HORIZONS 2020 PROGRAMME

Research and Innovation Action – FIRE Initiative

Call Identifier:	H2020–ICT–2014–1
Project Number:	643943
Project Acronym:	FIESTA-IoT
Project Title:	Federated Interoperable Semantic IoT/cloud Testbeds and Applications

Analysis of IoT Platforms and Testbeds

Document Id:	FIESTA-IoT-WP2- D22-AnalysisOfIoTPlatformsAndTestbeds-150811-V23
File Name:	FIESTA-IoT-WP2- D22- AnalysisOfIoTPlatformsAndTestbeds-150811-V23.doc
Document reference:	Deliverable 2.2
Version:	V23
Editor:	Konstantinos Bountouris, Claude Hary
Organisation:	Com4Innov
Date:	11 / 08 / 2015
Document type:	Deliverable
Dissemination level:	PU, CO

DOCUMENT HISTORY

Rev.	Author(s)	Organisation(s)	Date	Comments
V01	Martin Serrano	NUIG-Insight	2015/02/01	Initial Draft Proposal
V02	Konstantinos Bountouris / Claude Hary	Com4Innov	2015/06/01	Table of Contents / FIESTA-IoT Testbeds Analysis Table / Mapping-translating to IoT-ARM naming
V03	Francois Carrez	UNIS	2015/06/09	Restructuring / ARM background and overall methodology / UNIS test-bed contribution
V04	Konstantinos Bountouris / Claude Hary	Com4Innov	2015/06/10	Com4Innov Testbed Analysis (section 3.6), Com4Innov Testbed Functional View list (section 6.3)
V05	Luis Sanchez / David Gomez	UNICAN	2015/06/16	UNICAN Testbed description (section 2.2)
V06	Francois Carrez	UNIS	2015/06/23	Adding on Domain Model, Information Model, Information View
V07	Konstantinos Bountouris / Claude Hary	Com4Innov	2015/06/25	Remove the UI, FOKUS testbeds, update of table in sec.5, addition of flowcharts for C4I side
V08	François Carrez / Tarek El Saleh	UNIS	2015/07/01	UNIS Testbed Analysis (section 3.3), section 7 moved to 3.3 “Data format & Ontologies”, section 4.1 FV, IV and DOV
V09	Jaeseok Yun / Miao Ting	KETI	2015/07/02	Adding KETI’s testbed Analysis and Functional View (section 3.4 & 6.2)
V10	Luis Sanchez / David Gomez	UNICAN	2015/07/06	Section 3.1, UNICAN Testbed description update (section 3.2) and Functional View (section 6.4)
V11	Konstantinos Bountouris	Com4Innov	2015/07/10	Mapping of C4I services to Functional Model, Introduction
V12	Luis Sanchez	UNICAN	2015/07/13	Update of SmartSantander content
V13	UNPARALLEL / Martin Serrano	UNPARALLEL / NUIG-Insight	2015/07/15	Section 1 and Bibliography
V14	Paul Grace / Juan Echevarria	ITInnov / SDR	2015/07/15	Quality Review
V15	Konstantinos Bountouris	Com4Innov	2015/07/22	Conclusion, Terms and Acronyms
V16	Konstantinos Bountouris	Com4Innov	2015/07/28	Explanation of mapping of C4I services in Functional Model

V17	Francois Carrez	UNIS	2015/08/02	Changes in UNIS testbed section 3.3 / General Review of the doc
V18	Franck Le Gall	EGM	2015/08/03	Technical review
V19	Konstantinos Bountouris	Com4Innov	2015/08/07	Section 6 changes: conclusion, summary of all Functional Views, section 6.5
V20	Jaeho Kim / Jaeseok Yun / Miao Ting	KETI	2015/08/10	Update of KETI's testbed (section 3.4 and 5.1) and Functional View (section 6.2)
V21	Konstantinos Bountouris / Claude Hary	Com4Innov	2015/08/10	NGSI adding (section 3.5) and final review of the document before submitting
V22	Paul Grace & Vadims Krivcovs/ Luis Sanchez & David Gómez / Tarek Elsaleh / Konstantinos Bountouris	ITINNOV / UNICAN / UNIS/ Com4Innov	2015/08/11	State-of-the-Art Experimentation Tools
V23	Konstantinos Bountouris	Com4Innov	2015/08/11	Final review and version of the deliverable ready for submission

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TERMS AND ACRONYMS

ACL	Access Control List
ACP	Access Control Policies
ADA	Active Digital Artefact
AKA	Authentication and Key Agreement
AM	Aggregation Manager
API	Application Program Interface
APN	Access Point Name
ARM	Architecture Reference Model
CoAP	Constrained Application Protocol
DM	Domain Model
DOV	Deployment and Operation View
EaaS	Experiment-as-a-Service
EU	European Union
FC	Functional Component
FDD	Frequency Division Duplex
Fed4FIRE	Federation For Future Internet Research and Experimentation
FG	Functional Group
FI	Future Internet
FIESTA	Federated Interoperable Semantic IoT/cloud Testbeds and Applications
FM	Functional View
FTP	File Transfer Protocol
FV	Functional View
GW	Gateway
GUI	Graphical User Interface
GPRS	General Packet Radio Service
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IEEE	Institute of Electrical and Electronics Engineers
IERC	IoT European Research Cluster
IM	Information Model
IMS	IP-Multimedia Subsystem
IoT	Internet of Things

IV	Information View
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
LED	Light Emitting Diode
LTE	Long Term Evolution
M2M	Machine-to-Machine
MQTT	Message Queuing Telemetry Transport
NFC code	Near Field Communication code
OTAP	Over-the-Air Provisioning
OWL	Web Ontology Language
PCRF	Policy and Charging Rules Function
PDA	Passive Digital Artefact
PE	Physical Entity
PIR	Passive InfraRed
QR code	Quick Response code
RA	Reference Architecture
RAN	Radio Access Network
RCS	Rich Communication Suite
REST	REpresentational State Transfer
RDF	Resource Description Framework
RFC	Request For Comments
RFID	Radio Frequency IDentification
RM	Reference Model
SAO	Stream Annotation Ontology
SFA	Slice-based Facility Approach
SME	Small and Medium-sized Enterprise
SSL	Secure Sockets Layer
SSN	Semantic Sensor Network (Ontology)
STP	Security, Trust and Privacy
TDD	Time Division Duplex
TTEO	Things Talk to Each Other
VE	Virtual Entity
ViLTE	Video over LTE
VoLTE	Voice over LTE

VoWiFi	Voice over WiFi
UML	Unified Modelling Language
UMTS	Universal Mobile Telecommunication System
URN	Uniform Resource Name
WP	Work Package
WSN	Wireless Sensor Network
XML	eXtensible Mark-up Language

1 POSITIONING

1.1 FIESTA-IoT

Recent advances in the Internet of Things (IoT) area have progressively moved in different directions (i.e. designing technology, deploying the systems into the cloud, increasing the number of inter-connected entities, improving the collection of information in real-time and not less important the security aspects in IoT). IoT Advances have drawn a common big challenge that focuses on the integration of the IoT generated data. The key challenge is to provide a common sharing model or a set of models organizing the information coming from the connected IoT services, IoT technology and systems and more important able to offer them as experimental services in order to optimise the design of new IoT systems and facilitate the generation of solutions more rapidly.

In FIESTA-IoT we focus on the problem of formulating and managing Internet of Things data from heterogeneous systems and environments and their entity resources (such as smart devices, sensors, actuators, etc.), this vision of integrating IoT platforms, testbeds and their associated silo applications within cloud infrastructures is related with several scientific challenges, such as the need to aggregate and ensure the interoperability of data streams stemming from different IoT platforms or testbeds, as well as the need to provide tools and techniques for building applications that horizontally integrate diverse IoT Solutions. The convergence of IoT with cloud computing is a key enabler for this integration and interoperability, since it allows the aggregation of multiple IoT data streams towards the development and deployment of scalable, elastic and reliable applications that are delivered on-demand according to a pay-as-you-go model.

The activity in FIESTA-IoT is distributed in 7 work packages WP1 is dedicated to the project activities coordination, considering consortium administration, financial management, activity co-ordination, reporting and quality control. In FIESTA-IoT one of the main objectives is to include experimenters and new testbeds to test and feedback the platform and tools generated, thus open calls for those tenders will be issued that are also part of the WP1 activity and is called selection of third-parties.

WP2 focuses on stakeholder's requirements and the analysis on IoT Platforms and Testbeds in order to define strategies for the definition and inclusion of Experiments, Tools and KPIs. The activities in this WP2 are focused on studying the IoT Platforms and Testbeds and the specification of the Experiments, the detail of the needed tools for experimentation and the KPIs for validate the proposed solutions. This WP will conduct the design and development of the Meta-Cloud Architecture (including the relevant directory of IoT resources) and will define the technical specification of the project. WP2 also focuses on analysing the Global Market Confidence and establishes the Certification Programme Specifications that will drive the global market confidante and certification actions around IoT experimentation model.

WP3 package focuses on providing technologies, interfaces, methods and solutions to represent the device and network nodes of the test-beds as virtualized resources. The virtualized resources will be represented as services and will be accessible via common service interfaces and APIs (i.e. the FIESTA-IoT Testbed interfaces/APIs). The virtualized resources and their capabilities and interfaces will be also described

using semantic metadata to enable (semi-) automated discovery, selection and access to the test-bed devices and resources.

WP4 will implement an infrastructure for accessing data and services from multiple distributed diverse testbeds in a secure and testbed agnostic way. To this end, it will rely on the semantic interoperability of the various testbeds (realized in WP3) and implement a single entry point for accessing the FIESTA-IoT data and resources in a seamless way and according to an on-demand EaaS model. The infrastructure to be implemented will be deployed in a cloud environment and will be accessible through a unified portal infrastructure.

WP5 focuses on designing deploy and deliver a set of experiments, so as to assess the feasibility and applicability of the integration and federation techniques, procedures and functions developed during the project lifetime. It would define a complete set of experiments to test the developments coming from other WPs (mainly WP3 and 4), covering all the specifications and requirements of WP2. Developments will be tested over available IoT environments and/or smart cities platforms. WP5 would also provide evaluation of the key performance indicators defined for every experiment/pilot. The final deployed experiments will include a subset of those coming from WP2, 3 and 4, as well as those provided by FIESTA-IoT Open Calls.

WP6 focuses on the establishment and validation of the project's global market confidence on IoT interoperability, which will provide a vehicle for the sustainability and wider use of the project's results. The main activity in this WP focuses on specifying and designing an IoT interoperability programme, including a set of well-defined processes that will facilitate the participation of researchers and enterprises. WP6 works on providing a range of certification and compliance tools, aiming at auditing and ensuring the openness and interoperability of IoT platforms and technologies. WP6 also focuses on Interoperability testing and validation and to provide training, consulting and support services to the FIESTA-IoT participants in order to facilitate platforms and tool usability but also to maximize the value offered to them by using FIESTA-IoT suite and tools.

WP7 work package focuses on ensuring that FIESTA-IoT suite, models and tools engages well with the community outside of the project; from promotion and engagement of new customers, to the front line support of current users, and the long-term exploitation of results and sustainability of the facility itself. This will be carried out in a coordinated manner such that a consistent message and professional service is maintained. Dissemination activities and the KPI to measure the impacts will be studied and used in this WP. An ecosystem plan including the specification of processes, responsibilities and targets will be generated and the evaluation and effectiveness of the operating model will be evaluated within this WP. In this WP the successes of stakeholder engagement and report on their satisfaction with the services offered in FIESTA-IoT will be put in place at the end of the project.

1.2 WP2 Overview

This Work Package covers the FIESTA-IoT requirements engineering activities and will produce the requirements associated with testbed-agnostic experimentation, as well as with the Experiment-as-a-Service model to designing and conducting

experiments. WP2 is composed of five different tasks (depicted in Figure 1), which tackle distinct aspects of the FIESTA-IoT EaaS Experimental Infrastructure:

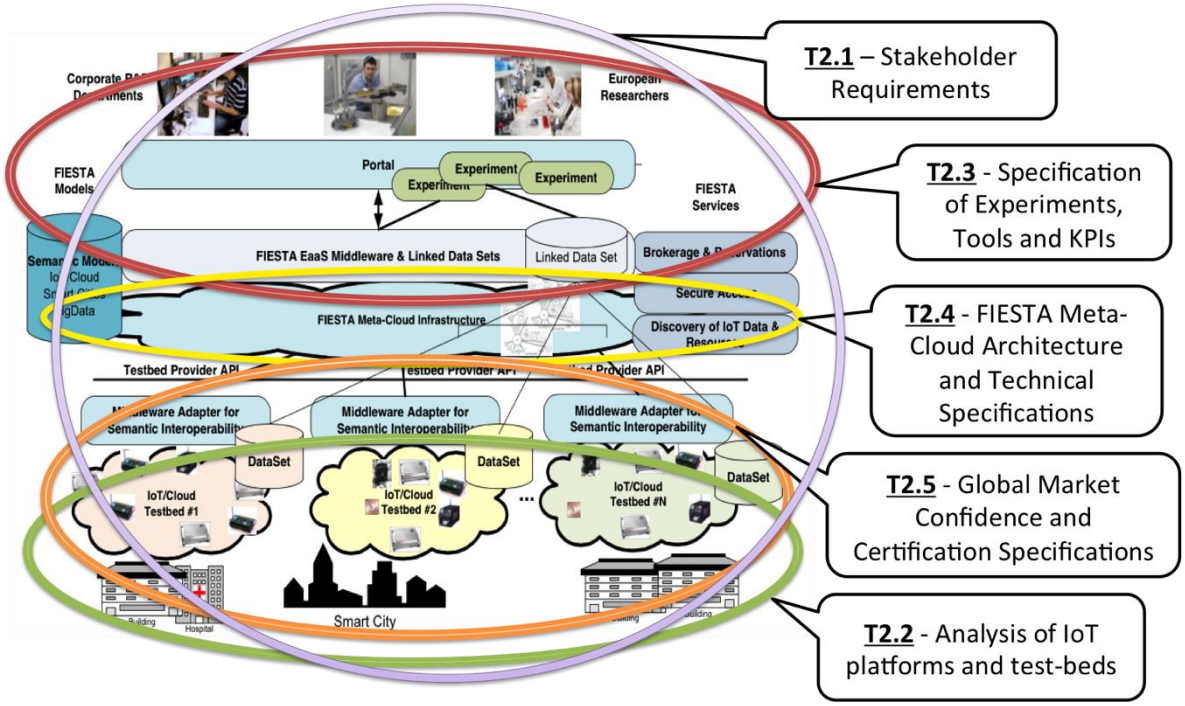


Figure 1. WP2 Overview

The WP2 Tasks cross all aspects of the FIESTA-IoT Infrastructure. They are:

Task 2.1. Stakeholder Requirements: This task is responsible for gathering and processing all Stakeholder requirements (using the Volere Requirements specifications). The involved stakeholders include: the IoT test-beds to be integrated, the experiment providers, and also researchers and experimenters. Also external projects (such as Open-IoT and Fed4Fire) will provide requirements so, to prepare FIESTA-IoT for the Open-calls. This task will produce a set of requirements that will be used by all other WP2 tasks.

Task 2.2. Analysis of IoT platforms and Test-beds: This task is focused on the Test-beds and IoT Platforms, analysing and describing what they do and how they do it. It will also use the set of test-bed requirements produced in T2.1 to better understand if each test-bed can fulfil the stakeholders' requirements. This task will then, model the Test-beds and IoT Platforms in functional blocks using the IoT-A ARM model (IoT-A, 2013). It will gather what type of information they provide, and how they provide this information so that Task 2.4 can take this into account when developing the FIESTA-IoT Architecture. The outcome of this task will provide a basis for WP3.

Task 2.3. Specification of Experiments, Tools and KPIs: This task will specify all planned experiments and extrapolate from it the needed tools to execute those experiments. It will use the experiment related requirements produced in T2.1 and analyse them in terms of the tools that need to be provided from FIESTA-IoT to the experimenters. It will also specify the KPIs of each

experiment so that later validation can occur. The result of this Task will be used as input to WP5.

Task 2.4. FIESTA-IoT Meta-Cloud Architecture and Technical Specifications: This Task will define the FIESTA-IoT Meta-Cloud Architecture, leveraging on the IoT-A ARM, and the technical specifications that will drive all the development work of the project. It will use information from previous tasks to identify the main building blocks, design & technology choices, and specify the functional blocks of the FIESTA-IoT architecture needed for achieving FIESTA's technical objectives. This architecture will serve as a base for all of the development phase of the project and more specifically for WP4.

Task 2.5. Global Market Confidence and Certification Specifications: This task is intended to study and define the global market confidence and certification specification. This means that this task is responsible to define the certification process, and the set of requirements that are required for a test-bed to comply, in order to be integrated into FIESTA-IoT. The outcome of this task will be used in WP6.

As described in the previous tasks description, the outcomes of each task will be used by other tasks of this WP2, or be used as inputs for the work in other WPs.

These relations between WP2 tasks and other WPs are depicted in Figure 2.

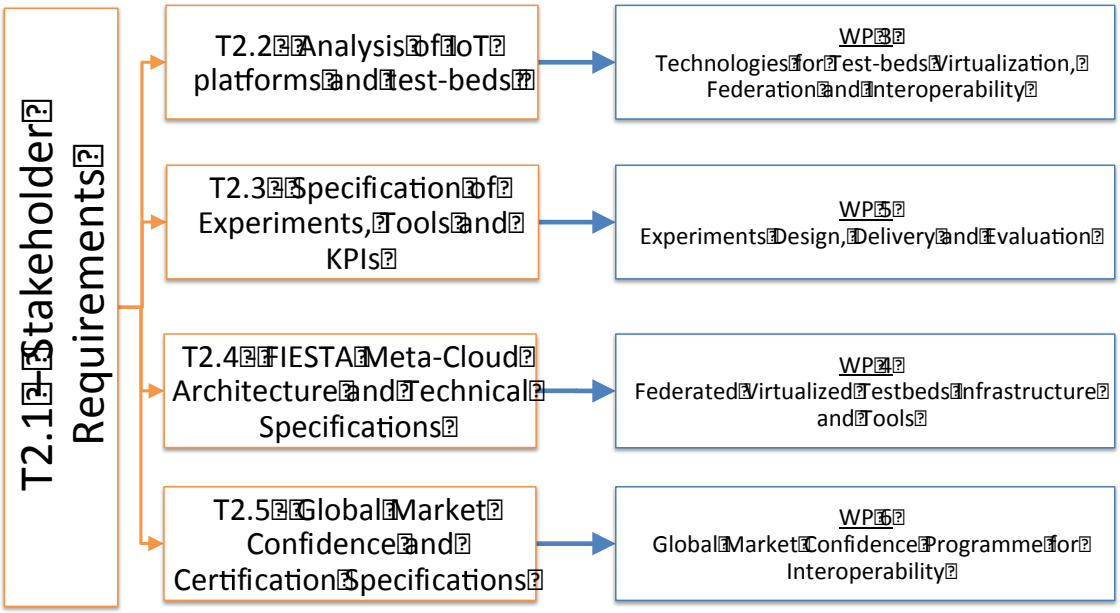


Figure 2. Relationship between WP2 tasks and with other WPs

In reference to the Fiesta-IoT project general objective(s), WP2 has a set of sub-objectives defined activities that are described as follow:

1) Determination of stakeholders' requirements.
2) Description of IoT Platforms and test-beds in order to facilitate their integration into FIESTA-IoT infrastructure.
3) Specification of planned experimentation and its executing tools, and the KPIs that will be used for validation.

4) Definition of the FIESTA-IoT Meta-Cloud architecture and the technical specifications required for the development WPs
5) Definition of the Global market confidence and Certification specifications

The Work Package 2 will also result in five deliverables, which will be directly linked with the objectives and tasks of the WP. Each Deliverable will be an outcome of each Task, meaning that Deliverable D2.1 will be provided at the end of T2.1 with the results of that specific task. The following table details the set of deliverables to be expected from WP2, with reference to the related tasks, the responsible partner for each deliverable and all other contributors.

Table 1. WP2 Deliverables

No.	Deliverable	Responsible Partner	Contributors
D2.1	Stakeholders Requirements	UNPARALLEL	NUIG-DERI, NEC, UNICAN, SODERCAN, SDR
D2.2	IoT Platforms and Testbeds Analysis	Com4Innov	KETI, UNICAN, UNPARALLEL, AIT, NUIG-DERI, INRIA, NEC
D2.3	Experiments, Tools and KPIs Specification	UNPARALLEL	UNICAN, INRIA, NEC, NUIG-DERI, AIT, ITINNOV, SODERCAN
D2.4	FIESTA Meta-Cloud Architecture and Technical Specifications	UNIS	AIT, NUIG-DERI, UNICAN, ITINNOV, KETI
D2.5	Global Market Confidence and Certification Programme Specifications	EGM	AIT, SODERCAN

1.3 Audience

This deliverable addresses the following audiences:

- **Researchers and engineers within the FIESTA-IoT consortium**, which will take into account the various requirements in order to research, design and implement the architecture of the FIESTA-IoT Meta-Cloud Architecture.
- **Researchers on Future Internet Research and Experimentation (FIRE) focused on IoT and cloud computing systems experimenters at large**, given that the present deliverable could be a useful reading for researchers studying alternative IoT technologies and applications, along with indications and requirements towards building/establishing experimental architectures.

- **Members of other Internet-of-Things (IoT) communities and projects (such as projects of the IERC cluster)**, which can find in this document a readily available requirements analysis for experimentation-like IoT services and tools. For these projects the document could provide insights into requirements and technological building blocks enabling the convergence between utility/cloud computing and the Internet-of-Things for enabling experimentation as a service.

1.4 Terminology and Definitions

This sub-section is intended to clarify the terminology used during this project. This initial step is intended to clarify all the important terms used, in order to minimise misunderstandings when referring to specific parts involved in the generation of data and the FIESTA-IoT concepts. The following definitions were set regarding the domain area of FIESTA-IoT, and so are aligned with terminologies used in FIRE community and in reference IoT-related projects (such as IoT-A).

Table 2. Terminology and Definitions table

Term	Definition
Characteristic	An inherent, possibly accidental, trait, quality, or property of resources (for example, arrival rates, formats, value ranges, or relationships between field values).
Device	Technical physical component (hardware) with communication capabilities to other Information technology (IT) systems. A device can be either attached to, or embedded inside a physical entity, or monitor a physical entity in its vicinity (IoT-A, 2013) (Haller et al., 2013). The device could be: • **Sensor**, A sensor is a special device that perceives certain characteristics of the real world and transfers them into a digital representation (IoT-A, 2013). • **Actuator**, An actuator is a mechanical device for moving or controlling a mechanism or system. It takes energy, usually transported by air, electric current, or liquid, and converts that into some kind of motion (IoT-A, 2013).
Discovery	Discovery is a service to find unknown resources/entities/services based on a rough specification of the desired result. It may be utilized by a human or another service. Credentials for authorization are considered when executing the discovery (IoT-A, 2013).

