simpli·city

The Road User Information System Of The Future

WP4 – Mobility-related Data as a Service

D4.1.2: Data Model Version II

Deliverable Lead: IBM

Contributing Partners: IBM, TUDA, CRF

Delivery Date: 07/2014

Dissemination Level: Public

Version 1.00

This document gives details of the second and final version of the SIMPLI-CITY Data Model. In particular the focus is on mobility-related data i.e. how to access, represent, model and transform mobility-related data. This document addresses all these questions and presents the architecture behind the transformation model.

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 318201

Document Status	
Deliverable Lead	Robert Tucker, IBM
Internal Reviewer 1	Stefan Schulte, TUV
Internal Reviewer 2	Jan Reehuis, ASC
Type	Deliverable
Work Package	WP4: Mobility-related Data as a Service
ID	D4.1.2: Data Model Version II
Due Date	30.06.2014
Delivery Date	21.07.2014
Status	Approved

Document History	
Contributions	V0.1, Robert Tucker, IBM, 23.05.2014, Added document structure V0.2, Robert Tucker, IBM, 10.06.2014, Added Open Data section Input. V0.3, Robert Tucker, IBM, 16.06.2014, Integrated input from CRF received 10.06. V0.4, Robert Tucker, IBM, 18.06.2014, Changed second internal reviewer to Ascora. V0.5, Robert Tucker, IBM, 24.06.2014, Updated section for general sensor data access (TUDA). V0.6, Robert Tucker, IBM, 24.06.2014, Created reviewable version V0.7, Daniel Burgstahler, TUDA, 25.06.2014, Updated sections for general sensor data access and End User data access V0.8, Robert Tucker, IBM, 26.06.2014, updates in response to RFR1 review comments V0.9, Daniel Burgstahler, TUDA, 27.06.2014, resolved comments from reviewers V0.10, Robert Tucker, IBM, 02.07.2014, further updates in response to RFR1 review comments V0.11, Robert Tucker, IBM, 03.07.2014, integration of TUDA RFR1 responses V0.12, Robert Tucker, IBM, 03.07.2014, integration of CRF RFR1 responses V0.13, Robert Tucker, IBM, 04.07.2014, RFR2 versionV0.14, Daniel Burgstahler, TUDA, 10.07.2014, resolved last reviewer comments for TUDA V0.14, Daniel Burgstahler, TUDA, 10.07.2014, resolved last reviewer comments V0.15, Robert Tucker, IBM, 11.07.2014, integrated reviewer responses for IBM, CRF V1.00, Stefan Schulte, TUV, 21.07.2014
Final Version	July 21st, 2014

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

Disclaimer

The views represented in this document only reflect the views of the authors and not the views of the European Union. The European Union is not liable for any use that may be made of the information contained in this document.

Furthermore, the information is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user of the information uses it at its sole risk and liability.

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

Project Partners

Vienna University of Technology (Coordinator), Austria

Ascora GmbH, Germany

TIE Nederland B.V., The Netherlands

Technische Universität Darmstadt, Germany

IBM Research – Ireland
Smarter Cities Technology Centre

Forschungsgesellschaft Mobilität, Austria

Talkamatic AB, Sweden

Atos Worldline, Spain

Centro Ricerche FIAT, Italy

SRM – Reti e Mobilità, Italy

Executive Summary

This document finalizes the SIMPLI-CITY Data Model (Data Model Version II) based on the Data Model Version I deliverable (D4.1.1) submitted in month 12. It details some implementations which have been done so far. These will enable access for end users and service developers to the SIMPLI-CITY data sources using the SIMPLI-CITY Unified Data Model. These implementations reinforce the prototype work done in the recent WP4 deliverables D4.3.1 and D4.4.1 which helped to solidify and test the Data Model defined in D4.1.1. The focus of the SIMPLI-CITY Data Model is on mobility-related data and how to access, represent, model and transform this mobility-related data coming from multiple different and heterogeneous data sources making it available to SIMPLI-CITY services.

The Data Model and access framework is key to the SIMPLI-CITY technical work packages (WP4-6) as it provides the mechanism for connecting applications, sensors, user data and Open Data with services and components and allowing seamless integration of the mobility-related Data Model. From this perspective in this deliverable the Data Model Version II prototype gives further examples and implementations and also references the other recent prototype deliverables in WP4, in particular D4.3.1 (Sensor Abstraction and Interoperability Interfaces Prototype I) and D4.4.1 (User-centric and Open Data Management Prototype I) thus proofing and fixing the Data Model. Full implementations connecting all the relevant components and data exchange APIs will be completed in the final prototypes for WP4, namely D4.2 (Cloud-based Intelligent Infrastructure Prototype), D4.3.2 (Sensor Abstraction and Interoperability Interfaces Prototype II), D4.4.2 (User-centric and Open Data Management Prototype II) and D4.5.2 (Media Data Streams and Data Prefetching Prototype II).

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

Table of Contents

1 Introduction

SIMPLI-CITY – The Road User Information System of the Future – is a project funded by the Seventh Framework Programme of the European Commission under Grant Agreement No. 318201. It provides the technological foundation for bringing the “App Revolution” to road users by facilitating data integration, service development, and end user interaction.

Within this document, the SIMPLI-CITY Data Model Version II prototype will be presented. The document accompanies the corresponding software prototype, which is the main content of the deliverable.

1.1 SIMPLI-CITY Project Overview

Analogously to the “App Revolution”, SIMPLI-CITY adds a “software layer” to the hardware-driven “product” mobility. SIMPLI-CITY will take advantage of the great success of mobile apps that are currently being provided for systems such as Android, iOS, or Windows Phone. These apps have created new opportunities and even business models by making it possible for developers to produce new apps on top of the mobile device infrastructure. Many of the most advanced and innovative apps have been developed by players formerly not involved in the mobile software market. Hence, SIMPLI-CITY will support third party developers to efficiently realise and sell their mobility-related service and app ideas by a range of methods and tools, including the Mobility Services and App Marketplaces.

In order to foster the wide usage of those services, a holistic framework is needed which structures and bundles potential services that could deliver data from various sources to road user information systems. SIMPLI-CITY will provide such a framework by facilitating the following main project results:

- Mobility Services Framework: A next-generation European Wide Service Platform (EWSP) allowing the creation of mobility-related services as well as the creation of corresponding apps. This will enable third party providers to produce a wide range of interoperable, value-added services, and apps for drivers and other road users.
- Mobility-related Data as a Service: The integration of various, heterogeneous data sources like sensors, cooperative systems, telematics, Open Data repositories, people-centric sensing, and media data streams, which can be modelled, accessed, and integrated in a unified way.
- Personal Mobility Assistant: An end user assistant that allows road users to make use of the information provided by apps and to interact with them in a non-distracting way – based on a speech recognition approach. New apps can be integrated into the Personal Mobility Assistant in order to extend its functionalities for individual needs.

To achieve its goals, SIMPLI-CITY conducts original research and applies technologies from the fields of Ubiquitous Computing, Big Data, Media Streaming, the Semantic Web, the Internet of Things, the Internet of Services, and Human-Computer Interaction. For more information, please refer to the project website at http://www.simpli-city.eu.

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

1.2 Deliverable Purpose, Scope and Context

The purpose of this document is to finalize the definition and means to make use of the SIMPLI-CITY Data Model for the different available data types i.e. Open Data, user-centric data and sensor data including car sensor data. For this, the scope and requirements of the Data Model and any implemented prototype software are detailed along with any requirements and preparations for users and developers where necessary and in particular with regard to further SIMPLI-CITY work package and use case development. This includes any installation and usage guides as well as a brief overview of any limitations.

The Data Model Version II is the outcome of the discussions and implementation work done in project months 12 to 21 in relation to the Data Modelling and Access Framework. It provides some preliminary implementations based on the mechanisms of the SIMPLI-CITY Data Model Version I and any adaptations that have been found necessary in the interim period since Data Model Version I.

It also references the public SIMPLI-CITY deliverables D3.2.1 (Functional Specification), and D3.2.2 (Technical Specification). This deliverable D4.1.2 builds on and fixes the Data Model given in deliverable D4.1.1.

1.3 Document Status and Target Audience

This document is listed in the Description of Work (DoW) as “Public” and it details and refers to prototypical work done to proof the SIMPLI-CITY Data Model defined in deliverable D4.1.1 (Data Model Version I).

While the document primarily is aimed at the project partners, this public deliverable can also be useful for the wider scientific and industrial community. This includes other publicly funded projects, which may be interested in collaboration activities.

1.4 Abbreviations and Glossary

A definition of common terms and roles related to the realization of SIMPLI-CITY as well as a list of abbreviations is available in the supplementary document “Supplement: Abbreviations and Glossary”, which is provided in addition to this deliverable.

Further information can be found at http://www.simpli-city.eu.

1.5 Document Structure

This deliverable is broken down into the following sections as detailed below:

Section 1 provides an introduction for this deliverable including a general overview of the project, and outlines the purpose, scope, context, status, and target audience of this deliverable. Section 2 provides an overview of the scope and relationship of the Data Model, showing how to use it within the overall SIMPLI-CITY software framework. Furthermore, an assessment of the requirements covered by the Data Model is given. Section 3 presents any requirements and preparations to be done by software developers wanting to make use of the SIMPLI-CITY Data Model. Section 4 states information about the installation and deployment of the provided software package. Section 5 describes how software developers can use the provided functionalities. Section 6 discusses the current limitations of the SIMPLI-CITY Data Model. Finally, Section 7 provides a summary of the document.

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

2 Prototype Scope and Requirements Coverage

2.1 Data Model Version II – General Information

The diagram in Figure 1 positions the Data Modelling and Access Framework (components highlighted in red) in the SIMPLI-CITY Global Architecture (see deliverable D3.1: Global Architecture Definition). The Data Modelling and Access Framework is complementary to the Data Processing component as it offers data modelling and access, but not processing facilities such as data merging or contextualization (all offered by Data Processing).

The main objective of the SIMPLI-CITY Data Model is to provide easy and transparent access to Open Data, sensor data (including car sensor data) and end user data, following the security and privacy-related guidelines of SIMPLI-CITY (see deliverable D3.3: Holistic Security and Privacy Concept). In the context of SIMPLI-CITY, the term “mobility-related” data is used to identify relevant transportation-aware open, sensor and end user data. One of the main characteristics of such data is the importance of temporal and spatial descriptions i.e., data is evolving on a time and space basis. For instance both bus information and road weather conditions are dynamic in both terms of time and space. In addition to handle a seamless data access, the Data Modelling and Access Framework also provides a Unified Data Model, called SIMPLI-CITY Data Model for representing mobility-related data across multiple heterogeneous data sources. Figure 1 shows the location of the software components handling the SIMPLI-CITY Data Modelling and Access.

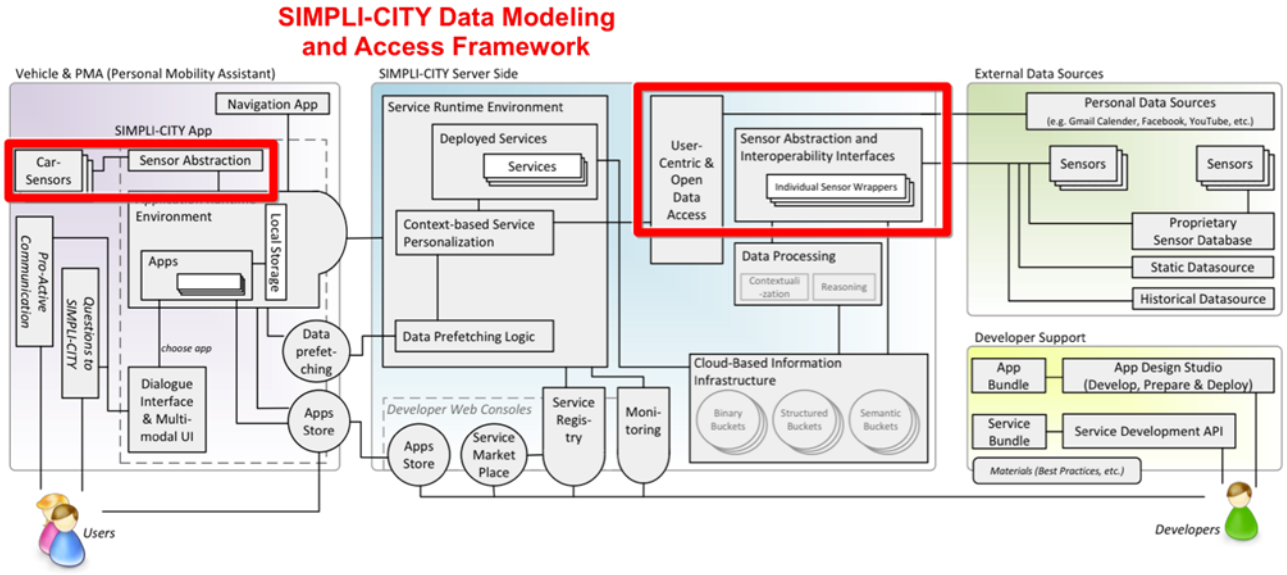


Figure 1: Location of Data Model and Access Framework Components in the SIMPLI-CITY Global Architecture

2.2 Scope of the Data Modelling and Access Framework

In the following subsections the scope of the SIMPLI-CITY Data Modelling and Access Framework implementations for different aspects of the Data Model will be detailed. In each subsection a diagram of the relevant SIMPLI-CITY components or subcomponents will be shown.

The status of the implementation will be shown using the following colour codes:

1. Green: Fully implemented.
2. Orange: Partially implemented.
3. White: No implementation so far.

The scope and status for different data sources and transformations will be discussed in more detail in the following subsections. For the Functional Specification and Technical Specification of these subcomponents, refer to SIMPLI-CITY deliverables D3.2.1 and D3.2.2, respectively. While the prototype implementations in this deliverable proof the completeness of the Data Model, the full implementation of components will be completed in the future deliverables D4.2, D4.3.2, D4.4.2 and D4.5.2. Hence, most components are marked as partially complete.

