerName: Device's manufacturer name.
    - Attribute type: Text
    - See also: https://schema.org/model
    - Mandatory
- name: Name given to this device model.
    - Normative References: https://schema.org/name
    - Mandatory
- description: Device's description
    - Normative References: description
    - Optional
- documentation: A link to device's documentation.
    - Attribute type: URL
    - Optional
- image: A link to an image depicting the concerned device.
    - Normative References: https://schema.org/image
    - Optional
- dateModified: Last update timestamp of this entity.
    - Attribute type: DateTime
    - Optional
- dateCreated: Entity's creation timestamp.
    - Attribute type: DateTime
    - Optional

The next examples show the instantiation of Device/DeviceModel by these applications:

➔ Camera Device in Smart Security.

| **Device** | **DeviceModel** |
|---|---|
| { "id": "device-1234", "type": "Device", "category": ["multimedia"], "controlledAsset": ["Building8"], "name": "BuildingCamera1", "ipAddress": "192.14.56.78", "serialNumer": "9845A", "refDeviceModel": "cameraModel-345", "deviceState": "ok", "configuration": { "encodingFormat": "mpeg4 ", "minVideoFrameWidth": "300 ", … }, "softwareVersion": "5.4.49", "dateFirstUsed": "2014-09-11", "dateLastValueReport": "2016-04-05" } | { "id": "cameraModel-345", "type": "DeviceModel", "category": ["multimedia"], "supportedProtocol": ["rtsp"] "brandName": "vicon", "modelName": "XWE23", "manufacturerName": "Device Inc", "documentation": "http://www.camera.html", "dateCreated":"2014-09-11" } |

➔ refDevice property in Smart City (see Alert Data Model).

```
{
  "id": "Alert:1",
  "type": "Alert",
  "alertType": "Weather condition",
  "eventObserved": "Heat alert",
  "location": {
    "type": "Point",
    "coordinates": [-3.712247222222222, 40.423852777777775]
  },
  "dateTime": "2017-01-02T09:25:55.00Z",
  "description": "41–54 °C Heat cramps and heat exhaustion are likely; heat
stroke is probable with continued activity.",
  "refuser": "…",
  "refDevice": "Device1"
}
```

➔ Smartphone Device in Smart Health and Smart Security

The Smart Health and Smart Security scenarios proposed to extend the *Device* data model by integrating a new attribute (i.e consistOf) used to connect a device's entity with other entities of same type. The extension relies on the need of describing attributes values from each sensor component embedded into a Device. Thus, a Device instantiation allows to create a compound device with internal self-contained devices (as a smartphone).

Also, an inclusion of two specific devices type as part of the attribute list *category* are required.

**Requirements:**

- consistsOf: Reference to other entity.
  - Attribute type: List of references to an entity of type Device.
  - Optional.

  It is worth to mention that this attribute is included in both SAREF and oneM2M ontologies, on which FIWARE Device data models is based.
- medicalDevice: A medical device that is especially targeted at acute and continuing health data, such as patient monitors, ventilators, infusion pumps, ECG devices, etc. https://www.iso.org/obp/ui/#iso:std:iso-ieee:11073:en.
  - metadata:
    - Allowed value: (bloodPressureMonitor, glucoseMeter, pulseOximeter, weighingScale, thermometer). http://standards.ieee.org/news/2013/ieee_11073_medical-device_communication.html
- smartphone: A specific product line (such as iPhone) that performs many of the functions of a computer, typically having a touchscreen interface, Internet access, and an operating system capable of running downloaded apps. https://schema.org/ContactPoint

| **Device (Smartphone)** | **DeviceModel** |
|---|---|
| <pre>{<br>"id": "sensor-345A",<br>"type": "Device",<br>"category": "smartphone",<br>"osVersion": "Android 4.0",<br>"softwareVersion": "MA-Test 1.6",<br>"hardwareVersion": "GP-P9872",<br>"firmwareVersion": "SM-A310F",<br>"consistOf":[<br>"sensor-9845A",<br>"sensor-9845B",<br>"sensor-9845C"<br>],<br>"refDeviceModel": "myDevice-345",<br>"dateCreated": "2016-08-22T10:18:16Z"<br>}</pre> | <pre>{<br>"id": "myDevice-345",<br>"type": "DeviceModel",<br>"category": "smartphone",<br>"brandName": "mySensor",<br>"modelName": "S4Container 345",<br>"manufacturerName": "mySensor Inc.",<br>"dateCreated": "2016-08-22T10:18:16Z",<br>}</pre> |

**Device (Accelerometer)**


```
{
  “id”: “sensor-9845A”,
  “type”: “Device”,
  “category”: “sensor”,
  “function”: [“sensing”],
  “controlledProperty”: [“accelerometer”],
  “hardwareVersion”: “SMT-P9872”,
  “firmwareVersion”: “SMT-A310F”,
  “value”: “-69.895,72.0493,4.90137,2017-01-18T20:45:43.765Z-0800 -
69.844,72.0726,4.85817,2017-01-18T20:45:43.799Z-0800...”,
  “configuration”: {
    “data”: {
   “format”: “csv”
    },
    “sampleRate”: {
   “value”: “60”,
   “type”: “hz”
    }
  },
  “dateCreated”: “2016-08-22T10:20:16Z”
}
```


Smartphone Device in Smart Security.

**Device (Smartphone)**


```
{
  “id”: “smartphone-345”,
  “type”: “Device”,
  “category”: “smartphone”,
  “osVersion”: “Android 4.0”,
  “softwareVersion”: “MA-Test 1.6”,
  “hardwareVersion”: “GP-P9872”,
  “firmwareVersion”: “SM-A310F”,
  “consistOf”:[
    “accelerometer-smartphone-345”,
    “gyroscope-smartphone-345”
  ],
  “location”: [18.876567, -99.63876],
  “refDeviceModel”: “deviceModel-
smartphone-345”,
  “dateCreated”: “2016-08-
22T10:18:16Z”
}
```


**DeviceModel**


```
{
  “id”: “deviceModel-smartphone-
345”,
  “type”: “DeviceModel”,
  “category”: “smartphone”,
  “brandName”: “mySensor”,
  “modelName”: “S4Container 345”,
  “manufacturerName”: “mySensor
Inc.”,
  “dateCreated”: “2016-08-
22T10:18:16Z”,
}
```

| Device (Accelerometer) | Device (Gyroscope) |
|---|---|
| {<br>"id": "accelerometer-smartphone-345",<br>"type": "Device",<br>"category": "sensor",<br>"function": ["sensing"],<br>"controlledProperty": ["accelerometer"],<br>"hardwareVersion": "SMT-P9872",<br>"value": "-3.895\|2.0493\|9.87\|2017-01-18T20:45:43.765Z-0800",<br>"dateCreated": "2016-08-22T10:20:16Z"<br>} | {<br>"id": "gyroscope-smartphone-345",<br>"type": "Device",<br>"category": "sensor",<br>"function": ["sensing"],<br>"controlledProperty": ["gyroscope"],<br>"hardwareVersion": "SMT-P5672",<br>"value": "45.895\|7.0493\|12.901\|2017-01-18T20:45:43.765Z-0800",<br>"dateCreated": "2016-08-22T10:20:16Z"<br>} |

#### 3.1.1.2 Vehicle/VehicleModel

These models belong to the Transportation Harmonized data models.

➔ Vehicle: represents a vehicle with all its individual characteristics [3][4].

**Data Model:**

- id: Entity's unique identifier.
- type: Entity type. It must be equal to Vehicle.
- name: Name given to this vehicle.
  - Normative References: https://schema.org/name
  - Optional
- description: Vehicle description.
  - Normative References: https://schema.org/description
  - Optional
- vehicleType: Type of vehicle from the point of view of its structural characteristics. This is different than the vehicle category (see below).
  - Attribute type: Text
  - Allowed Values: The following values defined by *VehicleTypeEnum* and *VehicleTypeEnum2*, DATEX 2 version 2.3:
    - (agriculturalVehicle, bicycle, bus, minibus, car, caravan, tram, tanker, carWithCaravan, carWithTrailer, lorry, moped, tanker, motorcycle, motorcycleWithSideCar, motorscooter, trailer, van, caravan, constructionOrMaintenanceVehicle)
    - (trolley, binTrolley, sweepingMachine, cleaningTrolley)
  - Mandatory
- category: Vehicle category(ies) from an external point of view. This is different than the vehicle type (car, lorry, etc.) represented by the vehicleType property.
  - Attribute type: List of Text
  - Allowed values:
    - (public, private, municipalServices, specialUsage).
    - (tracked, nonTracked). Tracked vehicles are those vehicles which position is

permanently tracked by a remote system.
        - Or any other needed by an application They incorporate a GPS receiver together with a network connection to periodically update a reported position (location, speed, heading ...).
    - Mandatory
- location: Vehicle's last known location represented by a GeoJSON Point. Such point may contain the vehicle's *altitude* as the third component of the coordinates array.
    - Attribute type: geo:json.
    - Normative References: https://tools.ietf.org/html/rfc7946
    - Attribute metadata:
        - timestamp: Timestamp which captures when the vehicle was at that location. This value can also appear as a FIWARE TimeInstant
        - Type: DateTime or ISO8601 (legacy).
        - Mandatory
    - Mandatory only if category contains tracked.
- previousLocation: Vehicle's previous location represented by a GeoJSON Point. Such point may contain the previous vehicle's *altitude* as the third component of thecoordinates array.
    - Attribute type: geo:json.
    - Normative References: https://tools.ietf.org/html/rfc7946
    - Attribute metadata:
        - timestamp: Timestamp which captures when the vehicle was at that location.
            - Type: DateTime
            - Mandatory
    - Optional
- speed: Denotes the magnitude of the horizontal component of the vehicle's current velocity and is specified in Kilometers per Hour. If provided, the value of the speed attribute must be a non-negative real number. null *MAY* be used if speed is transiently unknown for some reason.
    - Attribute type: Number
    - Default unit: Kilometers per hour
    - Attribute metadata:
        - timestamp: Timestamp which captures when the vehicle was moving at that speed. This value can also appear as a FIWARE TimeInstant
            - Type: DateTime or ISO8601 (legacy).
            - Mandatory
    - Mandatory only if category contains tracked.
- heading: Denotes the direction of travel of the vehicle and is specified in decimal degrees, where $0° \leq \text{heading} < 360°$, counting clockwise relative to the true north. If the vehicle is stationary (i.e. the value of the speed attribute is 0), then the value of the heading attribute must be equal to null. null *MAY* be used if heading is transiently unknown for some reason.
    - Attribute type: Number
    - Attribute metadata:
        - timestamp: Timestamp which captures when the vehicle was heading towards such direction. This value can also appear as a FIWARE TimeInstant
        - Type: DateTime or ISO8601 (legacy).
        - Mandatory
    - Mandatory only if category contains tracked.
- cargoWeight: Current weight of the vehicle's cargo.
    - Attribute type: Number
    - Attribute metadata:
        - timestamp: Timestamp associated to this measurement. This value can also