Information	Content of communication; data and metadata describing data. The material basis is raw data, which is processed into relevant information, including source information (e.g., analogue and state information) and derived information (e.g., statistical and historical information) (IEEE, 2007).
Measurement	The important data for the experimenter. It represents the minimum piece of information sent by a specific resource, which the experimenter needs in order to fulfil the objective of the experiment
Metadata	The metadata is the additional information associated with the measurement, facilitating its understanding.
Physical Entity	Any physical object that is relevant from a user or application perspective. (IoT-A, 2013) (Haller et al., 2013). Physical Entities are the objects from the Real-World that can be sensed and measured and they are virtualized in the cyber-space using Virtual Entities.
Requirement	A quantitative statement of business need that must be met by a particular architecture or work package. (Haren, 2009)
Resource	Computational element that gives access to information about or actuation capabilities on a Physical Entity (IoT-A, 2013) (Haller et al., 2013).
Stakeholder	An individual, group, or organization, who may affect, be affected by, or perceive itself to be affected by a decision, activity, or outcome of a project (Project Management Institute, 2013)
Testbed	A testbed is an environment that allows experimentation and testing for research and development products. A testbed provides a rigorous, transparent and replicable environment for experimentation and testing (Gavras, 2010)
Federated testbeds	A testbed federation or federated testbeds is the interconnection of two or more independent testbeds for the creation of a richer environment for experimentation and testing, and for the increased multilateral benefit of the users of the individual independent testbeds (Gavras, 2010)
Interoperability	The ability of two or more systems or components to exchange information and use the information that has been exchanged (IEEE, 1990)
Experimentation facility	An experimentation facility can be understood as an environment with an associated collection of tools and infrastructure that sits on top of one or several testbeds and can be used to conduct experiments to assess and evaluate new paradigms, architectural concepts and applications (MyFIRE, 2011)
Experiment	Experiment is a test under controlled conditions that is made to demonstrate a known truth, examine the validity of a hypothesis, or determine the efficacy of something previously untried (Soukhanov, Ellis, & Severynse, 1992)

1.5 Executive Summary

This deliverable analyzes the main testbeds and platforms participating in the FIESTA-IoT project. The testbeds and platforms that will be interconnected are: SmartSantander, University of Surrey, Com4Innov and KETI. This interconnection will enable the deployment of IoT experiments and allow services within federated utility-based cloud computing environments.

In particular, this deliverable consists of an analysis and description of the resources that each of the aforementioned testbeds includes i.e. the IoT devices and sensors, gateways, APIs. It is of importance to have an exact knowledge of all the resources that could be part of the interconnection of the different testbeds around Europe and the entire world. Except the material resources written above, there are recorded as well communication protocols, applications as well as security and authentication mechanisms that are used.

Moreover, at the later chapters in this deliverable, there is an introduction on the IoT ARM. This model will be used in order to translate the available resources of each testbed into a common language. There will be a mapping of the gateways, the IoT devices, the APIs etc in a common model, so that the interconnection of the testbeds and platforms becomes easier. Finally, there is the translation of the resources into the IoT ARM naming, e.g. the last chapter provides detailed analysis and description of the services of each Testbed and Platform based on both the Functional Model and Functional View approach from IoT ARM.

In the end of the document a State-of-the-Art on Experimentation Tools is present, in order to present a set of already existing experimentation tools - collected from the in-house testbeds and from external projects, which can provide an idea of the kind of functionalities that are normally present in this kind of tools.

2 INTRODUCTION

2.1 Motivation and objectives

The main goal of the FIESTA-IoT project is to open new horizons in the development and deployment of IoT applications and experiments at an EU (and global) scale, based on the interconnection and interoperability of diverse IoT platforms and testbeds. To this end, FIESTA-IoT provides a blueprint experimental infrastructure, tools, techniques, processes and best practices enabling IoT testbed/platforms operators to interconnect their facilities in an interoperable way, while at the same time facilitating researchers and solution providers in designing and deploying large scale integrated applications (experiments) that transcend the (silo) boundaries of individual IoT platforms or testbeds. FIESTA-IoT enables researchers and experimenters to share and reuse data from diverse IoT testbeds in a seamless and flexible way, which opens up new opportunities in the development and deployment of experiments that exploit data and capabilities from multiple testbeds. The blueprint experimental infrastructure to be provided by FIESTA-IoT includes middleware for semantic interoperability, tools for developing/deploying and managing interoperable applications, processes for ensuring the operation of interoperable applications, as well as best practices for adapting existing IoT facilities to the FIESTA-IoT interoperability infrastructure.

The project's experimental infrastructure provides European experimenters in the IoT domain with the following unique capabilities:

- **Access to and sharing of IoT datasets in a testbed-agnostic way.** FIESTA-IoT provides researchers with tools for accessing IoT data resources (including Linked sensor data sets) independently of their source IoT platform/testbed.
- **Execution of experiments across multiple IoT testbeds**, based on a single API for submitting the experiment and a single set of credentials for the researcher.
- **Portability of IoT experiments across different testbeds**, through the provision of interoperable standards-based IoT/cloud interfaces over diverse IoT experimental facilities.

Our chosen common language is the IoT ARM model (http://www.iot-a.eu/arm, proposed by UNIS). In section 3 there is a detailed description of the IoT ARM. It is important that all the testbeds and platforms "speak" the same language and that the terminology is the same throughout the different elements of the federated platforms.

2.2 Overall goal of interconnecting the IoT platforms

One of the main objectives of the FIESTA-IoT cloud will be to enable European experimenters/researchers to design, implement, execute and evaluate IoT experiments based on data from various IoT testbeds all over Europe. To this end, FIESTA-IoT will also offer a wide range of tools facilitating experimenters in the above tasks. These include: (A) A portal infrastructure serving as a single entry point for setting up and submission of IoT experiments and the monitoring of their

progress, (B) Tools for designing and enacting experiments in terms of IoT/cloud services and workflows, (C) Tools for sharing, linking and accessing datasets in a testbed agnostic way, (D) Tools and techniques for monitoring and managing the FIESTA-IoT cloud, including monitoring of all the necessary aspects of the underlying testbeds and (E) Tools and techniques for monitoring the status of experiments and collecting data for evaluating the experiments. These tools are an integral element of the project's Experiment-as-a-Service paradigm for the IoT domain. In order to accomplish its goals, the project will issue, manage and exploit a range of open calls towards involving third-parties in the project. The objective of the involvement of third-parties will be two-fold:

- To ensure the design and integration (within FIESTA) of more innovative experiments, through the involvement of additional partners in the project (including SMEs). The additional experiments focus on demonstrating the added-value functionalities of the FIESTA-IoT experimental infrastructure.

- To expand the FIESTA-IoT experimental infrastructure on the basis of additional testbeds. In this case the new partners will undertake to contribute additional testbeds and to demonstrate their blending and interoperability with other testbeds (already adapted to FIESTA). As part of this blending, the owners of these testbeds will also engage with the project's global market confidence program, which will provide them with the means to auditing the interoperability and openness of their platforms.

The involvement of third-parties therefore plays an instrumental role for the large scale validation of the FIESTA-IoT experimental infrastructure, but also for the take-up of the project's global market confidence program on IoT interoperability. It is also a critical step to the gradual evaluation of FIESTA-IoT towards an infrastructure/ecosystem for global IoT experimentation.

Beyond the validation of the FIESTA-IoT infrastructure on the basis of practical experiments and the integration of additional IoT testbeds, the project specifies concrete best practices for the federation of testbeds (addressed to testbed owners/administrators) wishing to become part of the virtualized meta-cloud infrastructure of the project. Similar best practices are also produced for European researchers and enterprises (including SMEs) wishing to design and execute experiments over the FIESTA-IoT EaaS infrastructure. These best practices are disseminated as widely as possible, as part of the project's efforts to achieve EU-wide/global outreach. The attraction and engagement of researchers and enterprises in the use of the FIESTA-IoT EaaS infrastructure is another vehicle for the sustainability and wider use of the project's results, which complements the global market confidence programme outlined above.

FIESTA-IoT is perfectly in-line with the directions identified and prioritized as part of recent FIRE roadmaps in the areas of IoT and its convergence with cloud computing and smart city applications. The project addresses the challenges identified in recent support actions (e.g., the AmpliFIRE Support Action) for the FIRE domain.

3 TESTBEDS AND PLATFORMS ANALYSIS

3.1 Testbed features analysed

Understanding the underlying infrastructure that will be federated into the FIESTA-IoT platform is of key importance before any kind of integration is started. This section introduces (in an abstract manner) the key features and characteristics that will focus the testbed description and analysis. Some of these features are qualitative but as far as possible they are quantitative so that objective analysis is possible.

Before delving deeper into the key testbed features used for the testbed analysis, it is important to highlight, for the sake of completeness, which are the main functionalities and related challenges that underlie the deployment and setup of IoT experimentation infrastructures. In this sense, the experimentation possibilities that must be supported in order to fulfil the requirements imposed to perform experimentally-driven research when moving from islands of sensor networks to a global networked infrastructure (as envisioned by the Internet of Things) open up new challenges that demand new capabilities and features from suitable testbeds (Gluhak, 2011), (Tonneau, 2015).

Key aspects that must be observed to support the experimentation when moving from sensor networks to IoT are:

- Scale: Real-world experimentation in a target deployment environment also requires experimentation at adequate scale. While smaller-scale testbeds with populations of tens up to hundreds of nodes were sufficient for most sensor networks experiments, many IoT experiments demand an order of magnitude larger scale.
- Heterogeneity: Future Internets of Things will consist of a wide variety of devices integrated with other FI infrastructure and service provisioning platforms. For reasons of applicability, it is expected that the development and evaluation of protocols and other IoT technologies be undertaken under conditions that is representative of the degree of heterogeneity inherent in the Internet of Things.
- Mobility: The IoT is composed of fixed and mobile devices which can also interact with each other in real life scenarios. While some indoor testbeds offer robot-controlled mobility, it is often difficult to reproduce real life mobility patterns in such testbeds.
- Experimentation realism: Live testbeds provide a degree of experimentation realism that even the most detailed simulation cannot achieve. We argue that, for IoT-technology experimentation, even lab-based testbeds do not suffice to evaluate research prototypes under realistic conditions and to facilitate their transfer into real world deployments. IoT technologies are heavily dependent on ambient environmental conditions in which they are deployed, including the service logic of the diverse IoT applications.
- Data-centricity: Data-centricity is another key differentiator between sensor networks experimentation and IoT one. While for the first one, focus is put on the devices and how they operate (i.e. communicate, inter-network, etc.), IoT builds on top of devices but provides a higher level of abstraction in which the

focus is put on the services that objects provide (generally data gathering and reporting).

- Concurrency: Experimentation as a Service model has to be adopted so that concurrent experimentation can be handled. This way the underlying infrastructure is decoupled from the experimenters' requests guaranteeing scalability both in terms of enlarging the deployed infrastructure (i.e. new IoT devices should be seamlessly incorporated to the testbed on a plug & play manner), as well as in terms of increasing the number of concurrent experiments.
- Autonomy: With the scale and variety of testbed management events to track, one cannot assume human intervention alone is sufficient to provide timely response to events and remediation to faults. Testbed management automation must be incorporated keeping the human in the loop only for decision-making and policy-specification. Moreover, there are other aspects, more related to the deployment and networking that should be put forward in order to maximize testbed autonomy.

Table 3 summarises the key testbed features that have been taken into account during the testbed description and analysis.

Table 3. Key testbed features included in the analysis

Feature name	Feature description	Related IoT experimentation support aspect
Number of resources	The absolute number of devices deployed in the testbed. The concept of resource is sometimes different from one testbed to another (there are even different meanings within some testbeds) so it is important to describe the hardware deployed thoroughly.	Scale, Heterogeneity
Location of resources	Geographic situation of the deployed infrastructure indicating whether they are indoor or outdoor, fixed or mobile.	Scale, Heterogeneity, Mobility
Type of sensors	Specification of the phenomenon that each of the deployed sensors is able to monitor. In case a particular feature of the sensor (accuracy, range, etc.) is of special relevance, it should be highlighted.	Heterogeneity, Experimentation realism
Type of actuators	Specification of the actuation possibilities offered by the deployed device. Moreover, it is relevant to know if there are resources on which is possible to actuate (e.g. RFID, QR or NFC tag, etc.)	Heterogeneity, Experimentation realism
Communication protocols	Communication technologies used and network architecture.	Mobility, Autonomy
Datasets or data-streams	Possibility to access real-time and/or historic information.	Experimentation realism, Data-centricity
APIs	Description of the methods available for accessing the experimentation services throughout the whole experiment lifecycle.	Concurrency

Applications	Summary of applications or experiments developed on top of the testbed. It does not have to be an extensive review but rather an exemplifying list.	Experimentation realism
Openness / Security	Relationship that has to exist between the experimenter and the testbed maintainer if any. Methods to guarantee access control to the testbed experimentation services.	Concurrency, Autonomy
Semantics	Support of annotated information.	Concurrency, Autonomy
Federation	Links established with other experimentation facilities. Main interest is pertaining to the Future Internet community in general and FIRE in particular.	Scale, Data-centricity, Heterogeneity

It is important to mention that this abstract definition of key features will represent the basis on which selection for additional testbeds will be done through the respective Open Calls to be carried out during the project.

The following sections describe, capturing all of the abovementioned key features, what the currently available testbeds, brought by four of the consortium partners, do and which tools they provide for supporting the IoT experimentation. Also, in SURREY test-bed Functional View (section 6.1) the Domain Ontologies and the General Purpose Ontologies for the testbed of University of Surrey are provided.

3.2 SmartSantander Testbed

3.2.1 Technical architecture overview

The SmartSantander testbed (www.smartsantander.eu) is an experimental test facility for the research and experimentation of architectures, key enabling technologies, services and applications for the Internet of Things in the context of a city. Additionally, in order to manage the testbed, SmartSantander can work as a platform and is able to be increased in size with new heterogeneous sensors.

The testbed architecture follows a three-tiered approach, as shown in Figure 3:

1. **IoT tier**: Responsible for sensing the corresponding parameter (e.g. temperature, CO, noise, light, car presence, soil temperature, soil humidity). The majority of these sensors are integrated in the network devices named repeaters, whilst the others are standalone and wirelessly communicate with the corresponding repeaters (it is the case for the parking sensor buried under the asphalt). For these devices, due to the impossibility of powering them with electricity, they must be fed with batteries. Repeaters are high-rise placed in street lights, public buildings, etc. in order to behave as forwarding nodes to transmit all the information associated to the different measured parameters. The communication between repeaters and IoT nodes is carried out through a proprietary protocol based on IEEE 802.15.4 called Digimesh (DigiMesh, 2015). Additionally, an IEEE 802.15.4 native interface is available for experimentation purposes in the repeaters.
2. **Gateway tier**: Both IoT nodes and repeaters are configured to send all the information (through Digimesh protocol), service provision and network management to the gateway. Once this information is received by the gateway, it forwards the information to the SmartSantander upper layers, through the different interfaces provided by it (GPRS/UMTS or Ethernet). Furthermore, most of the gateways contain enough intelligence to manage and control the network with different tools (OTAP, scan node, etc.).
3. **Platform tier**: Top layer in the SmartSantander architecture. This layer exposes the interfaces to inject new data coming from the city infrastructure as well as to retrieve data from them. Several tools for the control and management of the platform can be found in this tier, as well as the internal repositories for registered devices and sensor data. The federation wrapper is also implemented within the platform tier. This layer also accepts data from no physical sensors deployed within the SmartSantander testbed.

Currently, SmartSantander architecture is integrating new enablers to fulfil the requirements of Fed4FIRE federation. Additionally, SmartSantander also supports FI-WARE (initiative, 2015), which is used to store all the data retrieved by the sensors in their instances and be accessed from FIWARE-LAB (initiative, 2015).

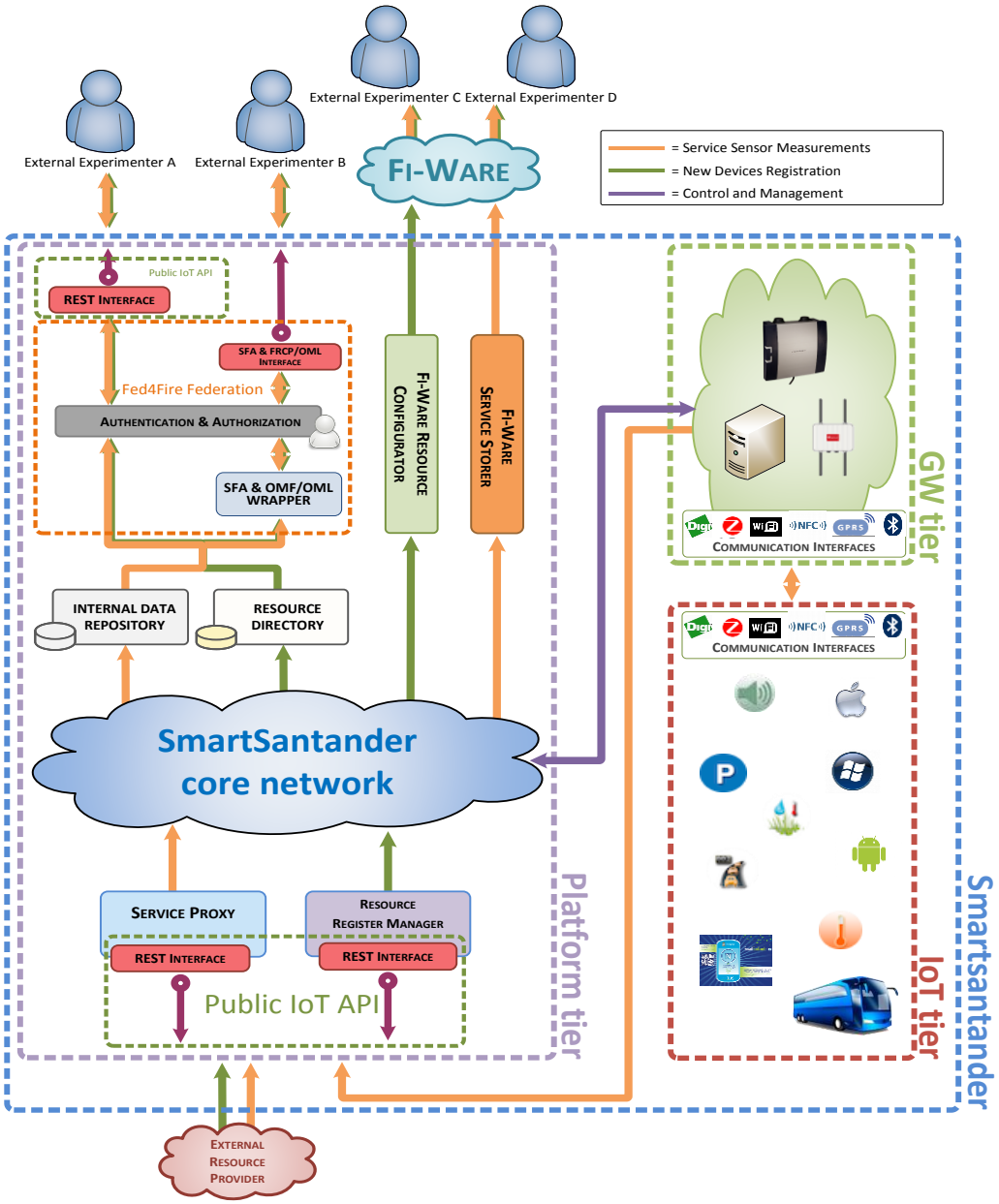


Figure 3. SmartSantander architecture and components

3.2.2 Physical Resources

The SmartSantander testbed is composed of around 3000 IEEE 802.15.4 devices, 200 GPRS modules and 2000 joint RFID tag/QR code labels deployed both at static locations (e.g. streetlights, facades, bus stops, etc.) as well as on-board of mobile vehicles (e.g. buses, taxis, etc.). Over the deployed platform, a number of use cases have been implemented, whose main highlights are briefly described below:

- **Environmental Monitoring (static nodes)**.

The city of Santander is trying to carry out an effective policy for environmental management through the signing of agreements that aid improvements in air quality and quality of life for its citizens. Key elements in undertaking this task are:

- Monitoring of pollutants (e.g. CO, NO_2, etc.)
- Noise and temperature measurement

With the goal of developing this environmental monitoring policy, temperature, CO index, noise and luminosity sensors, among others, have been installed in street lamps and facades. Namely, there are more than 1000 IoT devices already deployed and active (mainly at the city centre). All these devices send their information, in a multihop fashion (if needed), to the corresponding gateway, which gathers all the received information making it available to the SmartSantander backbone. Figure 4 shows several examples of the installation of these devices, at different streetlamps and facades of the city:

Figure 4. Examples of repeaters installation

- **Mobile Environmental Monitoring**:

In order to extend the aforementioned environmental monitoring use case, apart from measuring parameters at static points (thus leading to a costly solution), SmartSantander has installed additional devices, located at vehicles, which retrieve environmental parameters associated to determined parts of the city. Namely, they are installed in 150 public vehicles, including buses, taxis, police cars and parks and gardens adapted vehicles. This way we are able to cover a much wider area in a much more efficient way. Mobile nodes send the collected information to the internet/intranet, and also, interact with the corresponding static nodes placed at streetlamps and facades. Figure 5 shows the sensors installation in both buses, as well as how the park and gardens vehicles and the buses.

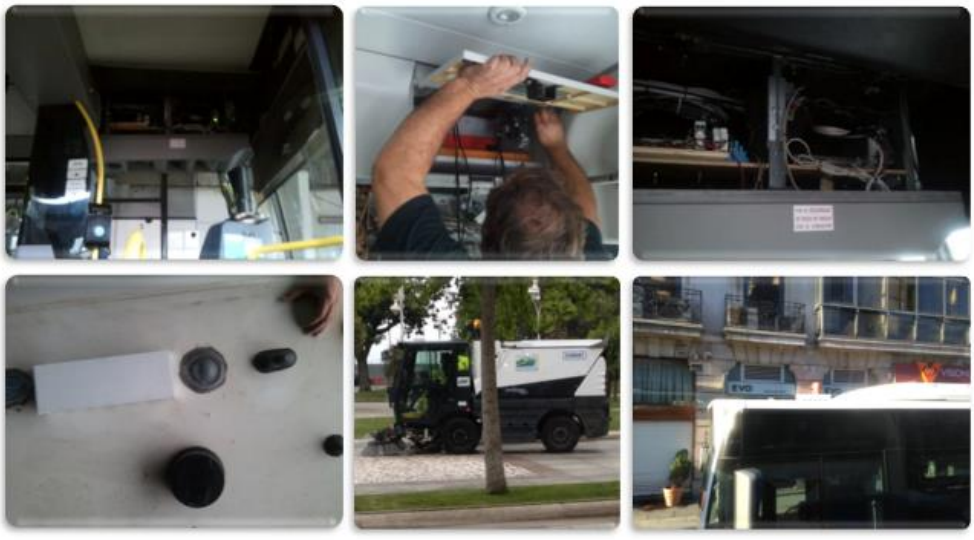


Figure 5. Mobile environmental monitoring sensors

- **Outdoor parking**

Numerous parking sensors (based on ferromagnetic technology) have been deployed in the downtown Santander, i.e. buried under the asphalt, to indicate the parking spots that are available or occupied. In order to detect the occupancy state of a determined parking spot, ferromagnetic sensors have been installed, thus sending the corresponding information (free or occupied) associated to a determined parking space. This information is forwarded to the central device (gateway), through the corresponding set of environmental monitoring repeaters. In Figure 6, the installation process of the parking sensors is shown:

Figure 6. Examples of parking sensors installation

To be more precise, 350 parking sensors have been installed within the “30 Zone”, in order to inform the occupancy degree of determined parking lots. All the information gathered from these sensors is sent to the SmartSantander core platform.

- **Traffic Intensity Monitoring**

Nowadays, the tracking and classification of vehicles in road traffic is accomplished by inductive loops placed under the pavement. These inductive loops allow monitoring vehicle passing by means of different configurations which provide a number of data in order to control several parameters of the traffic (e.g. vehicle speed, traffic congestion, traffic accidents, etc.).

Figure 7. Traffic sensors installation in the main roads of the city

However, these systems have several problems and disadvantages like their deployment, maintenance, high cost, and put into gear, among others. In this sense,

within the SmartSantander project a solution based on Wireless Sensor Network was deployed (see Figure 7 for an explicit overview of the operation of installing an arbitrary device), monitoring parameters such as cars speed, occupancy and count in the road lanes in the two main entrances of the city. Expressed in figures, around 60 devices located at the main entrances of the city of Santander have been deployed.

- **Guidance to free parking lots**

From the information gathered by the deployed parking sensors, 10 panels located at the main streets intersections have been installed in order to guide drivers towards the available free parking lots. An illustrative example of a panel is shown in Figure 8.