2.2.1 Open Data Access

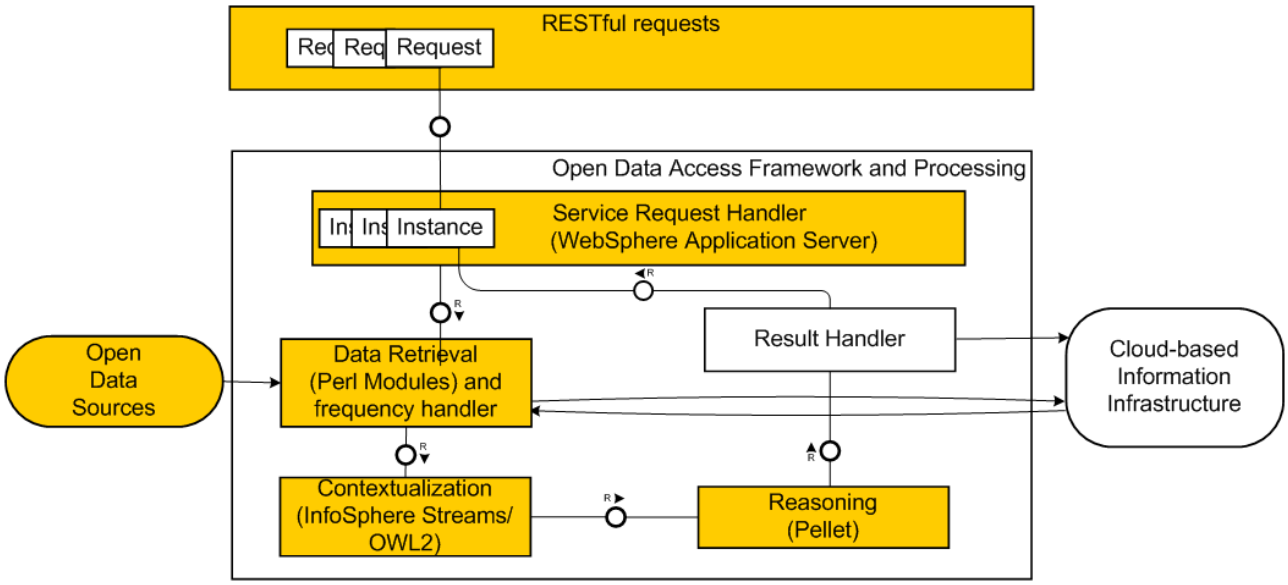


Figure 2: Scope of the SIMPL-CITY Data Model Version II – Open Data

The Open Data Access Framework and Processing provides the transformation of the SIMPLI-CITY Open Data sources into the SIMPLI-CITY Unified Data Model as described in the deliverable Data Model Version I (D4.1.1). It also provides RESTful interfaces to be called by the SIMPLI-CITY backend services which allows individual data sources to be queried returning data in an unprocessed format (Comma-Separated Value – CSV), in transformed format (RDF – Resource Description Format) and also feeding into diagnosis and prediction results which have been requested. In this prototype the mechanisms are demonstrated for particular data sources which have been implemented based on deliverables D4.1.1, D3.2.1, D3.2.2 and D3.3, and in conjunction with building the overall Data Processing infrastructure as described in deliverable D4.4.1. Because of that there is

an overlap in the descriptions and overall infrastructure of Data Processing components but with a focus and highlighting the Open Data aspects in this prototype.

Figure 2 depicts the status of implementation of the SIMPLI-CITY Data Model for Open Data access and processing. In Figure 2 the focus is on Open Data source access and processing. Transformation of other data source types for user-centric and sensor data will follow the same mechanism once they are fully integrated with the second prototype deliverables of the Sensor Abstraction and Interoperability Interfaces (D4.3.2) and User-Centric and Open Data Management (D4.4.2) software components.

2.2.2 General Sensor Data Access

The Sensor Abstraction and Interoperability Interfaces component provides the transformation of external sensor data sources into the Unified Data Model as described in D4.1.1. The data access is realized by a RESTful interface provided by a service within the Service Runtime Environment. All interfaces as defined in the SIMPLI-CITY Technical Specification (deliverable D3.2.2) are provided to other SIMPLI-CITY components, although the interfaces to request historical data are without functionality for now. This functionality will be added in deliverable D4.3.2. Furthermore, the prototype realizes the interaction between the server side and the PMA side of the Sensor Abstraction and Interoperability Interfaces. The prototype provides the server side access to all local PMA sensors, including sensors of a connected car, as well as access to user data from the contacts and the calendar of the user. In addition a Java interface is provided on the PMA side that can directly be used by the Application Runtime Environment to access directly local sensors, sensors of a connected car and user data. Thus the component provides a unified access and data format to request all sensor and user data.

Figure 3 depicts the status of implementation of the SIMPLI-CITY Data Model for General Sensor Data access. Light grey marked components are not part of this component but make use of it. Components inside a light grey box form a logical unit. Components that are not implemented until now will be completely covered in deliverable D4.3.2. The scope for the current state of the general sensor data access prototype is comprehensively treated for all subcomponents in Section 2.2 of deliverable D4.3.1 (Sensor Abstraction and Interoperability Interfaces Prototype I) for both server side and PMA side access.

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

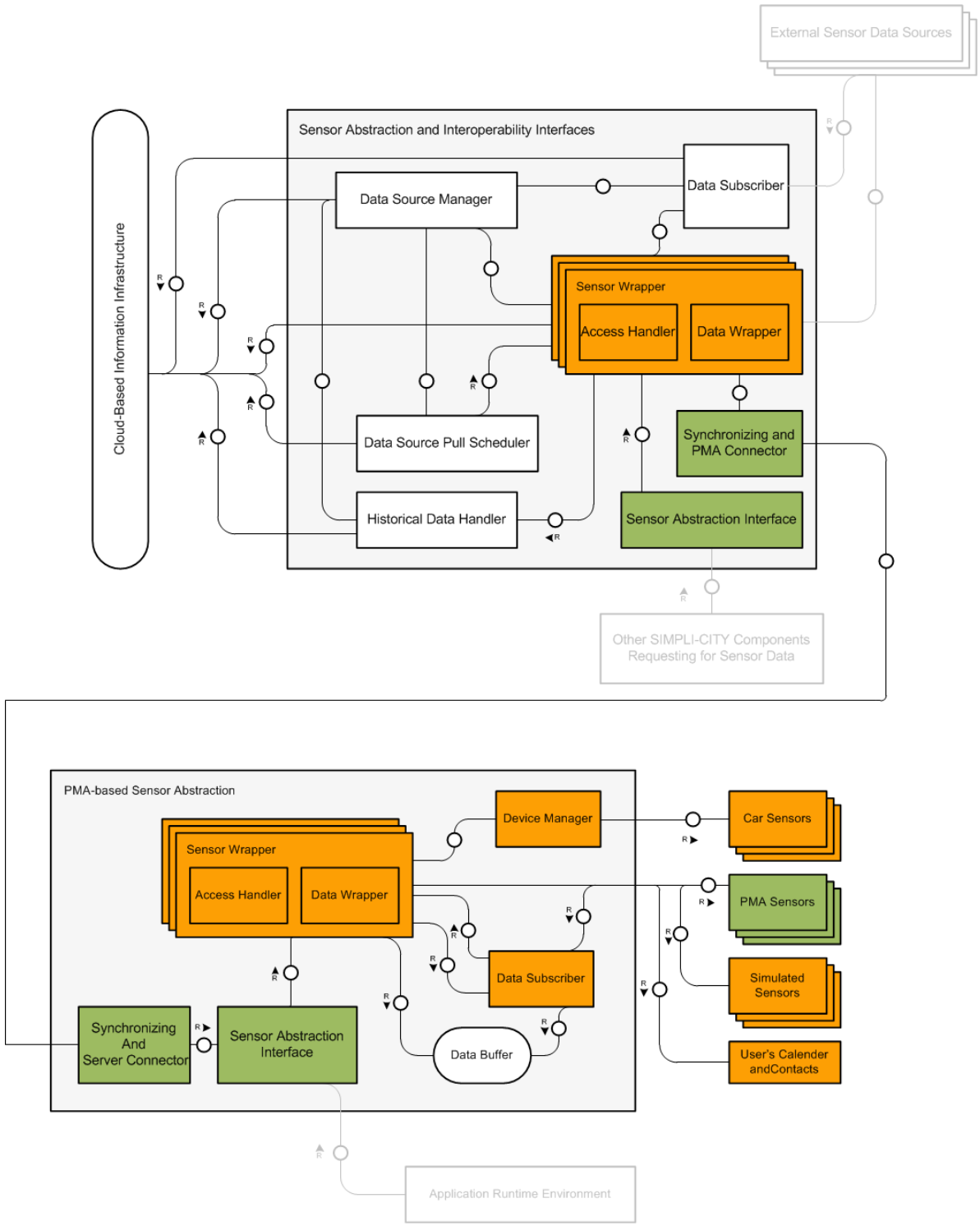


Figure 3: Scope of the SIMPL-CITY Data Model Version II – General Sensor Data

2.2.3 Car Sensor Data Provision

SIMPLI-CITY provides two kinds of car sensors:

- OBD port based: allows using car data available from the On-board Diagnostics (OBD) port from each car. Of course only a limited set of signals is available from the diagnostic port. Other Original Equipment Manufacturer (OEM)-specific messages are not provided from this more general sensor.
- OEM-specific car sensor (i.e. the one implemented by CRF for FIAT cars): allows using OEM-specific messages; it should be implemented by specific OEMs following the SIMPLI-CITY specification for the provision of SIMPLICITY sensor data (see Section 5 of D4.1.1).

From a service provider point of view, on the basis of the type of car data needed, a service can subscribe to the first or the second type of car sensors. Of course, if needing OEM-specific messages, the service can be used only for vehicles whose manufacturers have implemented the OEM-specific car sensor specification.

In this document the solution provided by CRF for a specific car model, using the SIMPLI-CITY Data Model outlined in D4.1.1, is described in detail. Other OEMs following the SIMPLI-CITY specifications could provide similar specific car sensors to be used by services and apps implemented as part of the SIMPLI-CITY Service Runtime Environment and Application Runtime Environment, respectively.

The process followed for providing data from a FIAT vehicle is summarized in the activity diagram shown in Figure 4:

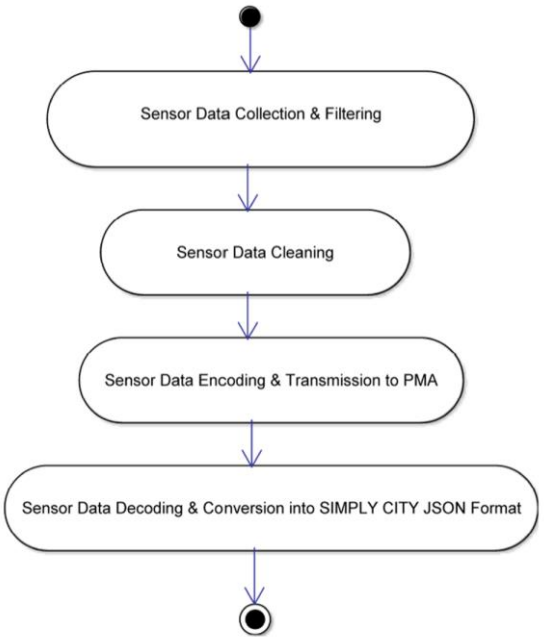


Figure 4: Car Sensor Data Provision – Activity Diagram

The four activities reported in this diagram are briefly described here:

1. Sensor Data Collection and Filtering: During this activity sensor data are gathered from the vehicle and then filtered. Table 1 lists the car data collected from the 500 L

trekking, the vehicle that will be used for the provision, the demonstration and the validation of the SIMPLI-CITY Use Cases in Turin.

Table 1: Car Sensor Data Acquired from the 500 L Trekking

Signal	Description
BatteryVoltageLevel	Value of the voltage of the battery
ExternalTemperature	External temperature
ExternalTemperatureFailSts	Check on the external temperature fail status
TemperatureUnit	Current temperature unit °C/F
DaysToService	Days left before service
DaysToServiceValidData	Check if the DaysToService value is valid
DistanceToService	Distance left before service
DistanceToServiceValidData	Check if the DistanceToService value is valid
DriverDoorSts	Indication about the status of the driver's door (open/close)
FuelLevel	Indication about the level of the fuel tank in percentage
FuelLevelFailSts	Check if the FuelLevel value is valid
LowFuelWarningSts	Indication about the emergency fuel level is detected
PsngrDoorSts	Indication about the status of the passenger’s door (open/close)
RHRDoorSts	Indication about the status of the rear passenger’s door (open/close)
EngineSpeed	Indication about the engine speed in rpm
VehicleSpeed	Indication about the vehicle speed
VehicleSpeedFailSts	Check if the VehicleSpeed value is valid
InstantFuelConsumptionValidData	Check if the InstantFuelConsumption values are valid
InstantFuelConsumption3	3rd digit of the instant fuel consumption (BCD - Binary-coded decimal - coded)
InstantFuelConsumption2	2nd digit of the instant fuel consumption (BCD coded)
InstantFuelConsumption1	1st digit of the instant fuel consumption (BCD coded)
AutonomyDistanceValidData	Check if the AutonomyDistance value is valid
AutonomyDistance	Distance in kilometers/miles that car can perform with the current fuel level

http://www.simpli-city.eu/ Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

TotalOdometer	Total of the kilometers/miles performed by the car
KeySts	Status of the key (Off, On, CrankOn, Stop)
AverageFuelConsumptionA_Digit3	3rd digit of the average fuel consumption (BCD coded)
AverageFuelConsumptionA_Digit2	2nd digit of the average fuel consumption (BCD coded)
AverageFuelConsumptionA_Digit1	1st digit of the average fuel consumption (BCD coded)
AverageFuelConsumptionA_ValidData	Check if the AverageFuelConsumption values are valid
AverageSpeedA	Average speed calculated for Trip A
AverageSpeedAValidData	Check if the AverageSpeedA value is valid
PartialOdometerA	Total of the kilometers/miles performed by the car for Trip A
PartialOdometerAValidData	Check if the PartialOdometerA value is valid
AccelerationEcoIndex	Acceleration score (performance) calculated by the CRF/FIAT FIAT Eco:Drive algorithm (1 - bad value / 5 - great value)
DecelerationEcoIndex	Deceleration score (performance) calculated by the CRF/FIAT FIAT Eco:Drive algorithm (1 - bad value / 5 - great value)
GearShiftingEcoIndex	Gear change score (performance) calculated by the CRF/FIATFIAT Eco:Drive algorithm (1 - bad value / 5 - great value)
SpeedEcoIndex	Speed score (performance) calculated by the CRF/FIAT algorithm FIAT Eco:Drive algorithm (1 - bad value / 5 - great value)
TotalEcoIndex	Total score (performance) calculated by the CRF/FIAT algorithm FIAT Eco:Drive algorithm based on the performances of other EcoIndex value (1 - bad value / 5 great value)
GPSLat	GPS latitude
GPSLon	GPS longitude

Most of the signals listed above are managed by the Controller Are Network (CAN) bus. The EcoIndex related values are basically generated by the car In-Vehicle Infotainment (IVI) platform via software as results of the FIAT Eco:Drive algorithm that runs inside the IVI platform. Finally, the GPS data is emitted by the GPS antenna included in the IVI platform if equipped or by an external GPS antenna.