appear as a FIWARE TimeInstant
- Type: DateTime or ISO8601 (legacy).
- Mandatory

- Default unit: Kilograms
- Optional

- vehicleIdentificationNumber: The Vehicle Identification Number (VIN) is a unique serial number used by the automotive industry to identify individual motor vehicles.
  - Normative References: https://schema.org/vehicleIdentificationNumber
  - Mandatory if vehiclePlateIdentifier is not defined.
- vehiclePlateIdentifier: An identifier or code displayed on a vehicle registration plate attached to the vehicle used for official identification purposes. The registration identifier is numeric or alphanumeric and is unique within the issuing authority's region.
  - Normative References: DATEX II vehicleRegistrationPlateIdentifier
  - Attribute Type: Text
  - Mandatory if vehicleIdentificationNumber is not defined.
- dateVehicleFirstRegistered: The date of the first registration of the vehicle with the respective public authorities.
  - Normative References: https://schema.org/dateVehicleFirstRegistered
  - Optional
- dateFirstUsed: Timestamp which denotes when the vehicle was first used.
  - Attribute type: DateTime
  - Optional
- purchaseDate: The date the item e.g. vehicle was purchased by the current owner.
  - Normative References: https://schema.org/purchaseDate
  - Optional
- mileageFromOdometer: The total distance travelled by the particular vehicle since its initial production, as read from its odometer.
  - Normative References: https://schema.org/mileageFromOdometer
  - Attribute metadata:
    - timestamp: Timestamp associated to this measurement. This value can also appear as a FIWARE TimeInstant
      - Type: DateTime or ISO8601 (legacy).
      - Mandatory
  - Optional
- vehicleConfiguration: A short text indicating the configuration of the vehicle, e.g. '5dr hatchback ST 2.5 MT 225 hp' or 'limited edition'.
  - Normative References: https://schema.org/vehicleConfiguration
  - Optional
- color: Vehicle's color.
  - Normative References: https://schema.org/color
  - Optional
- owner: Vehicle's owner.
  - Attribute Type: https://schema.org/Person or https://schema.org/Organization
  - Optional
- feature: Feature(s) incorporated by the vehicle.
  - Attribute type: List of Text
  - Allowed values: (gps, airbag, overspeed, abs, wifi, backCamera, proximitySensor, disabledRamp, alarm, internetConnection) or any other needed by the application.
    - In order to represent multiple instances of a feature it can be used the following syntax: “<feature>,<occurences>“. For example, a car with 4

airbags will be represented by “airbag,4”.
    - Optional
- serviceProvided: Service(s) the vehicle is capable of providing or it is assigned to.
    - Attribute type: List of Text
    - Allowed values: (garbageCollection, parksAndGardens, construction, streetLighting, roadSignalling, cargoTransport, urbanTransit, maintenance, streetCleaning, wasteContainerCleaning, auxiliaryServices goodsSelling, fairground, specialTransport) or any other value needed by an specific application.
    - Optional
- vehicleSpecialUsage: Indicates whether the vehicle is been used for special purposes, like commercial rental, driving school, or as a taxi. The legislation in many countries requires this information to be revealed when offering a car for sale.
    - Normative References: https://auto.schema.org/vehicleSpecialUsage
    - Allowed values: (taxi, ambulance, police, fireBrigade, schoolTransportation, military)
    - Optional
- refVehicleModel: Vehicle's model.
    - Attribute type: Reference to a VehicleModel entity.
    - Optional
- areaServed: Higher level area served by this vehicle. It can be used to group vehicles per responsible, district, neighbourhood, etc.
    - Attribute type: Text
    - Optional
- serviceStatus: Vehicle status (from the point of view of the service provided, so it could not apply to private vehicles).
    - Allowed values:
        - parked: Vehicle is parked and not providing any service at the moment.
        - onRoute: Vehicle is performing a mission. A comma-separated modifier(s) can be added to indicate what mission is currently delivering the vehicle. For instance “onRoute, garbageCollection” can be used to denote that the vehicle is on route and in a garbage collection mission.
        - broken: Vehicle is suffering a temporary breakdown.
        - outOfService: Vehicle is on the road but not performing any mission, probably going to its parking area.
    - Attribute type: Text
    - Attribute metadata:
        - timestamp: Timestamp which reflects when the referred service status was captured.
            - Type: DateTime
            - Mandatory
    - Optional
- dateModified: Last update timestamp of this entity.
    - Attribute type: DateTime
    - Optional
- dateCreated: Creation timestamp of this entity.
    - Attribute type: DateTime
    - Optional

➔ VehicleModel: represents a model of vehicle, capturing its static properties such as dimensions, materials or features [3][4].

**Data Model:**

- id: Entity's unique identifier.
- type: Entity type. It must be equal to VehicleModel.
- name: Name given to this vehicle model.
  - Normative References: https://schema.org/name
  - Mandatory
- description: Vehicle model description.
  - Normative References: https://schema.org/description
  - Optional
- vehicleType: Type of vehicle from the point of view of its structural characteristics.
  - See definition at Vehicle.
  - Mandatory
- brandName: Vehicle's brand name.
  - Attribute type: Text
  - See also: https://schema.org/brand
  - Mandatory
- modelName: Vehicle's model name.
  - Attribute type: Text
  - See also: https://schema.org/model
  - Mandatory
- manufacturerName: Vehicle's manufacturer name.
  - Attribute type: Text
  - See also: https://schema.org/model
  - Mandatory
- vehicleModelDate: The release date of a vehicle model (often used to differentiate versions of the same make and model).
  - Normative References: https://schema.org/vehicleModelDate
  - Optional
- cargoVolume: The available volume for cargo or luggage. For automobiles, this is usually the trunk volume.
  - Normative References: https://schema.org/cargoVolume
  - Default Unit: Liters
  - Optional
  - Note: If only a single value is provided (type Number) it will refer to the maximum volume.
- fuelType: The type of fuel suitable for the engine or engines of the vehicle.
  - Normative References: https://schema.org/fuelType
  - Allowed values: one Of (gasoline, petrol(unleaded), petrol(leaded), petrol, diesel, electric, hydrogen, lpg, autogas, cng, biodiesel ethanol, hybrid electric/petrol, hybrid electric/diesel, other)
  - Optional
- fuelConsumption: The amount of fuel consumed for traveling a particular distance or temporal duration with the given vehicle (e.g. liters per 100 km).
  - Normative References: https://schema.org/fuelConsumption
  - Default unit: liters per 100 kilometer.
  - Optional
- height: Vehicle's height.
  - Normative References: https://schema.org/height
  - Optional
- width: Vehicle's width.

- Normative References: https://schema.org/width
- Optional

- depth: Vehicle's depth.
  - Normative References: https://schema.org/width
  - Optional
- weight: Vehicle's weight.
  - Normative References: https://schema.org/weight
  - Optional
- vehicleEngine: Information about the engine or engines of the vehicle.
  - Normative References: https://schema.org/vehicleEngine
  - Optional
  - Note: This property could be at vehicle level as well.
- url: URL which provides a description of this vehicle model.
  - Normative References: https://schema.org/url
  - Optional
- image: Image which depicts this vehicle model.
  - Normative References: https://schema.org/image
  - Optional
- dateModified: Last update timestamp of this entity.
  - Attribute type: DateTime
  - Optional
- dateCreated: Creation timestamp of this entity.
  - Attribute type: DateTime
  - Optional

The models are used in the Smart Security and Smart City Applications.

➔ Use in Smart Security

| **Vehicle** | **VehicleModel** |
|---|---|
| <pre>{<br>"id": "vehicle-1234",<br>"type": "Vehicle",<br>"category": ["private"],<br>"vehicleType": "car",<br>"name": "VehicleID1"<br>"location":{<br>"type": "point",<br>"coordinates": [-3.1644, 40.62234]<br>},<br>"speed": "50",<br>"color":"red",<br>"refVehicleModel":"car-345",<br>"dateModified": "2016-04-05",<br>"dateCreated": "2016-01-05"<br>}</pre> | <pre>{<br>"id": "car-345",<br>"type": "VehicleModel",<br>"description": "A standard car."<br>"vehicleType": "car",<br>"brandName": "Nissan",<br>"modelName": "XWE23",<br>"fuelType":"diesel"<br>}</pre> |

➔ Use in Smart City

```
{
  “id”: “vehicle:private:1”,
  “type”: “Vehicle”,
  “vehicleType”: “car”,
  “brandName”: “Fiat”,
  “defaultCar”: “yes”,
  “modelName”: “500”,
  “vehicleModelDate”: “2017”,
  “fuelType”: “diesel”,
  “fuelConsumption”: “liters per 100 km”,
  “vehiclePlateIdentifier”: “3456ABC”,
  “dateCreated”: “2017-01-02T09:25:55.00Z”,
  “dateModified”: “2017-02-02T011:13:55.00Z”
}
```

### 3.1.2 New data models

This section describes all those data models that were designed because of the new requirements discovered in the applications development were not satisfied by the data models available on FIWARE. The structure of the data models is described in detail.

#### 3.1.2.1 Alert

This new data model was first proposed by the Smart City scenario, but after discussion the Smart Security and Smart HealthCare scenarios decided to merge their original Notifications data model into the Alert Data Model proposed by the Smart City scenario.

The entity models an alert generated by a user in a given location. These models could be used to send alerts related to traffic jam, accidents, weather conditions, high level of pollutants and so on. The objective of this model is to define all the data that will send to Orion Context Broker. Then, an application could subscribe to OCB and to take this information and to generate notifications for a user or trigger other actions.

Alerts are context data generated by final users (humans) or devices, such as smartphones or software apps. Also, a Smart Spot or Cloudino could send alerts if these technologies can incorporate some type of processing device that could process data and to determine if the data contains some data to be communicated to users. For instance, a Smart Spot could produce an alert about high levels of Ozone and in the same way the Cloudino could send alerts of strong rain to users.

An alert is generated by a specific situation that trigger an alert. The main features of an alert is that it is not predictable and it is not a recurrent data. That means that an alert could be an accident or a high level of pollutants measure, additionally it could be the fall down of a patient or a car driving in the opposite direction.

The alerts generated could be when a user or device detects an irregular situation sends an alert to specific data model. Some examples of context data are: type of alert (traffic, suspicious activities, and pollution, etc.), severity, location and so on.

**Data model:**

- id: Entity's unique identifier.
- type: Entity type. It must be equal to Alert.
- alertType: Define the type of alert (Traffic jam, accidents, weather conditions, high level of pollutants)
  - Attribute type: text
  - Normative References: https://schema.org/Event
  - Mandatory
- eventObserved: Describe the subtype of alerts (Alert: Traffic jam, Subtypes of alerts: Standstill Traffic Jam, Heavy Traffic Jam, Car Stopped On Road, and so on).
  - Attribute type: text
  - Normative References: https://schema.org/subEvent
  - Mandatory
- location: Location of alert represented by a GeoJSON geometry.
  - Attribute type: geo:json.
  - Normative References: https://tools.ietf.org/html/rfc7946
  - Mandatory
- address: Civic address of Alert
  - Normative References: https://schema.org/address
  - Mandatory if location is not present.
- dateTime: The date and time of this alert in ISO8601 UTCformat. It can be represented by an specific time instant or by an ISO8601 interval.
  - Attribute type: DateTime or an ISO8601 interval represented as Text.
  - Mandatory
- description: A description of alert (Traffic jam in Paseo de la Reforma. Emergency services at place).
  - Attribute type: text
  - Normative References: https://schema.org/description
  - Optional
- refUser: reference to the User that generated an alert. Within FIWARE, this reference could point to the end of the FIWARE Identity Manager, where FIWARE user profiles are stored.
  - Attribute type: string
  - Mandatory if refDevice is not present.
- refDevice: reference to device that generate an alert.
  - Attribute type: string
  - Mandatory if refUser is not present.

➔ Use in Smart city.

```
{
  “id”: “Alert:1”,
  “type”: “Alert”,
  “alertType”: “Traffic jam”,
  “eventObserved”: “Standstill traffic jam”,
  “location”: {
    “type”: “feature”,
    “coordinates”: [-3.712247222222222, 40.423852777777775]
  },
```

```
  "dateTime": "2017-01-02T09:25:55.00Z",
  "description": "There are emergency services at place",
  "refUser": "https://account.lab.fiware.org/users/8"
}
```

➔ Use in Smart Security and Smart Health.

The Smart Health and Smart Security scenarios proposed to extend the *Alert* data model designed by Smart City scenario. The proposal is to add **severity** and **severityValue** attributes taking into account our need to classify different types of alerts for different levels of attention by creating rules in the OCB for different subscriptors.

**Requirements:**

- **severity** is an attribute intended to store the scale defined for their application. The designer of the application will be responsible for designing their scale according to their application's needs. This attribute is important to give context to the **severityValue.**
- **severityValue** is a number attribute marking a value within the range defined in **severity,** intended for faster filtering in the OCB.