Figure 8. Parking lots panel indicator

- **Parks and gardens irrigation**

Within the SmartSantander IoT facility, around 50 agricultural IoT devices and weather stations have been deployed in three green zones of the city, i.e. the Las Llamas Park, La Marga Park and *Finca Altamira.* A total number of 48 IoT sensor nodes, covering an area of 55000 m^2, were deployed at key positions inside these three areas, equipped with special agricultural sensors measuring parameters like: air temperature and humidity, soil temperature and moisture, atmospheric pressure, solar radiation, wind speed/direction, rainfall, etc. All these sensors transmit wirelessly the data acquired to a gateway device located within the corresponding park and then, this device will manage all the information, sending it to the SmartSantander core platform with its inbuilt GPRS/3G module. It is worth illustrating the process followed in a node installation, as shown in Figure 9.

Figure 9. Soil moisture sensors installation in parks and gardens

- **Presence and luminosity sensors**

SmartSantander resources have been used as a testbed environment by other European initiatives that have increased its initial capabilities. One of these projects is OutSmart (project, http://fi-ppp-outsmart.eu/), which has provided SmartSantander with an application to control the street lighting of a located area of the city (Figure 10), deploying also a set of pedestrians' presence detectors and luminosity sensors. This control application keeps the street lighting off while the captured ambient light level is high (e.g. daylight) and triggers, during the night, a control process based on the pedestrian detection.

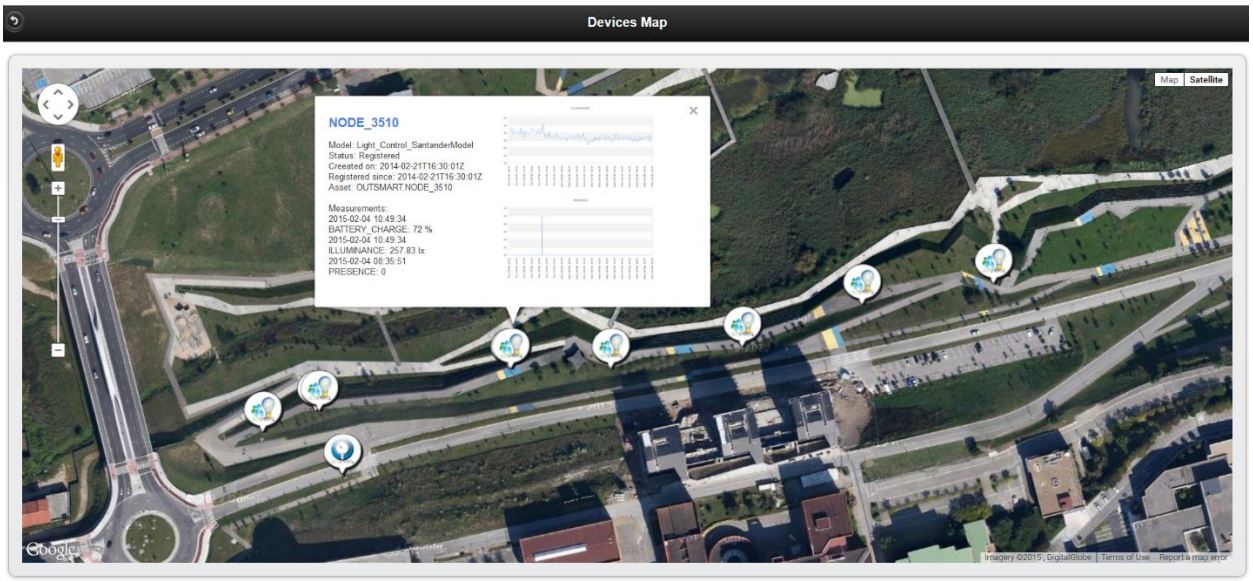


Figure 10. Presence detectors and Luminosity sensors deployed in Santander Las Llamas Park

- **NFC/QR tags**

More than 2000 tags have been deployed and distributed to different strategic places in the city centre of Santander. These tags are mainly at transportation points of interest (bus stops, taxis, etc…), cultural elements (monuments, etc…) and shops. All the information provided is online and can be updated at any time. A sample of one of these tags is shown in Figure 11.

Figure 11. NFC and QR tag located at a shopwindow

- **Testbed overview**

SmartSantander facility is based on a real IoT deployment in an urban setting. Its core is located in the city of Santander and surroundings, encompassing IoT deployments (Figure 12) in different key areas of the city infrastructure, ranging from public transport, key logistics facilities, public places and buildings, work places and residential areas, thus creating the basis for the development of a Smart City. This deployment exhibits the diversity, dynamics and scale that are essential for the evaluation of advanced protocol solutions.

Figure 12. Devices deployment within the SmartSantander IoT facility

- **Resources, observations and measurements**

Each one of the aforementioned devices represents a context information source, which is known within the SmartSantander terminology as a *resource*. They generate

observations, which are composed of one or several sensor *measurements*. Each of these measurements conveys information pertaining to a *phenomenon*. The taxonomy of available phenomenon within the SmartSantander testbed is expanded as shown in Table 1 (it is worth highlighting that “position:latitude”, “position:longitude” and “timestamp” are observed in every resource, hence they will not be included in the table below):

Table 4. Relationship between the different resources, measurement and phenomenon of the SmartSantander platform

▪ Noise monitoring: ▪ soundPressureLevel:ambient ▪ batteryLevel	▪ Luminosity monitoring: ▪ temperature:ambient ▪ iluminance ▪ batteryLevel ▪ acceleration:instantaneous	▪ CO monitoring: ▪ temperature:ambient ▪ chemicalAgentAtmosphericConcentration:CO ▪ batteryLevel ▪ acceleration:instantaneous
▪ Temperature monitoring: ▪ temperature:ambient ▪ batteryLevel ▪ acceleration:instantaneous	▪ Traffic: vehicle counter ▪ trafficIntensity ▪ trafficCongestion	▪ Traffic: vehicle speed ▪ trafficIntensity ▪ trafficICongestion ▪ speed:median ▪ speed:average
▪ Parking: ▪ presenceState:parking ▪ batteryLevel	▪ Park and gardens: Environmental Station ▪ batteryLevel ▪ temperature:ambient ▪ relativeHumidity ▪ solarRadiation ▪ atmosphericPressure ▪ rainfall ▪ windSpeed ▪ windDirection	▪ Park and gardens: Agriculture ▪ batteryLevel ▪ temperature:ambient ▪ relativeHumidity ▪ soilMoinstureTension ▪ temperature:soil
▪ Park and gardens: Irrigation ▪ batteryLevel ▪ temperature:soil ▪ temperature:ambient ▪ relativeHumidity	▪ Presence and Iluminance ▪ presenceState:people ▪ batteryLevel ▪ illuminance	▪ Automatic Metering and Managing System ▪ activePower ▪ reactivePower

▪ Power Regulator ▪ activePower ▪ reactivePower ▪ electricPotential ▪ electricCurrent	▪ Environmental monitoring: Mobile node with CANBUS ▪ CANBUS-related ▪ temperature:engine ▪ fuelConsumption:total ▪ fuelConsumption:instantaneous ▪ fillLevel:gasTank:1 ▪ fillLevel:gasTank:2 ▪ speed:instantaneous ▪ vehicleOverSpeed ▪ mass ▪ mileage:total ▪ motionState:vehicle ▪ presenceState:driverCard:1 ▪ presenceState:driverCard:2 ▪ direction:azimuth ▪ timeRelatedState:driver:1 ▪ timeRelatedState:driver:2 ▪ External sensors: ▪ temperature:ambient ▪ relativeHumidity ▪ chemicalAgentAtmosphericConcentration:airParticles ▪ chemicalAgentAtmosphericConcentration:CO ▪ chemicalAgentAtmosphericConcentration:NO2 ▪ chemicalAgentAtmosphericConcentration:O3 airQuality	▪ Air quality monitoring: Mobile node without CANBUS ▪ temperature:ambient ▪ relativeHumidity ▪ chemicalAgentAtmosphericConcentration:airParticles ▪ chemicalAgentAtmosphericConcentration:CO ▪ chemicalAgentAtmosphericConcentration:NO2 ▪ chemicalAgentAtmosphericConcentration:O3 ▪ airQuality ▪ position:altitude ▪ direction:azimuth ▪ mileage:total

3.2.3 Federation

SmartSantander is federated within the Fed4FIRE project (Vandenberghe, 2013). Indeed, enabling components for Fed4FIRE compliance are implemented on the top of the SmartSantander core platform. The Fed4FIRE project aims at providing a unified access to heterogeneous testbeds in order to perform experimentation on the available resources. The project takes into account three kinds of testbeds:

1. Testbeds with fully controllable resources, which are part of the *advanced federation*, where the experimenters can modify their resources behaviour for experimentation purposes. E.g. A testbed providing virtual machines with SSH access.
2. Testbeds with API-based access, which are part of the *light federation*, where the functionality is restricted by an API, and resources, cannot be modified out of the scope of the provided API. E.g. A testbed that provides sensor data access and has the possibilities of changing specific parameters, for example the measurement frequency of a resource, but does not provide unrestricted access to the resources.
3. Associated testbeds. These ones do not require technical integration and they just need to be included in the Fed4FIRE webpage with documentation.

Namely, SmartSantander is being federated following the second model at the time being. Future works foreseen federation under the first model.

Whilst associated testbeds do not require more than available documentation, light and advanced testbeds require technical integration. Common requirements are described as following:

- Documented REST API for testbed access.
- Support of Fed4FIRE credentials in a client-based SSL API

Advanced testbed federation requires, in addition, the implementation of a component called Federation Aggregate Manager (AM). This component implements XML-RPC over SSL connections to expose an API that follows specific Fed4FIRE guidelines.

As a central federation component, Fed4FIRE keeps a credential server to authorize the access to the different federated testbeds. Those which belong to the Fed4FIRE federation must accept this credential to access their resources. Additionally, an openly accessible web server with all the testbeds documentation is maintained.

3.2.4 Internal management methods and APIs

SmartSantander platform enables two different kinds of experimentation. Service Experimentation (SEL) consists of running experiments and/or applications based on the data gathered by SmartSantander sensor infrastructure and stored in a shared repository. Therefore, these services will be mainly based on data retrieval from this repository. In contrast, Native Experimentation (NEL) requires a thorough knowledge of the SmartSantander infrastructure and how the different nodes actually work. This type of experimentation is considered a low level experimentation as it directly

accesses the nodes and its hardware completely adapting its behaviour to the experiment requirements.
As part of the FIESTA-IoT project, the focus will be put on federating the SEL functionalities offered by the platform. In this sense, SmartSantander provides a set of mechanisms to work with the information and resources available in the testbed in an Experimentation-as-a-Service manner.
SmartSantander implements a REST interface which gives access to the different resources/devices, including characteristics and available data (i.e. historical and last values). In order for the data and the resources to be stored in the SmartSantander platform, repositories are a major part of the implementation of the testbed.

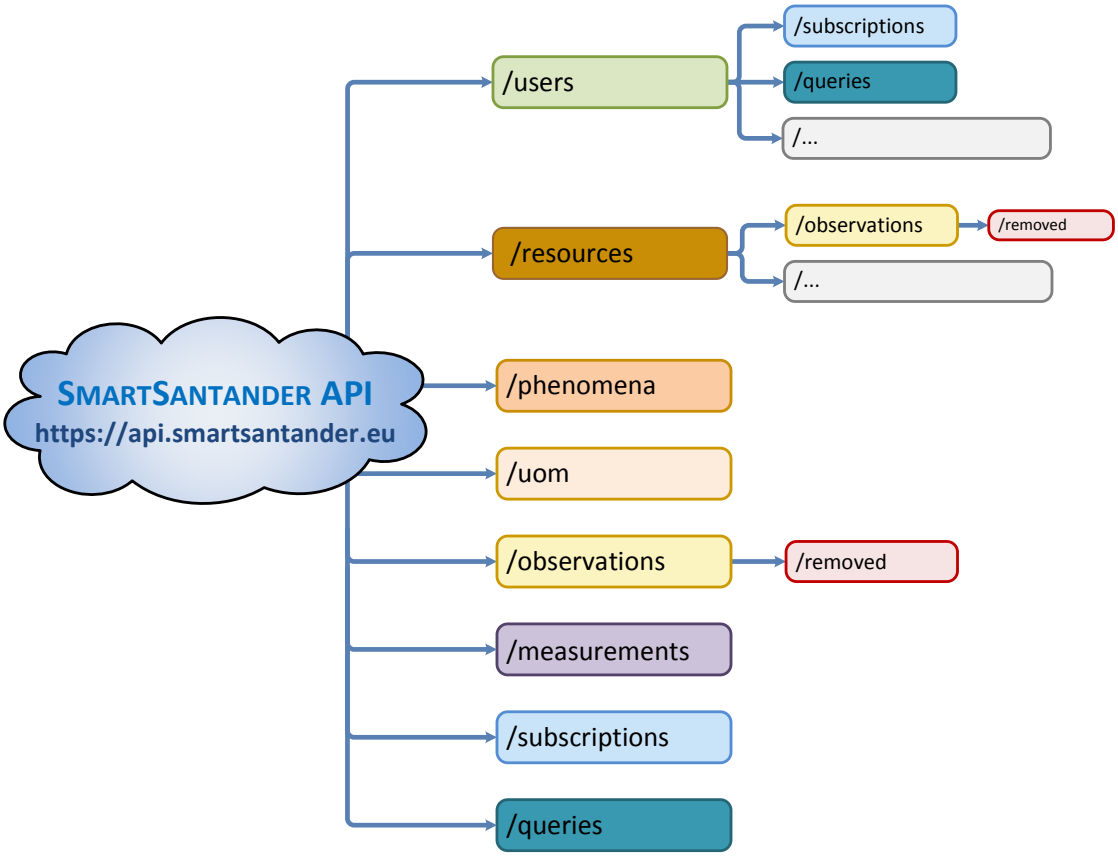


Figure 13. SmartSantander IoT API resources tree

Each of the REST resources mapped in the tree representation shown in Figure 13 enables access to the different SmartSantander testbed functionalities.

Most significantly, the API enables operations over the testbed resources as well as over the measurements and observations they produce.

The organization of the API allows multiple ways of accessing the information depending on the necessities and focus from the experimenters.

The API supports both synchronous (i.e. “queries” following request-response model) as well as asynchronous (i.e. “subscriptions” following publish-subscribe model) access.

The same API is used for inserting and retrieving information.

Last but not least, synchronous queries do not only provide access to last values but also allow accessing historic information as well.

Complete API documentation is available at (https://api.smartsantander.eu/docs).

IoT API

SmartSantander implements a REST interface which gives access to the different resources/devices, including characteristics and available data (i.e. historical and last values). As part of the SmartSantander platform, implementation depends on the repositories to store data and resources.

3.2.5 Data format

SmartSantander platform has defined information models both for the observations generated by the deployed IoT devices and for the resource descriptions. These models rule the content of the queries and responses used when accessing the SmartSantander platform in its native manner through the IoT API. For both cases a JSON schema has been specified. However, dependency on JSON is just an implementation decision, whereas the framework architecture is not language dependant and would support any other representation.

- **Monitoring info data format for experiments/services:**

As it has been already introduced, each one of the aforementioned devices represents a context information source. They generate observations, which are composed of one or several sensor measurements, each of them referring to one phenomenon.

Design of the observations model has been mainly driven by the following requirements:

- Native representation format is JSON although it can be easily translated to other formats.
- Devices reporting information to the SmartSantander platform MUST generate their observations according to the observation model.
- Observations MUST identify its originator.
- Observations SHOULD be positioned in terms of location
- Observations MUST be positioned in terms of time.
- A device can be equipped with multiple sensors so that observations can contain information pertaining to multiple different phenomena.
- Metadata of the observations can be accessed through the capabilities defined within the resource description of the observation originator (i.e. service-related section).

Table 5. SmartSantander's Service observation JSON format

Description:	JSON format to manage the measurements that reach the SmartSantander platform.
Standard /	Proprietary. JSON schema has been defined. (the format, the content

proprietary	is defined for SmartSantander exclusively)

Example:	{
	"urn": "urn:x-iot:smartsantander:u7jcfa:f3001",
	"timestamp": "2015-05-06T18:55:50+02:00",
	"measurements": [
	{
	"value": 17.2,
	"uom": "degreeCelsius",
	"phenomenon": "temperature:ambient"
	},
	{
	"value": 28,
	"uom": "percent",
	"phenomenon": "relativeHumidity"
	},
	{
	"value": 0.1,
	"uom": "microgramPerCubicMetre",
	"phenomenon": " chemicalAgentAtmosphericConcentration:CO"
	},
	{
	"value": 0.83,
	"uom": "microgramPerCubicMetre",
	"phenomenon": "
	chemicalAgentAtmosphericConcentration:airParticles"
	},
	{
	"value": 120,
	"uom": "microgramPerCubicMetre",
	"phenomenon": " chemicalAgentAtmosphericConcentration:O3"
	},
	{
	"value": 116,
	"uom": "microgramPerCubicMetre",
	"phenomenon": " chemicalAgentAtmosphericConcentration:O3NO2"
	},
	{
	"value": 13,
	"uom": "metre",
	"phenomenon": "position:altitude"
	},
	{
	"value": 24,
	"uom": "kilometrePerHour",
	"phenomenon": "speed:instantaneous"
	},
	{
	"value": 285,
	"uom": "degreeAngle",
	"phenomenon": " direction:azimuth"
	},
	{
	"value": 47536,
	"uom": "kilometre",
	"phenomenon": " mileage:total"
	}
	],
	"location": {
	"coordinates": [-3.80961, 43.4612],
	"type": "Point"
	}
	}

- **Resources naming:**

Unique naming scheme has been defined within the SmartSantander platform. Every resource in the testbed is univocally identified through a name assigned according to this scheme.

Table 6. SmartSantander's resource naming

Description:	URN for each resource
Standard / proprietary	It is based on the RFC's RFC2141 and RFC3406. The name is divided in a Prefix, a Domain, an index for the provider, manufacturer or owner, which is a random alphanumeric value, and a local Id (chosen by the owner of the resource)
Example:	As an example of URN: "urn:x-iot:smartsantander:u7jcfa:t235

- **Resources description format**

IoT devices deployed within the SmartSantander testbed are described following a proprietary model. Resource descriptions are available and contain information of interest for the experimenters indicating which capabilities the device has.

Thanks to this, resource discovery is possible through the IoT API. Look-up queries can ask for any of the properties defined as part of the resource description model.

Design of the resource model has been mainly driven by the following requirements:

- Resource model should be scalable and extendable to support the already existing heterogeneity as well as the foreseen increase of such heterogeneity.
    - Resource description structure is pre-defined; resources' features are mostly not pre-defined.
    - Only a reduced set of mandatory parameters are included on every resource description.
    - There is a set of pre-defined parameters that might be included on the resource descriptions.
    - Key-value pairs are mostly used to introduce new features for the different resource descriptions.
- Native representation format is JSON although it can be easily translated to other formats.
- Resource model should mainly support service experimentation, native experimentation and infrastructure management.

Table 7. SmartSantander resource description JSON format

Description:	JSON (see example below).
Standard / proprietary	Proprietary. JSON schema has been defined.

Example:	{
	"identification": {
	"urn": "urn:x-iot:smartsantander:u7jcfa:t355",
	"parent_id": "urn:x-iot:smartsantander:u7jcfa:M05",
	"resource_type": "SERVICE_NODE",
	"virtual_device": false,
	"topics": ["t_filab"]
	},
	"description": {
	"attributes": [


```
      {"node_desc":"Temperature and light sensor"},
      {"golden_image":"sms-v5.4"}
    ]
  },
  "service": {
    "capabilities": [
      {
        "phenomenon": {
          "name": "temperature:ambient",
          "ref":
"http://purl.oclc.org/NET/ssnx/qu/quantity#temperature"
        },
        "uom": {
          "name": "degreeCelsius",
          "ref":
"http://purl.oclc.org/NET/ssnx/qu/unit#degreeCelsius"
        },
        "metadata": [
          {
            "name": "Frequency",
            "value": 300,
            "ref": "http://purl.oclc.org/NET/ssnx/ssn#Frequency"
          },
          {
            "name": "Accuracy",
            "value": 1.5,
            "ref": "http://purl.oclc.org/NET/ssnx/ssn#Accuracy"
          }
        ]
      },
      {
        "phenomenon": {
          "name": "illuminance",
          "ref":
"http://purl.oclc.org/NET/ssnx/qu/quantity#illuminance"
        },
        "uom": {
          "name": "lux",
          "ref": " http://purl.oclc.org/NET/ssnx/qu/unit#lux"
        },
        "metadata": [
          {
            "name": "Frequency",
            "value": 300,
            "ref": "http://purl.oclc.org/NET/ssnx/ssn#Frequency"
          },
          {
            "name": "Accuracy",
            "value": 10,
            "ref": "http://purl.oclc.org/NET/ssnx/ssn#Accuracy"
          }
        ]
      }
    ]
  },

  "management": {
    "status": "READY",
    "created": "2014-04-30T11:41:01.123+02:00",
    "updated": "2015-05-03T01:24:21.583+02:00",
    "logs": [
      {
        "event": {
          "type": "FAILURE"
        },
        "timestamp": "2014-07-13T10:31:21.312+02:00"
      },
      {
```

```
          "event": {
            "type": "FLASH",
            "description": "Golden image v0.3.5"
          },
          "timestamp": "2014-11-21T14:01:41.012+02:00"
        },
        {
          "event": {
            "type": "DISCONNECTION"
          },
          "timestamp": "2015-02-11T07:51:34.702+02:00"
        },
        {
          "event": {
            "type": "FAILURE"
          },
          "timestamp": "2015-01-13T10:31:21.312+02:00"
        }
      ]
    }
}
```


3.2.6 Security

IoT API is provided through REST services on top of HTTPS.

Authentication and access control is provided through two different means:

- SmartSantander authenticates and authorizes accesses to its services by using a Certification Authority (CA) certificate, based on the X.509 v3 standard (Cooper D., 2008).
- Additionally, SmartSantander platform issues API keys that can be used to get access to the platform services.

3.2.7 Applications and GUIs

SmartSantander implements three applications to make the citizens participants of the testbeds and provides a gateway to access the SmartSantander data.

- **Participatory Sensing**

Citizens can become a proactive part of a Smart City. From this affirmation, SmartSantander exploits their smartphones capabilities, thus making people able to share and contribute with data and observations. In this scenario, citizens, Santander City Council and the local newspaper “El Diario Montañés” are connected into a common platform where they can report, share and be notified of events happening in the city. As an example, a user walking in the city centre who finds a hole in the pavement can take a picture, write a text and finally share this incidence with other users of the application. The Santander City Council will therefore be notified of the occurrence of the event and proceed accordingly by sending an employee to the location in order to fix this problem.

Figure 14. SmartSantander Participatory Sensing smartphone snapshots (iOS)

- **Augmented Reality (AR) service**

Developed as part of the SmartSantander project, includes information about more than 2700 places in the city of Santander, classified in different categories: beaches, parks and gardens, monuments, buildings, tourist information offices, shops, art galleries, libraries, bus stops, taxi ranks, bicycle hire points, parking lots and sports centres. Furthermore, it allows real-time access to traffic and beach cameras, weather reports and forecasts, public bus information and bike hire service, generating a unique ecosystem for end users when moving around the city.

Figure 15. SmartSantander Augmented reality smartphone snapshots (iOS)

- **Online map**

Last, but not least, SmartSantander provides an online map to see the installed resources online. It can be accessible following next link: http://maps.smartsantander.eu/ .