2. Sensor Data Cleaning: in this activity, there are two steps:
    a. Sensor Data Validation: The purpose of this task is the validation of the signals for which there exists a corresponding control signal, used for checking if the signal value is valid (e.g., for the signal “AutonomyDistance” there is the coupled control signal “AutonomyDistanceValidData”).In particular if the given signal has an invalid value, the value is forced to a predefined control value (e.g. -999).
    b. Sensor Data Combination & Conversion: in this sub-activity, numeric-type signals coming from the vehicle are combined if needed and then converted

http://www.simpli-city.eu/ Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

using a fixed measurement unit associated with them. Considering the list of selected signal, the Data Combination is performed only for the InstantFuelConsumption signal. This signal is obtained by combination of three signals coming from vehicle: InstantFuelConsumption1, InstantFuelConsumption2 and InstantFuelConsumption3.

3. Sensor Data Encoding & Transmission to PMA: after filtering and cleaning, a car signal is encoded to be transmitted to a mobile device (i.e., the PMA).
   For this purpose, Google ProtocolBuffer is used. The defined ProtocolBuffer requires that, for every car signals to be transmitted, the following information are specified:
   a. Sensor Type
   b. Sensor Subtype
   c. Sensor Value Accuracy Name
   d. Sensor Value Accuracy Unit

In Table 2 the information encoded for each signal is listed.

Table 2: Car Sensor Data Encoding Information

Signal	SENSOR_ TYPE	SENSOR_ SUBTYPE	SENSOR_VALUE_ ACCURACY_VALUE	SENSOR_VALUE_ ACCURACY_UNIT
AutonomyDistance	Vehicle	Distance	Not Defined	km
DistanceToService	Vehicle	Distance	Not Defined	km
AverageFuelConsumption APlus	Vehicle	FuelConsumption	Not Defined	km/l
InstantFuelConsumption	Vehicle	FuelConsumption	Not Defined	km/l
AverageSpeedAPlus	Vehicle	Speed	+/- 0.1	km/h
VehicleSpeed	Vehicle	Speed	Not Defined	km/h
BatteryVoltageLevel	Vehicle	BatteryVoltageLevel	0.05	Volt
DaysToService	Vehicle	DaysToService	Not Applicable	Not Applicable
EngineSpeed	Vehicle	EngineSpeed	+/- 1	Rpm
FuelLevel	Vehicle	FuelLevel	+/- 2.5 at 17.5%	Percentage
KeySts	Vehicle	KeySts	Not Applicable	Not Applicable
LowFuelWarningSts	Vehicle	LowFuelWarningSts	Not Applicable	Not Applicable
DriverDoorSts	Vehicle	DoorStatus	Not Applicable	Not Applicable
PsngrDoorSts	Vehicle	DoorStatus	Not Applicable	Not Applicable

http://www.simpli-city.eu/ Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

Signal	SENSOR_ TYPE	SENSOR_ SUBTYPE	SENSOR_VALUE_ ACCURACY_VALUE	SENSOR_VALUE_ ACCURACY_UNIT
RHRDoorSts	Vehicle	DoorStatus	Not Applicable	Not Applicable
TemperatureUnit	Vehicle	Temperature	Not Applicable	Not Applicable
ExternalTemperature	Vehicle	Temperature	+/- 0,5	°C
TotalOdometer	Vehicle	Odometer	Not Defined	km
PartialOdometerAPlus	Vehicle	Odometer	+/- 0,1	km
AccelerationEcoIndex	Vehicle	EcoIndex	Not Applicable	Not Applicable
DecelerationEcoIndex	Vehicle	EcoIndex	Not Applicable	Not Applicable
GearShiftingEcoIndex	Vehicle	EcoIndex	Not Applicable	Not Applicable
SpeedEcoIndex	Vehicle	EcoIndex	Not Applicable	Not Applicable
TotalEcoIndex	Vehicle	EcoIndex	Not Applicable	Not Applicable
GPSLat	Vehicle	GPSLocation	Not Defined	decimal degrees
GPSLon	Vehicle	GPSLocation	Not Defined	decimal degrees

After encoding, data are transmitted to the mobile device using the Bluetooth protocol.

4. Sensor Data Decoding & Conversion into SIMPLY-CITY JSON Format: During this activity, sensor data encoded and then transmitted by the vehicle are received by the mobile device. Sensor data are decoded according with the message types defined in the ProtocolBuffer and then transformed in the SIMPLY-CITY JSON format. The listing below reports an example of how, at the end of this process, the numeric type signal ExternalTemperature=18, coming from the vehicle is converted into the SIMPLY-CITY JSON format.

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

Listing 1: Example of Car Sensor Data Conversion into SIMPLI-CITY JSON Format

```
{
   "object": "sensor",
   "sensorType": {
      "type": "Vehicle",
      "subtype": "Temperature"
   },
   "sensorValue": [
      {
         "name": "External Temperature",
         "value": 18,
         "unit": "C",
         "accuracy": {
            "unit": "C",
            "value": 0.5
         }
      }
   ],
   "objectID": "...",
   "timeStamp": "...",
   "parent": "..."
}
```

2.2.4 End User (PMA) Data Access

This section describes the Data Model prototype for user-centric data that is per deliverable D4.4.1 based on data extracted from the users' PMA. On the PMA side, i.e. an Android enabled smartphone, the user can add several online accounts, e.g., different Google or Facebook accounts. All these accounts may contain a kind of address book or calendar. All these data sources are merged within the mobile operating system, i.e. Android that is also responsible for data synchronisation. This is a basic functionality of all current major mobile operating systems, i.e., Google Android, Windows Phone and Apple iOS. On the PMA side a background service of the Sensor Abstraction and Interoperability Interfaces is responsible for accessing this data. This background service provides a Java interface to provide data access to the SIMPLI-CITY apps running in the Application Runtime Environment. Furthermore this background service communicates with the server side of the Sensor Abstraction and Interoperability Interfaces that provide an interface to other SIMPLI-CITY services to access this user data. These server side interfaces are implemented as Java interfaces for services executed on the same physical machine and additionally as RESTful interfaces to provide access to SIMPLI-CITY services executed on other machines. The data format representing user data is exactly the same as used for general sensor data.

The provision of this data in the SIMPLI-CITY Data Model in prototype D4.4.2 (User-centric and Open Data Management Prototype II) will be via a REST API call to the Sensor Abstraction and Interoperability Interfaces. This will return the relevant user-centric data, e.g. calendar events, to the caller in the JSON schema format corresponding to sensor data. In this case the caller is the Data Processing component.

2.3 Covered Requirements

This section describes the degree of fulfilment of the requirements to be covered by the SIMPLI-CITY Data Model as specified in the Requirements Analysis Deliverable (D2.3) and the Functional Specification (D3.2.1).

Table 2: Requirements Related to SIMPLI-CITY Data Model and their Degree of Fulfilment

Requirement	Degree of Fulfilment	Comment
Must Have Requirements		
U80: Profile in the cloud	0%	The Cloud Infrastructure APIs are not available yet (T4.2). This facility will be completed in deliverable D4.2.
U85: Interaction with car sensors	100%	Completely covered. Some car sensors are available via the OBD interface of the car. A selected set of car sensors are available for specific car models from CAN bus via on-board telematics platform.
U87: Confidentiality. Does not give away data to third parties	0%	User ID's which are unique per app and service will be implemented as per D3.3 within the functionalities for user-centric data provided in the D4.3.2 and D4.4.2 deliverables.
U89: Certification. Only certified apps are allowed to access users data	0%	Users linked to app IDs for certified apps will be used to determine access within functionalities developed for the D4.3.2 prototype.
U107: Access to sensors of the vehicle	100%	Completely covered. Some car sensors are available via the OBD interface of the car. A selected set of car sensors are available for specific car models from CAN bus via on-board telematics platform.
U108: Access to smart device sensors	100%	Covered in the D4.3.1 prototype. The sensors are available to the Application Runtime Environment in the PMA and the access is forwarded to the server side.
U109: Access to remote sensors, e.g. traffic sensors	100%	Traffic and weather sensors are being accessed for Dublin and Bologna.

Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

Requirement	Degree of Fulfilment	Comment
U114: Configuration of the frequency of update of the data from data sources	80%	Some frequency manipulation is possible via REST APIs. Full implementation in interaction with services still to do and will be completed in D4.4.2.
U115: Transformation support	70%	Open Data sources mainly transformed. Other user-centric and sensor data to be completed in D4.3.2 and D4.4.2.
U116: Unified data model	100%	Data Model Version II finalizes the Unified Data Model defined in Data Model Version I (D4.1.1).
U117: Data filtering U118: Data correlation	25%	To be implemented as per definition in D3.2.1 in D4.4.2.
U119: Data summarisation	0%	Functionality will be covered by the D4.4.2 prototype.
U120: Handle of data streams	75%	Stream operators implemented for some Open Data sources. This will be completed in D4.4.2.
U189: Unified interface for accessing sensors	100%	The Sensor Abstraction Interface provides data in a Unified Data Model. The Interfaces and the data model are already implemented since D4.3.1.
U202: Diagnosis of abnormal traffic condition in real-time	50%	Development of this feature is ongoing and will be completed in the D4.4.2 prototype.
U203: Prediction of abnormal traffic condition	20%	Development of this feature is ongoing and will be completed in the D4.4.2 prototype.
U204: Support for querying diagnosis historic	40%	Development of this feature is ongoing and will be completed in the D4.4.2 prototype.
U205: Support for querying impact factor on traffic condition	0%	Development of this feature will be completed in the D4.4.2 prototype.

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

Requirement	Degree of Fulfilment	Comment
Should Have Requirements		
U107: Access to sensors of the vehicle	100%	A general access to a subset of car sensors is already realized via the OBD interface of the car.
Will Not Have For Now Requirements		
U110: Remote control of car components when the functionality is available, e.g. air conditioning, heating, battery charge timing	0%	Writing to a car's data bus from an external device is not possible for now. Thus this functionality will not be implemented.

http://www.simpli-city.eu/ Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

3 Preparations

This section provides information about what potential users (both administrators and software developers) need to prepare in order to use the SIMPLI-CITY Data Modelling and Access Framework.

3.1 Server Side (System Administrators)

3.1.1 Open Data Access

As mentioned in Section 2.2.1 and deliverable D4.4.1 (User-Centric and Open Data Management Prototype I), the overall infrastructure required for this prototype also forms the basis of infrastructure required for the components contributing to the 'Data Processing' parts of SIMPLI-CITY. The Open Data Management Infrastructure is deployed within Virtual Machines (VMs) running on servers within IBM Ireland's Research Lab setup as described below. Access to the raw (CSV) and transformed (RDF) data corresponding to the SIMPLI-CITY datasets is via RESTful interfaces. This is shown in the figure below.

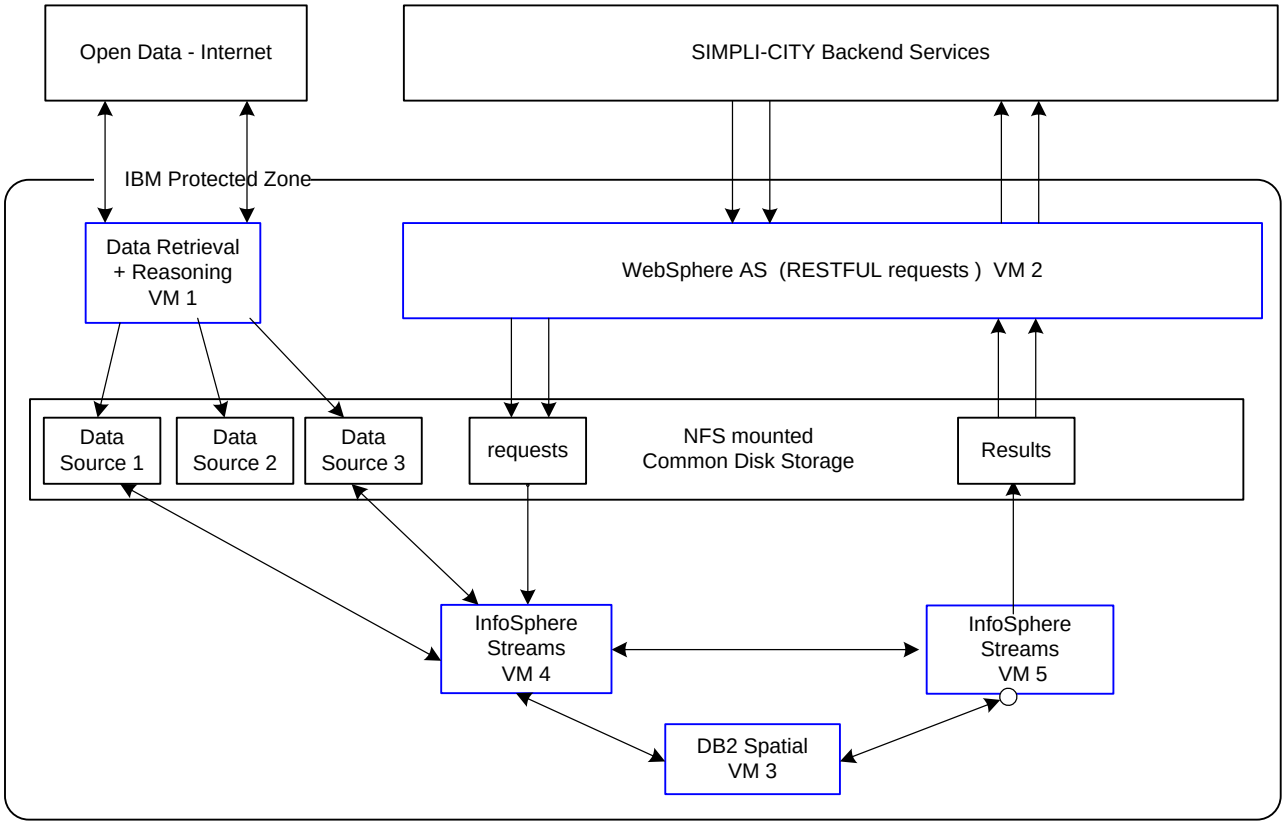


Figure 5: Open Data Access Framework and Data Processing Infrastructure

The setup of the above system shown in Figure 5 requires a system administrator to execute the following actions (each VM is represented by a blue box in the figure):

- Setup 5 VMs running Red Hat® Enterprise Linux® Server version 6.4. Each VM has 2 Intel Xeon CPUs running at 2.0GHz and 32Gb memory. This is the necessary and sufficient VM spec for the current prototype; however this may change for the final prototype for Open Data using this infrastructure (part of D4.4.2).