For example, in Smart Security context, the severity dictionary could be defined as *{"informational", "low", "medium", "high", "critical"}*, but this dictionary can be customized to any application's requirements. In our case, "informational" would be used for the internal system messages between cloud servers for data that needs further processing and all other alerts will be used to filter information to inform different security agents depending on their security roles.

```
{
  "id": "VisualEvent123",
  "type":"Alert",
  "alertType":"Accidents",
  "eventObserved":"Car Accident",
  "location":{
    "type":"feature",
    "coordinates":[-3.712247222222222,40.423852777777775]
  },
  "dateTime": "2017-04-25T09:25:55.00Z",
  "description": "Car collision",
  "refuser":"…",
  "refDevice": "Camera1234",
  "data": {
    "videoURL": "www.smartsecurity.com/video123.mp4",
    "initialFrame": 80,
    "finalFrame": 120
  },
  "severity": ["informational", "low", "medium", "high", "critical"],
  "severityValue": 4
}
```

Please, take into account that proposed attributes (**severity** and **severityValue**) will be able to model both: discrete and continuous values. In the scope of health, the attributes will allow numerical measurements within a specific interval. For instance, to report / notify that a particular patient has reached a dangerous blood pressure level that overtake a recommended threshold, as illustrated in next

example:

```
{
  “id”: “health-notification-1”,
  “type”: “Alert”,
  “alertType”: “Health report”,
  “eventObserved”: “High blood pressure”,
  “dateTime”: “2017-04-25T09:25:55.00Z”,
  “refuser”: “http://2001.23.13.1:1234/user/1”,
  “refDevice”: “Sphygmomanometer”,
  “data”: {
    “bloodPressure”: 130
  },
  “severity”: [0, 5],
  “severityValue”: 4.5
}
```

#### 3.1.2.2 Questionnaire

This data model was previously designed to cover qualitative feedback required in the health application; for example, to collect clinician observation / evaluation after a physical test is performed[1]. However, due a potential use in other domains of interest, we decided to generalize it.

The model is part of a three layer-elements: (1) Questionnaire; a root source that concentrate a collection of questions, (2) Question; a specific sentence that request an input, and (3) Answer; a specific input to reply a specific question. Where a Questionnaire entity is allowed to be connected to a number of Question entities, as a Question is allowed to be connected to a number of Answers.

The flexibility of the models allows the user to develop a questionnaire based on a predefined set of questions or the inclusion of new questions into a questionnaire.

On the other hand, restriction on the entity attributes facilitate the appropriate management of the creation of entities. For instance, the population of Answer's entities strictly depend on the pre-instanced Question; since no answer can exist without respective question.

In context of the physical test scenario, a questionnaire is being created by integrating questions related to the participant / patient performance, for instance: Did the participant/patient required assistance when performing the test? Did the participant / patient followed directions as requested? and so on. Thus, once the physical test has concluded, the mobile phone application will trigger a survey including respective questionnaire’ questions, for the observer to provide a respective answer to each question.

This model has been implemented based to cover the need of collecting qualitative feedback after controlled physical test are performed. Thus, no model has being referenced.

**Data Model:**

- id: Unique identifier.
  - Mandatory.
- type: Entity type. It must be equal to Questionnaire.

[1] https://github.com/netzahdzc/oHealth-Context/tree/master/schemas/Questionnaire

    - Mandatory.
- questionnaireType: Unique value to contextualize a test.
    - Attribute type: string.
    - Allowed values: The parameter is open to any descriptor that might bring a significant meaning.
    - Mandatory.
- refQuestion: List of questions included into the questionnaire.
    - Attribute type: Question.
    - Mandatory.
- description: A brief description regarding the purpose of the questionnaire.
    - Attribute type: string.
    - Mandatory.
- dateModified: Last entity's update timestamp.
    - Attribute type: DateTime.
    - Mandatory.

**Example of use:**

```
 {
  “id”: “fffffffff9cbbf4465f0ef30033c587-questionnaire-7118”,
  “type”: “Questionnaire”,
  “questionnaireType”: “Timed Up and Go”,
  “refQuestion”: [“fffffffff9cbbf4465f0ef30033c587-question-7118”,
“fffffffff9cbbf4465f0ef30033c587-question-7119”,
“fffffffff9cbbf4465f0ef30033c587-question-7120”],
  “description”: “Simple test used to assess a person's mobility.”,
  “dateModified”: “2017-01-18T20:45:42.697Z”
}
```

#### 3.1.2.3 Questionnaire/Question

A sentence worded or expressed to provide an inquiry, as elaborated en previous section 3.1.2.2.

**Data Model:**

- id: Unique identifier.
    - Mandatory.
- type: Entity type. It must be equal to Question.
    - Mandatory.
- refQuestionnaire: Reference to a questionnaire.
    - Attribute type: string.
    - Mandatory.
- category: A unique category value to specify the domain of the question.
    - Attribute type: string
    - Allowed value: (health). Please note that other option values can be included.
    - Mandatory.
- value: A single question written to provide specific information.
    - Attribute type: string.
    - Mandatory
- language: Language in which the questions is written.

- Attribute type: string.
- Allowed values: (eng, es, etc., and those included into the ISO: 639-4:2010).
- Mandatory.

- dateModified: Last entity's update timestamp.
  - Attribute type: DateTime.
  - Mandatory.

**Example of use:**

```
 {
  "id": "fffffffff9cbbf4465f0ef30033c587-question-7118",
  "type": "Question",
  "refQuestionnaire": "fffffffff9cbbf4465f0ef30033c587-questionnaire-
7118",
  "category": "health",
  "value": "Did the participant require physical assistance to perform the
test?",
  "language": "en",
  "dateModified": "2017-01-18T20:45:42.697Z"
}
```

#### 3.1.2.4 Questionnaire/Answer

A sentence worded or expressed to provide an inquiry, as elaborated en previous section 3.1.2.2.

**Data model:**

- id: Unique identifier.
  - Mandatory.
- type: Entity type. It must be equal to Answer.
  - Mandatory.
- refQuestion: Reference to a Question.
  - Attribute type: string.
  - Mandatory.
- refUser: Reference to the actual User sheltered by an independent service.
  - Attribute type: string.
  - Mandatory.
- answer: Information given to answer respective question.
  - Attribute type: string.
  - Mandatory.
- dateModified: Last entity's update timestamp.
  - Attribute type: DateTime.
  - Mandatory.

**Example of use:**

```
{
  “id”: “fffffffff9cbbf4465f0ef30033c587-question-7118”,
  “type”: “Answer”,
  “refQuestion”: “fffffffff9cbbf4465f0ef30033c587-question-7118”,
  “refUser”: “http://207.249.127.162:1234/users/1”,
  “answer”: “true”,
  “dateModified”: “2017-01-18T20:45:42.697Z”
}
```

## 3.2 Data Models required by scenario

### 3.2.1 Smart Health

Smart health data-model consists of a data structures designed to bring a meaningful interpretation of data by reducing its complexity through a standardized set of schemas. It aims to facilitate writing applications that can address mobile sensor data for health.

#### 3.2.1.1 Reused data models

The Smart Health scenario does not reuse any existing FIWARE data model beyond the Device and DeviceModel discussed in Section 3.1.1.1.

#### 3.2.1.2 New data models

This model has been inspired by the already accepted open source data model for health: Open Mobile Health [5], which consist on a set of tools and specifications designed to handle medical data.

##### 3.2.1.2.1 MotorPhysicalTest

This model represents the sensor data collected while the participant carried / worn a sensor during the performance of a physical test. The structure allows the inclusion of a variety of multisensory devices which provide a detailed description from each sensor embedded into a device. Thus, sensor data could be referenced to its respective source of data producer.

Although, this model does not appropriately integrate any of the Open Mobile Health schemas defined within the model, is has been inspired by the aforementioned project by following respective design principles.

**Data Model:**

- id: Entity's unique identifier which must follow a specific format (i.e., <DEVICE UNIQUE ID>-<TEST NUMBER>; without blank spaces in between).
  - Attribute type: Identifier.
  - Mandatory.
- type: Entity type. It must be equal to MotorPhysicalTest.
  - Attribute type: Text.
  - Mandatory.
- testType: Name of physical test.
  - Attribute type: Text.

    - Allowed values: One of the following of any other meaningful to the application.
        - Timed Up and Go, 30 second sit to stand test, 4-Stage Balance Test.
    - Mandatory.
- subCategoryTestType: This field helpd to specify testType value, by allowing to provide a specific subcategory if needed.
    - Attribute type: Text.
    - Allowed values: One of the following of any other meaningful to the application.
        - Side by Side, Semi-Tandem, Tandem (Full), Single-Leg Stance.
    - Optional.
- refUser: Reference to the actual User, sheltered by an independent service.
    - Attribute type: Text.
    - Mandatory.
- refDevice: Reference to the device(s) that consist on a collection of sensors.
    - Attribute type: Device.
    - Mandatory.
- configuration: Description to enrich provided information along the Device references. This attribute is intended to be a dictionary of properties which capture parameters related with the test's design.
    - Attribute type: StructuredValue.
    - Optional.
- dateTestStarted: Timestamp to denotes when the test started.
    - Attribute type: DateTime.
    - Mandatory.
- dateTestEnded: Timestamp to denotes when the test ended.
    - Attribute type: DateTime.
    - Mandatory.

**Example of use:**

```
{
  “id”: “fffffffff9cbbf4465f0ef30033c587-7118”,
  “type”: “MotorPhysicalTest”,
  “testType”: “Timed Up and Go”,
  “refUser”: “http://207.249.127.162:1234/users/1”,
  “refDevice”: [
    “device-9845A”,
    “device-9845B”,
    “device-9845C”
  ],
  “configuration”: {
    “relationship”: “device-limbs”,
    “data”:[
      {
        “device”: “device-9845A”,
        “position”: “right-leg”
      },
      {
        “device”: “device-9845B”,
```

```
        "position": "left-leg"
      },
      {
        "device": "device-9845C",
        "position": "lower-back"
      }
    ]
  },
  "dateTestStarted": "2017-01-18T20:45:58.447Z",
  "dateTestEnded": "2017-01-18T20:45:42.697Z"
}
```

### 3.2.2 Smart City

In this scenario, a set of new data models have been proposed. Moreover, we have reused the existing data model from FIWARE to send data of air quality, devices and private vehicles.

#### 3.2.2.1 Reused Data Models

The Smart City application scenario reuses the following FIWARE Data Models:

##### 3.2.2.1.1 AirQualityObserved

**Data Model [3][4]**

- id: Unique identifier.
- type: Entity type. It must be equal to AirQualityObserved.
- dateModified: Last update timestamp of this entity.
  - Attribute type: DateTime
  - Optional
- dateCreated: Entity's creation timestamp.
  - Attribute type: DateTime
  - Optional
- location: Location of the air quality observation represented by a GeoJSON geometry.
  - Attribute type: geo:json.
  - Normative References: https://tools.ietf.org/html/rfc7946
  - Mandatory if address is not defined.
- address: Civic address of the air quality observation. Sometimes it corresponds to the air quality station address.
  - Normative References: https://schema.org/address
  - Mandatory if location is not present.
- dateObserved: The date and time of this observation in ISO8601 UTCformat. It can be represented by an specific time instant or by an ISO8601 interval.
  - Attribute type: DateTime or an ISO8601 interval represented as Text.
  - Mandatory
- source: A sequence of characters giving the source of the entity data.
  - Attribute type: Text or URL
  - Optional
- refDevice: A reference to the device(s) which captured this observation.
  - Attribute type: Reference to an entity of type Device
  - Optional

- refPointOfInterest: A reference to a point of interest (usually an air quality station) associated to this observation.
  - Attribute type: Reference to an entity of type PointOfInterest
  - Optional
- measurand: An array of strings containing details (see format below) about *each air quality measurand* observed.
  - Attribute type: List of Text.
  - Allowed values: Each element of the array must be a string with the following format (comma separated list of values): <measurand>, <observedValue>, <unitcode>, <description>, where:
    - measurand: corresponds to the chemical formula (or mnemonic) of the measurand, ex. CO.
    - observedValue: corresponds to the value for the measurand as a number.
    - unitCode: The unit code (text) of measurement given using the UN/CEFACT Common Code (max. 3 characters). For instance, GP represents milligrams per cubic meter and GQ represents micrograms per cubic meter.
    - description: short description of the measurand.
    - Examples: "CO,500,GP,Carbon Monoxide" "NO,45,GQ,Nitrogen Monoxide" "NO2,69,GQ,Nitrogen Dioxide" "NOx,139,GQ,Nitrogen oxides" "SO2,11,GQ,Sulfur Dioxide"
  - Mandatory
- In order to enable a proper management of the *historical evolution* of the concentrations of the different pollutants, *for each* element described by the measurand array list there *MAY* be an attribute which name *MUST* be exactly equal to the measurand name described on the measurand array. The structure of such an attribute will be as follows:
  - Attribute name: Equal to the name of the measurand, for instance CO.
  - Attribute type: Number
  - Attribute value: Exactly equal (same unit of measurement) to the value provided in the measurand array.
  - Attribute metadata:
    - timestamp: optional timestamp for the observed value in ISO8601 format. It can be ommitted if the observation time is the same as the one captured by the dateObserved attribute at entity level.
    - Type: DateTime