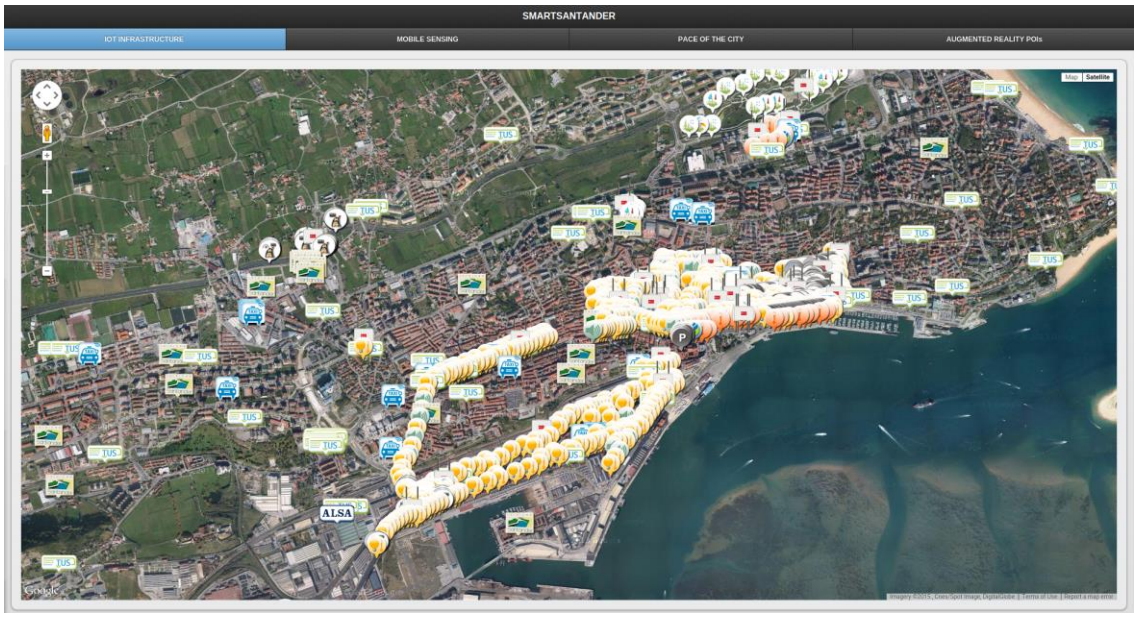


Figure 16. SmartSantander online map snapshot (web browser)

3.3 UNIS Testbed

3.3.1 Technical architecture overview

Figure 17 provides an overview of the network architecture of the Smart Campus testbed.

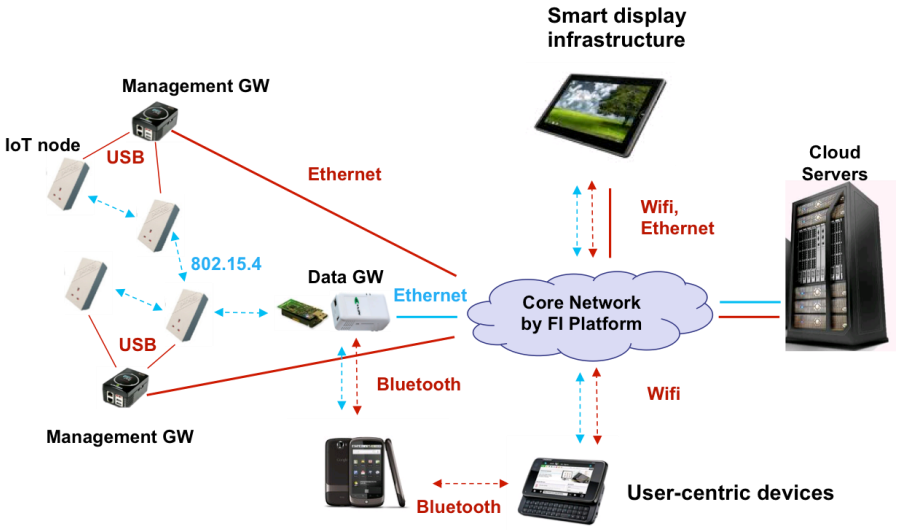


Figure 17. Network Architecture of Smart Campus testbed

Three tiers can be identified:

i) A **server** tier that hosts all the back-end functionalities of the testbed and provides the entry point for an experimenter to access the testbed,
ii) An embedded **Gateway** (GW) tier which forms the testbed infrastructure and allows the
iii) **IoT** tier which is connected and reachable to a backbone network through WiFi or Ethernet

Although all the tiers can be involved in each experimentation phase, the IoT tier represents the user-centric component of the testbed, merging embedded IoT nodes with sensing capabilities together with more higher-end user-centric devices such as Smartphones and Smart Displays. Each IoT node provides two forms of connectivity:

i) Wireless communication capabilities (IEEE 802.15.4, and through the GW devices WiFi and Bluetooth) that can be exploited during an experiment in order to form different kinds of networks,
ii) A wired USB connection to a dedicated GW for management purposes

3.3.2 Physical Resources

With the exception of QR-code/NFC stickers deployed outdoors, all the testbed resources are currently deployed over the three floors of the 5G ICS building covering almost every desk present. Figure 18 shows an overview of the deployment.

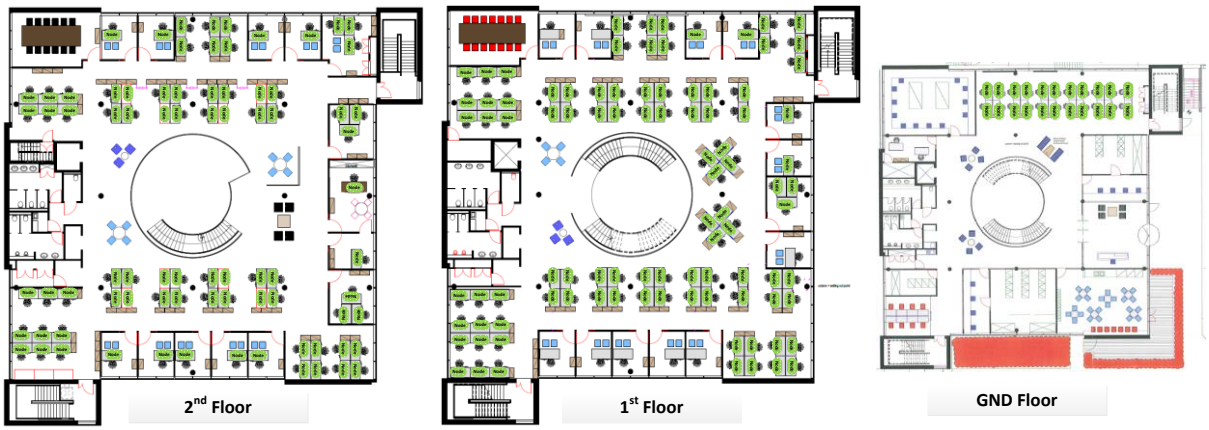


Figure 18. Indoor testbed deployment overview

The main hardware components of the testbed are described in the following.

- **IoT nodes**. Figure 19(left) shows the main elements composing an IoT node. The main design principle of this device was to provide a vast set of sensing modalities for smart building environments. For this purpose, a TelosB mote has been interfaced with an off-the-shelf energy meter (Plogg) allowing the monitoring of energy use of user-related appliances in their work environment and basic on/off actuation. A custom designed multi-modal sensor board is connected to top of the IoT node casing. The board provides additional sensing modalities in the form of relative amount of light, relative noise level, temperature and motion within the range of the IoT node through a PIR sensor. A vibration sensor has also been included to determine tampering with the device, while an LED provides feedback to the user about the device state. 200 of such IoT nodes have been deployed at each work desk in the SmartCCSR building, connecting utilized user appliances (desktop computer, LCD screen, lamp, fan, charger and laptop) via a multi-socket.

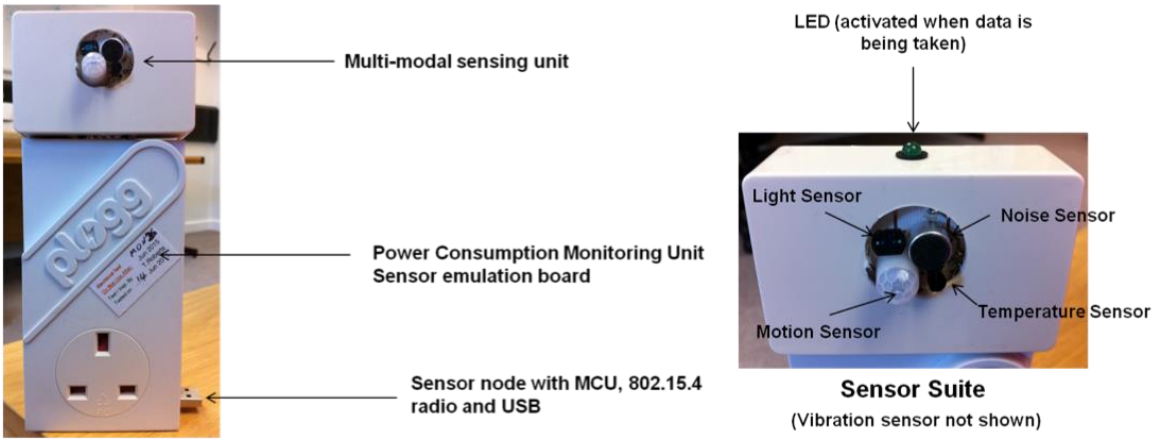


Figure 19. IoT node

- **Embedded GWs**. The GWs tier wires the IoT nodes to the testbed and guarantees the realization of a reliable management plane. It consists of 100 embedded Linux GWs deployed across all rooms of the building. The selected platform is a GuruPlug providing a 1.2GHz Marvell Embedded CPU, coupled with 512MB of RAM and 512MB of storage space available for deploying user software. While allowing wired USB connectivity of external devices, Ethernet, WiFi and Bluetooth technologies are also available for network connectivity. The ratio of GW nodes to IoT nodes is between 1:1 to 1:4, depending on the number of IoT nodes that are deployed in a room and availability of Ethernet ports in the office space for the connection of GWs to the backbone network.
- **Smart Displays**. The testbed provides also a public display infrastructure offering enhanced user interaction capabilities inside of the 5G IC building. The testbed relies on 10 Android tablets (Samsung 10.1) deployed at strategic locations of the building as part of a permanent installation. These 'Smart Displays' are linked through WiFi to the testbed infrastructure and provide Bluetooth for localized device interactions. Example use cases that utilized these displays are indoor guidance systems for visitors and for emergency evacuation.
- **Servers**. A server cloud hosts the testbed management servers and allows the on-demand creation of other application servers and data management tools. It consists of 10 High End Servers (12 Xeon Cores, 24GB RAM each) with 8TB of storage and a VMware Cloud Computing Platform.

Testbed upgrades are currently planned and in the process to be developed. Each desk or cluster of them will be provided with a so called *Smart Egg*, based on Seeduino Arch platform (ARM Cortex M-0 and M-3) featuring the following capabilities:

- Bluetooth and WiFi connectivity
- Wide range of sensing capabilities: light, humidity, temperature, noise, Co2 level, ranging and proximity sensors;
- Wide range of feedback/actuation capabilities: vibration motor, multi-colour LED. APIs to access such capabilities will be designed and implemented.

Figure 20 shows a concept image of the Smart Egg.

Figure 20. Smart Egg

3.3.3 Federation

Federation is performed using the SFA-wrapper approach. An Aggregation Manager (AM) has been implemented and deployed at each federated site. The AM exposes a homogenous set of APIs for resources retrieval and reservation across all the federated testbed. Internally the Aggregation Manager has been implemented by wrapping the required APIs among the natively provided testbed APIs (WISEBED web services, http://www.wisebed.eu/).
Every testbed implementing the same set of SFA APIs will be compatible with this federation. Internal implementation might differ from testbed to testbed and each testbed can provide more or less functionalities.
Each federated testbed needs to provide an implementation of the functionalities required by the Aggregation Manager. The Aggregation Manager stub is implemented in Python. Smart Campus testbed provides a REST interface as a wrapper for the WISEBED web services APIs in order to provide required AM functionalities. AM APIs are implemented by calling dedicated REST APIs. This way the complexity of the implementation is moved outside the AM.
SFA doesn't require central components; however resources should be described in a homogenous format (i.e., using RSpec). A virtual machine can be provided. A Testbed manager will only need to implement the Aggregation Manager APIs.

3.3.4 Internal management methods and APIs

Testbed is able to perform the following:

- Monitor status of GWs and IoT nodes;
- Detect disconnection and failures of GWs and IoT nodes;
- Reset, reprogram IoT nodes;
- Check IoT nodes status, i.e., reserved or free for experimentation.

- The ability to register new devices/resources has been designed but implementation is not yet finalized.

- **API/functionality for exposure of existing and available resources/devices**

Description:	WISEBED Reservation APIs
Portability /reusability	Deployed as Java application on any web-server
External information source:	https://github.com/itm/testbed-runtime

- **API/functionality to retrieve generated information from the testbed**

Description:	homemade REST APIs
Portability /reusability	Installed on any participating testbed as part of any web-server environment (e.g. Tomcat). However current implementation is highly bounded to data format.
External information source:	Documentation uploaded on the SVN: https://subversion.deri.ie/fiesta-iot/WP2/Task%202.2/REDUCE-database-access.docx

- **Data format (Open data formats and Communication protocols format)**

The communication protocol is HTTP, and employs a REST-Easy framework for implementation of REST APIs. A Detailed description of the API can be found at: https://subversion.deri.ie/fiesta-iot/WP2/Task%202.2/REDUCE-database-access.docx

- **Monitoring info data format for experiments/services**

Description:	APIs to retrieve data about energy consumption at each desk. Data can be access per node_id, floor, room.
Standard:	Non-standard – JSON or XML format are available but data are not based on existing ontology. It provides: temperature, light, presence, noise level, watts
Example:	http://131.227.23.2:8080/SmartCCSR-testbed/restful-services/REDUCE/sensors/{node_id} http://131.227.23.2:8080/SmartCCSR-testbed/restful-services/REDUCE/json/sensors/{node_id}

- **Resources naming**

Description:	APIs to retrieve a map of deployed resources
Standard:	Non-standard – JSON or XML format are available but data are not based on existing ontology. It provides nodes id and room, floor id.
Example:	http://131.227.23.2:8080/SmartCCSR-testbed/restful-services/REDUCE/map (for XML data format)

	http://131.227.23.2:8080/SmartCCSR-testbed/restful-services/REDUCE/json/map

3.3.5 Data format & Ontologies

The Surrey Test bed uses the two following ontologies:

- **IoT-lite:** The IoT-lite (see Figure 21 below) is intended to be a "lite" ontology for describing devices, sensors, services and object (Virtual Entities). It is simpler than the IoT-A ontology but keeps its "spirit". It also references a fragment of SSN (dealing with Sensor description). The following figure shows the main concepts of the ontologies and how they relate to each other.

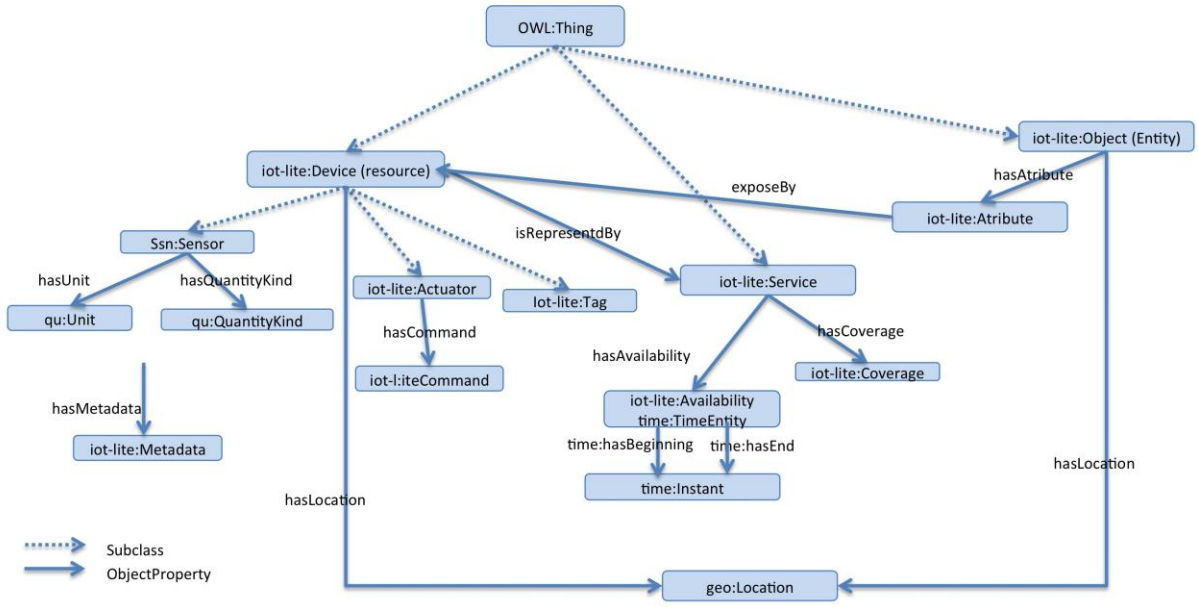


Figure 21. IoT-lite ontology

- SAO: The Stream Annotation Ontology (see Figure 22) allows the representation of real-time data streams and time series. It links to additional ontologies like OWL-time and SSN:observation.

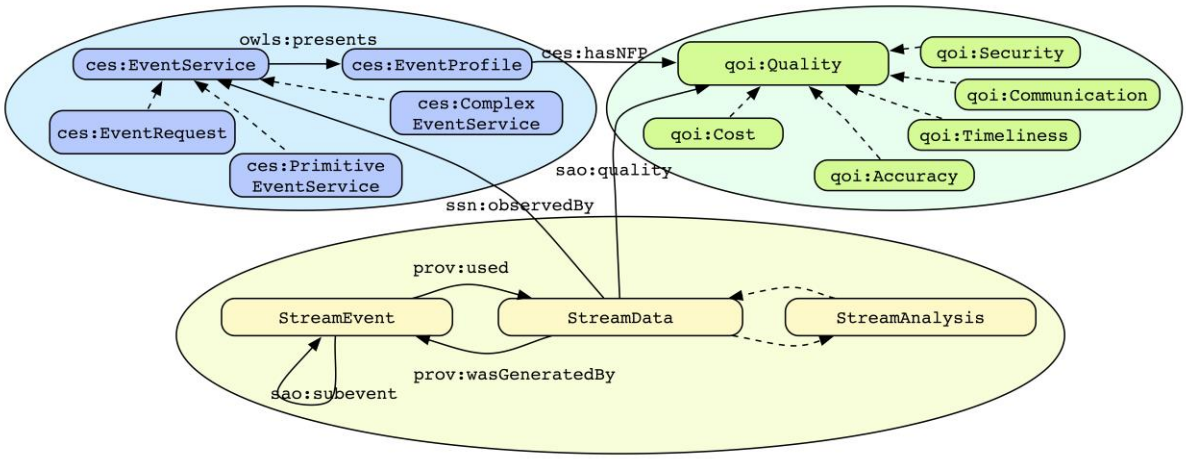


Figure 22. SAO ontology

3.3.6 Security

Access is granted by the testbed manager via email. With regards to authentication and authorization, the Shibboleth software package is used for accessing the testbed resources. It is completely transparent to the user, since only the username and password is required. It is also planned to have OAuth access to the data generated by the testbed. To ensure trust relations, any requests for new accounts must be generated from a valid provider's account.

3.3.7 Applications and GUIs

The MyGuardian and MyCampus app provide a means to access geo-localized services and information from students in the University of Surrey campus when tagging actions are performed towards the deployed QR-code/NFC stickers. User distribution data within the campus is collected and stored accordingly.

In particular the MyGuardian application allows tagging the path of people within the campus and share progress with a selected group of trusted friends using the app. A screenshot is provided in Figure 23.

Figure 23. MyGuardian app

For data visualization, the MyEcoFootprint GUI visualizes energy data related to energy consumption at work desk. A screenshot is provided in Figure 23.

3.4 KETI Testbed

3.4.1 Technical architecture overview

The KETI testbed is an experimental system originally developed to monitor the indoor environmental condition in a KETI's building like temperature, humidity, illumination, human presence, and then allow the building energy management system (BEMS) to make energy consumption efficiency. There are three tiers composing of the testbed: server platform (Mobius), device platform (&Cube), IoT devices:

- Server platform (Mobius): the Mobius platform is an open IoT service platform complying with oneM2M standards (i.e., oneM2M standards-defined service platform, IN-CSE) and released in 2014 by KETI. Figure 24 shows the overall architecture of the Mobius.
  - It supports indexing functionality to identify a vast number of IoT devices, but also searching functionality to easily discover IoT devices of interest, e.g., location- or tag-based search. Users (including service providers with IoT devices) register/deregister a set of IoT devices and configure access control right to the devices.
  - For data update/retrieval, the Mobius offers a suite of REST APIs complying with oneM2M standards. Details are described later.
  - All the functionalities are offered through REST APIs so that developers can use them to create new services and applications. As an example, KETI developed a browser app for smartphones using the open APIs related to indexing and searching functions, called IoT Browser, where IoT devices registered in the Mobius can be explored on smartphone maps (e.g., Google maps) in terms of search criteria like location and tag.
  - For IoT ecosystem, the Mobius is designed to work with app stores for IoT devices. For example, when a user chooses a device on the IoT Browser, it shows a list of smartphone apps for using the device, and then brings the user to the app store (e.g., Google Play) to install the selected application.
  - The KETI and Korean government has established an open source-based global partnership for IoT standards, OCEAN (Open alliance for iot standard) in 2014, and opened the source code and binary files of the platforms. OCEAN aims at providing development and commercialization services with coverage of IoT platforms, devices, and services, which are based on IoT standard-based open sources association.
  - The following web sites provide further information:
    - Mobius web portal: `www.iotmobius.com`
    - Mobius community portal: `www.open-iot.net`
    - OCEAN website: `www.iotocean.org`
    - Device API repository: `www.programmable-things.net`
    - IoT Browser (Google Play): `play.google.com/store/apps/details?id=com.einsware.iot_browser`
    - IoT Browser (Apple store): `itunes.apple.com/kr/app/iot-browser-search-explore/id727763099`

- The Gateway and device platform (&Cube): the &Cube is a software platform (i.e., a middleware) running on the Java runtime environment (JRE), and designed to serve as the oneM2M standards-defined platform for gateways and devices (MN-CSE and ASN-CSE).
  - The &Cube is developed to run on the JRE-based machine, so it could be ported to any systems like gateways and IoT devices installed with the Java virtual machine.
  - Two versions of the &Cube are developed: &Cube-Rosemary (MN-CSE, for gateway) and &Cube-Lavender (ASN-CSE, for IoT devices).
  - As can be shown in Figure 24, IoT devices can be connected with the Mobius through a gateway installed with the &Cube-Rosemary, but also directly via the &Cube-Lavender.
  - For network protocol binding, the &Cube supports a lightweight HTTP server for HTTP/CoAP protocols, and MQTT client for subscription/publication messaging with the MQTT broker residing in the Mobius.
  - The source code, binary fines, and developer guides written in Korean (English translation work is underway) are openly available from the OCEAN website.

- IoT devices: are responsible for collecting data about the indoor environmental condition of part of the KETI's building.
  - The majority of the IoT devices are wirelessly connected with a set of gateways through a ZigBee-based network. All the gateways are developed with Raspberry-Pis and installed with the &Cube-Rosemary.
  - Data is collected from IoT devices and sent to the Mobius in a standardized form, i.e., a oneM2M-defined protocol and data format (e.g., JSON or XML). Experimenters can retrieve the collected data using REST APIs offered by the Mobius. Details are described later.