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

- Configure a user and group ‘simplicity’ on each VM
- Mount the common disk storage over NFS from each VM
- Install IBM WebSphere® Application Server in VM 2
- Install IBM DB2® version 9.7 with Spatial Extender package on VM 3 and allow 40Gb database storage on the common mounted disk.
- Create the database SM_SPE
- Load spatial data into the database SM_SPE as follows:
  db2se enable_db SM_SPE
  db2se shape_info –filename dublin_highway.shp
- Install IBM InfoSphere® Streams version 3.1 in the VMs 4 and 5:
  the common ‘streams home’ area is created on the common mounted disk storage
- Create an instance called *sm_data_all* in IBM InfoSphere® Streams which can use both VM 4 and VM 5.
- Enable firewall rules for the Data Retrieval VM’s IP to be able to call out to the URLs associated with the listed datasets in Table 1
- On the Data Retrieval VM open the ports 80 (HTTP) and 443 (HTTPS)

3.1.2 General Sensor Data Access

Preparations for the current state of general sensor data access prototype is comprehensively treated in Section 3 of the D4.3.1 deliverable (Sensor Abstraction and Interoperability Interfaces Prototype I) for both server side and PMA side access and therefore not repeated here.

3.1.3 Car Sensor Data Provision

In order to realize car sensor data provision the vehicle has to be equipped with an IVI platform (e.g. uConnect Platform in a FIAT vehicle).

The IVI platform has to be able to gather sensor data from the CAN bus of the type described in Section 2.2.3 and to transmit them to a mobile device. Regarding data transmission to mobile device, sensor data will be serialized using Google ProtocolBuffer and then sent to the mobile device using a Bluetooth connection.

3.1.4 End User (PMA) Data Access

The extraction of user data on the PMA is handled within SIMPLI-CITY via the Sensor Abstraction and Interoperability Interfaces component. As per deliverable D4.4.1 for this prototype, user-personal data was extracted from Google Calendar. The data can be accessed via the Sensor Abstraction and Interoperability Interfaces. The schema and the sensor structures used are described in detail in the Technical Specification (D3.2.2) and Data Model Version I (D4.1.1) deliverables. The necessary preparations for the current state of user data access is the same as for general sensor data.

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

Listing 2: Subset of the Extracted User Calendar Data in Sensor JSON Format


```
{
   "sensorType": {
      "type": "userdata",
      "subtype": "calendar"
   },
   "sensorValue": [
      [
         {
            "name": "title",
            "value": "May Day"
         },
         {
            "name": "organizer",
            "value": "en-gb.irish#holiday@group.v.calendar.google.com"
         },
         {
            "name": "description",
            "value": ""
         },
         {
            "name": "dtstart",
            "value": "2015-05-04T00:00:00Z"
         },
         {
            "name": "dtend",
            "value": "2015-05-05T00:00:00Z"
         },
         {
            "name": "allDay",
            "value": "1"
         },
         {
            "name": "duration"
         },
         {
            "name": "eventLocation",
            "value": ""
         },
         {
            "name": "calendar_displayName",
            "value": "Holidays in Ireland"
         }
      ],
      [
         {
            "name": "title",
            "value": "GROUP Meeting"
         },
         {
            "name": "organizer",
            "value": "example.user.1@gmail.com"
         },
         {
            "name": "description",
            "value": ""
         },
```


http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

```
         {
            "name": "dtstart",
            "value": "2014-04-30T14:00:00Z"
         },
         {
            "name": "dtend",
            "value": "2014-04-30T16:00:00Z"
         },
         {
            "name": "allDay",
            "value": "0"
         },
         {
            "name": "duration"
         },
         {
            "name": "eventLocation",
            "value": ""
         },
         {
            "name": "calendar_displayName",
            "value": "example.user.1@gmail.com"
         }
      ]
   ],
   "object": "sensor",
   "timeStamp": "2014-05-06T07:54:08Z",
   "objectID": "11b04404-b0b4-4cb3-931a-f94e54cd4004",
   "parent": "4b168d73-3722-43df-896b-482ac4aba724"
}
```


The current version of the PMA-based Sensor Abstraction and Interoperability Interfaces component is implemented as an Android background service and operates on every device running Android version 4.1 or higher. The Sensor Abstraction and Interoperability Interfaces provide a RESTful interface to retrieve contact and calendar information of a SIMPLI-CITY user. The input parameter is the SIMPLI-CITY user ID. To make use of the RESTful interfaces no further preparations have to be done. The service can be called from a standard Web browser.

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

4 Installation (Deployment)

This section provides guidelines on how to install and deploy the developed implementations relating to the SIMPLI-CITY Data Model and Access Framework.

4.1 Server Side (System Administrators)

This section provides guidelines on how to install and deploy the developed implementations of the Data Model Version II prototype.

4.1.1 Open Data Access

The Open Data Access Framework is implemented within the overall Data Processing component and it handles retrieving Open Data sets directly. The software solutions chosen for transforming data into the Unified Data Model in the Technical Specification D3.2.2 was IBM InfoSphere® Streams. Operators developed in IBM InfoSphere Streams to transform different data sources are deployed as follows:

- Deploy data stream operators into IBM InfoSphere® Streams for transformation in VM 4 as follows:

Listing 3: Stream Operator Deployment Call

```
streamtool submitjob -i sm_data_all@simplicity
Demo2.SimplicityDemoDuTrips_1_min.adl
```

The RESTful services which give access to Open Data in raw and transformed formats are made available by deploying a Web Archive file into VM 2:

- Load the simplicity.war file into WebSphere Application Server using the management console in VM 2

4.1.2 General Sensor Data Access

General Sensor Data Access is provided by the Sensor Abstraction and Interoperability Interfaces component. The server side component of the Sensor Abstraction and Interoperability Interfaces is executed as a bundle within the Service Runtime Environment. Thus in a first step the Service Runtime Environment has to be started as described in deliverable D5.3.1. Due to completeness and replicability this is also explained in the following Section 4.1.2.1. In the next step the necessary components have to be loaded within the Service Runtime Environment. The installation for the current state of general sensor data access prototype is comprehensively treated in Section 4 of the D4.3.1 deliverable (Sensor Abstraction and Interoperability Interfaces Prototype I) for both server side and PMA side access. Due to completeness and replicability this is also explained in the following Section 4.1.2.2.

4.1.2.1 Installation of the Service Runtime Environment

In a first step some preparations are necessary. The server side part of the Service Runtime Environment will be executed by administrators of the SIMPLI-CITY Mobility Services Framework. In order to make use of the first prototype of the Service Runtime Environment, Oracle's Java SE Development Kit 7 and Apache Maven 3.0.5 need to be installed on the system. In addition, port 8080 has to be available for the Service Runtime

Environment as this port will be used for the Service Registry. If this port is not available, the Service Runtime Environment will not work properly. Further, the Service Runtime Environment requires a valid Internet connection; otherwise, some of the services may fail.

The current prototype was tested on a Linux machine running Ubuntu 12.04, Java(TM) SE Runtime Environment (build 1.7.0_51-b13) and Apache Maven 3.0.5.

In addition, it is required that the environment variable JAVA_HOME is defined as per Oracle guidelines.[1]

To install and deploy the prototype of the Service Runtime Environment the following has to done:

Although a bash script is provided for Ubuntu (as well as its counterpart for Windows), a manual installation is described in the following section:

1. Unpack the provided archive file and change into the resulting folder, where the following directory structure should appear:
   bin/ data/ deploy/ etc/ instances/ lib/ lock log/ system/
2. Navigate into the “bin” folder
3. Type into the terminal: “./karaf”. This will start the Service Runtime Environment. If the output is similar to Figure 6, the Service Runtime Environment should be up and running.

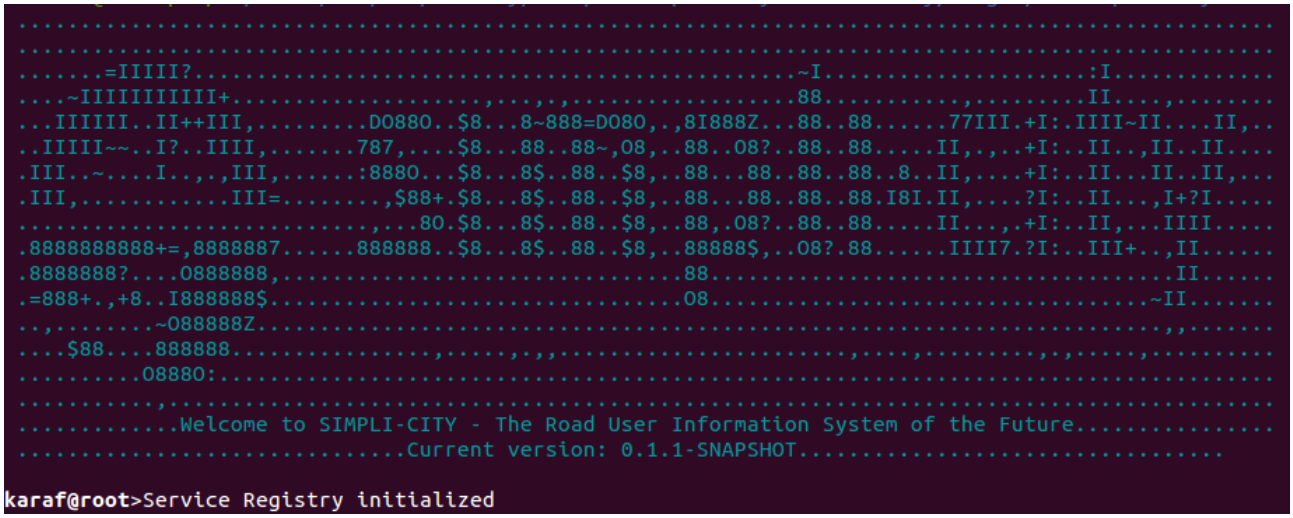


Figure 6: Service Runtime Environment Start Up Output

If no error was printed, the Service Runtime Environment has been started successfully. In addition, the last line indicates that the Service Registry was successfully started.

4.1.2.2 Installation of the Sensor Abstraction and Interoperability Interfaces

The server side component of the Sensor Abstraction and Interoperability Interfaces is executed as a bundle within the Service Runtime Environment. Thus in a first step the Service Runtime Environment has to be started as described in Section 4.1.2.1. In the next step the necessary components have to be loaded within the Service Runtime Environment. This is done by the following four commands, to be executed within the Service Runtime Environment terminal:

[1] http://docs.oracle.com/cd/E19182-01/820-7851/inst_cli_jdk_javahome_t/index.html

Listing 4: Installation of the Server Side Components of the Sensor Abstraction and Interoperability Interfaces within the Service Runtime Environment

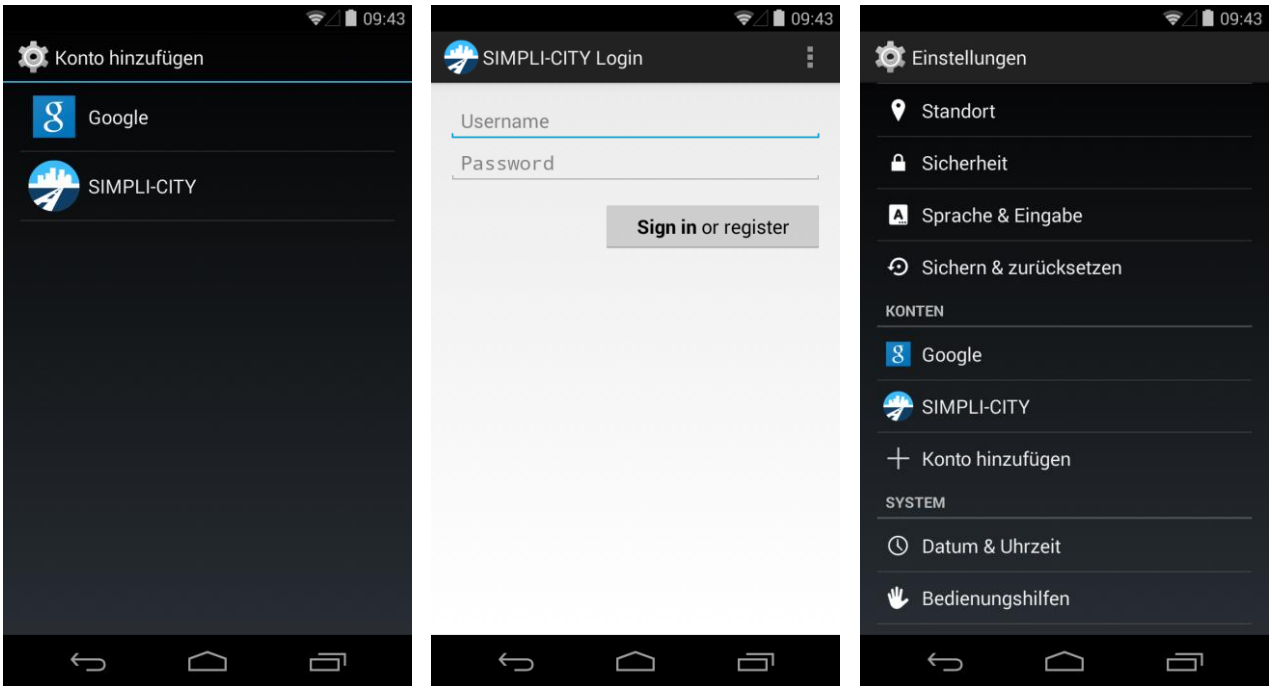
```
install -s mvn:com.google.code.gson/gson/2.0

install -s mvn:eu.simpli-city/eu.simpli-city.sensorabstraction.scsensors/0.0.1-SNAPSHOT

install -s mvn:eu.simpli-city/eu.simpli-city.sensorabstraction.DataBaseConnector/0.0.1-SNAPSHOT

install -s mvn:eu.simpli-city/eu.simpli-city.sensorabstraction.pmaConnector/0.0.1-SNAPSHOT
```

On the PMA side in a first step the app has to be deployed to the Android smartphone. For this the debugging mode has to be activated within Android. To deploy the prototype app the Eclipse IDE can be used. After the code project is loaded into Eclipse the option to execute the app on the connected smartphone is offered once the project is selected to be started as Android application.

The SIMPLI-CITY user account has to be added to the Android smartphone. This can be done in the android settings in the menu accounts as depicted in Figure 7.

In the next step the app has to be started. The app has the SIMPLI-CITY logo as start icon and provides only a very simple user interface, since the app is only a container to start the PMA-based Sensor Abstraction and Interoperability Interfaces component that consists of just one background service. This app only provides a *start* button, to start the mentioned background service, and the opportunity to configure the server address (see Figure 8). After the start of the background service, all sensors are available via the RESTful interfaces provided by the server side component of the Sensor Abstraction and Interoperability Interfaces. Thus PMA sensors can be accessed via the server side. The prototype app is just for demo purposes, in the second prototype the functionality will be available via the Application Runtime Environment.