**Example of use:**

```
{
  "id":"CDMX-AmbientObserved-484150020109",
  "type":"AirQualityObserved",
  "address": {
    "type": "StructuredValue",
    "value": {
      "addressCountry": "MX",
      "addressLocality": "Ciudad de México",
      "streetAddress": "Acolman"
    }
  },
  "dateObserved":{
    "type":"DataTime",
    "value":"2016-03-14T17:00:00-05:00"
  },
```

```
  "location": {
    "type": "geo:json",
    "value": "19.431768, -99.122984"
  },
  "source":{
    "type":"text",
    "value":"http://www.aire.cdmx.gob.mx/"
  },
  "temperature":{
    "type":"text",
    "value":"12.2"
  },
  "relativeHumidity":{
    "type":"text",
    "value":"0.54"
  },
  "measurand": {
    "type":"text",
    "value": [
      "CO, 500, PPM, Carbon Monoxide",
      "03, 45, PPB, Nitrogen Monoxide",
      "NO2, 69, PPB, Nitrogen Dioxide",
      "SO2, 11, PPB, Sulfur Dioxide",
      "PM10, 139, GQ, Particle Pollution"
    ]
  },
  "CO":{
    "type":"number",
    "value":"500"
  },
  "O3":{
    "type":"number",
    "value":"45"
  },
  "NO2":{
    "type":"number",
    "value":"69"
  },
  "SO2":{
    "type":"number",
    "value":"11"
  },
  "PM10":{
    "type":"number",
    "value":"139"
  }
}
```


#### 3.2.2.1.2 Public vehicle model

**Data model [3][4]:**

- id: Entity's unique identifier.
- name: Name given to this vehicle model.
    - Normative references: https://schema.org/name

    - Mandatory
- description: Vehicle's model description
    - Normative references: https://schema.org/description
    - Optional
- brandName: Vehicle's brand name.
    - Attribute type: Text
    - See also: https://schema.org/brand
    - Mandatory
- modelName: Vehicle's model name.
    - Attribute type: Text
    - See also: https://schema.org/model
    - Mandatory
- passengersTotal: The total number of passenger that the vehicle can transport (including the driver).
    - Attribute type: Number
    - Optional
- fuelType: The type of fuel suitable for the engine or engines of the vehicle.
    - Normative references: https://schema.org/fuelType
    - Allowed values: (One of the followings)
        - gasoline
        - petrol(unleaded)
        - petrol(leaded)
        - petrol
        - diesel
        - electric
        - hydrogen
        - lpg
        - autogas
        - cng
        - biodiesel
        - ethanol
        - hybrid electric/petrol
        - hybrid electric/diesel
        - other
- fuelConsumption: The amount of fuel consumed for traveling a particular distance or temporal duration with the given vehicle (e.g. liters per 100 km).
    - Normative references: https://schema.org/fuelConsumption
    - Default unit: liters per 100 kilometer
    - Optional
- height: Vehicle's height.
    - Normative references: https://schema.org/height
    - Optional
- width: Vehicle's width.
    - Normative references: https://schema.org/width
    - Optional
- depth: Vehicle's depth.
    - Normative references: https://schema.org/depth
    - Optional
- weight: Vehicle's weight.
    - Normative references: https://schema.org/weight

- Optional
- dateModified: Last update timestamp of this entity.
  - Attribute type: DateTime
  - Optional
- dateCreated: Creation timestamp of this entity.
  - Attribute type: DateTime
  - Optional

**Example of use:**

```
{
    “id”: “publicVehicleModel001”
    “name”: “vehicleName”,
    “brandName”: “vehicleBrandName”,
    “modelName”: “vehicleModelName”,
    “passengersTotal”: 65,
    “fuelType”: “diesel”
}
```

#### 3.2.2.2 New Data Models

This section details new FIWARE Data Models developed in the context of the Smart City scenario of SmartSDK.

##### 3.2.2.2.1 SmartSpot

Smart Spots are devices which provide the the technology which allows users to get access to smart points of interaction so that they can obtain extra information (infotainment, etc.), provide suggestions (suggestions mailbox, etc.) or generate new content (co-creation, etc.). The data model contains resources to configure the interaction service such as the broadcasted URL (typically shortened), the period between broadcasts, the availability of the service, transmission power depending on the area to be covered, etc.
This model was born for the necessity of adding bluetooth low energy announcement capabilities to a specific device, it was modeled in collaboration with the FIWARE Foundation who proposed adopting an inheritance relationship for the first time in a data model, thanks to this relation the Smart Spot can extend the Device data model for adding the necessary bluetooth capabilities to the device. In this kind of relationship, the extended data model derives all his fields to his child. This kind of relation has many benefits, new data models are able to be defined seamless, this models can be created only extended the existing ones providing them new capabilities and allowing to the user to create bigger catalog of specialized models, on the other hand, not all is a bed of roses, for being able to use all the benefits of this new capabilities, inheritance rules must to adopted by the current data models, now a days all the data models contains a huge number of optionals fields that use to break the Liskov substitution principle, also using inheritance in this way, huge data models can be generated, this data models can be difficult to handle by users and new developers. A good solution for this would be split the current data models in more specialized entities, these entities could contain only the necessary fields for its specialized behaviour, for sure, we would have many more entities, but all of the would be more reusable.

**Data model:**

- id: Unique identifier.
- type: Entity type. It must be equal to SmartSpot.
- announcedUrl: URL broadcasted by the device.
  - Attribute type: URL
  - Mandatory
- signalStrenght: Signal strength to adjust the announcement range.
  - Attribute type: Text
  - Allowed values: “lowest”, “medium” or “highest”.
  - Mandatory
- bluetoothChannel: Bluetooth channels where to transmit the announcement.
  - Attribute type: Text
  - Allowed values: “37”, “38”, “39”, “37,38”, “38,39”, “37,39” or “37,38,39”.
  - Mandatory
- areaCoveredRadius: Radius of the spot coverage area in meters.
  - Attribute Type: Number
  - Default unit: Meters.
  - Optional
- announcementPeriod: Period between announcements.
  - Attribute Type: Number
  - Default unit: Milliseconds.
  - Mandatory
- availability: Specifies the functionality intervals in which the announcements will be sent. The format is an structured value which must contain a subproperty per each required functionality interval, indicating when the functionality is active. If nothing specified (or null) it will mean that the functionality is always on. The syntax must be conformant with schema.org openingHours specification. For instance, a service which is only active on dayweeks will be encoded as “availability”: “Mo,Tu,We,Th,Fr,Sa 09:00-20:00”.
  - Attribute type: StructuredValue
  - Mandatory. It can be null.
- refSmartPointOfInteraction: Reference to the Smart Point of Interaction which includes this Smart Spot.
  - Attribute type: Reference to an entity of type SmartPointOfInteraction
  - Optional

**Example of use:**

```
{
  “id”: “SSPOT-F94C51A295D9”,
  “type”: “SmartSpot”,
  “announcedUrl”: “https://hpoi.info/325531235437”,
  “signalStrenght”: “high”,
  “bluetoothChannel”: “37-38-39”,
  “areaCoveredRadius”: 30,
  “announcementPeriod”: 500,
  “availability”: “Tu,Th 16:00-20:00”,
  “refSmartPointOfInteraction”: “SPOI-ES-4326”
}
```

#### 3.2.2.2.2 SmartPointOfInteraction

A Smart Point of Interaction defines a place with technology to interact with users, for example, through Beacon technology from Apple, Eddystone/Physical-Web from Google or other proximity-based interfaces. Since the interactive area could be composed by more than one device providing the technology, this model encompasses a group of SmartSpot devices.

The data model includes information regarding the area/surface covered by the technology (i.e., the area covered by Bluetooth Low Energy-based Beacon), a way to specify the functionality intervals (i.e. when interactive points are available) and the link to the multimedia resource where users will interact (i.e. Web Apps, etc.). Additionally, the data model may reference to another NGSI entity such as a Parking, a Point of Interest (POI), etc. with enriched interaction provided by this Smart Point of Interest.

**Data model:**

- id: Unique identifier.
- type: Entity type. It must be equal to SmartPointOfInteraction.
- category: Defines the type of interaction.
  - Attribute type: List of Text
  - Allowed values: information, entertainment, infotainment, co-creation or any other extended value defined by the application.
  - Mandatory
- areaCovered: Defines the area covered by the Smart Point of Interaction using geoJSON format.
  - Attribute type: geo:json.
  - Normative References: https://tools.ietf.org/html/rfc7946
  - Optional
- applicationUrl: This field specifies the real URL containing the solution or application (information, co-creation, etc) while the SmartSport 'announcedUrl' field specifies the broadcasted URL which could be this same URL or a shortened one.
  - Attribute type: URL
  - Mandatory
- availability: Specifies the time intervals in which this interactive service is available, but this is a general information while Smart Spots have their own real availability in order to allow advanced configurations. The format is an structured value which must contain a subproperty per each required functionality interval, indicating when the functionality is active. If nothing specified (or null) it will mean that the functionality is always on. The syntax must be conformant with schema.org openingHours specification. For instance, a service which is only active on dayweeks will be encoded as “availability”: “Mo,Tu,We,Th,Fr,Sa 09:00-20:00”.
  - Attribute type: StructuredValue
  - Mandatory. It can be null.
- refRelatedEntity: List of entities improved with this Smart Point of Interaction. The entity type could be any such as a “Parking”, “Point of Interest”, etc.
  - Attribute type: List of entities.
  - Optional
- refSmartSpot: References to the “Smart Spot” devices which are part of the Smart Point of Interaction.

- Attribute type: Reference to one or more entity of type SmartSpot
- Optional

**Example of use:**

```
{
  "id": "SPOI-ES-4326",
  "type": "SmartPointOfInteraction",
  "category": ["co-creation"],
  "areaCovered": {
    "type": "Polygon",
    "coordinates": [[
      [25.774, -80.190],
      [18.466, -66.118],
      [32.321, -64.757],
      [25.774, -80.190]
    ]]
  },
  "applicationUrl": "https://www.fiware.org",
  "availability": "Tu,Th 16:00-20:00",
  "refRelatedEntity": "POI-PlazaCazorla-3123",
  "refSmartSpot": [ "SSPOT-F94C58E29DD5", "SSPOT-F94C53E21DD2", "SSPOT-F94C51A295D9"]
}
```

#### 3.2.2.2.3 Service Network

The Service Network defines a set of data models with the main purpose of collecting information about the trips made by users of the public transportation services. This way, it is possible to map the user's trips and collect useful information regarding their habits when using public transportation, such as the most used stops, the most used types of transports, and other useful metrics.
The data models described below are the Trip, Stop, Route and Agency entities, which interact in a hierarchical way to map the user's trips of a service network.
These data models were defined based on the General Transit Feed Specification [6], which is a standard that defines a common format for public transportation schedules and associated geographic information, based on CSV files. It is one of the most popular and used standards for public transportation [10], with many entities already using it to describe their services. This means there is a lot of data already available, and there are many applications developed using this standard. These facts lead to the adoption of this standard as the base of the models defined here.

##### Agency

This entity models a public transport agency, including all properties which are common to multiple trip instances belonging to such model. An agency refers to a company or entity which provides a public transportation service.

**Data model:**

- id: Entity's unique identifier.
- Type: Entity type. It must be equal to Agency.
- name: Name given to this agency.
  - Normative References: https://schema.org/name

- Mandatory
- url: URL which provides information about this agency.
  - Normative references: https://schema.org/url
  - Optional
- timezone: Timezone where this agency belongs to.
  - Attribute type: Text
  - Optional

**Example of use:**

```
{
  "id": "CC",
  "type": "Agency",
  "name": "Corredores Concesionados"
}
```

### Route

This entity models a public transport route, including all properties which are common to multiple trip instances belonging to such model. A route is a way to identify a journey that is regularly made by a given transport in an agency.