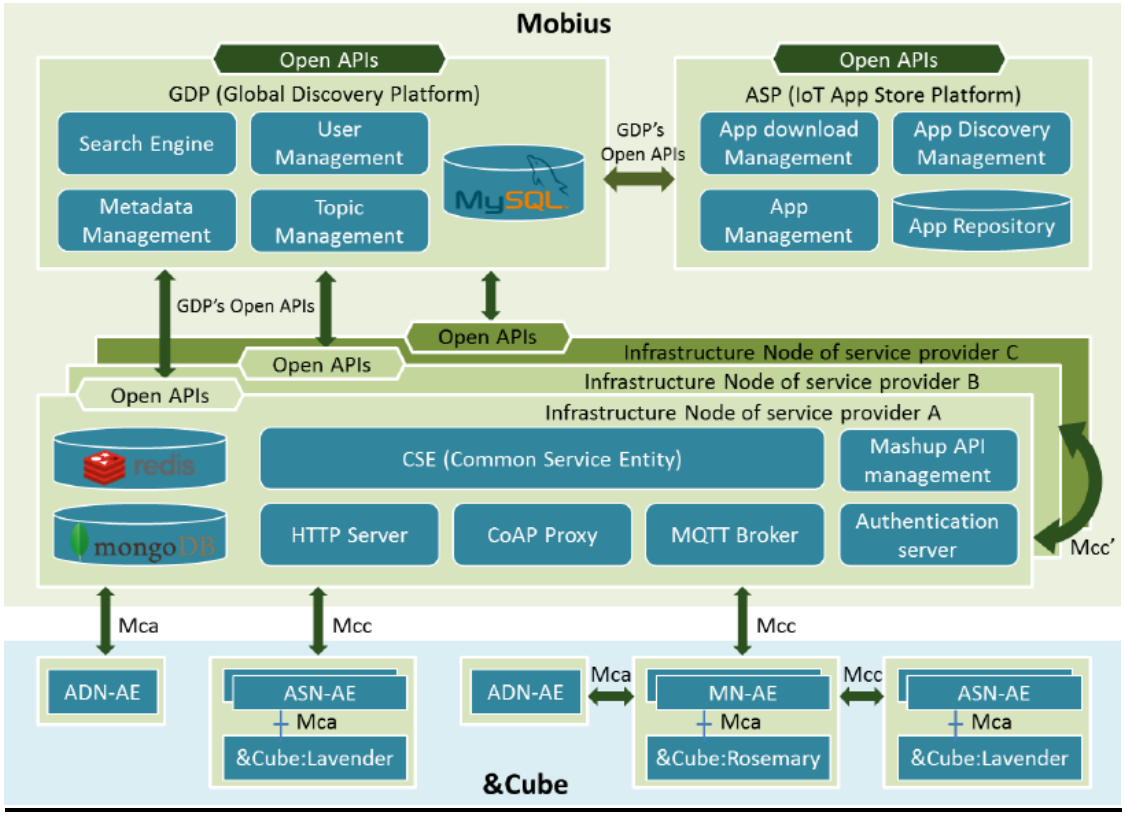


Figure 24. Overview of Architecture of KETI' IoT Platform

3.4.2 Physical Resources

The KETI testbed has been deployed on the 5th floor of a KETI building for various experimental studies. The detailed descriptions on the KETI testbed are follows:

- The KETI testbed is established to monitor and collect sensing data from a set of areas of offices including meeting area, relaxing area and work area in an office and hallway. Through the deployed sensors, data such as occupant presence in the office, indoor environment condition (e.g., temperature, humidity, illumination, CO_2), and electricity consumption could be easily collected and transferred to the Mobius server platform for further processing and analysis.
- The KETI testbed is composed of 160 compound sensors, evenly shared by 4 kind of sensors (temperature, humidity, illumination and presence sensor), 10 CO_2 level detection sensors, and 10 smart sockets designed to control power on and off to reduce the electrical power consumption, 180 sensors in total.
- The deployment of the KETI testbed is shown in Figure 25. The 180 sensors are deployed within 7 offices located in KETI's fifth floor.
  - 2 compound sensors and one CO_2 sensor are deployed in the seminar room located on the left upper side;
  - 6 compound sensors, 2 CO_2 sensors, and 2 smart sockets are placed in the hallway of the KETI's 5th floor;

- For each of four middle-sized offices, 4 compound sensors and one CO_2 sensor are evenly deployed on the ceiling of each office;
- For one large-sized office, 6 compound sensors, 5 CO_2 sensors and one smart sockets are evenly deployed;
- 10 compound sensors, 3 smart sockets and 2 CO_2 sensors are evenly deployed in an extra-large-sized office.

- Two principle monitoring tasks can be undertaken using the KETI testbed.
  - Indoor Environmental Monitoring
    - CO^2: 10 CO_2 sensors are installed to detect indoor CO_2 concentration;
    - Temperature, Humidity and Illumination information: 40 compound sensors for each environmental parameter monitoring with a total of 120 compound sensors;
    - Office workers' presence: 40 compound sensors are deployed to monitor and collect the presence of office workers as a data resource for further analysis.
  - Electricity Monitoring: 10 smart sockets are installed in hallway and two relatively large-sized offices (shown as Figure 25) to monitor the working state of office equipment such as laptops, printers, scanners, projectors or even coffee machines etc. and collect the data on their energy consumption during a specified period.
- The sensing data of indoor temperature, humidity, illumination, employee's occupancy, and office equipment electricity consumption can be measured at a regular interval. This data can be stored for further analysis.

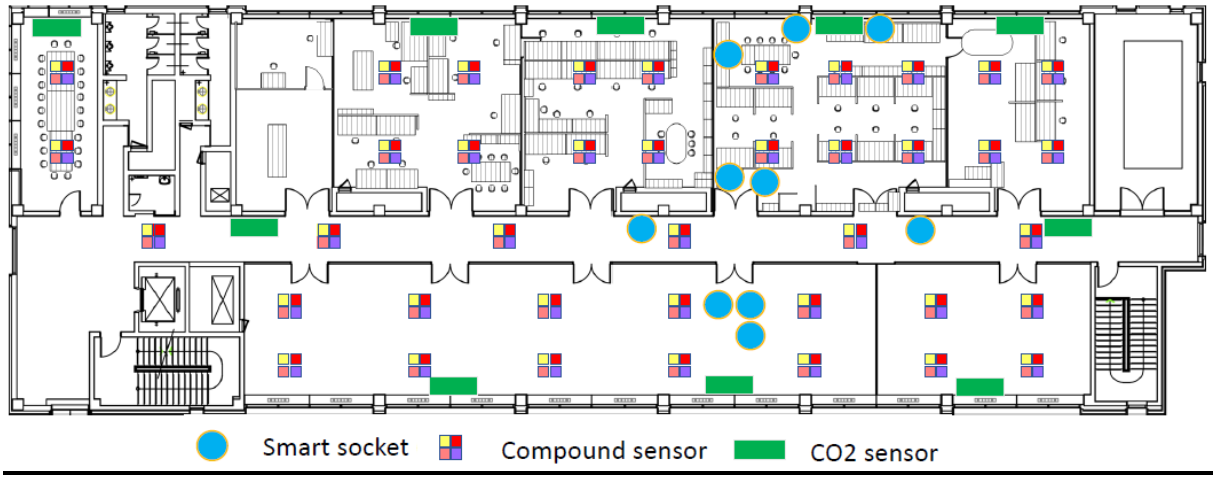


Figure 25. Overview of deployment of KETI testbed

3.4.3 Federation

The KETI testbed is currently not federated with other testbeds. However, with the use of REST APIs supported by the Mobius, light federation could be achieved. In other words, the data collected from the KETI testbed could be easily accessed from

federated members through REST API-based access, even though creation and update of resources will be restricted according to appropriate access credentials. The requirements for the future development of the federation approach shall include the following:

- Documented REST API for our testbed access.
- Support of oneM2M standard interfaces (Mca, Mcc').
- Support of oneM2M resource description format.
- Processing access key

3.4.4 Internal management methods and APIs

The KETI testbed is designed to be able to implement the following functionalities:

- Real-time monitoring functionalities
  - Monitoring the working status of the deployed sensors and actuators;
  - Detect disconnection and malfunction of the deployed sensors and actuators;
    - A software module in the Mobius is implemented for scanning active sensors through checking the active frequency for the environmental monitoring. For example, if a sensor seems inactive and cannot be accessed, it would be regarded as malfunctioning sensors and will be reported to the central control module.
    - The functionality of reprogramming remote sensors will be added to KETI testbed.
  - Monitoring of the presence of workers in the offices will work at almost real-time manner.
- Indoor environmental monitoring including temperature, humidity, illumination or electricity consumption is conducted for a specified period of time (1 min at present).
- Provision of API for exposure of existing and available resources/devices.

Availability:	Yes. It is available through oneM2M-defined APIs.
Description:	It is a REST interface which provides access to the existing data resources including data attributes and available data including historical and latest state values.
Is it portable and/or reusable (license):	It is a part of the Mobius. Repositories could be accessed by the authenticated users having appropriate issued keys.
External information source (English):	For more details, please see oneM2M API sections.

- Provision of API for retrieval of the generated information from the testbed.

Availability:	Yes. It is available through oneM2M-defined APIs.
Description:	OneM2M provides subscription/notification interface to access to the historical data resources asynchronously. In addition, oneM2M supports data retrieval including application entity (AE), AE container, and AE content

	instance retrieving, which are terms defined in oneM2M standard. Here we provide an example to show some rules we have to follow when we send a RESTful HTTP GET request to retrieve the data from Mobius platform: Request: **GET/Mobius/remoteCSE-0.2.481.1.0001.001.975/container-temperature/latest?countPerPage=1&startIndex=1 HTTP/1.1** **Host**: open.iotmobius.com **Accept**: application/onem2m-resource+xml **Content-Type**: application/onem2m-resource+xml **uKey**: YTFlMWQ1YWMtYTEyOC00MzhiLTg5ZTAtOTkxYjlhMzZkNjAx **From**: mobiusX-M2M-RI: 12345 Response: HTTP/1.1 200 OK **Content-Length**: 651 **Content-Type**: application/onem2m-resource+xml;charset=UTF-8 http://open.iotmobius.com/Mobius/remoteCSE-0.2.481.1.0001.001.975/container-temperature/latest-CI00000000002354345 **X-M2M-RI**: 12345 <? xml version="1.0" encoding="UTF-8" standalone="yes"?> <m2m:listOfResources xmlns:m2m="http://www.onem2m.org/xml/protocols" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="contentInstance"> <contentInstance> <resourceType>4</resourceType> <resourceID>CI00000000008588599</resourceID> <parentID>CT0000000171</parentID> <creationTime>2015-03-18T17:19:03+09:00</creationTime> <lastModifiedTime>2015-03-18T17:19:03+09:00</lastModifiedTime> <stateTag>0</stateTag> <typeOfContent>celsius</typeOfContent> <contentSize>4</contentSize> <content>MA==</content> // Base-64 encoded temperature value, e.g., 24^0C </contentInstance> </m2m:listOfResources>
Is it portable and/or reusable (license):	OneM2M APIs provide functionalities for data resource retrieving. It's a part of oneM2M standard.
External information source (English):	For more details on oneM2M, please visit oneM2M official website.

- Provision of adding new IoT devices through a REGISTRATION process. The sensing data from the new added IoT devices can be collected and stored in the Mobius platform. The Mobius can also support to combine the new collected data resources with existing data resources for providing mash-up services.

3.4.5 Data format (Open data formats and communication protocols format)

Open data formats and IoT network communication protocols formats are supported by the Mobius platform.

- The Mobius platform is implemented complying with the oneM2M standard which applies to a RESTful architecture so that the Mobius platform uses a standardized data format for communicating and exchanging data with other platforms which also conform to the oneM2M standard.
- A standardized XML Schema Definition Language (XSD) is used to describe the sensing data collected from IoT devices, to define the resource naming conventions, and to describe the resources in terms of <AE>, <CSE>, <container>, and <contentInstance>, shown in the following tables, respectively.

Description:	Simple Data types incorporated from XML Schema are used to describe data resources collected from IoT devices registered with Mobius platform. Resource common attributes specified in oneM2M standard are also described by W3C XML Schema Definition Language (XSD).
Is it a standard?	Yes. W3C XML Schema Definition Language (XSD) is a W3C Recommendation.
Example:	**<? xml version="1.0" encoding="UTF-8" standalone="yes"?>** <m2m:contentInstance **xmlns:m2m="http://www.onem2m.org/xml/protocols" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"**> <resourceType>4</resourceType> <resourceID>CI00000000008034166</resourceID> <parentID>CT0000300566</parentID> </m2m:contentInstance>

Description:	Three resource type<AE> (application entity), <container>, and <contentInstance> are defined in oneM2M standard to describe three kind of resources.
Is it a standard?	Yes. Each type of resource specified in the oneM2M standard is also described by W3C XML Schema Definition Language (XSD), which is a W3C Recommendation.
Example:	<? xml version="1.0" encoding="UTF-8" standalone="yes"?> <**m2m:contentInstance** xmlns:m2m="http://www.onem2m.org/xml/protocols" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <resourceType>4</resourceType> <resourceID>CI00000000008034166</resourceID> <parentID>CT0000300566</parentID> <creationTime>2015-03-02T11:01:34+09:00</creationTime> </**m2m:contentInstance**>

3.4.6 Security

Several security approaches are deployed in the Mobius platform to guarantee secure and authenticated access to the KETI testbed, and the integrity of the data resources during collection and storage of the sensor data in the following:

- Any users who register their IoT devices into the Mobius are granted to access the data resources associated with their registered IoT devices using a user-

access key called u-key, and to execute specified operations associated with the functionalities supported by the Mobius platform.

- An authorization mechanism based on Access Control List (ACL) is used to guarantee the authorized access to and modification on data resources, according to the Access Control Policies (ACPs) defined as a <accessControlPolicy> resource.
- The security mechanism of the Mobius will be changed as complying with oneM2M standards.

3.4.7 Applications and GUIs

KETI has released a series of applications along with the Mobius platform to demonstrate services offered by the Mobius platform in the following:

- Indoor environment & energy use monitoring:
  - Aims to develop a monitoring system which can measure the indoor building environment including: temperature, humidity, illumination, CO_2, occupancy, as well as electrical energy consumed in office utilities.
  - Based on the measured data, a mobile application (called C4 solution) is developed, which allows users to monitor the indoor environment and electricity consumption in terms of specific areas of the room.

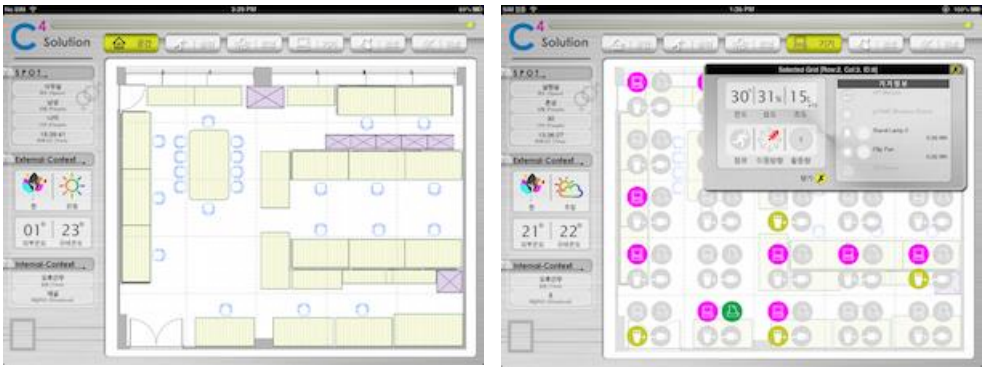


Figure 24. Indoor Environmental Monitoring including temperature (indoor/outdoor), humidity, illumination, employees presence, office equipment electricity consumption

- iThing
  - aims to develop a voice command based home appliance controlling system
  - iThing is a mobile application which enables users to control home appliances with their verbal command through smartphone microphones, as shown in Figure 26.
  - The device control is performed with smart sockets we have explained above.
  - For watching the demo video, https://youtu.be/6pe1HdpUOnA.

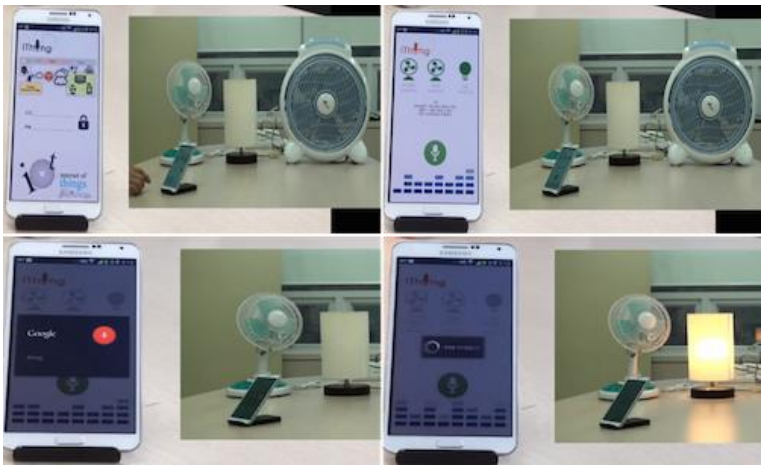


Figure 26. iThing Demo Captures

- TTEO: rule-based autonomous home appliance control
  - TTEO (Things Talk to Each Other) is a smartphone application with which users can set rules for home appliances that enable them to work cooperatively, performing daily tasks autonomously without human intervention, as shown in Figure 27.
  - For watching the demo video, https://youtu.be/9Veka6C2FrE

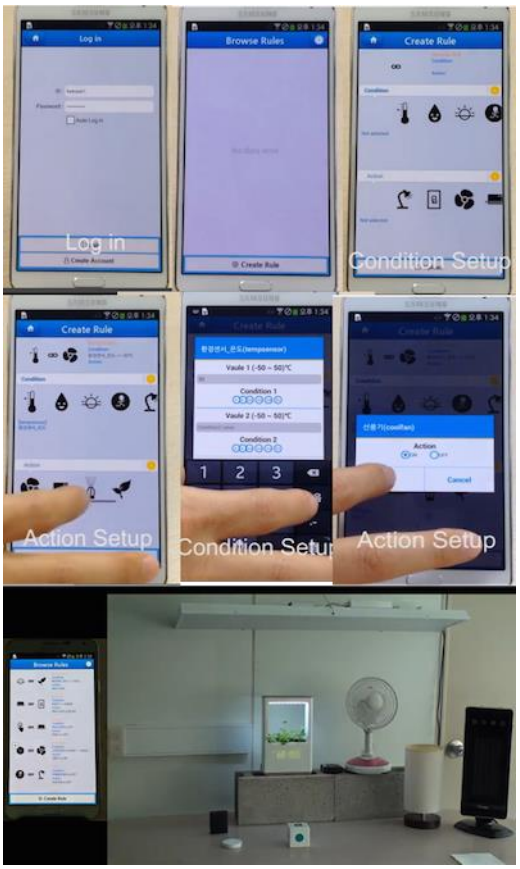


Figure 27. TTEO Pilot Service Demo Captures

The GUI visualization of the connected IoT devices and data resources is not available in current version of the Mobius platform.

3.5 Com4Innov Testbed

3.5.1 Technical architecture overview

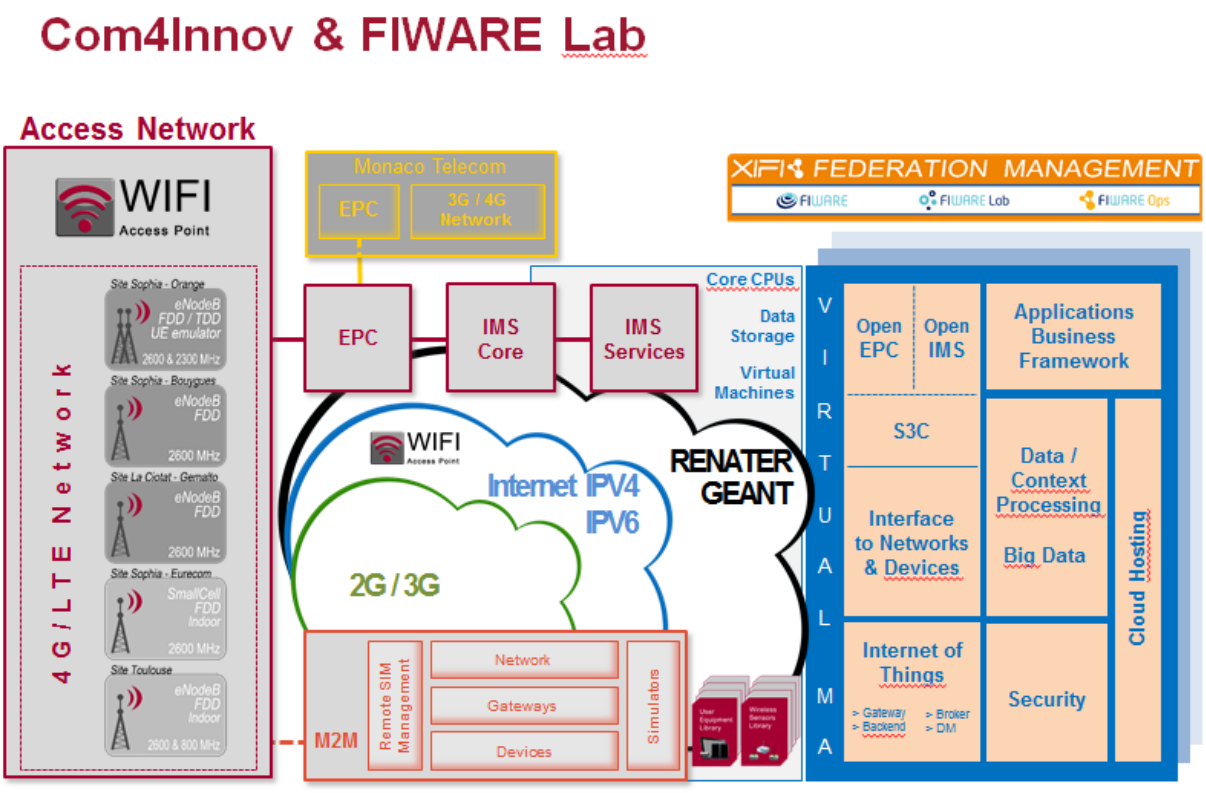


Figure 28. Com4Innov & Fiware Lab

Based on the infrastructure and the services of Com4Innov (http://www.com4innov.com), the Telecom Platform Association operates and provides infrastructure, means and unique skills to market players (SMEs, large enterprise or industrial groups, researchers). Com4Innov is a unique shared environment providing the market with the means to develop and validate their technologies, applications or services in the domains of networks and mobile services of the future (4G/LTE/IMS), M2M (Machine to Machine) and communicating objects / smart devices for the future IoT. These domains are complemented with computing and data hosting in place for our users. Com4Innov is a real full-scale laboratory, ready to use environment providing advanced wireless technology and services, prior to their market availability.
FIESTA-IoT project is an exciting opportunity to drive Com4Innov through the 5th Generation of cellular communications. Interconnecting 4G/LTE network with IoT infrastructure and services leads to a creation and exploitation of big amounts of data and information over a cellular network. At the Com4Innov datacenter, the servers can host several types of services, like Mobius from KETI testbed (http://iotmobius.com/IoTPortal/main/changeLocale?locale=en) and OpenIoT.

An overview of the main elements of the testbed follows.

3.5.2 Physical Resources

Technologies and Networks: (a) 4G/LTE Core and access network with 3 eNodeB, (b) 2 antennas sites in Sophia Antipolis , one in La Ciotat – Marseille), (c) IPv4/IPv6 support, (d) Roaming between Com4innov 4G and Monaco Telecom 4G network and access to 4G/3G Monaco Telecom network, (e) Backhaul network using High speed Hertzian Relays and VPN on Optical Fiber, (f) End-to-End traffic optimization, (g) M2M gateways library for short range network including 6LowPAN, ZigBee, Newsteo, WmBus and GPRS, 3G, UMTS, 4G mobile network, (h) Decentralized architecture for M2M sensors network in an LTE environment, (i) Measurement Methods & Tools.