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201



(a) (b) (c)

Figure 7: SIMPLI-CITY User Account Management on the PMA

Figure 7 shows the SIMPLI-CITY user account management on the PMA. The account type the SIMPLI-CITY user is listed in the menu to add new user accounts (a). To add a new account, the SIMPLI-CITY user name and the respective password has to be entered (b). Afterwards, the user account is listed in the systems settings menu (c).

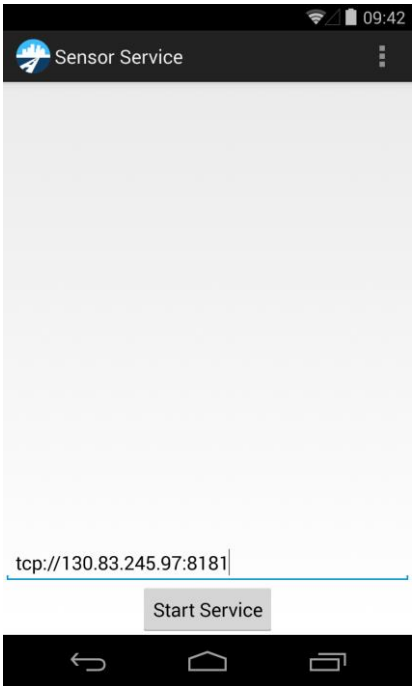


Figure 8: Simple Demonstrator App of the PMA-based Sensor Abstraction and Interoperability Interfaces Component

4.1.3 Car Sensor Data Provision

The software modules to be installed in the IVI platform for realizing car sensor data provision are:

- Car Sensor Data Provision XML file: This file filters the signals set to be sent to the mobile device from all signals generated by the vehicle. It refers to list all car signals sent to SIMPLI-CITY platform coded in XML Format. An extract of how this file is structured is shown in Listing 5.

Listing 5: Example of Car Sensor Data Provision XML File

```
<?xml version="1.0" encoding="UTF-8"?>
<F2CAPSignalCollection xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <SignalCollection>
      ...
      <F2CAPSignal>
         <F2CAP_Signal>INFO_DriverPresent</F2CAP_Signal>
         <F2CAPSignalId>5000</F2CAPSignalId>

<AssociatedSignal>SigRxNetClassBSBR1RowDriverSeatConfigSts</AssociatedSignal>
      </F2CAPSignal>
      <F2CAPSignal>
         <F2CAP_Signal>INFO_LeftRearPsngrPresent</F2CAP_Signal>
         <F2CAPSignalId>5002</F2CAPSignalId>

<AssociatedSignal>SigRxNetClassBSBR2RowDriverSeatConfigSts</AssociatedSignal>
      </F2CAPSignal>
      <F2CAPSignal>
         <F2CAP_Signal>INFO_CentralRearPsngrPresent</F2CAP_Signal>
         <F2CAPSignalId>5003</F2CAPSignalId>

<AssociatedSignal>SigRxNetClassBSBR2RowCentralSeatConfigSts</AssociatedSignal>
      </F2CAPSignal>
   </SignalCollection>
</F2CAPSignalCollection>
```

- Car Sensor Data Provision PROTO file: This .proto file defines messages and services that will be used for interactions between the mobile device and the IVI platform. The code generated by compiling .proto file using the protocol buffer compiler *protoc* has to be installed on both the vehicle and mobile device to manage data marshalling and un-marshalling. In particular, the compiled modules installed on the in-vehicle platform allow managing and serializing the vehicle data that are then sent over the Bluetooth Serial Port Profile to the nomadic device where the PMA instance is deployed. At the same time also the nomadic device can send messages to the IVI platform (e.g. control messages). For the vehicle data, marshalling, i.e. the translation from manufacturer dependent proprietary formats into aggregated and structured ProtocolBuffer data, is performed by the vehicle gateway and un-marshalling, i.e., the translation from ProtocolBuffer data into single sensor readings represented in the common JSON data format, is performed at the target activity (e.g. Android smartphone). Contrariwise for the control messages marshalling is performed by the source activity and un-marshalling is performed by the vehicle gateway.

All defined message types are shown in the listings below:

Listing 6: Car Sensor Data Provision PROTO File – PMA to Head Unit

```
package it.crf.simplicity;
/*
       This message is sent from PMA to Head Unit
*/
message FromPmaMessage {
       // Command to restart the communication
       optional bool restart = 5;
}
```

Listing 7: Car Sensor Data Provision PROTO File – Head Unit to PMA (Part I)

```
/*
       This message is sent from Head Unit to PMA
*/

message FromHuMessage {
       // It is possible to send a SensorMessage or a DeviceMessage
       optional SensorMessage sensor_msg=1;
       optional DeviceMessage device_msg=2;

       /*
              SensorValueMessage are sent after that a SensorMessage has sent.
              In this way when values are updated frequently is not necessary to
build a whole SensorMessage.
       */
       repeated SensorValueMessage value_updates = 3;

       // Milliseconds from 1/1/1970
       optional uint64 timestamp = 4;
}

message AccuracyMessage{
       optional GenericValue value = 1;
       optional string unit = 2;
}

message SensorMessage{
       optional SensorTypeMessage sensor_type_msg=1;
       repeated SensorValueMessage sensor_value_msg=2;
       optional string objectId = 3;
       optional string parent = 4;

}
```


http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

Listing 8: Car Sensor Data Provision PROTO File – Head Unit to PMA (Part II)

```
message GenericValue {
	optional int64 int_val = 1;
	optional double float_val = 2;
	optional string text_val = 3;
	optional bytes bin_val = 4;
}

message DeviceMessage {
	optional string deviceType =1;
	repeated SensorMessage connectedSensors = 2;
	repeated DeviceMessage connectedDevices = 3;
	optional string objectId = 4;
	optional string parent = 5;
}

/*
	This message is sent from Head Unit to PMA
*/

message SensorValueMessage{

	//handle is necessary to map the value to the correct SensorMessage
	required uint32 handle = 1;

	optional string name = 2;
	optional GenericValue value = 3;
	optional string unit = 4;
	optional AccuracyMessage accuracy = 5;
}
message SensorTypeMessage{
	optional string type=1;
	optional string sub_type=2;
}
```

4.1.4 End User (PMA) Data Access

The end user (PMA) or user-centric data extraction from personal data sources is provided via the RESTful interface of the Sensor Abstraction and Interoperability Interfaces and data is returned in JSON sensor schema format. The data format is described in detail in Section 5.1.2.1.

The sensor abstraction mechanism for extracting data from the personal data sources and producing the data in the JSON sensor schema has the following deployment description.

The user data is extracted from the PMA, i.e., the users' Android smartphone, by the use of the PMA-based components of the Sensor Abstraction and Interoperability Interfaces. For the first prototype of the Sensor Abstraction and Interoperability interfaces in D4.3.1 these components are provided as Android application (app).

The PMA based Sensor Abstraction and Interoperability Interfaces are an Android project named *SensorAbstractionService*. This project has a dependency to the project SCSensors. Both projects are provided as software packages in a zip archive with this deliverable. To deploy the background service prototype to an Android device one has to use an Eclipse IDE with Android plugin and import both projects. Then the background

service can easily be deployed on the mobile device as an Android app. For this, the debugging mode has to be activated within Android on the smartphone. In a next step the SIMPLI-CITY user account has to be added to the Android system settings as described in in Section 4.1.2.2. Then the app has to be started. The app has the SIMPLI-CITY logo as start icon and provides only a very simple user interface, since the app is only a container to start the PMA-based Sensor Abstraction and Interoperability Interfaces component that consists of just one background service. This app only provides a *start* button, to start the mentioned background service, and the opportunity to configure the server address (see Section 4.2 in deliverable D4.3.1). After the start of the background service, all sensors are available via the RESTful interfaces provided by the server side component of the Sensor Abstraction and Interoperability Interfaces. Thus PMA sensors and personal user data as virtual sensors can be accessed via the server side. The prototype app is just for demo purposes, in the second prototype of the Sensor Abstraction and Interoperability Interfaces the functionality will be available via the Application Runtime Environment.

To make use of the RESTful interfaces no installations have to be done. The service can be called from a standard Web browser. The input parameter is the SIMPLI-CITY user ID. A detailed description of the usage of these interfaces is given in Section 5.1.2.3.

5 Execution and Usage of the Software

This section describes how to use the different subcomponents of the prototype.

5.1 Server Side

5.1.1 Open Data Access

The REST APIs for calling current values of a dataset in raw (CSV) and transformed (RDF) format using the respective parameters *plain* and *transformed* which have been developed and expanded over the project so far for Dublin and Bologna cities are listed below. Note that the IP listed is currently only accessible internally to IBM but will be made accessible to the SIMPLI-CITY backend services.

Dublin City Council Trip Times (DCC_TRIPS) in CSV format:

- http://9.162.92.201:8080/rest/read/data/plain?sources=DCC_TRIPS&frequency=1
- http://9.162.92.201:8080/rest/read/data/plain?sources=DCC_TRIPS&frequency=10

Dublin Weather (DU_WEATHER) in CSV format:

- http://9.162.92.201:8080/rest/read/data/plain?sources=DU_WEATHER&frequency=10
- http://9.162.92.201:8080/rest/read/data/plain?sources=DU_WEATHER&frequency=30

Dublin City Council Trip Times (DCC_TRIPS) in RDF format:

- http://9.162.92.201:8080/rest/read/data/transformed?sources=DCC_TRIPS&frequency=1
- http://9.162.92.201:8080/rest/read/data/transformed?sources=DCC_TRIPS&frequency=10
- Dublin Weather (DU_WEATHER) in RDF format
- http://9.162.92.201:8080/rest/read/data/transformed?sources=DU_WEATHER&frequency=30

Bologna Trip Times (BO_TRIPS) in CSV format:

- http://9.162.92.201:8080/rest/read/data/plain?sources=BO_TRIPS&frequency=10
- http://9.162.92.201:8080/rest/read/data/plain?sources=BO_TRIPS&frequency=30

Bologna Weather (BO_WEATHER) in CSV format:

- http://9.162.92.201:8080/rest/read/data/plain?sources=BO_WEATHER&frequency=10
- http://9.162.92.201:8080/rest/read/data/plain?sources=BO_WEATHER&frequency=30

Bologna Trip Times (BO_TRIPS) in RDF format:

- http://9.162.92.201:8080/rest/read/data/transformed?sources=BO_TRIPS&frequency=10
- http://9.162.92.201:8080/rest/read/data/transformed?sources=BO_TRIPS&frequency=30

Bologna Weather (BO_WEATHER) in RDF format:

- http://9.162.92.201:8080/rest/read/data/transformed?sources=BO_WEATHER&frequency=10
- http://9.162.92.201:8080/rest/read/data/transformed?sources=BO_WEATHER&frequency=30

The following two screenshots show examples of what the returned data looks like also for the Bologna city 'elaborated' trip data (identified by BO_TRIPS') in CSV and transformed formats.

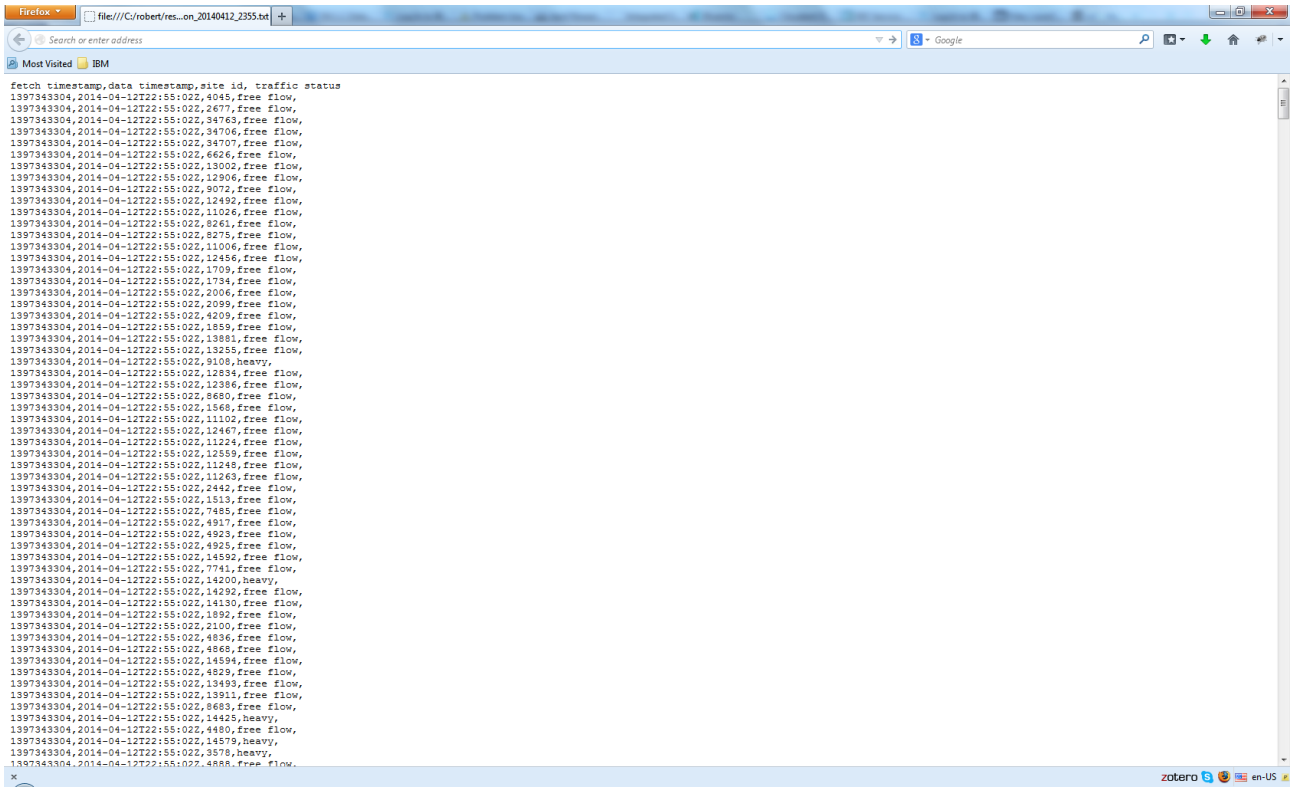


Figure 9: Selection of Raw (Plain) Data for Bologna City Trips Info

Figure 9 shows the fetched data for the BO_TRIPS data stream parsed into the 'plain' CSV format.