**Data model:**

- id: Entity's unique identifier.
- type: Entity type. It must be equal to Route.
- short_name: Short name of a route, often a short and abstract identifier like "7", "34", or "Blue" that riders use to identify a route.
  - Normative References: https://schema.org/name
  - Mandatory
- long_name: Full name of a route, more descriptive than the route_short_name and often include the route's destination or stop
  - Normative References: https://schema.org/name
  - Mandatory
- description: Description about this route.
  - Attribute type: https://schema.org/Text
  - Optional
- type: Describes the type of transportation used on a route
  - Attribute type: Number
  - Allowed values:
    - 0: Tram, Streetcar, Light rail. Any light rail or street level system within a metropolitan area.
    - 1: Subway, Metro. Any underground rail system within a metropolitan area.
    - 2: Rail. Used for intercity or long-distance travel.
    - 3: Bus. Used for short- and long-distance bus routes.
    - 4: Ferry. Used for short- and long-distance boat service.
    - 5: Cable car. Used for street-level cable cars where the cable runs beneath the car.
    - 6: Gondola, Suspended cable car. Typically used for aerial cable cars where

the car is suspended from the cable.
        - 7: Funicular. Any rail system designed for steep inclines.
    - Mandatory
- color: Color that corresponds to the route. The color must be provided as a six-character hexadecimal number, for example, 00FFFF.
    - Attribute type: Text
    - Optional
- url: The URL of a web page about that particular route
    - Normative references: https://schema.org/url
    - Optional

**Example of use:**

```
{
  "id": "routeID",
  "type": "Route",
  "short_name": "SAUSA"
  "long_name": "Metro Tacubaya - La Valenciana"
  "type": 3
}
```

### Stop

This entity models a public transport stop, including all properties which are common to multiple trip instances belonging to such model. A stop is defined by its geographical coordinates and its type, and represents a place where a given type of public transports stops for boarding and unloading passengers.

**Data model:**

- id: Entity's unique identifier.
- type: Entity type. It must be equal to Stop.
- code: Number that uniquely identifies the stop for passengers.
    - Attribute type: Text
    - Optional
- name: Name given to this stop.
    - Normative references: https://schema.org/name
    - Mandatory
- description: Description of this stop.
    - Attribute type: Text
    - Optional
- location: Stop location represented by a GeoJSON Point.
    - Attribute type: geo_json
    - Normative References: https://tools.ietf.org/html/rfc7946
    - Mandatory
- location_type: Identifies whether this stop represents a stop or station.
    - Attribute type: Number
    - Allowed values:
        - 0: Stop. A location where passengers board or disembark from a transit

vehicle.
        - 1: Station. A physical structure or area that contains one or more stop.
    - Optional
- wheelchairBoarding: Wheelchair accessibility information for the stop.
    - Attribute type: Number
    - Allowed values:
        - 0: Indicates that there is no accessibility information for the stop.
        - 1: Indicates that at least some vehicles at this stop can be boarded by a rider in a wheelchair.
        - 2: Wheelchair boarding is not possible at this stop.
    - Optional

**Example of use:**

```
{
  "id": "stopID",
  "type": "Stop",
  "name": "M. Tacubaya",
  "location": [19.402325646816475,-99.18885111808775]
}
```

### Trip

This entity models a trip. A trip consists of one or more segments, which can belong to different routes within a type of public transport, or even different types of public transports, belonging to different agencies.

**Data model:**

- id: Entity's unique identifier.
- type: Entity type. It must be equal to Trip.
- weekdays: The weekdays that this trip refers to.
    - Attribute type: List of Text
    - Mandatory
- segments: An a array of segments containing details(see list below) about each segment that completes the trip. Each trip can have one or more segments.
    - Attribute type: List
    - id: Entity's unique identifier.
    - name: Name given to this trip segment.
        - Normative references: https://schema.org/name
        - Mandatory
    - headsign: Text that identifies the trip's segment destination to passengers.
        - Atttribute type: Text
        - Optional
    - wheelchairAccessible: Wheelchair accessibility information for the trip segment.
        - Attribute type: Number
        - Allowed values:

        - 0: Indicates that there is no accessibility information for the trip.
        - 1: Indicates that the vehicle being used on this particular trip can accommodate at least one rider in a wheelchair.
        - 2: Indicates that no riders in wheelchairs can be accommodated on this trip.
      - Optional
  - bikesAllowed: Bicycle accomodation information for this trip segment
    - Attribute type: Number
    - Allowed values:
      - 0: Indicates that there is no bike information for the trip.
      - 1: Indicates that the vehicle being used on this particular trip can accommodate at least one bicycle.
      - 2: Indicates that no bicycles are allowed on this trip.
    - Optional
  - agency: Agency the trip segment belogs to.
    - Attribute type: Reference to a Agency entity.
    - Mandatory
  - route: Route the trip segment belongs to.
    - Attribute type: Reference to a Route
    - Mandatory
  - stop_departure: Stop where the trip segment start.
    - Attribute type: Reference to a Stop
    - Mandatory
  - stop_arrival: Stop where the trip segment finish.
    - Attribute type: Reference to a Stop
    - Optional
  - departure_timestamp: Timestamp which captures when the user started the trip segment. This value can also appear as a FIWARE TimeInstant
    - Attribute type: Time or ISO8601 (legacy).
    - Mandatory
  - arrival_timestamp: Timestamp which captures when the user finished the trip segment. This value can also appear as a FIWARE TimeInstant
    - Attribute type: Time or ISO8601 (legacy).
    - Optional

**Example of use:**

```
{
  “id”: “tripID”,
  “type”: “Trip”,
  “weekdays”: [“monday”, “tuesday”],
  “segments”: [
    “segment1”: {
      “id”: “segmentID”,
      “name”: “Tacubaya - La Valenciana por Eje 3 Sur”,
      “agency”: “CC”,
      “route”: “routeID”,
      “stop_departure”: “stopID”,
      “stop_arrival”: “stopID”,
      “departure_timestamp”: “08:30:11Z”,
```

```
      "arrival_timestamp":"08:50:30Z"
    }
  ]
}
```

### Affect Transit Service

This entity models a affected transit service, including all properties which can be used to specify exactly which parts of the public transport network are affected by the alert. This data model is based on the Realtime Transit Service Alerts [7] that defines a common format for public transportation service alerts.

**Data model:**

- id: Entity's unique identifier.
- type: Entity type. It must be equal to AffectedTransitService.
- refAgency: Spcecifies the agency that will be affected. This means that all the agency network will be affected.
  - Attribute type: Reference to an Agency
  - Optional
- refRoute: Specifies the particular route that will be affected.
  - Attribute type: Reference to a Route
  - Mandatory only if no other type of entity is defined
- route_type: Specifies the type of route that will be affected. This means that all toutes of this type will be affected.
  - Attribute type: Number
  - Allowed values:
    - 0: Tram, Streetcar, Light rail. Any light rail or street level system within a metropolitan area.
    - 1: Subway, Metro. Any underground rail system within a metropolitan area.
    - 2: Rail. Used for intercity or long-distance travel.
    - 3: Bus. Used for short- and long-distance bus routes.
    - 4: Ferry. Used for short- and long-distance boat service.
    - 5: Cable car. Used for street-level cable cars where the cable runs beneath the car.
    - 6: Gondola, Suspended cable car. Typically used for aerial cable cars where the car is suspended from the cable.
    - 7: Funicular. Any rail system designed for steep inclines.
  - Mandatory only if no other type of entity is defined
- refTrip: Specifies the particular trip that will be affected
  - Attribute type: Reference to Trip
  - Mandatory only if no other type of entity is defined
- refStop: Specifies the particular stop that will be affected
  - Attribute type: Reference to Stop
  - Mandatory only if no other type of entity is defined

**Example of use:**

```
{
  "id": "entityID0",
  "type": "AffectedTransitService",
  "refAgency": "agencyID0",
  "refRoute": "routeID001"
}
```

### Transit Service Alerts from public transport

This entity models a public transport service alert model, including all properties which can be used to specify a public transport service alert whenever there is a disruption on the public transport network. This data model is based on the Realtime Transit Service Alerts [7] that defines a common format for public transportation service alerts.

**Data model:**

- id: Entity's unique identifier.
- type: Entity type. It must be equal to TransitServiceAlert.
- header: Summary of the alert.
  - Attribute type: Text
  - Mandatory
- description: A complete description of the alert.
  - Attribute type: Text
  - Mandatory
- timeRange: The time range that the alert will be active. The time range can refer to one or multiple time ranges. It's represented as an array of time intervals.
  - Attribute type: DateTime or an ISO8601 interval represented as Text.
  - Mandatory
- refAffectedTransitService: Specify exactly which parts of the network this alert affects.
  - Attribute type: List of references to entities of type AffectedTransitService
  - Mandatory
- cause: The cause of the alert. Can only exist one cause for each alert.
  - Attribute type: Text
  - Allowed Values:
    - Unknown cause
    - Other cause (not represented by any of these option)
    - Technical problem
    - Demonstration
    - Accident
    - Holiday
    - Weather
    - Maintenance
    - Construction
    - Police activity
    - Medical emergency
  - Mandatory
- effect: What effect does this problem have on the specified entity. Cal only exist one effect for each alert.

- Attribute type: Text
- Allowed values:
  - No service
  - Reduced service
  - Significant delays
  - Detour
  - Additional service
  - Modified service
  - Stop moved
  - Other effet
  - Unknown effect
- Mandatory

- dateModified: Last update timestamp of this entity.
  - Attribute type: DateTime
  - Optional
- dateCretated: Creation timestamp of this entity.
  - Attribute type: DateTime
  - Optional

**Example of use:**

```
{
  "id": "Alert0000",
  "type": " TransitServiceAlert ",
  "header": "Holiday",
  "description": "Complete description of what will happen",
  "timeRange": [
    [
      2017-04-05T08:15:30+01:00,
      2017-04-06T08:15:30+01:00
    ],
    [
      2017-04-08T08:15:30+01:00,
      2017-04-09T08:15:30+01:00
    ]
  ],
  "refAffectedTransitService": ["entityID0"],
  "cause": "Holiday",
  "effect": "Modified Service"
}
```

### 3.2.3 Smart Security

#### 3.2.3.1 Reused Data Models

Smart Security will reuse the OffStreetParking, Building and UserContext data models.

##### 3.2.3.1.1 OffStreetParking

For parking representation FIWARE data model offers several data models to encoding data. The Smart Security application will use the OffStreetParking model [3][4]. The model represents a site, off street, intended to park vehicles, managed independently and with suitable and clearly marked access

points (entrances and exits).