Applications and Services: (a) IMS Core and Applications servers: Multi Media Telephony (e.g. supporting VoLTE), Presence Management, Instant Messaging, (b) Video content server, Rich Communications Services enabled, (c) Access to the European Platform PlanetLab, (d) M2M cloud for federation and data management between sensors-gateways-craft applications, and for gateways and objects management, (e) ETSI M2M services gateway, (f) M2M pre-deployment test tools with remote SIM Cards solution, data traffic simulation at gateway level and sensor radio traffic simulation.

Terminals: (a) Library of 4G/LTE terminals and prototypes, (b) SIM Cards to access the 4G/LTE/IMS services, (c) Library of ready to use M2M sensors, (d) Polyvalent M2M sensor for prototyping, (e) SIM Cards for M2M tests on real network.

4G/IMS Testbed

Radio Network FDD / TDD – Macro and Smallcells
IPV4 & IPV6 supported
Test drive area with 2 eNodeB in the area of Sophia Antipolis
4G/IMS roaming with Monaco Telecom Network
IMS / RCS 5.1 (VoLTE/IR92, Video/IR94)
Support Emergency call
WIFI Calling
SMS over IMS
User Equipment Simulation on RAN network (up to 500 to 1 500 UE)
Monitoring / Traces
Servers and Virtual Machine with connectivity to 4G equipment
Traces, Logs
Probes
OSS
4G User Equipment Library (phone, tablets, USB dongles): Around 30 devices

Overview of the 4G/IMS installation:

Figure 29. 4G/IMS installation

Figure 30. 4G/IMS installation (2)

Sensors & Gateways:
Around 60 sensors are available to build M2M test case or proof of concept

Wireless Sensors library examples
- Inside or outside Temperature
- Humidity & Temperature
- High Temperature (Thermocouple)
- Low Temperature PT1000
- Thermal flow
- Sun light measurement
- Rain measurement
- Wind measurement
- Fissure / cracks measurement in concrete
- Concrete temperature sensor
- Strain Gauge
- Gas CO2 Detection
- Noise Pollution detection
- Remote meters

Gateways
- Gateways / Collectors 2G/ 3G / 4G / Internet

Data Traffic simulation
- Gateway level
- Sensor level

3.5.3 Federation

Com4Innov is one of the 17 FIWARE lab nodes that provide the infrastructure to test or develop FIWARE applications or to deploy Generic Enablers.

Infrastructure:

The generic FIWARE infrastructure	
Quantity	**Description**
1	Dell Poweredge R820 32 CPU Intel XEON E5-4620 at 2,2 GHz 64 GB of RAM, 1,6 TB RAID 5 ESXi 5,5 > Virtual Machine ITBox 1.3.4.1 2 cores / 2 GB RAM / 120 GB HDD
1	Server Controller : Dell Poweredge R820 32 CPU Intel XEON E5-4620 at 2,2 GHz 64 GB of RAM, 1,6 TB RAID 5, 4x1GB Nic
1	Cinder & Compute server: Dell Poweredge R820 32 CPU Intel XEON E5-4620 at 2,2 GHz 64 GB of RAM, 1,6 TB RAID 5, 4x1GB Nic Volume Cinder: 1 GB Volume Compute: 600 GB
1	Cinder & Compute server: Dell Poweredge R820 32 CPU Intel XEON

	E5-4620 at 2,2 GHz 64 GB of RAM, 1,6 TB RAID 5, 4x1GB Nic Volume Compute: 600 GB
1	Switch Pica8 Pronto 3290 48 ports overflow

Here is an example of the set of IoT related GEs that are part of the FIWARE catalog and can be deployed on the Com4Innov XIFI Node. More Generic Enablers can be found on the FIWARE catalog http://catalogue.fiware.org/enablers.

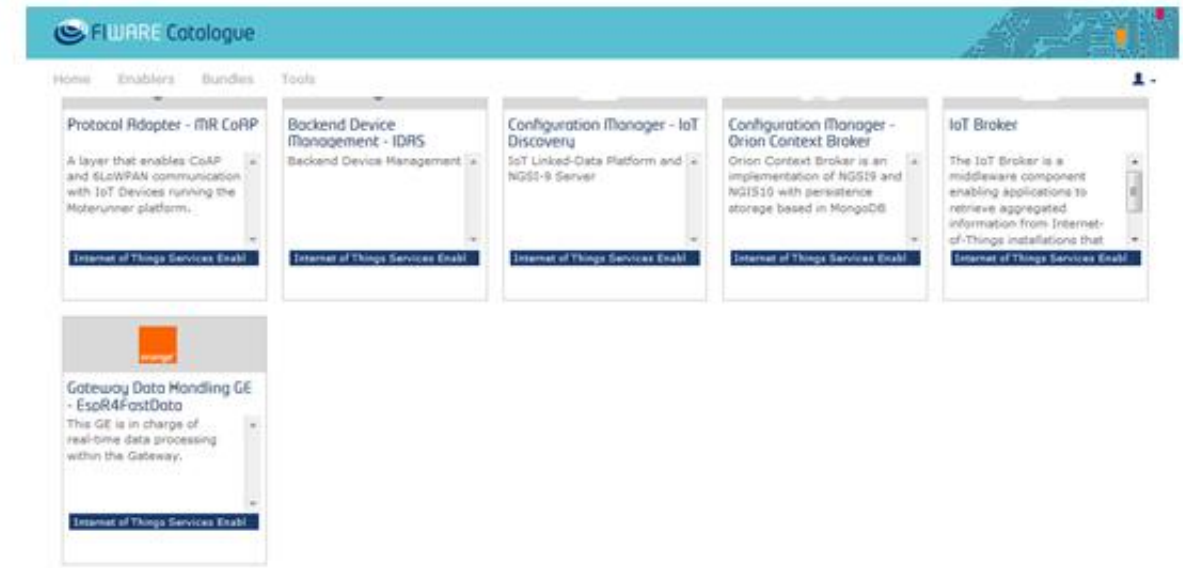


Figure 31. IoT-related Generic Enablers

3.5.4 Internal management methods and APIs

As Com4Innov is one of the 17 FIWARE nodes it makes use of the FIWARE NGSI Specification. The FIWARE specification is a modification of the OMA NGSI specification which aims to maximize the role of NGSI for massive data collection from the IoT world, where a myriad of IoT resources are providing context elements occurrence/updates involving low level protocols. In other words FIWARE NGSI is mainly the binding over OMA NGSI, however some small differences have been implemented in FIWARE NGSI with respect to OMA specification.

OMA NGSI (Next Generation Service Interface) Operations are grouped into two major interfaces:

- NGSI-10
  - updateContext
  - queryContext
  - subscribeContext / unsubscribeContext / updateContextSubscription
  - notifyContext

- NGSI-9

- registerContext
- discoverContextAvailability
- subscribeContextAvailability / unsubscribeContextAvailability / updateContextAvailabilitySubscription
- notifyContextAvailability

Com4Innov has several means offered to manage the environment:

- Capabilities to register new object
- Capabilities to trace and monitor traffic messages at different level of the network
- Capabilities to simulate traffic at different level (gateway level and sensor level)

3.5.5 Data format

Com4Innov infrastructure is free of data format. It rather transports data and information between the core network and the devices and between the devices themselves. This means that it can host multiple and different types of data.
As Com4Innov is a FIWARE collaborator, it makes use of the NGSI API, which among others has the following data structures (in the Context Management part – show in Figure):

- ContextElement structure: Context in FIWARE is in turn represented through context elements. A context element extends the concept of data element by associating an EntityId and EntityType to it, uniquely identifying the entity (which in turn may map to a group of entities) in the FIWARE system to which the context element information refers. In addition, there may be some attributes as well as meta-data associated to attributes that we may define as mandatory for context elements as compared to data elements.

Element name	**Element type**	**Optional**	**Description**
EntityId	EntityId	No	Identifies the Context Entity for which the Context Information is provided.
AttributeDomainName	xsd:string	Yes	Name of the attribute domain that logically groups together set of Context Information attributes. Examples of attribute domain are: device info (battery level, screen size, ...), location info (position, civil address, ...).
ContextAttribute	ContextAttribute [0...unbounded]	Yes	List of Context Information attributes. Note: In case of the attributeDomainName is specified all contextAttribute have to belong to the

			same attributeDomainName.
DomainMetadata	ContextMetadata [0..unbounded]	Yes	Metadata common to all attributes of the logical domain (related to the AttributeDomain)

- ContextAttribute structure

Element name	**Element type**	**Optional**	**Description**
Name	xsd:string	No	Name of the Context Information attribute
Type	xsd:anyURI	Yes	Indicates the type of the value field
ContextValue	xsd:any	No	The actual value of the Context Information attribute
Metadata	ContextMetadata [0..unbounded]	Yes	Metadata about the Context Information attribute

- ContextMetadata structure

Element name	**Element type**	**Optional**	**Description**
Name (Timestamp, Expires, Source, ID)	xsd:string	No	Name of the metadata
Type	xsd:anyURI	Yes	Indicates the type of the value field
Value	xsd:any	No	The actual value of the metadata

- ContextRegistration structure: This structure is used in the ContextRegistrationList parameter of registerContext operation and the ContextRegistration field of the ContextRegistrationResponse structure. This structure can be used either to register/update the information about ProvidingApplication or to register/update the availability of ContextEntities and their related attributes. If the ContextRegistration structure is used to update the registered information about the ContextEntities, EntityIdList SHALL be present together with ContextRegistrationAttribute parameter. If the ContextRegistration structure is used to register the metadata about ProvidingApplication, RegistrationMetadata SHALL be present. The registration/update of ContextEntities and ProvidingApplication can be done at the same time**.**

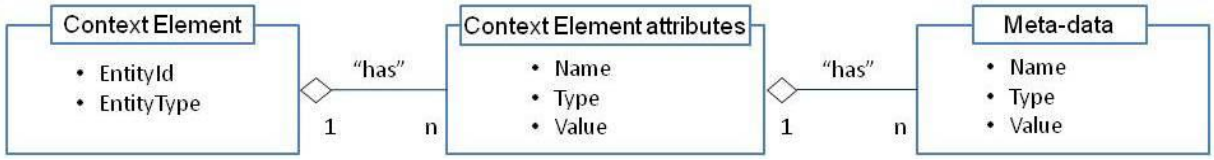


Figure 32. Context Element Structure Model

3.5.6 Security

4G/IMS infrastructure uses SIM Card and strong LTE Authentication and Key Agreement (AKA) for subscriber authentication. Also, it uses Digest Authentication. In order for an external user to use the platform and the services that are provided he can access them only by downloading a security key and by that way he will be authorized and authenticated.

3.5.7 Applications and GUIs

Currently Com4Innov does not provide generic IoT application but they can be smoothly added to the network and services infrastructure. Nevertheless some applications from Com4Innov partners in IoT can be used e.g. Newsteo, Web Monitoring.

4 THE IOT ARCHITECTURAL REFERENCE MODEL

4.1 Introduction to the methodology

A primary decision of the Fiesta project is to follow as much as possible the IoT *Architectural Reference Model* (ARM) as defined in the IoT-A project (the lighthouse FP7 European Project on IoT Architecture) (IoT-A, 2013). Following the IoT ARM, in particular it means adopting and following as closely as possible the overall methodology that defines the steps which lead to the actual architecture and to stick to the Reference Model as defined in the ARM (especially Domain Model and Information Model which are considered as fixed). The architecting process will then consist of (in order):

- Requirement collection;
- Requirement processing (called requirement mapping);
- Elaboration of the Physical-Entity and Context Views;
- Elaboration of the Functional View;
- Elaboration of the Information View;

Before going into the detail of each individual step, we recall hereafter what the ARM (natively) is made of.

The IoT ARM consists of three inter-related parts, which are the IoT Reference Model (RM), the IoT Reference Architecture (RA) and a comprehensive Guidance Chapter. Those aware of the Rozansky and Woods' work on software architecture activity will feel comfortable reading and applying the ARM: indeed the ARM borrows the main principles introduced by Rozansky and Woods (Rozanski&Woods) i.e. Views, Viewpoints and Perspectives and tailors them to the IoT Domain. The constituents of those three parts are detailed further below:

1. The IoT *Reference Model* (RM): Consists of a set models **which the architect must comply to**:
    - Domain Model: the *Domain Model* (DM) (Haller et al., 2013) (IoT-A, 2013) defines the main concepts of IoT (e.g. Physical, Virtual and Augmented Entities, Devices, Resources and Services); excluding general and basic concepts used in the IT world and also defines relationships between those concepts. It consists of an UML class diagram and can be considered as a semi-formal definition of what IoT is (from the IoT-A point of view at least);
    - Information Model: the *Information Model* (IM) provides the basic structure of VE, VE attributes, meta-data and the different levels of descriptions (Service, device and resources) and finally association between VEs and devices. The information model (which is a meta-model) does not make any assumption on how the various information is encoded and stored/managed, and consequently many complying options / implementations are possible;
    - Functional Model: the *Functional Model* (FM) proposes a decomposition of functions an IoT system should implement into 7+2 Functional Groups. The

objective of the requirement mapping activity (see Section 4.2 below) is to identify Functional Components and to assign those components to Functional Groups that match their purpose and nature. It is worth noting that the IoT ARM Functional View proposes a set of functional components (so-called Functional Decomposition) that result from the requirement mapping of the IoT-A Unified Requirements (IoT-A UNIs);

- Communication Model: The IoT Communication Model aims at defining the main communication paradigms for connecting elements, as defined in the IoT Domain Model. It provides a reference set of communication rules to build interoperable stacks, together with insights about the main interactions among the elements of the IoT Domain Model;

- Security, Trust and Privacy (STP) Model: that defines rules and ways to mitigate them (e.g. anonymization for privacy) for the three domains of Security, Trust and Privacy;

2. The IoT *Reference Architecture* (RA): consists of a set of Views and Perspectives as defined by Rozansky and Woods (Rozanski&Woods, 2011)

    - Functional View (FV): The Functional View provides a Functional Decomposition of the IoT System Functionalities in term of Functional Components (main viewpoint) that are assigned to the Functional Groups already identified in the Functional Model. Section 4.4.2 gives more detail about the FV and in particular introduces the “native” FV from IoT-A. Section 6 shows individual FV per test-bed. Those individual FVs will be used as starting point when designing the FIESTA-IoT functional architecture. The FIESTA-IoT Functional View will be elucidated in Deliverable D2.4;

    - Information View (IV): the Information View provides information about the information (in the broad sense) as it is handled in the system. Several Viewpoints focus on storage, information flows between components (which will be later on referred as System Use cases in D2.4), sequence diagrams, information about storage and finally ontologies. There is no Information View in this document, but the test-bed descriptions do provide pieces of information that will be used in order to build the FIESTA-IoT IV. Section 4.4.3 gives more information about the viewpoints used when building the IV. The FIESTA-IoT Information View will be elaborated in Deliverable D2.4;

    - Deployment and Operation View (DOV): So far we have introduced different models and Views which are rather abstract in the sense they are decoupled from any implementation / realisation concerns. When it comes to realizing an IoT System operating in the Real-World it is necessary to understand and show how and where the different parts of the system, i.e., IoT devices, Resources, Virtual Entities and all kinds of Services (IoT- or not) which were, until now talked about in a rather abstract way will be deployed and how they will communicate; this is the exact aim of the DOV. Part of the Deployment view concerns also non-IoT aspects of the system like servers, micro-controllers, gateways, cloud, databases etc…

3. Guidance: that defines the overall process used to derive a concrete architecture out of the ARM. The process part defines all elementary steps, the order in which they should be addressed and how they impact one another. It also provides, per perspective, a large (but obviously non-exhaustive) set of tactics and associated design choices, which can be used in order to address non-functional requirements.

In the following sub-sections we provide more detail about the requirement process and then the different models (RM section) and Views (RA section) we are considering in FIESTA.

4.2 Requirement Engineering

The Fiesta platform is not designed from scratch. On the contrary it is expected to provide a single point of entry to a constellation of pan-European/Asian (to start with) testbeds so that running Fiesta experiments becomes testbed agnostic. It is envisaged that the collection of requirements goes beyond typical functional requirements that would apply to a single testbed. Indeed many requirements will define the way FIESTA-IoT is expected from the experimenters' point of view to handle the federation of those multiple existing underlying heterogeneous testbed platforms in order to perform testbed agnostic experiments.

The main objectives from this first phase are:

1. Identification of the main functionalities which need to be provided by the FIESTA platform. These include:
    - New core functionalities which are possibly not part of any of the underlying test-beds e.g. orchestration engine.
    - Federation functionalities that eventually strongly rely on a set of testbed specific and proprietary functionalities. Typical example would be a federated resource registry.
2. To understand which are the existing functionalities (possibly redundant) already available and to map them to the IoT-A Functional Groups as defined in the IoT Functional Model.
3. To understand which similar functionalities need to be federated and which ones need to be elected a representative instance
4. To determine to what extent existing test-beds need to be adapted and modified according to the requirements.

The requirement engineering phase consists of 3 phases (Volere):

- Requirement collection: Functional and non-functional requirements are expressed and collected via a VOLERE (http://www.volere.co.uk/) template (excel sheet) that consists mainly of two parts:
    - The left most part of it relates to the description of requirements: identifier (unique), priority (must, should and could), type (functional /

non-functional), description (what?), rationale (why?), fit criteria (cross check) and comment

- The right most part of it relates to the requirement mapping activity (see below)

- Requirement mapping: the requirement mapping activity takes place right after the requirement collection phase and it consists of going into the requirements one by one and trying and identifying which aspects of the architecture they do impact. Depending on the type of the requirements there are two main possibilities:
  - Functional requirements: there is a need to identify which of the Functional and/or Information view is impacted by the requirement. In case the functional view is impacted it is necessary to identify which one of the Functional Group is concerned with the requirements and which Functional Component. If none of the pre-identified IoT-A Functional Component is fulfilling the requirement, a new component may be created.
  - Non-Functional Requirement: we have to identify which perspective is impacted. Later on, we will identify which tactic needs to be applied, and on which view.
- Cross-check: this last activity of requirement engineering consists of checking if all requirements were eventually handled properly. The system being designed it will consist of referring to the associated fit-criterion and checking it.

4.3 The IoT Reference Model

In this section we go in a bit more detail about the different models briefly introduced earlier that make the RM. We will focus mainly on the Domain, Information and Functional models.

4.3.1 Domain Model

The purpose of the IoT Domain Model (Haller et al., 2013) (IoT-A, 2013) as proposed by IoT-A is to introduce the concepts pertaining to the IoT Domain and the different relationships between those concepts.

Physical Entities (P-Es): Physical Entities are the objects from the real world that can be sensed and measured and they are virtualized in cyber-space using Virtual Entities. Examples of P-Es are for the SmartSantander testbed buses and traffic lights and for the Surrey testbed offices in the ICS building.

Virtual Entities (VEs): VEs are at the heart of an IoT system. They are a representation of physical objects in the cyber world as stated above. Properties of the VEs can be populated/instrumented using tags, sensors (incl. tag readers, positioning sensors, temperature sensors etc.) and actuators which are devices. Devices and hosted IoT resources are therefore used for bridging the Cyber and Physical worlds; Sensors do report information about the physical entity while actuators are used to modify the state or properties of the Physical Entity. The IoT

Domain Model makes a distinction between VEs being active (called Active Digital Artefacts) in the sense they are equipped with some “service or business logics” and VEs being passive (called Passive Digital Artefacts); typically a database record describing some P-E properties.

VEs are digital extensions of the Physical-Entities that are used to manage, monitor and control their physical counterparts P-Es. They may have their own goals and be equipped with an internal logic in order to achieve their goals. VEs get perception through accessing sensors readings via IoT Services (Resource-centric IoT Service) and can impact their environment or undertake physical actions using actuators (triggered via other IoT Services). Finally VEs may interact with other VEs for various purposes like:

- collaboration (sharing a common goal with other VEs);
- cooperation (getting help from other VEs in order to achieve their own objective);
- giving access to VEs properties/attribute (via access to VE-centric IoT Service);
- giving access to raw data (via access to r-Centric IoT Service);
- offering actuation service (via VE-Centric or r-Centric IoT Service).

A P-E can be associated with more than one VE, where for instance each VE would focus on a specific facet of the P-E (for a given Bus-P-E we could have one associated VE_{mngt} (Passive Digital Artefact – as mainly a database entry) focusing on management aspects while another associated VE_{op} (Active Digital Artefact) would be more focused on real-time properties of the bus during operation. In any case, a VE cannot be associated with more than one P-E.

VEs offer service interfaces to other VEs or higher-level services (like orchestration engine for instance).

From the Domain Model point of view, VEs interact with IoT Services, which expose using a standard protocol (e.g. REST), the functionality provided by Resources (which often rely on proprietary protocol stacks). Often Resources are running on IoT Devices. A look-up service can be used for a VE to identify which IoT Service it may access. And as seen earlier, interfacing with a VE is possible through IoT Services.

VEs –as software components- do not have to “run” at the P-E side. Discussing where the VE is or should be deployed is a deployment issue that can be made explicit within the Deployment View. Pragmatically and depending on the Devices supporting the P-E, a VE could be part of a micro-controller that also manage Sensing and actuating, could be partially deployed between a gateway and a micro-controller (smarter functionalities being part of the gateway while the IoT Resources would be hosted within the micro-controller), or hosted in the cloud; many options are possible depending on the nature of the P-E, the devices attached to it and the complexity of services it offers through its VE(s).

Augmented Entity (AE): Even if probably not directly used in FIESTA, it is worth mentioning that the IoT-A Domain Model introduces the concept of Augmented Entity (AE). The AU is the composition of a VE and the P-E it is representing in order to highlight the fact that the two entities belongs together and make a whole entity.

Augmented Entities are the “things” in the term “Internet of Things” as an AE represents both the physical and digital aspect of the thing.

IoT Devices: In FIESTA, IoT Devices are the hardware supporting the sensing and actuation functions. Micro-controllers, batteries, ROM memory etc. are Devices (without the IoT prefix).

IoT Resources: are the software embedded in IoT Devices that provides the raw readings (for sensors) and actuations. The IoT Domain Model advises not accessing directly resources, but on the contrary to access corresponding Resource-centric IoT Services (see below).

IoT Services: We can consider different kinds of IoT services depending on their level of abstraction:

- Resource-centric IoT services (r-IoT Service) are exposing the IoT Resources using standardized interfaces and possibly adding meta-data to the raw reading available at the resource level; They all connect to a sole resource (sensor or actuator);
- VE-centric IoT Services (ve-IoT Service) are associated to the VEs and are used for accessing VEs attributes/status or to access VE-level services not directly connected to VEs attribute or situation; In the Functional View the VE Service FC deals with such accesses;
- Integrated IoT Service (i-IoT Service) are combinations of the two above when combining different readings from different sensors (e.g. “secured” room can depend on lock/unlock status, presence indicators and light status).

IoT Services are associated with Service descriptions that can be used to discover particular Sensing/Actuation capabilities.

VE properties: All of the above mentioned service types are wrapped under the VE property concept. This provides a unified way of describing, retrieving and accessing these services. VE properties allow a richer and uniform semantic description despite the differences between these services. This is guaranteed by the mandatory attributes required to be filled in during the services’ endpoints description.

Services: Services (without IoT prefix) are associated to VEs but do not relate to specific Resources as illustrated in the example above. Services are not part of the IoT Domain Model but could be added to the global picture for the sake of clarity.

User: A user of the FIESTA-IoT platform will mainly interact through IoT Services and Services. A user can be a human person or a Digital Artefact (VE, Application, and Service). It is worth noting that the services offered by the platform itself are not represented in the Domain Model as they are not constrained to the IoT Domain, but are general enablers.