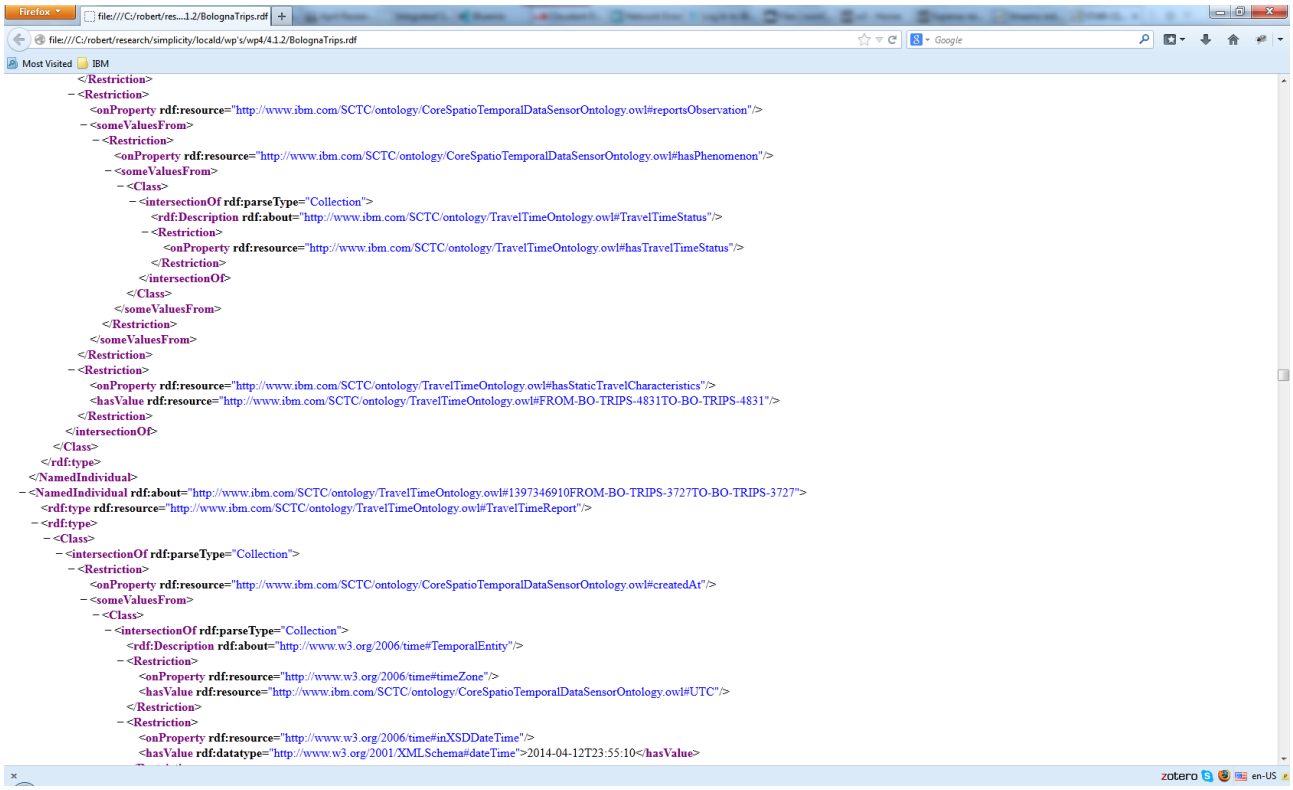


Figure 10: Selection of Transformed Data for Bologna Trips Info

Figure 10 shows the fetched data for the Bologna ‘Elaborated’ data stream identified by BO_TRIPS’, transformed into the SIMPLI-CITY Unified Data Model RDF format.

In addition the STAR-CITY GUI (**S**emantic **T**raffic **A**nalytics and **R**easoning for **CITY**) was developed to give a map-based representation of Open Data sources (traffic, weather, roadwork and event information) and some Twitter feeds allowing a visualization of analysis and diagnosis of road traffic anomalies. A screenshot of STAR-CITY for Bologna is given below adding to the detail described for Dublin City given in deliverable D4.4.1 (User-centric and Open Data Management). A limited version for Dublin City is publicly available at http://dublinked.ie/sandbox/star-city/, however at times it may not be accessible due to maintenance. The latest developed version is currently accessible only from an IBM machine, but will be made available for SIMPLI-CITY server access.

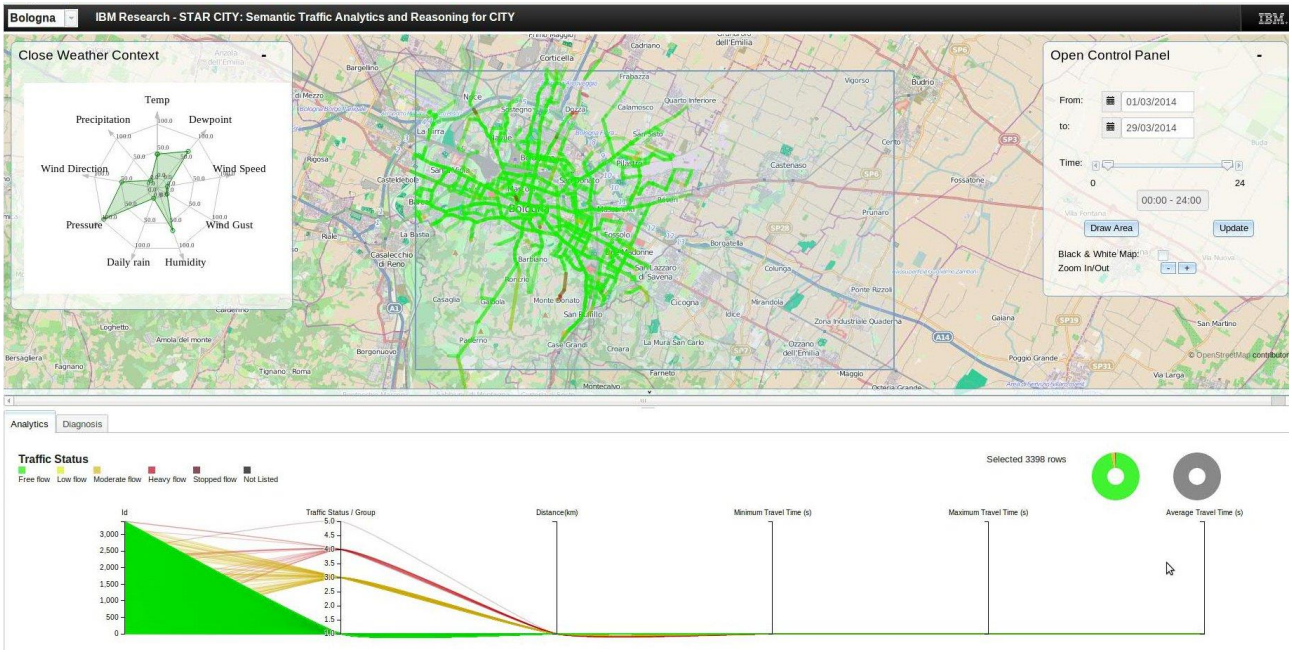


Figure 11: Screenshot from STAR-CITY for Bologna

The above figure shows a Bologna city map with coloured road segments that indicate the traffic status of the respective roads and a weather chart on the top left. This is followed by Traffic Status Analytics screen with the road-segment colour-coding given.

In addition some screenshots from STAR-CITY for Dublin are shown to demonstrate the diagnosis and prediction features developed so far by clicking on the diagnosis or prediction tabs.

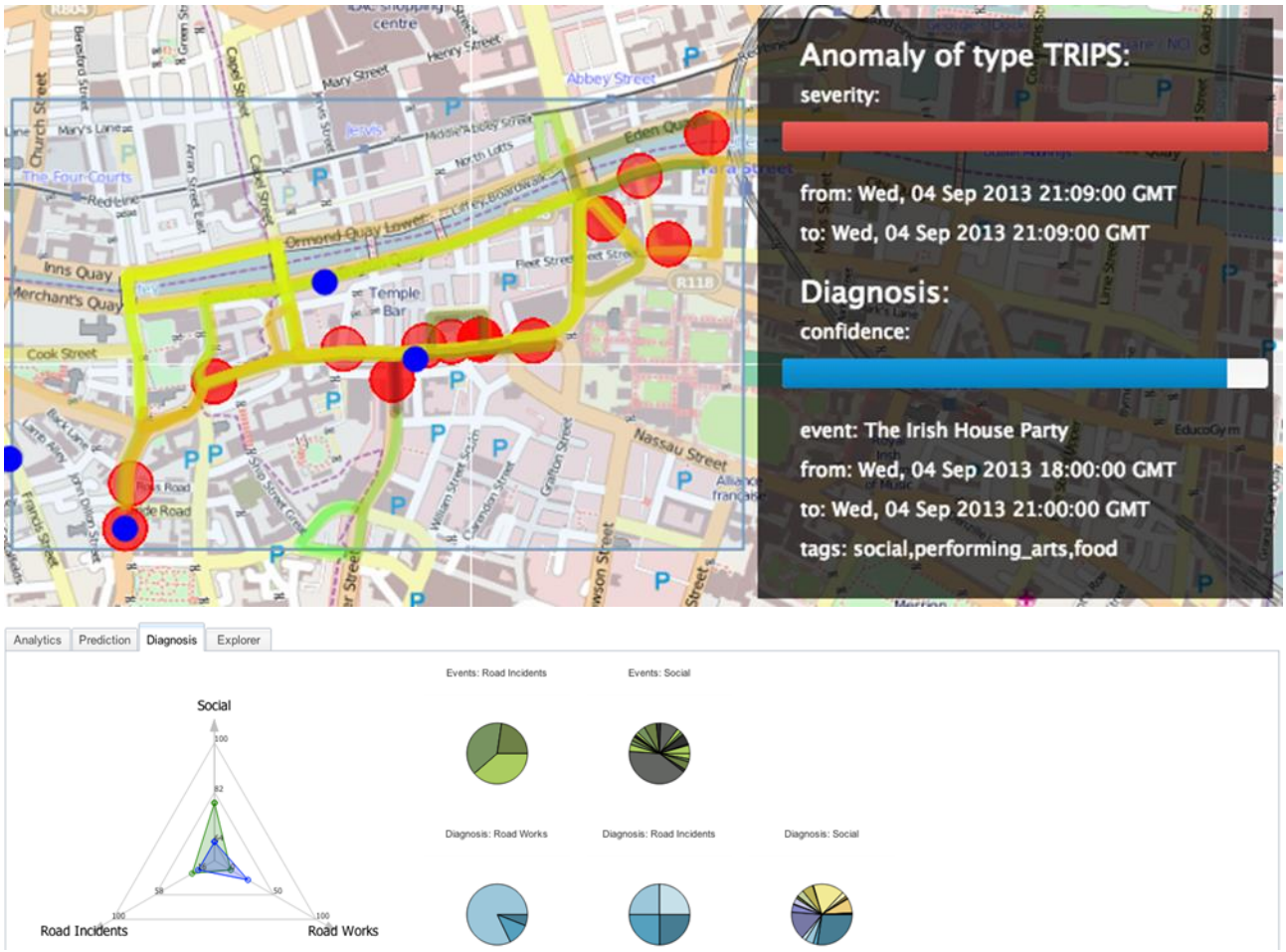


Figure 12: STAR-CITY Screenshots for Showing Traffic Anomalies and Diagnosis of Traffic Status

The above screenshots show the Diagnosis STAR-CITY view for a selected time window and spatial box on the Dublin city map. On the map the districts are outlined with numbers for anomalies and coloured dots for contributing factors. The lower part of the screenshot shows graphically the overall and relative contributions of different factors and incidents e.g. road-works and events occurring at that time.

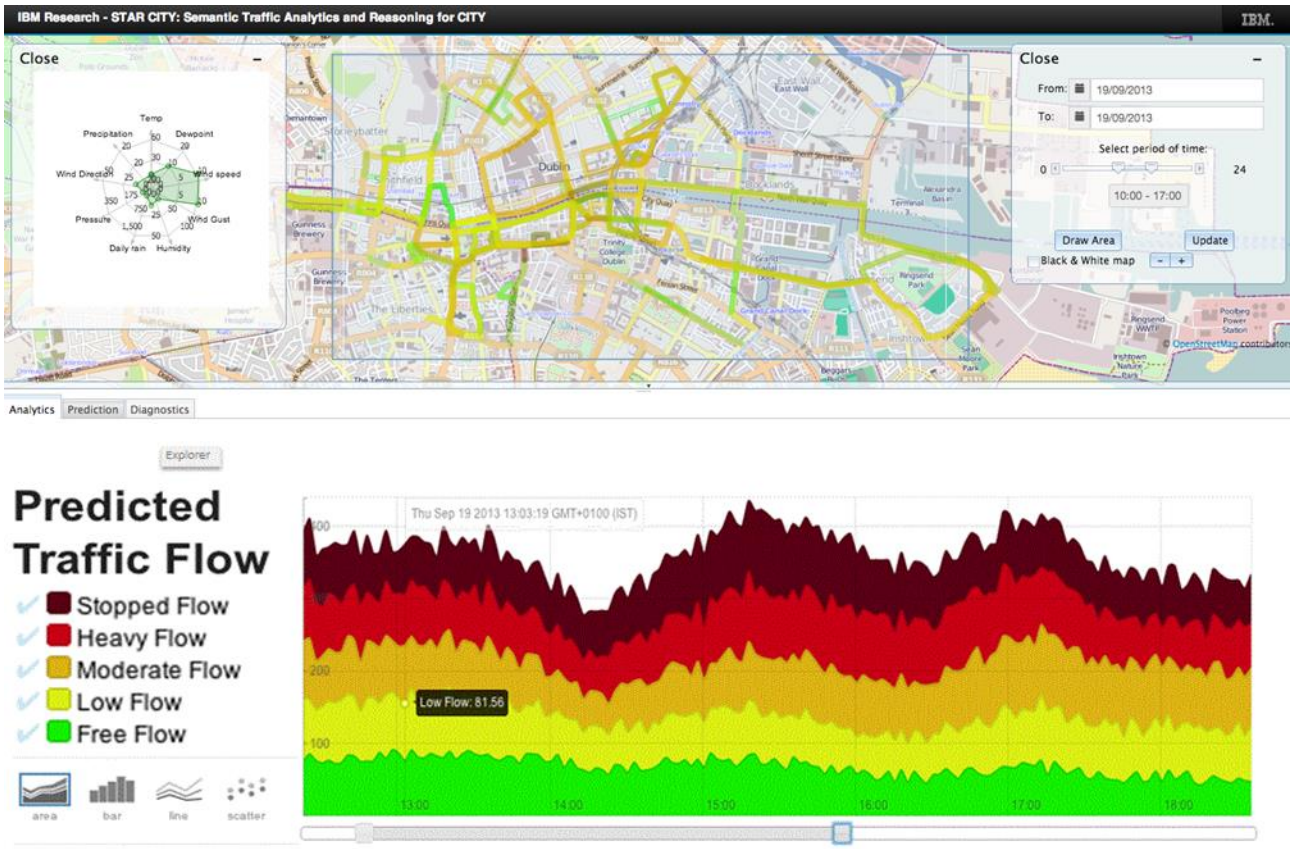


Figure 13: STAR-CITY Screenshots Related to 'Prediction' for Routes Shown on the Map Above

Some more screenshots are given above of the STAR-CITY implementation to demonstrate some developed aspects of the Data Processing component using the SIMPLI-CITY Unified Data Model.