**Data Model:**

- id: Unique identifier.
- type: Entity type. It must be equal to OffStreetParking.
  - dateCreated: Entity's creation timestamp
  - Attribute type: DateTime
  - Optional
- dateModified: Last update timestamp of this entity
  - Attribute type: DateTime
  - Optional
- location: Geolocation of the parking site represented by a GeoJSON (Multi)Polygon or Point.
  - Attribute type: geo:json.
  - Normative References: https://tools.ietf.org/html/rfc7946
  - Mandatory if address is not defined.
- address: Registered parking site civic address.
  - Normative References: https://schema.org/address
  - Mandatory if location is not defined.
- name: Name given to the parking site.
  - Normative References: https://schema.org/name
  - Mandatory
- category: Parking site's category(ies). The purpose of this field is to allow to tag, generally speaking, off street parking entities. Particularities and detailed descriptions should be found under the corresponding specific attributes.
  - Attribute type: List of Text
  - Allowed values:
    - (public, private, publicPrivate, urbanDeterrentParking, parkingGarage, parkingLot, shortTerm, mediumTerm, longTerm, free, feeCharged, staffed, guarded, barrierAccess, gateAccess, freeAccess, forElectricalCharging, onlyResidents, onlyWithPermit, forEmployees, forVisitors, forCustomers, forStudents, forMembers, forDisabled, forResidents, underground, ground)
    - The semantics of the forxxx values is that the parking offers specific spots subject to that particular condition.
    - The semantics of the onlyxxxvalues is that the parking only allows to park on that particular condition.
    - Other application-specific
  - Mandatory
- allowedVehicleType: Vehicle type(s) allowed. The first element of this array *MUST* be the principal vehicle type allowed. Free spot numbers of other allowed vehicle types might be reported under the attribute extraSpotNumber and through specific entities of type *ParkingGroup*.
  - Attribute type: List of Text
  - Allowed Values: The following values defined by *VehicleTypeEnum*, DATEX 2 version 2.3:
    - (agriculturalVehicle, bicycle, bus, car, caravan, carWithCaravan, carWithTrailer, constructionOrMaintenanceVehicle, lorry, moped, motorcycle, motorcycleWithSideCar, motorscooter, tanker, trailer, van, anyVehicle)
  - Mandatory

- chargeType: Type(s) of charge performed by the parking site.
  - Attribute type: List of Text
  - Allowed values: Some of those defined by the DATEX II version 2.3 *ChargeTypeEnum* enumeration:
    - (flat, minimum, maximum, additionalIntervalPrice seasonTicket temporaryPrice firstIntervalPrice, annualPayment, monthlyPayment, free, other)
    - Any other application-specific
  - Mandatory
- requiredPermit: This attribute captures what permit(s) might be needed to park at this site. Semantics is that at least *one of* these permits is needed to park. When a permit is composed by more than one item (and) they can be combined with a ",". For instance "residentPermit,disabledPermit" stays that both, at the same time, a resident and a disabled permit are needed to park. If empty or null, no permit is needed.
  - Attribute type: List of Text
  - Allowed values: The following, defined by the *PermitTypeEnum* enumeration of DATEX II version 2.3.
    - oneOf (employeePermit, studentPermit, fairPermit, governmentPermit, residentPermit, specificIdentifiedVehiclePermit, visitorPermit, noPermitNeeded)
    - Any other application-specific
  - Mandatory
- occupancyDetectionType: Occupancy detection method(s).
  - Attribute type: List of Text
  - Allowed values: The following from DATEX II version 2.3 *OccupancyDetectionTypeEnum*:
    - (none, balancing, singleSpaceDetection, modelBased, manual)
  - Or any other application-specific
  - Mandatory
- acceptedPaymentMethod: Accepted payment method(s).
  - Normative references: https://schema.org/acceptedPaymentMethod
  - Optional
- description: Description about the parking site.
  - Normative References: https://schema.org/description
  - Optional
- image: A URL containing a photo of this parking site.
  - Normative References: https://schema.org/image
  - Optional
- layout: Parking layout. Gives more details to the category attribute.
  - Attribute type: Text
  - Allowed values: As per the *ParkingLayoutEnum* of DATEX II version 2.3:
    - one Of (automatedParkingGarage, surface, multiStorey, singleLevel, multiLevel, openSpace, covered, nested, field, rooftop, sheds, carports, garageBoxes, other). See also OpenStreetMap.
    - Or any other value useful for the application and not covered above.
  - Optional
- usageScenario: Usage scenario(s). Gives more details to the category attribute.
  - Attribute type: List of Text
  - Allowed values: Those defined by the enumeration *ParkingUsageScenarioEnum* of DATEX II version 2.3:

        - (truckParking, parkAndRide, parkAndCycle, parkAndWalk, kissAndRide, liftshare, carSharing, restArea, serviceArea, dropOffWithValet, dropOffMechanical, eventParking, automaticParkingGuidance, staffGuidesToSpace, vehicleLift, loadingBay, dropOff, overnightParking, other)
        - Or any other value useful for the application and not covered above.
    - Optional
- parkingMode: Parking mode(s).
    - Attribute type: List of Text
    - Allowed values: Those defined by the DATEX II version 2.3 *ParkingModeEnum* enumeration:
    - (perpendicularParking, parallelParking, echelonParking)
    - Optional
- facilities: Facilities provided by this parking site.
    - Attributes: List of Text
    - Allowed values: The following defined by the *EquipmentTypeEnum* enumeration of DATEX II version 2.3:
        - (toilet, shower, informationPoint, internetWireless, payDesk, paymentMachine, cashMachine, vendingMachine, faxMachineOrService, copyMachineOrService, safeDeposit, luggageLocker, publicPhone, elevator, dumpingStation freshWater, wasteDisposal, refuseBin, iceFreeScaffold, playground, electricChargingStation, bikeParking, tollTerminal, defibrillator, firstAidEquipment fireHose fireExtinguisher fireHydrant)
    - Any other application-specific
    - Optional
- security: Security aspects provided by this parking site.
    - Attributes: List of Text
    - Allowed values: The following, some of them, defined by *ParkingSecurityEnum* of DATEX II version 2.3:
        - (patrolled, securityStaff, externalSecurity, cctv, dog, guard24hours, lighting, floodLight, fences areaSeperatedFromSurroundings)
    - Any other application-specific
    - Optional
- highestFloor: For parking sites with multiple floor levels, highest floor.
    - Attribute type: Number
    - Allowed values: An integer number. 0 is ground level. Upper floors are positive numbers. Lower floors are negative ones.
    - Optional
- lowestFloor: For parking sites with multiple floor levels, lowest floor.
    - Attribute type: Number
    - Allowed values: An integer number.
    - Optional
- maximumParkingDuration: Maximum allowed stay at site, on a general basis, encoded as a ISO8601 duration. A null or empty value indicates an indefinite duration.
    - Attribute type: Text
    - Optional
- totalSpotNumber: The total number of spots offered by this parking site. This number can be difficult to be obtained for those parking locations on which spots are not clearly marked by lines.

    - Attribute type: Number
    - Allowed values: Any positive integer number or 0.
    - Normative references: DATEX 2 version 2.3 attribute *parkingNumberOfSpaces* of the *ParkingRecord* class.
    - Optional
- availableSpotNumber: The number of spots available (*including* all vehicle types or reserved spaces, such as those for disabled people, long term parkers and so on). This might be harder to estimate at those parking locations on which spots borders are not clearly marked by lines.
    - Attribute type: Number
    - Allowed values: A positive integer number, including 0. It must lower or equal than totalSpotNumber.
    - Metadata:
        - timestamp: Timestamp of the last attribute update
        - Type: DateTime
    - Optional
- extraSpotNumber: The number of extra spots *available*, i.e. free. This value must aggregate free spots from all groups mentioned below: A/ Those reserved for special purposes and usually require a permit. Permit details will be found at parking group level (entity of type ParkingGroup). B/ Those reserved for other vehicle types different than the principal allowed vehicle type. C/ Any other group of parking spots not subject to the general condition rules conveyed by this entity.
    - Attribute type: Number
    - Allowed values: A positive integer number, including 0.
    - Metadata:
        - timestamp: Timestamp of the last attribute update
        - Type: DateTime
    - Optional
- openingHours: Opening hours of the parking site.
    - Normative references: http://schema.org/openingHours
    - Optional
- firstAvailableFloor: Number of the floor closest to the ground which currently has available parking spots.
    - Attribute type: Number
    - Metadata:
        - timestamp: Timestamp of the last attribute update
        - Type: DateTime
    - Allowed values: Stories are numbered between -n and n, being 0 ground floor.
    - Optional
- specialLocation: If the parking site is at a special location (airport, depatment store, etc.) it conveys what is such special location.
    - Attribute type: Text
    - Allowed values: Those defined by *ParkingSpecialLocationEnum* of DATEX II version 2.3:
        - (airportTerminal, exhibitonCentre, shoppingCentre, specificFacility, trainStation, campground, themePark, ferryTerminal, vehicleOnRailTerminal, coachStation, cableCarStation, publicTransportStation, market, religiousCentre, conventionCentre, cinema, skilift, hotel, other)
    - Optional

- status: Status of the parking site.
  - Attribute type: List of Text
  - Metadata:
  - timestamp: Timestamp of the last attribute update
  - Type: DateTime
- Allowed values: The following defined by the following enumerations defined by DATEX II version 2.3:
  - *ParkingSiteStatusEnum*
  - *OpeningStatusEnum*
  - (open, closed, closedAbnormal,openingTimesInForce, full, fullAtEntrance, spacesAvailable, almostFull)
  - Or any other application-specific
  - Optional
- reservationType: Conditions for reservation.
  - Attribute type: Text
  - Allowed values: The following specified by *ReservationTypeEnum* of DATEX II version 2.3:
  - one Of (optional, mandatory, notAvailable, partly).
  - Optional
- owner: Parking site's owner.
  - Attribute type: Text
  - Optional
  - provider: Parking site service provider.
  - Normative references: https://schema.org/provider
  - Optional
- contactPoint: Parking site contact point.
  - Normative references: https://schema.org/contactPoint
  - Optional
- averageSpotWidth: The average width of parking spots.
  - Attribute type: Number
  - Default unit: Meters
  - Optional
- averageSpotLength: The average length of parking spots.
  - Attribute type: Number
  - Default unit: Meters
  - Optional
- maximumAllowedHeight: Maximum allowed height for vehicles. If there are multiple zones, it will be the minimum height of all the zones.
  - Attribute type: Number
  - Default unit: Meters
  - Optional
- maximumAllowedWidth: Maximum allowed width for vehicles. If there are multiple zones, it will be the minimum width of all the zones.
  - Attribute type: Number
  - Default unit: Meters
  - Optional
- refParkingAccess: Parking site's access point(s).
  - Attribute type: List of references to ParkingAccess
  - Optional
- refParkingGroup: Parking site's identified group(s). A group can correspond to a zone, a

complete storey, a group of spots, etc.
    - Attribute type: List of references to ParkingGroup
    - Optional
- refParkingSpot: Individual parking spots belonging to this offstreet parking site.
    - Attribute type: List of references to ParkingSpot
    - Optional
- areaServed: Area served by this parking site. Precise semantics can depend on the application or target city. For instance, it can be a neighbourhood, borough or district.
    - Attribute type: Text
    - Optional
- aggregateRating: Aggregated rating for this parking site.
    - Normative References: https://schema.org/aggregateRating
    - Optional

**Example of use:**

```
{
  “id”: “diningHallParking”,
  “type”: “OffStreetParking”,
  “name”: “DiningHallParking-2016-10-25”,
  “dateCreated”: “2016-04-05”,
  “dateModified”: “2016-05-05”,
  “description”: “Off street parking located near of the dining hall “,
  “location”:{“type”: “Polygon”,
    “coordinates”: [
      [
        [-100.0, -100.0],
        [100.0, -100.0],
        [100.0, 100.0],
        [-100.0, -100.0]
      ],
      [
        [-100.0, -100.0],
        [100.0, -100.0],
        [100.0, 100.0],
        [-100.0, -100.0]
      ]
    ]
  },
  “category”: “private”,
  “parkingMode”:”echelonParking”,
  “security”:”securityStaff”,
  “totalSpotNumber”: 50
  “averageSpotWidth”: 2.3
  “averageSpotLenfth”: 3.5
}
```

#### 3.2.3.1.2 Building

This model is intended to use to represent information about vertical smart homes, smart cities [8].