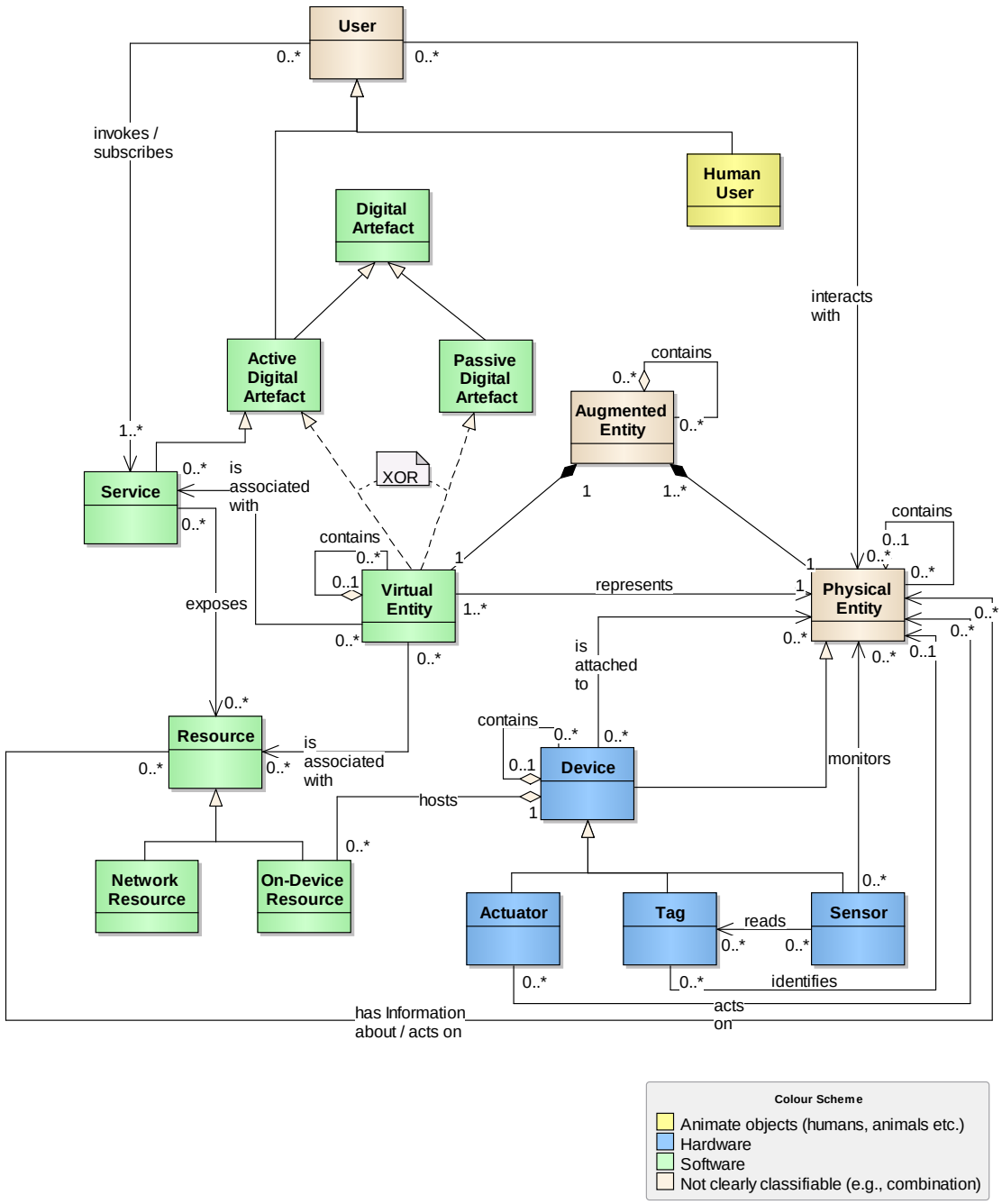


Figure 33. IoT Domain Model

4.3.2 Information Model

The IM focuses on the description of the structure of Virtual Entities as a representation in the cyber space of physical entities. The representation of the information (either it is encoded in XML, RDF, binary or any other) is kept away from the Information Model and left to the architect's choice.

The central part of the IM consists of the structure of the VE which is modelled using a set of attributes and which are associated (via the association relationship) to Service Description. These associations is essential in the IoT IM; it makes the binding between an attribute –which is at the VE (or Object) level- and a

corresponding Resource-centric IoT Service (meaning a service exposing a resource), which, as the name suggests, is at the resource level. This association must be managed in a dynamic way so that the binding between an object attribute and a resource (via a service) can vary along the time axis.

The Attribute is the aggregation of one to many Value containers. Each of those containers contains one single value and one to many metadata (e.g. time stamp, location, accuracy …).

VE are described through Service Description where each Service would be characterised e.g. by its interface or any useful information that a lookup service can exploit (e.g. the FIESTA-IoT Meta-directory).

As IoT Service is exposing Resources which are themselves hosted by Devices, the IM authorises Service Description to contain 0 to many Resource Description(s) and Resource Description to contain 0 to many device Description(s). The structure of Descriptions is not constrained by the IM and therefore left to the Architect's own choice.

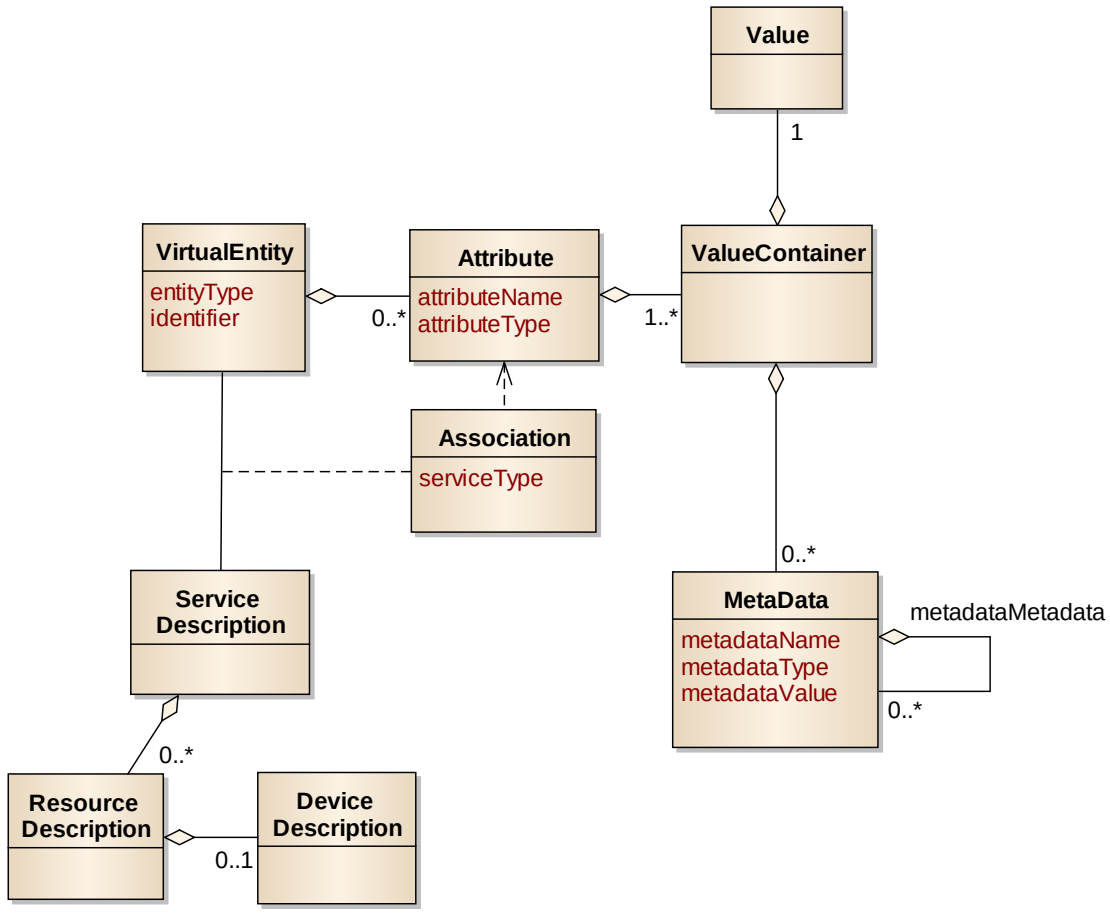


Figure 34. IoT Information Model

4.3.3 Functional Model

The Functional Model (see Figure 35 below) proposed by IoT-A corresponds to a service-oriented approach of IoT. It identifies 7 main Functional Groups and 2 additional ones that are kept outside the scope of the ARM.

The FM also identifies where inter-FG interactions should take place (yellow arrows). The Functional Groups are defined as follows (IoT-A, 2013):

- ***IoT Process Management FG:*** The purpose of the FG is to allow the integration of process management systems with the IoT platform. For example, the formal definition of an PAN-testbed experiment (as a process) would fall into this category;
- ***Service Organisation FG:*** This FG is responsible for composing and orchestrating services, acting as a communication hub between other FGs. The execution of an experiment described within the Process Management FG would take place in this FG, like any other kind of choreography/orchestration engines. A central message bus offering publish/subscribe functionalities would also be part of this FG;
- ***Virtual Entity FG:*** This FG relates to VEs as defined in the IoT Domain model, and contains functionalities such as discovering VEs and their associations with Resource-centric IoT-services. This FG also allows access to the VE-centric IoT Service offered (formally “associated with”) by a Virtual Entity;
- ***IoT Service FG:*** The IoT Service FG contains functions relating to Resource-centric Services. Those services expose the resources like sensors and actuators and provide the means for reading sensor values or actuating. It also contains storage capability functionality. More specifically the IoT ARM states that “A particular type of IoT Service can be the Resource history storage that provides storage capabilities for the measurements generated by resources”;
- ***Communication FG:*** The Communication FG is used to abstract the communication mechanisms used by the Devices. Communication technologies used between Applications and other FGs is out of scope for this FG as these are considered to be typical Internet technologies;
- ***Security FG:*** The Security transversal FG “is responsible for ensuring the security and privacy of IoT- compliant systems”;
- ***Management FG:*** The Management transversal FG contains components dealing with Configuration, Faults, Reporting, Membership and State. It should be mentioned here that this FG works in tight cooperation with the Security FG.

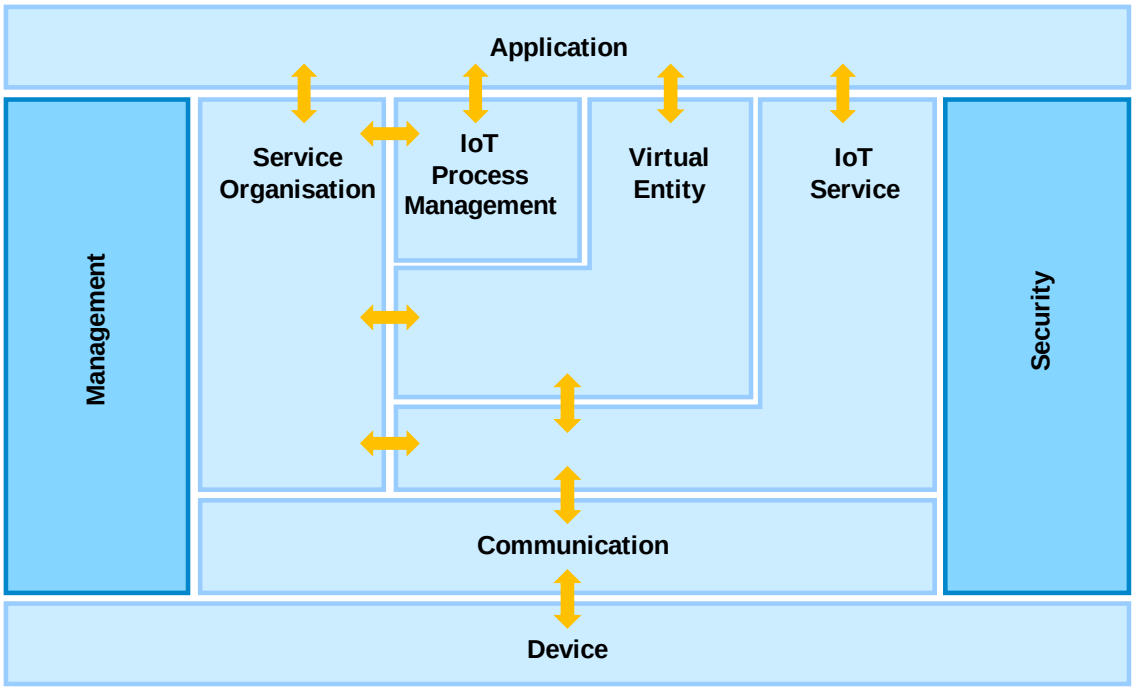


Figure 35. IoT Functional Model

4.4 IoT Reference Architecture

The ARM consists of different views as introduced earlier, i.e. Functional View, Information View and Deployment & Operation View. There are two additional Views which are not *stricto sensu* part of the ARM: Physical-Entity View and Context View; however they are part of the guidance chapter and those views are clearly essential when elaborating the IoT System architecture. Before explaining more about the Functional and Information Views, we remind briefly about the concept of viewpoints, which is essential as for describing views from different angles.

4.4.1 Viewpoints

Rozansky and Woods introduce the concept of viewpoint as follows (Rozanski&Woods, 2011):

> *"A viewpoint is a collection of patterns, templates, and conventions for constructing one type of view. It defines the stakeholders whose concerns are reflected in the viewpoint and the guidelines, principles, and template models for constructing its views."*

In practice, we shall use different notations like UML class diagrams, Message Sequence Charts, interaction diagrams, information flows, ontology diagrams, UML Use-Case notations etc… for describing different aspects of the Functional and Information views. The following picture taken from an IoT ARM introduction slide set illustrates this idea.

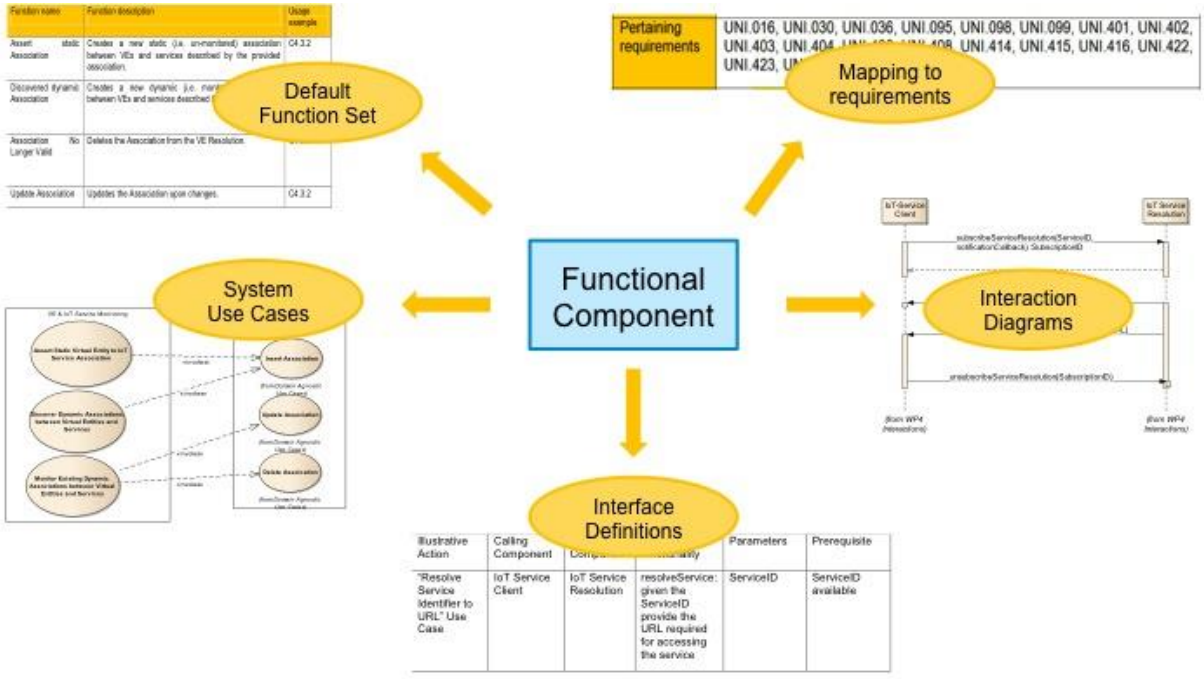


Figure 36. Viewpoint examples

4.4.2 Functional View

In this section we present the IoT-A Functional View as it shows a good example of functional decomposition resulting from a requirement analysis (in that particular case it was the analysis of the IoT-A Unified Requirements). This functional decomposition gives a collection of Functional Components (FC) assigned to the Functional Groups already identified within the Functional Model. We recommend referring to the IoT ARM final architecture deliverable D1.5 (IoT-A, 2013) in order to get more detail about each of those Functional Components. The IoT-A "native" Functional View is summarised in Figure 37 below:

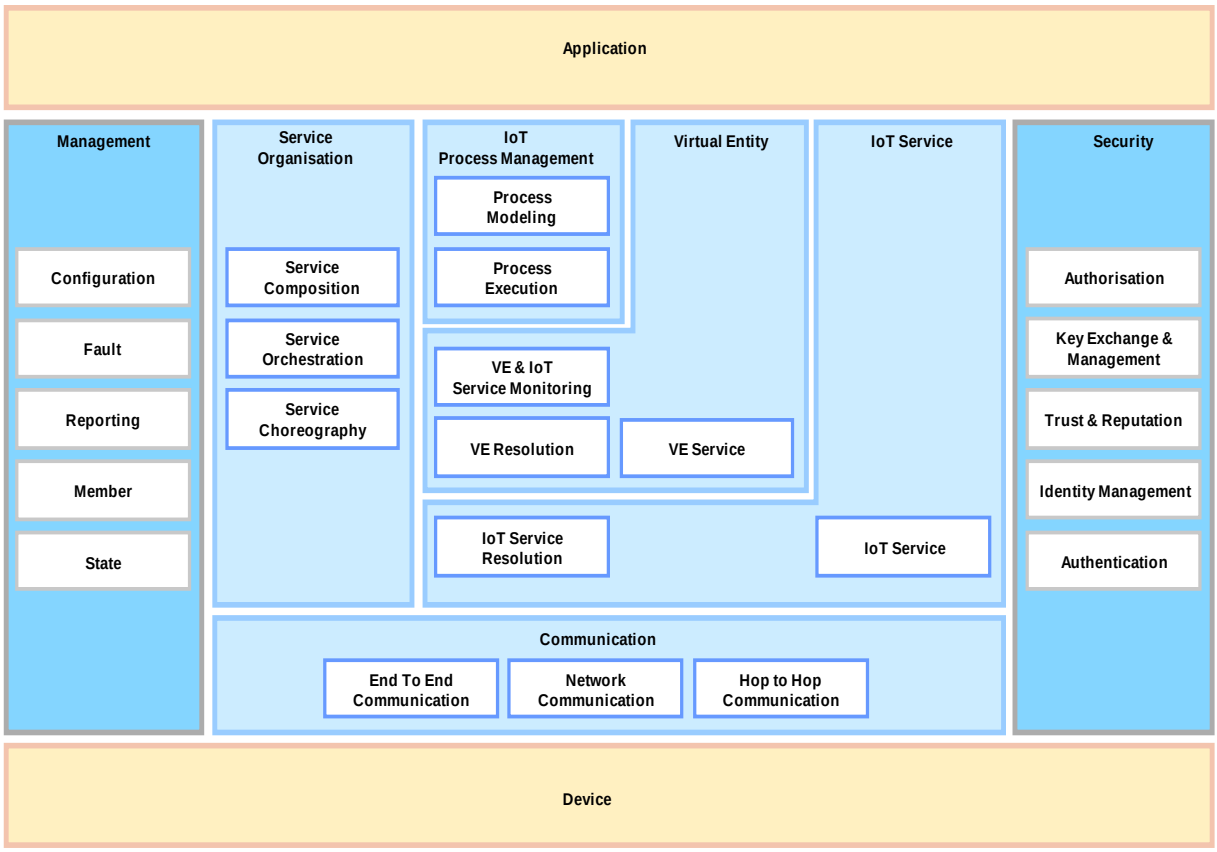


Figure 37. IoT "Native" Functional View

When elaborating the reverse-mapping from various existing FIESTA-IoT test-beds towards the IoT Functional View it is important to try keeping as much as possible to the list of components already identified, however it is perfectly possible and allowed to introduce new ones as well. Whenever a component with a different name –e.g. `newFunct-` covers an existing "native" functionality –e.g. `natFunct-`, it is useful to write something like "`newFunct:natFunct`" within the box.

Along with a description of the FCs within FGs it is equally important to get a concise –and still precise- description of the different FCs implemented in the various testbeds and to understand also very clearly how they interact with and position w.r.t. the other components. At this point in time we stick to a textual description, but in the architecture document System Use-Cases will formally elucidate those inter-component interactions.

4.4.3 Information View

In this section we don't provide the IoT Information View but rather explain what it should consist of, and what preliminary information is awaited from the reverse-mapping exercise as far as the Information View is concerned.

We use various viewpoints for describing the information view, but some of them will be used only for the elaboration of the FIESTA-IoT architecture. The ones which are particularly interesting at this point in time concern ontologies and the way information is formatted and annotated within FIESTA.

As a first approach it is therefore important to describe the ontologies used within the different testbeds and to understand how they are used.

4.4.4 Deriving other views

Other views like the Context View and Physical-Entity Views are of interest for FIESTA. They even ought to be derived before the Functional and Information Views. As we see later on, those views allow for understanding the relation of FIESTA-IoT to its environment, in particular the experimenters. Those two P-E and IoT Context Views will be part of the architecture document D2.4.

4.4.4.1Physical-Entity View

This view gives information about the Physical-Entities of interest for the IoT System and how those P-E are represented within this IoT system:

- List of P-Es and their associated properties that can be observed, in addition to the associated actuations that can be applied to them;
- List of devices that are instrumented in order to bridge the physical properties to the cyber world;
- How the sensors/actuators are associated to the P-Es and their location; if a P-E is in the scope of a camera or if a P-E is attached to a temperature sensor for instance.

4.4.4.2 IoT Context View

The IoT Context View consists of the Context View and instantiated Domain Model.

According to Rozansky and Woods (Rozanski&Woods, 2011) the Context view dependencies, and interactions between the system and its environment (the people, systems, and external entities with which it interacts)". This view focuses on formalizing the boundary between the system and its environment (outside world) and shows how the system interacts with other IT systems, organizations and end-users/administrators etc… using the IoT system.

As said earlier, the IoT Context view also includes an instantiated Domain Model. Of course the instantiated domain must comply with the IoT Domain Model (the "generic" one) but also aim at identifying the P-Es and associated VEs, Resources, Devices, Services and Users as they are instantiated in the concrete architecture.

5 FIESTA-IOT TESTBEDS VS IOT ARM

In this section the totality of the components and the resources that each IoT testbed and platform owns will be translated and mapped in the terms of the IoT ARM naming. This is a very important step that has to be done in the direction of building a common IoT platform, by interconnecting the different IoT platforms and testbeds around the Europe and the entire world.

The importance of this medium step lies to the following: all the different components and resources of the IoT testbeds and platforms participating in the aforementioned interconnection should be translated into a common language; this procedure would ease much more the task of the connectivity and interoperability of the platforms, which is the ultimate goal of the project.

In section 4 there was an extended review of the IoT ARM so in this section only the translation and mapping of the IoT platforms and testbeds in the Architecture Reference Model will be provided.

The following pages contain a table with an extensive classification of the existing components in the IoT platforms and testbeds which is very useful for the further analysis and the translation and mapping of them in the Architecture Reference Model as well as quantitative and qualitative analysis of them through radar graphs.

In the following tables there is a detailed description and analysis of the resources and facilities that each testbed provides. In the first row there are the IoT testbeds and platforms and in the first column we can see a presentation of the resources (software/hardware, protocols, security mechanisms etc) which will be used towards the completion of the goal of this project.