5.1.2 General Sensor Data Access

This section describes how to use the different subcomponents of the Sensor Abstraction and Interoperability Interfaces to request sensor data. The core scope of this prototype was the integration of the PMA side component and the server based component of the Sensor Abstraction and Interoperability Interfaces. This allows a unified access to the local sensor data sources of the PMA, a connected car and the user data available through the PMA, i.e., the users' calendar and contacts data. Once the app on the PMA is started, as described in Section 4.1.4, the PMA-based Sensor Abstraction and Interoperability Interfaces component automatically connects to the counterpart component on the server side and automatically maintains the connection. The server provides a RESTful interface to request sensor information from the PMA. For a simple test purpose, any Web browser can be used to request this RESTful interface.

5.1.2.1 SIMPLI-CITY Sensor Model

The JSON format is used as common data representation model for sensor data. A lightweight structure is used to consistently represent all sensors and user-related data. Figure 14 gives a hierarchical overview of the data structure. In the hierarchical view, on

the top level there will be an object class that brings the basic properties for devices and sensors. Each object brings the following properties:

- Object: The respective class descriptor (sensor, device)
- Timestamp: The point in time of object creation
- ObjectID: A unique 128 bit identifier
- Parent: The object ID of the parent object if available

However, the type object will never be directly initiated. Instead, objects of the derived classes sensor or device will be used. A device can be connected from 0 to n sensors and 0 to m other devices. That means there could be a cascade of devices, each in between can have sensors but do not have to. Sensors are always connected to devices and have 1 to n values. That means a sensor has at minimum one corresponding value. All sensors have a field describing their type. Sensor values are provided as array; so, complex sensor values can be represented as an array of single values. Each sensor value can have, but does not have to, an accuracy property that has a value describing the accuracy range and a corresponding unit. If a data object is not available the value *NULL* is provided.

To exemplify the previously described JSON sensor data object hierarchy in Figure 14, in the following a step-by-step example is provided.

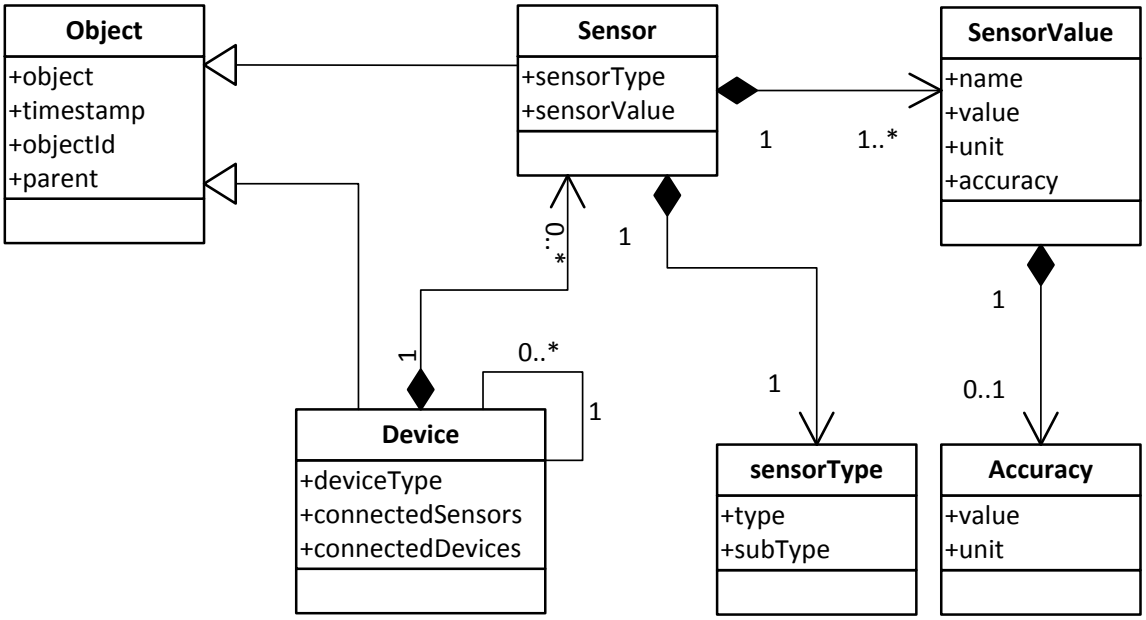


Figure 14: JSON Object Structure for Sensor-related Data

For simplification all 128-bit identifiers have been replaced by more readable elements. The example consists of the following devices:

- Device: PMA; ObjectID: ID-AAA
- Device: CAR; ObjectID: ID-BBB

These devices have the following sensors:

- PMA sensor: location GPS sensor; ObjectID: ID-CCC
- PMA sensor: location address virtual sensor; ObjectID: ID-DDD
- CAR sensor: engine speed; ObjectID: ID-EEE
- CAR sensor: accelerator pedal; ObjectID: ID-FFF

The top-level object is a device (ID-AAA) that consists of two sensors (ID-SENSOR-CCC, ID-SENSOR-DDD) and a connected device, the car (ID-BBB). Additionally a timestamp is provided and the device has no parent. In this context parent directly points to the physical object hierarchy and thus is a pointer to the parent object the device or sensor is connected to. The JSON expression is given in Listing 9.

Listing 9: JSON Expression of the Sample Device ID-AAA

```
{
   "object": "device",
   "deviceType": "PMA",
   "connectedSensors": [
      {
         "sensorType": {
            "type": "location",
            "subType": "coordinates"
         },
         "sensorID": "ID-SENSOR-CCC"
      },
      {
         "sensorType": {
            "type": "location",
            "“subType”": "address"
         },
         "sensorID": "ID-SENSOR-DDD"
      }
   ],
   "connectedDevices": [
      {
         "deviceType": "car",
         "deviceID": "ID-BBB"
      }
   ],
   "timeStamp": "2013-08-28T10:27:10.000Z",
   "objectID": "ID-AAA",
   "parent": null
}
```


http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

The first sensor connected to the device (ID-AAA) is a location sensor (ID-SENSOR-CCC) that provides the location in coordinates. The sensor has two values, one for longitude, one for latitude, and both with an accuracy value. The sensor also provides a timestamp and a reference to the parent device. The respective JSON expression is given in Listing 10.

Listing 10: JSON Expression of a GPS-based Location Sensor

```
{
   "object": "sensor",
   "sensorType": {
      "type": "location",
      "subtype": "coordinates"
   },
   "sensorValue": [
      {
         "name": "latitude",
         "value": "50.00526032632832",
         "unit": "degree",
         "accuracy": {
            "unit": "meters",
            "value": 24
         }
      },
      {
         "name": "longitude",
         "value": "8.647010091683981",
         "unit": "degree",
         "accuracy": {
            "unit": "meters",
            "value": 24
         }
      }
   ],
   "objectID": "ID-SENSOR-CCC",
   "timeStamp": "2013-08-28T10:27:10.000Z",
   "parent": "ID-AAA"
}
```


http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

The second sensor, connected to device (ID-AAA) is also a location sensor (ID-SENSOR-CCC), but this one provides the location address. The sensor has four values: street, house number, postal code, city and country, all without accuracy. The sensor also provides a timestamp and a reference to the parent device. The respective JSON expression is given in Listing 11.

Listing 11: JSON Expression of an Address-based Location Sensor

```
{
   "object": "sensor",
   "sensorType": {
      "type": "location",
      "subtype": "address"
   },
   "sensorValue": [
      {
         "name": "street",
         "value": "Rundeturmstraße",
         "unit": null,
         "accuracy": null
      },
      {
         "name": "houseNumber",
         "value": "10",
         "unit": null,
         "accuracy": null
      },
      {
         "name": "postalCode",
         "value": "64283",
         "unit": null,
         "accuracy": null
      },
      {
         "name": "city",
         "value": "Darmstadt",
         "unit": null,
         "accuracy": null
      },
      {
         "name": "country",
         "value": "Germany",
         "unit": null,
         "accuracy": null
      }
   ],
   "objectID": "ID-SENSOR-DDD",
   "timeStamp": "2013-08-28T10:27:10.000Z",
   "parent": "ID-AAA"
}
```

The second device (ID-BBB) is the car connected to the top-level device (ID-AAA). This has three sensors and no connected devices. Additionally a timestamp is provided and the reference to the parent device. The respective JSON expression is given in Listing 12.

Listing 12: JSON Expression of the Connected Device Car

```
{
   "object": "device",
   "deviceType": "car",
   "connectedSensors": [
      {
         "sensorType": {
            "type": "vehicle",
            "subtype": "engineSpeed"
         },
         "sensorID": "ID-SENSOR-EEE"
      },
      {
         "sensorType": {
            "type": "vehicle",
            "subtype": "acceleratorPedal"
         },
         "sensorID": "ID-SENSOR-FFF"
      },
      {
         "sensorType": {
            "type": "vehicle",
            "subtype": "vehicleSpeedOdometer"
         },
         "sensorID": "ID-SENSOR-GGG"
      }
   ],
   "connectedDevices": null,
   "timeStamp": "2013-08-28T10:27:10.000Z",
   "objectID": "ID-BBB",
   "parent": "ID-AAA"
}
```


http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

The third sensor is connected to device (ID-BBB) and is an engine speed sensor (ID-SENSOR-EEE). The value has also accuracy, additionally a timestamp is provided and the reference to the parent device. The respective JSON expression is given in Listing 13.

Listing 13: JSON Expression of a Speed Sensor

```
{
   "object": "sensor",
   "sensorType": {
      "type": "vehicle",
      "subtype": "engineSpeed"
   },
   "sensorValue": [
      {
         "name": "engineSpeedRPM",
         "value": 3254,
         "unit": "rpm",
         "accuracy": {
            "unit": "percentage",
            "value": 0.05
         }
      }
   ],
   "objectID": "ID-SENSOR-EEE",
   "timeStamp": "2013-08-28T10:27:10.000Z",
   "parent": "ID-BBB"
}
```

The fourth sensor is connected to device (ID-BBB) and is a sensor of the position of the accelerator pedal of the car (ID-SENSOR-FFF). The value has no accuracy, additionally a timestamp is provided and the reference to the parent device. The respective JSON expression is given in Listing 14.

Listing 14: JSON Expression of An Accelerator Pedal Sensor

```
{
   "object": "sensor",
   "sensorType": {
      "type": "vehicle",
      "subtype": "acceleratorPedal"
   },
   "sensorValue": [
      {
         "name": "acceleratorPedalPosition",
         "value": 76,
         "unit": "percentage",
         "accuracy": null
      }
   ],
   "objectID": "ID-SENSOR-FFF",
   "timeStamp": "2013-08-28T10:27:10.000Z",
   "parent": "ID-BBB"
}
```


http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

The fifth sensor is connected to device (ID-BBB) and is a vehicle speed odometer sensor of the car (ID-SENSOR-GGG). The value has no accuracy, additionally a timestamp is provided and the reference to the parent device. The respective JSON expression is given in Listing 15.

Listing 15: JSON Expression of a Speed Odometer Sensor

```
{
   "object": "sensor",
   "sensorType": {
      "type": "vehicle",
      "subtype": "vehicleSpeedOdometer"
   },
   "sensorValue": [
      {
         "name": "vehicleSpeedOdometer",
         "value": 76,
         "unit": "km/h",
         "accuracy": null
      }
   ],
   "objectID": "ID-SENSOR-GGG",
   "timeStamp": "2013-08-28T10:27:10.000Z",
   "parent": "ID-BBB"
}
```

The hierarchical structure of this concrete example is likewise depicted in Figure 15.

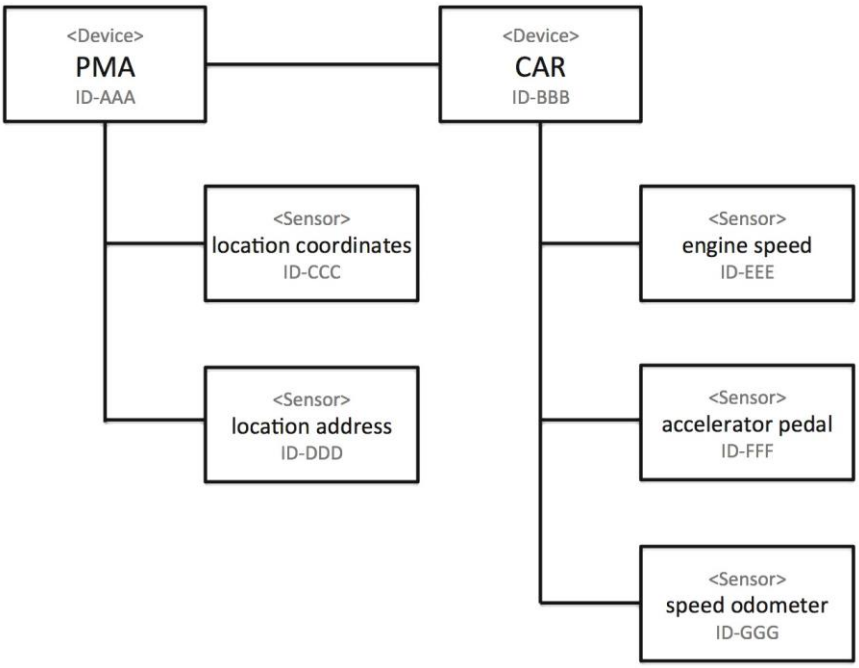


Figure 15: Hierarchy of the Sensor Model Example

5.1.2.2 JSON Data Format Skeleton

In the following examples, the JSON data format skeleton for a device is given in Listing 16 and for a sensor reading is given in Listing 17.

Listing 16: JSON Skeleton for a Device

```
{
   "object": "device",
   "deviceType": "…",
   "connectedSensors": [
      {
         "sensorType": {
            "type": "…",
            "subType": "…"
         },
         "sensorID": "…"
      }
   ],
   "connectedDevices": [
      {
         "deviceType": "…",
         "deviceID": "…"
      }
   ],
   "timeStamp": "…",
   "objectID": "…",
   "parent": "…"
}
```

Listing 17: JSON Skeleton for Sensor Data

```
{
   "object": "sensor",
   "sensorType": {
      "type": "…",
      "subtype": "…"
   },
   "sensorValue": [
      {
         "name": "…",
         "value": "… ",
         "unit": "…",
         "accuracy": {
            "unit": "…",
            "value": "…"
         }
      }
   ],
   "objectID": "…",
   "timeStamp": "…",
   "parent": "…"
}
```

A description of the used terms is given in the following:

- object: It describes if the data object represents a sensor or a device. The only possible values are *sensor* and *device*.