**Data Model:**

- id: Unique identifier.
- type: Entity type. It must be equal to Building.
  - dateCreated: Entity's creation timestamp
  - Attribute type: DateTime
  - Optional
- dateModified: Timestamp of of the last modification of this entity
  - Attribute type: DateTime
  - Optional
- dateCreated: Entity creation timestamp
  - Attribute type: DateTime
  - Optional
- containedInPlace:The geo:json encoded polygon of the building plot in which this building sits.
  - Attribute type: geo:json.
  - Normative References: https://tools.ietf.org/html/rfc7946
  - Optional.
- location: The geo:json encoded polygon of this building.
  - Attribute type: geo:json.
  - Normative References: https://tools.ietf.org/html/rfc7946
  - Mandatory.
- address: The building PostalAddress encoded as a Schema.org PostalAddress.
  - Normative References: https://schema.org/PostalAddress
  - Optional.
- category: List of one or more categories relevant to the building.
  - Normative references: with choices based on for example http://wiki.openstreetmap.org/wiki/Map_Features#Building.
  - Optional
- refOwner: List of references to person(s) or organization(s) owner(s) of the building.
  - Normative References: https://schema.org/Person , https://schema.org/Organization
  - Optional
- refOccupier: List of references to person(s) or organization(s) occupier(s) of the building.
  - Normative References: https://schema.org/Person , https://schema.org/Organization
  - Optional
- floorsAboveGround: The number of floors above ground level in this building.
  - Attribute type: Number
  - Optional
- floorsBelowGround: The number of floors below ground level in this building.
  - Attribute type: Number
  - Optional
- description: An optional description of the entity
  - Attribute type: Text
  - Optional
- mapUrl: A URL to a mapping service which shows the location of the building.
  - Attribute type: URL
  - Optional
- notes: List of free format notes relating to the building e.g. published occupants, opening hours etc.
  - AttributeType: Text
  - Optional

**Example of use:**

```
{
  “id”: “Building8-12345”,
  “type”: “Building”,
  “name”: “Building8-2016-10-25”,
  “dateCreated”: “2016-04-05”,
  “dateModified”: “2016-05-05”,
  “category”’:[“university”, “public”],
  “description”: “ Computer Science department Building 8 “,
  “containedInPlace”:{“type”: “Polygon”,
    “coordinates”: [
      [
        [-200.0, -200.0],
        [200.0, -200.0],
        [200.0, 200.0],
        [-200.0, -200.0]
      ],
      [
        [-200.0, -200.0],
        [200.0, -200.0],
        [200.0, 200.0],
        [-200.0, -200.0]
      ]
    ]
  },
  “location”:{“type”: “Polygon”,
    “coordinates”: [
      [
        [-100.0, -100.0],
        [100.0, -100.0],
        [100.0, 100.0],
        [-100.0, -100.0]
      ],
      [
        [-100.0, -100.0],
        [100.0, -100.0],
        [100.0, 100.0],
        [-100.0, -100.0]
      ]
     ]
   },
   “floorsAboveGround”: 2 ,
   “floorsBelowGround”: 0 ,
}
```

#### 3.2.3.1.3 UserContext

This model is intended to represent high level information about a person detected or identified in the scene. This data model describes the Context of a User [9]. No personal data is encoded in the model. The actual User data are stored in a different endpoint, as identified by the refUser property.

**Data Model:**

- id: Unique identifier.
- type: Entity type. It must be equal to Building.

- location
- refDevice: reference to the device who is in the context.
  - Attributte type: Device
  - Optional
- refUser: A user url reference.

**Example of use:**

```
{
  "id": "UserContext1",
  "type": "UserContext",
  "location": {
  "type": "Point",
  "coordinates": [
    -4.75444444,
    41.640833333
  ]
  },
  "refDevice": "Device1",
  "refUser": "user1"
}
```

#### 3.2.3.2 New Data Models

This section describes the new data models for use in Smart Security application.

##### 3.2.3.2.1 VideoObject

The model is based on the schema VideoObject entity [11]. As the VideoObject entity in schema is related to video objects in the context of "youtube videos" and movies, only common and specific attributes are selected for the VideoObject proposed in this document. The attributes proposed are valid to capture metadata of the main video standards. A refDevice attribute is added to identify the camera source.

**Data model:**

- id: Entity's unique identifier.
- type: Entity type. It must be equal to VideoObject.
- name: Common name given to this video.
  - Normative References: https://schema.org/name
  - Optional
- description: Video description.
  - Normative References: https://schema.org/description
  - Optional
- location: Location where the video was recorded represented by a GeoJSON geometry.
  - Attribute type: geo:json.
  - Normative References: https://tools.ietf.org/html/rfc7946
  - Optional
- refDevice: The camera device source.
  - Attribute type: Device.
  - Optional.
- contentSize: File size in (mega/kilo) bytes.

    - Attribute Type: Text
    - Optional
- videoFrameWidth: The frame width size of the video.
    - Attribute Type: Text
    - Mandatory.
- videoFrameHeight: The frame height size of the video.
    - Attribute Type: Text
    - Mandatory
- bitRate: the bitrate of the video in kbps.
    - Attribute Type: Text
    - Mandatory.
- encodingFormat: mpeg4, etc..
    - Attribute type: Text.
    - Mandatory.
- duration: the duration of the video.
    - Attribute Type:Text in ISO 8601 date format.
    - Mandatory.
- dateCreated: The date on the which the video was created.
    - Attribute type: Date
    - Mandatory.
- dateExpires: Date the content expires and is no longer useful or available .
    - Attribute type: Date
    - Optional.
- contentURL: Actual bytes of the video object.
    - Attribute type: URL
    - Mandatory.

**Example of use:**

The model will be used by Smart Security application to encoding high level data of videos recorded when movement is detected in the scene. One instance of this model is created every time that the basic movement event is detected by the streaming processing module.

```
{
  “id”: “videoObject-1234”,
  “type”: “VideoObject”,
   “name”: “Building8-2016-10-25”,
  “description”: “Video recorded in the library.“
  “location”: {
     “type”: “point”,
     “coordinates”: [-3.1644, 40.62234]
  },
  “refDevice”: “camera12345”,
  “contentSize”:”450 M”,
  “duration”:”T1H5M12S”,
  “dateCreated”: “2016-04-05”,
  “dateExpires”: “2016-05-05”,
  “contentUrl”:”www.surveillance.com/Building8-2016-10-25.mp4”
}
```

#### 3.2.3.2.2 VisualObject

This model is intended to capture information about the objects detected on video. From this point of view all elements (in a first stage) are an object instance. The model proposed is based on ViSOR [12] and the research described in [13]. Both references are based on ontologies where visual concepts and the surveillance domain are present. ViSOR is a repository that contains a large set of multimedia data and the corresponding annotations. On the other side, [13] describes a surveillance approach based on ontology for video event analysis. From video analysis perspective, all visual elements are objects which means that backpacks, buildings, vehicles, glasses, etc. are (visual) objects.

**Data Model:**

- id: Entity's unique identifier.
- type: Entity type. It must be equal to VisualObject.
- visualCategory: Category that the object belongs like book, backpack, hat, wall and so on.
  - Mandatory
- description: VisualObject description.
  - Normative References: https://schema.org/description
  - Optional
- location: Location that is represented by a GeoJSON geometry.
  - Attribute type: geo:json.
  - Normative References: https://tools.ietf.org/html/rfc7946
  - Optional
- movementCapacity: Defines if the object is a fixed object or not.
  - Attribute type: Text
  - Allowed values:
    - movable.
    - fixed.
  - Optional
- contextType: Defines if the object belongs to the context.
  - Attribute type: Text
  - Allowed values:
    - contextual.
    - noContextual.
  - Mandatory if the movementCapacity is Movable.
- dateCreated: The date when the mapObject was created.
  - Attribute type: Text.
  - Optional.

**Example of use:**

After the Smart Security application has processed the video streaming, visual objects are identified. Thus, every time that the recognition modules detect/identify a new object an instance of this data model will be created.

```
{
  "id": "visualObject-1234",
  "type": "VisualObject",
  "visualCategory": "Backpack",
```

```
  “description”: “Movable object in the scene “
  “location”:{“type”: “Polygon”,
    “coordinates”: [
      [
        [-100.0, -100.0],
        [100.0, -100.0],
        [100.0, 100.0],
        [-100.0, -100.0]
      ]
    ]
  },
  “movementCapacity”:”movable”,
  “contextType”:”noContextual”,
  “dateCreated”: “2016-04-05”,
}
```

# 4 DATA PRIVACY AND SECURITY ANALYSIS

The scenarios tackled in SmartSDK may involve the usage of private or sensitive data (e.g. the health record of a patient), as such it is important to analyse them and decide how to handle them in such a way that such data are protected in line with Data Privacy and Data Protection regulations. This is important due to the new technological advances that have allowed a paradigm shift in the use of computational applications: from desktop software to software as a service (or cloud computing).

Cloud computing delivers existing services through servers that are in the open Internet. Because of this, it is essential to apply ethical rules regarding the data privacy and security of the final use of data. Accordingly, we describe the possible issues about this topic in the SmartSDK applications and their data models.

- Smart Security application. In the application, we notice three components to be susceptible to data privacy and security issues: user management, video data management and video analysis.
  - User management. First of all, in order that a user (usually a guard) should able to use the system, he should be a registered user. This information will be managed in private by the system using the Identity Manager.The ContextUser data model will be secured and will be anonymised: it will not include and data that will allow direct identification of the person to which the data are linked.
  - Video management. Although the Device and VideoObject model does not contain sensitive data, the streaming contains images with people that have not given his consent to be recorded. Thus, the user management is in charge of allow access to registered user only that can see the cameras online. For video saving on external devices an administrator user is required. The streaming is managed by Kurento and is not sent to the Context Broker.
  - Video Analysis. As the video analysis and video event detection is online, the scene changes over the time, so, information about people and vehicle contain generic information. For example, if a person is recognized an anonymous label is created as name (person1). For vehicle detection/classification we do not save specific information about the owner, and we will protect and/or encrypt sensitive data such as the plate number.

  In general, such approach will be enough to comply with Data Privacy and Security legislation in Mexico (where the security trial will take place).

- Smart City Application: In the application, we notice three components to be susceptible to data privacy and security issues: user management, user alerts and user routes.
  - User management All information of registered users will be managed in private by the system, using secure connections through the Identity Manager, and also by creating regular and private databases with information of the users. This type of information includes his personal data (name, last name, age, and so on) and the information from his type of vehicle and diseases. Such data will be only accessible to the user himself and will be used to compute anonymised route recommendations.
  - User alerts. New data models were created to send alerts related to traffic jam, accidents cars, weather conditions, high level of pollutants, pollen, and asthma attacks. This information is sent to the Context Broker because it represents context data useful to app users. Such information will be anonymized to ensure that users generating the alert will not be traceable.
  - User routes management. All the information related to routes generated by users

traveling by transit, bus and rail will be secured and only accessible to the user himself (eventually through an encryption layer). Such information will be anonymized to ensure that users generating the alert will not be traceable.

These data management procedures will be enough to comply with Data Privacy and Security legislation in Mexico (where the trial will take place).

➔ Smart Health Application. Developed components and data-models contributed in this chapter have been developed in consideration of three key elements related to security and privacy.

- Secure access. Public repository in which data is shared through the Orion Context Broker, has enabled a restricted access limited by previous OAuth authentication and FIWARE's token.
- Privacy. There are two components taking into account:
  - User management. Information will be managed by a private data-manager, sheltered by using OAuth authentication mechanism through the Identity Manager, and by creating a private database. User information (main owner) will be based on unique identification data, such as: name and e-mail. Participant information (direct interactor of devices) will consist on demographic data such as: age, date of birth, and gender.
  - Sensor data (used as part of the motorPhisycalTest data-model). Although metadata will include descriptive information from the mobile devices, it will avoid sensitive data such as IMEI number[2]. Thus, no data have to be encrypted.
- Ethical considerations. Sensitive data will be removed from mobile devices (after secure them on FIWARE's cloud) and remote servers (after concluding trial) in order to fulfil ethical legislation from the countries in which the trial will be conducted (i.e., Mexico, Italy).

[2] https://en.wikipedia.org/wiki/International_Mobile_Equipment_Identity

# 5 CONCLUSIONS

One of the most important elements to facilitate the development of applications in FIWARE are data models. Data models are a formalized data structure that facilitates the exchange of information between the components of an application.

Smart City, Smart Health and Smart Security are three smart Applications that have guided the exploration process to identify new data model requirements.

The Device/DeviceModel models reused by the three applications. In order to have access to specific settings/values of the sensors contained in a Device instance, the consistOf attribute is proposed as an extension by SmartSecurity and SmartHealth applications.

The new data models Alert, Questionnaire, Questionnaire/question, Questionnaire/answer are data models that would be useful in the three scenarios. The Alert data model was designed by the Smart City Application and extended to be common by the Smart Security and Smart Health scenarios.

Four new data models (MotorPhysicalTest, Questionnaire, Questionnaire/Question and Questionnaire/Answer) are proposed by Smart Health application and reused some properties coming from Device.

Smart City reuses five data models (Air quality, Device, DeviceModel and Public vehicle Model) and propose ten new data models (Alert, SmartSpot, SmartPointOfInteraction, Agency, Route, Stop, Trip, TimeRange, Entity and ServiceAlert).

The Smart Security reuse six data models (Parking, Vehicle, Building, UserContext, Device and DeviceModel) and proposes three new data models (VideoObject and VisualObject).

Finally, as the data model design in FIWARE is an incremental process this document could be considered as a snapshot of this process. Thus, new data models could be added in next version of this report.

“Additional” data models, namely data models that are not sent to the Orion Context Broker, are described in the appendix A.

# REFERENCES

[1] FIWARE NGSIv2 (2016, October 31). fiware-ngsiv2-rc-2016_10_31. Retrieved 2017, March from http://telefonicaid.github.io/fiware-orion/api/v2/stable/

[2] SmartSDK Consortium. Description of Action. July 2016. SmartSDK project is co-funded by the EU's Horizon2020 programme under agreement number 723174 - ©2016 EC and by CONACYT agreement 737373

[3] Fiware Data Models. Retrieved 2017, March from http://fiware-datamodels.readthedocs.io/en/latest/index.html

[4] Github Repository. (2017, March 6). "smartsdk/dataModels". Retrieved 2017, March from https://github.com/smartsdk/dataModels

[5] Open Mobile Health. Retrieved 2017, March from http://www.openmhealth.org/

[6] General Transit Feed Specification. (2016, March 31). Retrieved 2017, April from https://developers.google.com/transit/gtfs/

[7] Service Alerts. (2016, July 12). Retrieved 2017, April from https://developers.google.com/transit/gtfs-realtime/guides/service-alerts

[8] Github Repository. "GSMADeveloper/HarmonisedEntityDefinitionsA". Retrieved 2017, March from https://github.com/GSMADeveloper/HarmonisedEntityDefinitionsA/

[9] Github Repository-UserContext. "tree/user". Retrieved 2017, March from https://github.com/smartsdk/dataModels/tree/user

[10] Google Transit Feed Specification - TransitWiki. Retrieved 2017, May from https://www.transitwiki.org/TransitWiki/index.php/General_Transit_Feed_Specification

[11] Schema. Retrieved 2017 March from http://schema.org/

[12] Video Surveillance Online Repository. Retrieved 2017, May from http://www.openvisor.org/config_schema.asp

[13] J. M. M. Á. G. Juan Carlos San Miguel, "An ontology for Event Detection and its Application in Surveillance Video," in *Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2009.

[14] D 3.1 SmartSDK Reference Models and Recipes. Retrieved 2017, May from https://drive.google.com/file/d/0B7ZXNoZldhmeaENsRktYMGVna1U/view?usp=sharing

# APPENDIX A - ADDITIONAL DATA MODELS

## A1. Introduction

In SmartSDK Generic Enablers, Reference Architectures and Data Models are the key elements that combined help the development of smart applications. In FIWARE smart applications are applications that are context-aware and rely on data context management services to provide "smart" decisions and information. Such models are encoded using NGSI formalism as described in D3.1 and, specifically for the applications developed in SmartSDK in D2.1.

Of course, not all data models part of an application are related to the "context". For example, a user profile, is required to manage users inside an application, but a user profile, including email, name and so on it is not what we usually consider contextual data. Contextual data are all such type of data that frequently vary over time, for example the location of a given user. Thus, besides the data models provided by FIWARE [1] and the new data models proposed for the different SmartSDK [2] scenarios we are including additional data models that are not context related but are needed for the development of concrete applications in the context of SmartSDK project.

This appendix complements the set of data models reported in this document.

## A2. Data Model for specific use by application

### A2.1. Smart Health

#### A2.1.1. Clinical Control

Patient's basic health data that might influence a physical-test performance. This model is build based on the Open Mobile Health [5]. Thus, the model has being harmonised to make them part of FIWARE data-model.

**Data model:**

- id: Unique identifier.
  - Mandatory.
- type: Entity type. It must be equal to ControlTest.
  - Mandatory
- refUser: Reference to the actual User sheltered by an independent service.
  - Attribute type: string.
  - Mandatory.
- omh:body_weight: Represents body weight.
  - Attribute type: body-weight.
  - Allowed values: (kg, g, lbs, oz).
  - Mandatory.
- omh:body_height: Represents body height.
  - Attribute type: body-height.
  - Allowed values: (m, cm, ft, in).
  - Mandatory.
- waistCircumference: This schema represents a person's waist circumference, either a single

body weight measurement, or for the result of aggregating several measurements made over time (see Numeric descriptor schema for a list of aggregate measures).
    - Attribute type: waist-circumference.
    - Mandatory.
- omh:heart_rate: Represents a person's heart rate.
    - Attribute type: heart-rate.
    - Allowed values: (beats/min).
    - Mandatory.
- omh:systolic_blood_pressure: It should be combined with diastolic blood pressure schema to create the full schema.
    - Attribute type: systolic-blood-pressure.
    - Allowed values: (mmHg).
    - Mandatory.
- omh:diastolic_blood_pressure: It should be combined with systolic blood pressure schema to create the full schema.
    - Attribute type: diastolic-blood-pressure.
    - Allowed values: (mmHg).
    - Mandatory.
- dateModified: Date and time measurements are taken.
    - Attribute type: DateTime.
    - Mandatory.

**Example of use:**

```
{
  "id":"fffffffff9cbbf4465f0ef30033c587-control-4",
  "type":"ControlTest",
  "refUser":"http://207.249.127.162:1234/users/1",
  "omh:body_weight": {
    "value":89,
    "unit":"kg"
  },
  "omh:body_height":{
    "value":180,
    "unit":"cm"
  },
  "waistCircumference":{
    "value":100,
    "unit":"cm"
  },
  "omh:heart_rate":{
    "value":70,
    "unit":"beats/min"
  },
  "omh:systolic_blood_pressure":{
    "value":120,
    "unit":"mmHg"
  },
  "omh:diastolic_blood_pressure":{
    "value":120,
    "unit":"mmHg"
  },
  "dateModified":"2017-01-18T20:45:42.697Z"
```

```
}
```

## A2.2. Smart City

### A2.2.1. User profile

This entity models a user. In addition, this model considers the addresses of home and workplace of a user. The model considers these data due to that these could be used to save favorites places. It is important to mention that this model is saved in Identity Manager.

**Data model:**

- id: Entity's unique identifier.
- type: Entity type. It must be equal to user.
- name: The name of user.
  - Attribute type: text
  - Normative References: https://schema.org/name
  - Mandatory
- family name: The last name of user
  - Attribute type: text
  - Normative References: https://schema.org/familyName
  - Mandatory
- gender: The gender of user
  - Attribute type: text
  - Normative References: [https://schema.org/gender] (https://schema.org/gender)
  - Optional
- birthDate:The date of birth of user
  - Attribute type: Date
  - Normative References: [https://schema.org/birthDate] (https://schema.org/birthDate)
  - Optional
- homeaddress: Civic address of a user´s home.
  - Normative References: [https://schema.org/address]
  - Optional
- workaddress: Civic address of the workplace user.
  - Normative References: https://schema.org/address
  - Optional
- dateCreated: Entity's creation timestamp
  - Attribute type: DateTime
  - Optional
- dateModified: Last update timestamp of this entity
  - Attribute type: DateTime
  - Optional

**Example of use:**

```
{
  “id”: “user:1”,
  “type”: “user”,
  “name”: “Iker”,
  “family name”: “Smith”,
   “gender”: “Smith”,
  “birthDate”: {
    “year”:”1988”,
    “month”:”april”,
    “day”:”8”
  }
  “homeaddress”: {
    “addressCountry”: “Mexico”,
    “addressRegion”: “Ciudad de México”,
    “addressLocality”: “Coyoacán”,
    “streetAddress”: “Puente de Piedra 150”,
    “postalCode”: “14090”
  },
  “workaddress”: {
    “addressCountry”: “Mexico”,
    “addressRegion”: “Ciudad de México”,
    “addressLocality”: “Tlalpan”,
    “streetAddress”: “San Fernando 37”,
    “postalCode”: “14050”
  },

   “dateCreated”: “2017-01-02T09:25:55.00Z”,
  “dateModified”: “2017-02-02T011:13:55.00Z”
}
```

### A2.2.2. Disease

This entity models the health condition of a user, including properties such as pathology, symptom and risk factor. This information could be useful to determine the best route to follow to reach a destination, taking into account the user health conditions. We use this model in Smart City App to propose an ideal route for the user, avoiding high levels of pollution, floods or pollen, etc., allowing for instance, to obtain the preferred routes for people with respiratory diseases. It is important to mention that this models is saved in Identity Manager.

**Data model:**

- id: Entity's unique identifier.
- type: Entity type. It must be equal to disease.
- pathology: Illness physical or mental of user
  - Attribute type: Text
  - Normative References: https://health-lifesci.schema.org/Pathology
  - Optional
- symptom: A sign or symptom of user condition.
  - Attribute type: Text
  - Normative References: http://health-lifesci.schema.org/signOrSymptom
  - Optional

- PhysicalActivity: Any bodily activity that enhances or maintains physical fitness and overall health and wellness. Includes activity that is part of daily living and routine, structured exercise, and exercise prescribed as part of a medical treatment or recovery plan.
    - Attribute type: Text
    - Normative References: https://health-lifesci.schema.org/PhysicalActivity
    - Optional
- possibleComplication: A possible unexpected and unfavorable evolution of a medical condition. Complications may include worsening of the signs or symptoms of the disease, extension of the condition to other organ systems, etc.
    - Attribute type: Text
    - Normative References: http://health-lifesci.schema.org/possibleComplication
    - Optional
- riskfactor: A modifiable or non-modifiable factor that increases the risk of user contracting this condition.
    - Attribute type: Text
    - Normative References: https://health-lifesci.schema.org/riskFactor
    - Optional
- dateCreated: Entity's creation timestamp
    - Attribute type: DateTime
    - Optional
- dateModified: Last update timestamp of this entity
    - Attribute type: DateTime
    - Optional

**Example of use:**

```
{
  “id”: “disease:asthma”,
  “type”: “disease”,
   “pathology”: “Mild Intermittent Asthma”,
  “symptom”: “cough, wheeze, chest tightness or difficulty breathing less
than twice a week.”,
  “PhysicalActivity”: “Do not interfere with normal activities”,
  “possibleComplication”: “Nighttime symptoms less than twice a month”
  “dateCreated”: “2017-01-02T09:25:55.00Z”,
  “dateModified”: “2017-02-02T011:13:55.00Z”
}
```

## A2.3. Smart Security

### A2.3.1. MapObject

This model is intended to capture high level data of the file that contains a map.

**Data Model:**

- id: Entity's unique identifier.

- type: Entity type. It must be equal to MapObject.
- name: Common name given to this MapObject.
  - Normative References: https://schema.org/name
  - Mandatory
- description: MapObject description.
  - Normative References: https://schema.org/description
  - Optional
- location: Location that the map represents. It is represented by a GeoJSON geometry.
  - Attribute type: geo:json.
  - Normative References: https://tools.ietf.org/html/rfc7946
  - Optional
- contentURL: The URL of the map file.
  - Attribute type: URL
  - Mandatory
- dateCreated: The date when the mapObject was created.
  - Attribute type: Text.
  - Optional.

**Example of use:**

One of the objectives of the Smart Security Application is detect wrong way events. To do this, a reference map is required to capture information about the area monitored. An instance of this model should be created to know information reference where the camera has been installed.

```
{
  “id”: “mapObject-1234”,
  “type”: “mapObject”,
  “name”: “mapObject-2016-10-25”,
  “description”: “Map of the institute. “
  “location”: {“type”: “Polygon”,
    “coordinates”: [
      [
        [-100.0, -100.0],
        [100.0, -100.0],
        [100.0, 100.0],
        [-100.0, -100.0]
      ]
    ]
  },
  “contentUrl”:”www.surveillance.com/map-2016-10-25.map“,
  “dateCreated”: “2016-04-05”,
}
```

## A3. Conclusions

Although specific application data models are not sent to the OCB, it could be reused for specific purposes by other applications. So, this document is an annexed to the *D.2.1 Reference data models for data intensive and IoT based Smart City, Smart Health and Smart Security Applications* report.