- objectID: A unique 128 bit identifier, corresponding to RFC 4122, represented as 32 byte hexadecimal string
- timeStamp: A string that represents the time in UTC time corresponding to ISO_8601, e.g., 2013-08-28T10:27:10.000Z.
- parent: A unique 128 bit identifier that acts as pointer to a parent object, i.e., a pointer from a sub device to its parent device or from a sensor to its corresponding device.
- connectedSensors: An array of sensors connected to a device, containing a description name, derived from the sensor type and sub type, and the unique 128 bit identifier of the respective sensor.
- connectedDevices: An array of devices connected to a device, containing a description name and the unique 128 bit identifier of the respective device.
- sensorType: Describes the type of the respective sensor by the two strings type and sub type. The used vocabulary for sensor description will be derived from the SIMPLI-CITY Data Model. The Sensor Abstraction Services will provide interfaces to retrieve a list of all available sensor and value types.
- sensorValue: Since sensors, especially virtual sensors like calendar entries, can have multiple values, this is an array of sensor value objects. Each sensorValue consists of a description name, e.g., temperature, pressure, latitude, longitude, street, city, etc., and a corresponding value. If possible a unit is given, else the value of unit is NULL. Optionally also an accuracy is given with a value that describes the range in which the sensor value can differ, e.g., 2 degrees for a temperature value results in that the corresponding temperature value is in a range plus or minus 2 degrees of the given value, else the value of accuracy is null.

Finally it should be marked that data could also be a set of multiple sensor values, e.g., especially if a range of values from a start point in time to an end point in time is requested, that results in an array of sensor objects, each with a different timestamp.

5.1.2.3 Usage of the General Sensor Data Access

In the following, the workflow for a test request of data from the PMA is explained in all necessary steps. This example uses the server address 130.83.245.97 and the port for the RESTful interfaces is 4321.

The demonstration workflow for a test execution is as follows: In a first step the user ID is requested via a RESTful interface, as depicted in Listing 18. This is just necessary for test purposes to get knowledge about the user ID because all PMA-based requests are based on the user ID. The input parameters are the user name and the respective password. For simplification and ease during debugging, the password is transferred in plain text. In the final prototype (D4.3.2) this will be changed. Furthermore, in this prototype, users are manually entered to the server side database. A web-based user management will be provided with the implementation of deliverable D4.3.2.

Listing 18: Request of the User ID

```
http://130.83.245.97:4321/cxf/pmaconnector/register?username=simplicityuser1&password=gmsmplcty1
```

The response contains the unique user ID, which is necessary as input parameter to request data from the PMA of a specific user. With this knowledge another RESTful interface, as depicted in Listing 19, provides access to the device information of the users'

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

PMA. No data is cached, all requests are passed to the PMA, and the response is sent back to the server side. The data structure of devices and sensors is described in Sections 5.1.2.1 and 5.1.2.2 and in more detail in deliverable D4.1.1. The input parameter is the previously requested user ID.

Listing 19: Request of the User's PMA Information

```
http://130.83.245.97:4321/cxf/pmaconnector/pma?userID=ba9e2620-aac6-11e3-a5e2-0800200c9a66
```

The return value of this request is the device description in the SIMPLI-CITY JSON data format as described in deliverable D4.1.1. In the following Listing 20, the device description of a PMA based on a Nexus 4 is given. The description contains a unique ID of the PMA device. In this example, no sub device is connected to the PMA, i.e., no car is connected. The available sensors are listed as an Array of Sensor JSON objects that all provide a unique ID. This ID can be used to directly request a sensor.

Listing 20: Result to the Request of the User's PMA Information

```
{
   "connectedSensors": [
      {
         "sensorType": {
            "type": "userdata",
            "subtype": "calendar"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "11b04404-b0b4-4cb3-931a-f94e54cd4004"
      },
      {
         "sensorType": {
            "type": "movement",
            "subtype": "linear_acceleration"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "141fb8a1-7599-42c0-a287-b9ec06d2899a"
      },
      {
         "sensorType": {
            "type": "movement",
            "subtype": "rotation_vector"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "69b9e39f-6b48-4769-a9b9-abf5e07b93e0"
      },
      {
         "sensorType": {
            "type": "environment",
            "subtype": "light"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "13044b4d-9fab-46f1-9154-f087ad702575"
      },
      {
         "sensorType": {
            "type": "environment",
            "subtype": "proximity"
         },
```


http://www.simpli-city.eu/ | Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

```
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "793e2009-c220-436e-880f-583fd5201fa1"
      },
      {
         "sensorType": {
            "type": "environment",
            "subtype": "magnetic_field"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "1a90842d-872d-45d2-828e-bb5558882d65"
      },
      {
         "sensorType": {
            "type": "environment",
            "subtype": "air_pressure"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "c9891d78-6c8f-482b-be30-69889b77fba0"
      },
      {
         "sensorType": {
            "type": "userdata",
            "subtype": "contacts"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "fe550ee4-99ac-4284-bbc9-906c2b572240"
      },
      {
         "sensorType": {
            "type": "environment",
            "subtype": "raw_magnetic_field"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "69449fde-c9f4-4441-8ab3-e966b4de98b2"
      },
      {
         "sensorType": {
            "type": "movement",
            "subtype": "gyroscope"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "c74c058b-59e1-4102-aaf8-942ede4a3465"
      },
      {
         "sensorType": {
            "type": "movement",
            "subtype": "game_rotation_vector"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "ed8291f0-b849-40c9-9c42-de1e287ddd4b"
      },
      {
         "sensorType": {
            "type": "movement",
            "subtype": "gyroscope"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "f68d8f2b-2ffc-4fe2-8c9a-c29ebc827240"
```


http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

```
      },
      {
         "sensorType": {
            "type": "orientation",
            "subtype": "gravity"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "21146ea4-30e8-4ab6-93c6-05e760e8f6d5"
      },
      {
         "sensorType": {
            "type": "movement",
            "subtype": "accelerometer"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "a569726f-7108-4dde-ac42-40474f88c557"
      },
      {
         "sensorType": {
            "type": "orientation",
            "subtype": "device"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "5133cb4c-fe3c-435d-94ff-ee54c54da0a9"
      },
      {
         "sensorType": {
            "type": "movement",
            "subtype": "significant_motion"
         },
         "timeStamp": "2014-05-06T07:49:48Z",
         "objectID": "831abff5-ff29-4315-9208-a04f69c5d5ca"
      }
   ],
   "connectedDevices": [],
   "deviceType": "PMA",
   "object": "device",
   "timeStamp": "2014-05-06T07:49:48Z",
   "objectID": "4b168d73-3722-43df-896b-482ac4aba724"
}
```


These listed sensors contain all built-in sensors of the PMA device, as well as the virtual sensors *calendar* and *contacts*. A request of these virtual sensors directly allows requesting personal user data. To request one of the listed sensors, the RESTful interface can be used as depicted in Listing 21. The input parameter is the unique ID of the respective sensor.

Listing 21: Request of a Specific Sensor

```
http://130.83.245.97:4321/cxf/pmaconnector/sensor?sensorID=4e7224b1-a6bd-49f7-bfc0-
15b20b7db268
```

As an example, in Listing 22, the result of a request to the PMAs accelerometer sensor is given.

Listing 22: Result of a Request to the PMA's Accelerometer Sensor


```
{
   "sensorType": {
      "type": "movement",
      "subtype": "accelerometer"
   },
   "sensorValue": [
      [
         {
            "name": "name",
            "value": "0.28689575",
            "unit": "unit",
            "accuracy": {
               "unit": "unit",
               "value": "value"
            }
         },
         {
            "name": "name",
            "value": "0.023361206",
            "unit": "unit",
            "accuracy": {
               "unit": "unit",
               "value": "value"
            }
         },
         {
            "name": "name",
            "value": "9.624054",
            "unit": "unit",
            "accuracy": {
               "unit": "unit",
               "value": "value"
            }
         }
      ]
   ],
   "object": "sensor",
   "timeStamp": "2014-05-06T07:53:34Z",
   "objectID": "a569726f-7108-4dde-ac42-40474f88c557",
   "parent": "4b168d73-3722-43df-896b-482ac4aba724"
}
```


5.1.3 Car Sensor Data Provision

It should be noticed that any installation on the IVI platform should be done by adequately trained experts. Moreover any application installed in the car should follow a proper process of validation and quality assessment that can be different for each Vehicle manufacturer.

Given these premises, after the installation of the software modules generated by compiling `.proto` file described as described in Section 4.1.3, car sensor data provision from the vehicle to the PMA can be started.

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

This functionality can be invoked directly from the IVI platform.

Before enabling the car sensor data ürovision functionality, the IVI platform should be connected to the mobile device via Bluetooth.

Where necessary, a different set of car data to be transmitted can be possibly selected editing the XML Car Sensor Data Provision filter file, described in Section 4.1.3.

5.1.4 End User (PMA) Data Access

The user data can be retrieved from the respective RESTful interfaces, provided by the server based Sensor Abstraction and Interoperability Interfaces. These interfaces and the respective parameters are described in detail in deliverable D4.3.1.

In general, user data can be accessed in a similar way as already described in Section 5.1.2 for accessing sensors. The used data format is also completely the same JSON data format.

As an example for personal user data, the following Listing 23 depicts the result of a request of the virtual sensor *calendar*. The following example only contains two calendar events. A request to the RESTful interface will return an array with all events in the respective user calendar.

Listing 23: Result of a Request to the Calendar Data


```
{
   "sensorType": {
      "type": "userdata",
      "subtype": "calendar"
   },
   "sensorValue": [
      [
         {
            "name": "title",
            "value": "May Day"
         },
         {
            "name": "organizer",
            "value": "en-gb.irish#holiday@group.v.calendar.google.com"
         },
         {
            "name": "description",
            "value": ""
         },
         {
            "name": "dtstart",
            "value": "2015-05-04T00:00:00Z"
         },
         {
            "name": "dtend",
            "value": "2015-05-05T00:00:00Z"
         },
         {
            "name": "allDay",
            "value": "1"
         },
         {
            "name": "duration"
```


http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

```
          },
          {
             "name": "eventLocation",
             "value": ""
          },
          {
             "name": "calendar_displayName",
             "value": "Holidays in Ireland"
          }
       ],
       [
          {
             "name": "title",
             "value": "DSS GROUP Meeting"
          },
          {
             "name": "organizer",
             "value": "sensor.abstraction@gmail.com"
          },
          {
             "name": "description",
             "value": ""
          },
          {
             "name": "dtstart",
             "value": "2014-04-30T14:00:00Z"
          },
          {
             "name": "dtend",
             "value": "2014-04-30T16:00:00Z"
          },
          {
             "name": "allDay",
             "value": "0"
          },
          {
             "name": "duration"
          },
          {
             "name": "eventLocation",
             "value": ""
          },
          {
             "name": "calendar_displayName",
             "value": "sensor.abstraction@gmail.com"
          }
       ]
    ],
    "object": "sensor",
    "timeStamp": "2014-05-06T07:54:08Z",
    "objectID": "11b04404-b0b4-4cb3-931a-f94e54cd4004",
    "parent": "4b168d73-3722-43df-896b-482ac4aba724"
}
```

6 Limitations and Further Developments

All implementations will be completed in the respective final prototypes of WP4 i.e. D4.2, D4.3.2, D4.4.2 and D4.5.2, the Data Model now being fixed, with the limitations of this prototype (D4.1.2) implementation give below.

For Open Data sets the transformation into the SIMPLI-CITY Data Model has been handled for most datasets, however the transformation of sensor and user-centric data for contextualization is still to be implemented. Open Data source collection takes place at the frequency of data updates for the different data sources and also some aggregation takes place e.g. where the data update frequency is at 1 minute intervals a 10 minute and 30 minute frequency is available of aggregated data over the respective time period, however to provide other service-driven frequency values the frequency handler needs to be adapted. These will be implemented as part of D4.4.2.

The required provision of a transformation from OWL2 RDF format into JSON format is currently under investigation in order to define the implementation in relation to the SIMPLI-CITY Data Model. This will also be implemented fully as part of D4.4.2.

The current access to car sensors via the CAN bus is realized by a proprietary solution and will in this way only work with the FIAT test car. However, in addition the OBD interface is used to request car sensor data. This set of sensors is limited but supported by all cars produced within the last decade. It has to be mentioned that most cars do not support all signals specified in ISO15031. The implemented component checks the available signals and provides these signals within the device description as described in Section 5.1.2.

The user-centric data in this prototype is limited to the personal data sources that are natively supported by Android. This is currently mainly the Google account information that consists of calendar and contact information. This will be fully implemented in D4.3.2.

In addition, APIs for interaction and data exchange between components for Open Data, user data, general and car sensor data and also for the Cloud Infrastructure will be developed and completed in the final WP4 prototypes.

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201

7 Summary

In this document – SIMPLI-CITY Data Model Version II – it has been detailed what the scope of the deliverable is, what the requirements coverage is, how to prepare, deploy and execute the implementations and the limitations and further developments have been detailed.

Some implementations based on the SIMPLI-CITY Data Model Version I deliverable D4.1.1 are described and some further detail given where necessary e.g. car sensor data extraction information. The required heterogeneous datasets used in SIMPLI-CITY can be facilitated in the Unified Data Model so that the elements of the SIMPLI-CITY Data Model are now fixed.

In particular the Open Data management and overall Data Processing infrastructure has been detailed, including some additional data transformations and screenshots for Bologna using STAR-CITY, adding to those given in D4.4.1 for Dublin City. Thus raw (CSV format) Open Data has been exploited and transformed into the SIMPLI-CITY Unified Data Model (RDF format) so that contextualization and reasoning can take place over these heterogeneous datasets.

It has also been shown how user-centric data in the form of a personal data source Google account has been provided via the sensor JSON format and which can be used to drive user-centric diagnosis related to the user's calendar events as in D4.4.1.

For car sensor data access extraction into JSON sensor data format has been described. In the case of general sensor access prototype work, the recent deliverable for D4.3.1 (Sensor Abstraction and Interoperability Interfaces Prototype I**)** has been referenced.

Limitations and further development details have been provided. Further implementations of APIs for interaction between components Open Data, user data, general and car sensor data and the Cloud Infrastructure using the SIMPLI-CITY Data model will be developed for the next WP4 prototypes, D4.2, D4.3.2 and D4.4.2.

http://www.simpli-city.eu/	Copyright © SIMPLI-CITY Project Consortium. All Rights Reserved. Grant Agreement No.: 318201