5.1 Summary of IoT Testbeds and Platforms

Table 8. Testbeds' assessment summary

Capabilities	IoT-ARM mapping	Com4Innov (FR)	SmartSantander (SP)	KETI (KOR)	UNIS Surrey (ENG)
General Description of the Platform	**Domain Model**	4G/LTE/IMS core 3 eNBs 3 antennas M2M gateways (zigbee, 6LowPAN, GPRS, 3G, 4G) measurements tools MMTAS RCS	3-level Architecture: 1)IoT part: (i) IoT nodes, (ii) repeaters 2) GW part (UMTS/GPRS/Ethernet connectivity) 3) Platform part (Retrieve/inject data)	Testbed composed of: 1) IoT server platform - Mobius 2) Device platform - &Cube 3) Community ecosystem - OCEAN: offers source code to download complying oneM2M specs	3-levels of the testbed: 1) Server level: hosts the services and functionalities of testbed - experimenter accesses the testbed 2) embedded GW 3) IoT level that is connected to backbone network through the GW level
Physical Resources		4G/IMS: FDD-TDD 30 UEs (smartphones, dongles, tablets) 60 sensors (indoor/outdoor) temperature, humidity, light, CO2, noise GWs	3000 802.15.4 sensors 200 GPRS sensors	180 sensors 10 smart sockets - control power on/off	IoT nodes:Smartphones, 10 Smart displays QR/NFC tags 10 Cloud servers (12 cores/24gb RAM each) GWs (Management/Data)
Sensors/Actuators	**Device**	60 sensors	3000 IEEE 80.15.4 sensors	160 compound sensors 10 CO2 sensors	IoT nodes: TelosB interfaced with energy meter
2G/3G/4G modules/UEs	**Device**	30 4G UEs (smartphones, tablets, dongles)	200 GPRS modules	No	No
QR/NFC/RFID tags	**Device**	No	2000 RFID/QR tags	No	1000 QR-code/NFC stickers

Capabilities	IoT-ARM mapping	Com4Innov (FR)	SmartSantander (SP)	KETI (KOR)	UNIS Surrey (ENG)
Gateways	**Communication Functional Group**	Gateways Collectors 2G/ 3G / 4G / Internet	Technologies for communicating between the GWs and the IoT nodes: DIGImesh ZigBee NFC BT GPRS/UMTS	10 embedded Linux GWs	100 embedded Linux GWs
Type of data provided by Devices (in general)	**Virtual Entity attribute**	Temperature Humidity Thermal flow Sun light measurement Rain measurement Wind measurement Fissure / cracks measurement Concrete temperature sensor Strain Gauge Gas CO2 Detection Noise Pollution detection Remote meters	Noise monitoring Temperature monitoring Luminosity CO2 Air quality Presence Power regulator Road traffic conditions Weather conditions Soil temperature and moisture Participatory sensing etc.	Temperature Humidity Illumination Presence CO2 concentration/detection	Geo-localization services People's distribution in the campus Light Relative noise level Temperature and motion
Data Format	**Virtual Entity attribute** **Metadata**	NGSI	JSON format for data reaching the testbed URN for each resource XML -> JSON for RD Format	XML format	JSON format XML format
Database/Repository	**Passive Digital Artefacts**	No	ResourceDirectory Stores all the resource information within the SmartSantander platform in a database. New data inserted through a REST interface.	Not clear	10 Cloud servers (12 cores/24gb RAM each) 8TB storage VMWare cloud computing platform

Capabilities	IoT-ARM mapping	Com4Innov (FR)	SmartSantander (SP)	KETI (KOR)	UNIS Surrey (ENG)
Communication protocols	**Communication Functional Group**	LTE air interface WiFi Communication protocols from sensors: BT, ZigBee, 6LoWPAN	External communications for the platform and for FI-WARE uses REST	HTTP, MQTT, CoAP oneM2M conformation RESTful architecture	HTTP REST-Easy framework for implementation of REST APIs
Federation		FiWARE LAB - XiFi Federation	Fed4FIRE project	With oneM2M platforms (Fraunhofer FOKUS's openMTC, InterDigital's oneMPower)	SFA-wrapper approach Aggregation Manager wrapps the APIs among the provided testbed APIs Each testbed implementing the SFA-wrapper approach can be federated
Real Time Resources/Monitoring		No	Node Manager scans for active sensors by frequency of measurements	Indoor environmental monitoring Presence sensor monitor - is real time In Mobius - there is sw monitoring module	Yes: Monitor status of GWs and IoT nodes Detect disconnection and failures of GWs and IoT nodes Reset, reprogram IoT nodes Check IoT nodes status
Add new devices/resources		New devices can be added by registering them to the 4G/IMS infrastructure – using the C4I SIM card	Yes - Stores all the resource information within the SmartSantander platform in a database. New data can be inserted through a REST interface.	Can add new devices and resources - required a registering process via REST interfaces.	In the near future yes
APIs to expose existing resources/devices	**IoT Service**	NGSI	1) IoT API REST interface 2) Orion Context Broker GE (FIWARE)	REST interface - provides access to existing data attributes and historical data - oneM2M defined APIs Part of Mobius - to access it you need to be authenticated	WISEBED Reservation APIs - Java application on any webserver

Capabilities	IoT-ARM mapping	Com4Innov (FR)	SmartSantander (SP)	KETI (KOR)	UNIS Surrey (ENG)
APIs to retrieve data from testbed	**IoT Service**	NGSI	1) IoT API REST interface 2) Orion Context Broker GE (FIWARE) 3) Cosmos GE	OneM2M provides subscription interface to access the historical data. oneM2M supports to the data retrieving including application entity (AE), AE container, and AE content instance retrieving	Yes: homemade REST APIs - can be installed in any webserver environment
Authentication/ Authorization/Security	**Security Functional Group**	4G/LTE SIM cards with AKA for subscriber authentication Access granted by Com4Innov	CA (Certification Authority) certificate based on X.509 v3 standard Access provided by UNICAN	User-access key and HTTPS auth.	Shibboleth to access the testbed (Username and password) oAuth to access the data
Applications	**Resources**	Applications can be added in the environment. Some apps from Com4innov partners can be used e.g. Newsteo Web Monitoring.	1)Citizens-UEs-CityCouncil connected in a common platform 2) City tour guide + real time access to city cameras	1) iThing - home appliance control 2) Smart Flowerpot 3)iDrone 4) TTEO (Things Talk to Each Other)	2 smartphone applications that access geo-localized services and information from students in the campus when tagging actions are performed towards the deployed QR-code/NFC stickers: 1) MyGuardian: tag path of people within the campus and share progress with a selected group of friends using the app 2) MyCampus
Platform that the application runs		User Equipment (Smartphones / equivalent or Objects can use the network services) Virtual machines	1) Smartphones 2) Web	1) Smartphones 2) Web	Smartphones
GUI		No	http://maps.smartsantander.eu/	No	MyEcoFootprint webapplication for vizualition energy data

We now describe the translation of the resources of the IoT testbeds and platforms into the IoT ARM:

- **Sensors-Actuators / 2G-3G-4G modules / NFC-QR-RFID** tags are *Devices*: Devices are technical artefacts, i.e. hardware, for bridging the real world of Physical Entities with the digital world of the Internet. Often a Device hosts Resources, which represent the software counterpart. Example: Typical devices are sensors, like temperature, noise or light sensors, but also more complex ones like cameras – or actuators, like switches, door openers or more complex ones like air conditioning systems.
- **APIs** for exposing and retrieving data are *IoT services*: A Service provides a well-defined and standardized interface, offering all necessary functionalities for interacting with Physical Entities and related processes. Often it exposes a functionality provided by a Resource to the overall IoT system.
- **Database / Repositories** are *Passive Digital Artefacts*: They are passive software elements such as database entries that can be digital representations of the Physical Entity.
- **Data format / Types of data** are *Virtual Entity attributes*
- **Authentication/Authorization/Security** is mapped to *Secure/Trust/Privacy Model.*

6 FUNCTIONAL VIEWS (REVERSE-MAPPING PER TESTBED)

In this section we do a mapping of all existing testbed platforms to the IoT Functional Model in order to generate the testbed-centric Functional Views. It is important in this section to also have a list of components which is a concise –and still precise- description so that it becomes clear which FG the FC belongs to.

6.1 SURREY test-bed Functional View

- Resource Registry: Experiments executed at the Surrey testbed allow for accessing different kinds of resources distributed among the different offices of the Surrey ICS building. A database is structured in floors and rooms and provides a REST interface for querying and accessing resources and therefore reading data. The look-up component allows querying based on objects ID's (e.g. to retrieve room ID's on a given floor for instance) or based on resources (e.g. to retrieve the reading of Temperature sensor#xyz). It therefore spans both the Virtual Entity and IoT Service FG's. This component is currently implemented in the test-bed and is not semantic-enabled.

- SenseToWeb: is a web-tool and API that can be used to do registration of VEs, Resources and Services. It is compliant to the IoT-A ontology and is planned to be migrated to IoT-lite. It also offers simple search capabilities and of course enables any SPARQL queries. SenseToWeb will be upgraded in order to support FIESTA-IoT ontologies;

- SAOPY: is a tool that provides ontology-dependant API for annotating data following the ontology. As the API strictly follows the ontology structure, it becomes difficult to then make mistakes while annotating the data. SAOPY will be upgraded in order to support FIESTA-IoT ontologies;

- SSN Validation Service: This service checks if ontology complies with SSN. It will also return the list of un-used classes and properties, in addition to locating possible syntactic errors. SSN Validation Service will be upgraded in order to support FIESTA-IoT ontologies;

- KAT: The Knowledge Acquisition Toolkit is a toolkit that is able to extract and represent human understandable and/or machine interpretable information from raw data. The toolkit includes a collection of algorithms on each step of the acquisition workflow ranging from data and signal pre-processing algorithms such as Frequency Filters, dimensionality reduction techniques such as PCA, Wavelets, FFT, ACPA, SAX, and Feature Extraction and Abstraction and Inference methods such as Clustering, Classification and Logical Reasoning.

The following Figure 38 shows the above-described components within the ARM Functional View template (FM).

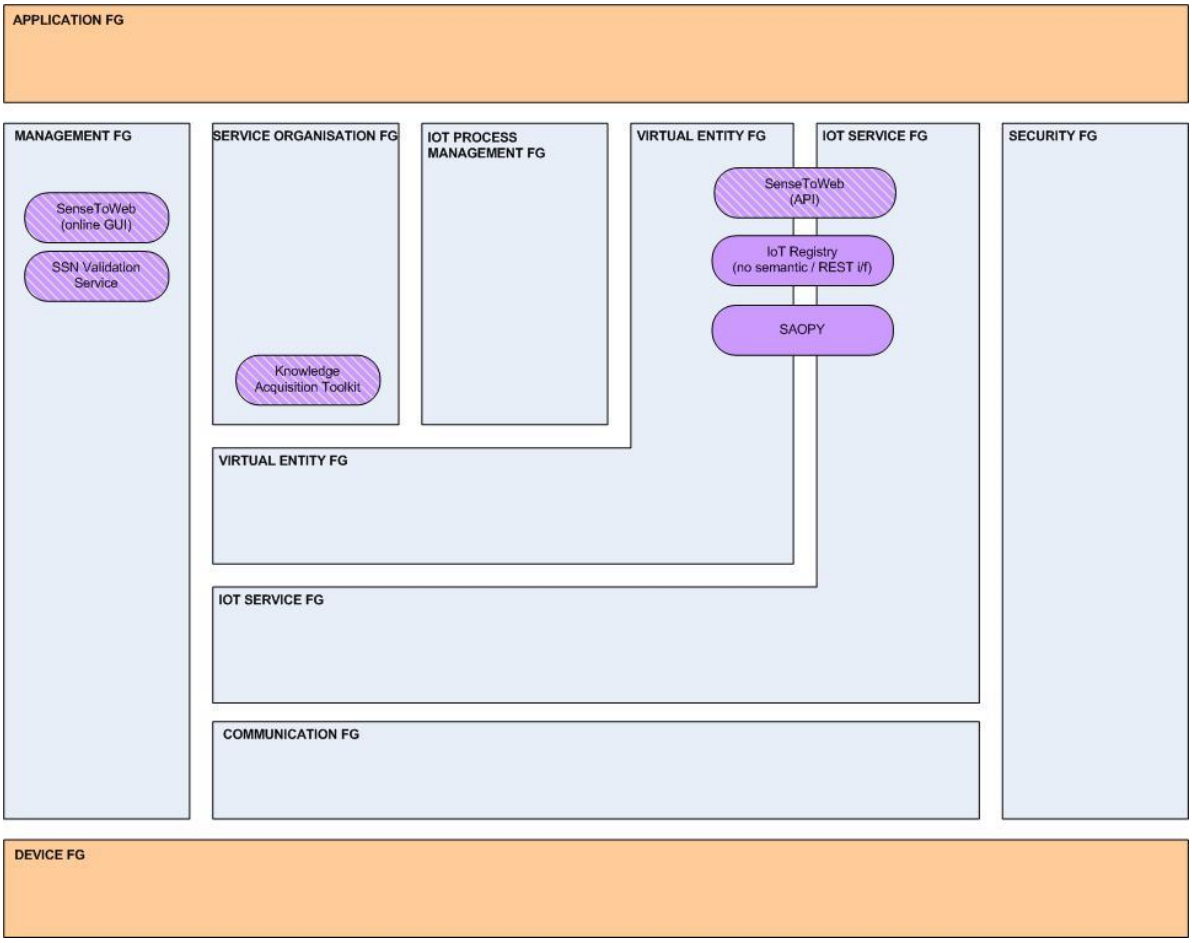


Figure 38. Surrey test-bed FV

6.2 KETI test-bed Functional View

- Common Services Functions: KETI's platforms and test-bed support oneM2M standards-defined CSFs (Common Service Functions)
    - Registration function: The Registration function processes a request from an AE (application entity) or another CSE to register with a Registrar CSE in order to allow the registered entities to use the services offered by the Registrar CSE.
    - Data management and repository function: The Data Management and Repository function is responsible for providing data storage and mediation functions. It includes the capability of collecting data for the purpose of aggregating large amounts of data, converting this data into a specified format, and storing it for analytics and semantic processing.
    - Discovery function: The Discovery function searches information about applications and services as contained in attributes and resources. The result of a discovery request from an Originator depends upon the filter criteria and is subject to access control policy allowed by Service Subscription. An Originator could be an AE or another CSE. The scope of the search could be within one CSE, or in more than one CSE. The discovery results are returned back to the Originator.
    - Subscription and notification function: The Subscription and Notification function provides notifications pertaining to a subscription that tracks

event changes on a resource (e.g. deletion of a resource). A subscription to a resource is initiated by an AE or a CSE, and is granted by the Hosting CSE subject to access control policies. During an active resource subscription, the Hosting CSE sends a notification regarding a notification event to the address(es) where the resource subscriber wants to receive it.

- Device management function: The Device Management function provides management of device capabilities on gateways and devices. Application Entities (AE) can manage the device capabilities on those Nodes by using the services provided by the Device management function alleviating the need for the AE to have knowledge of the management technology specific protocols or data models. While the AE does not require an understanding of the management technology specific protocols or data models, this information is provided to the AE so that an AE can utilize this information for administrative purposes (e.g. diagnostics, troubleshooting).
- Location function: The Location function allows AEs to obtain geographical location information of Nodes for location-based services.
- Group management function: The Group Management function is responsible for handling group related requests. The request is sent to manage a group and its membership as well as for the bulk operations supported by the group. Bulk operations include read, write, subscribe, notify, device management, etc.
- Security function: The Security function is providing management of access control policies which specify which node entity has appropriate privilege for accessing resources in the Mobius platform. As it has been introduced in Section 3.4, User-Access key, short for u-key, is provided to authenticate and authorize the access to resources and services. An authorization mechanism based on Access Control List (ACL) is also provided to guarantee the authorized access to requested resources, according to the Access Control Policies (ACPs) defined as a <accessControlPolicy> resource in oneM2M standard.

- Connectivity Support functions: KETI's platform and test-bed support three protocols, HTTP, MQTT and CoAP which are a widely accepted standard IoT protocols and also support a capability for binding between core protocol and those protocols.

- iotmobius.com: the Mobius web portal provides a web-based interface for registering IoT devices, as well as monitoring and updating devices status.

  - Mobius web portal home

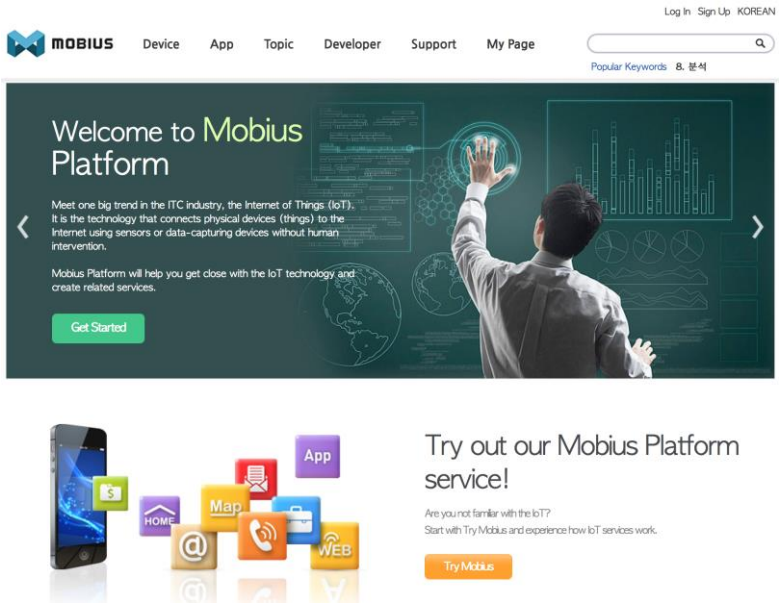
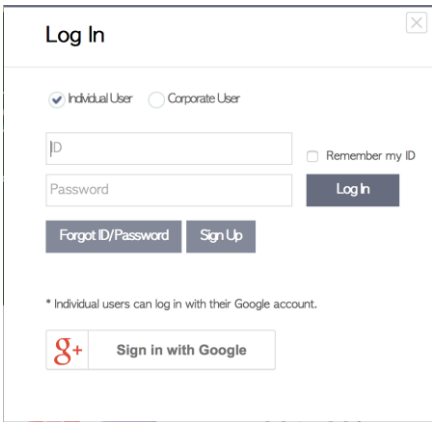
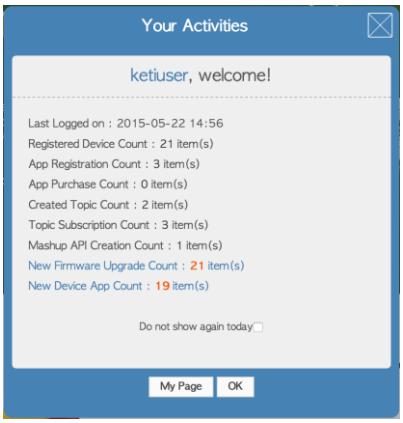


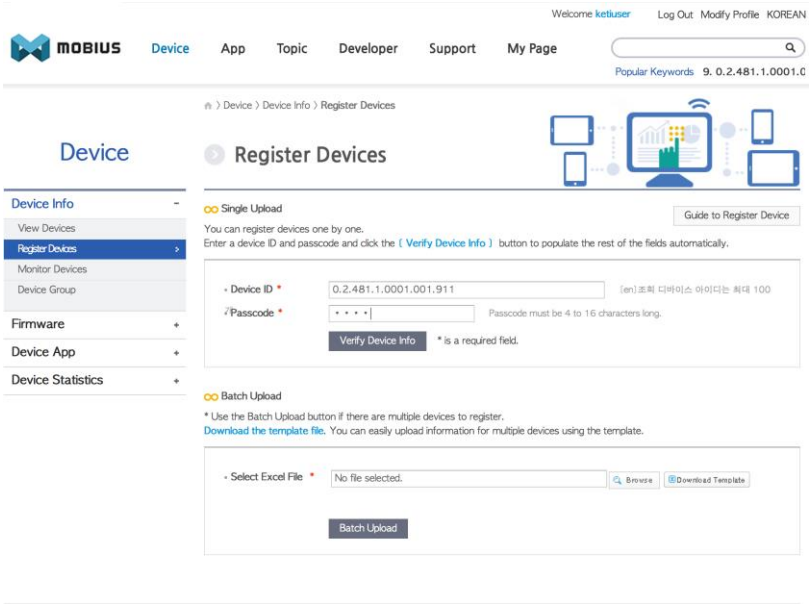
Figure 39. Mobius web home portal

- Mobius login page and user information
  Once logged in, a window pops up showing the user's activities including the last logged time, # of devices, topic subscription, etc.

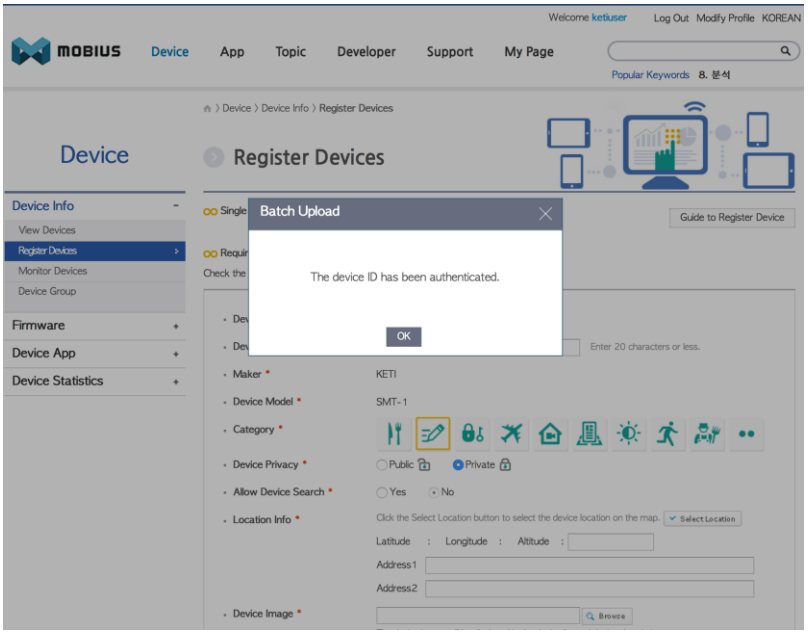




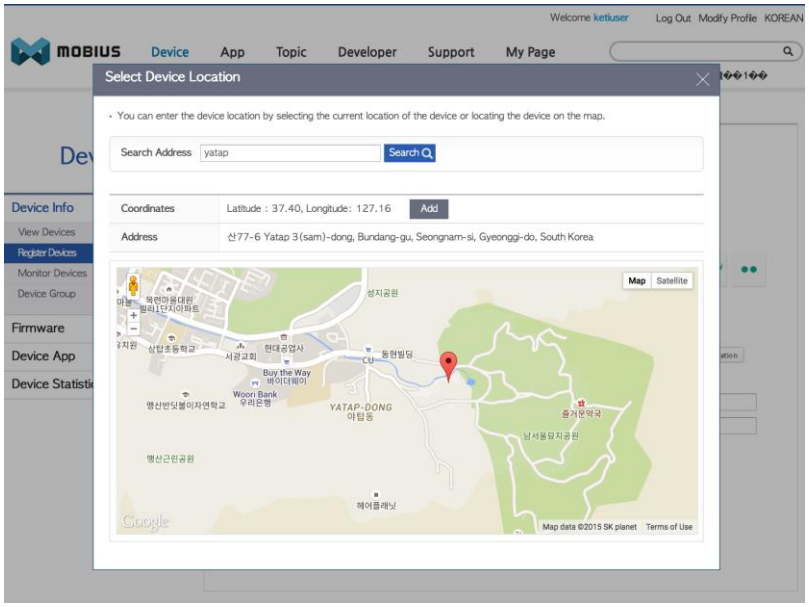
- Device registration page
  1. Device authentication:
     This page performs a verification process that ascertains the IoT device to be registered is a valid one. Assume that every IoT device has its globally unique ID and verification passcode.



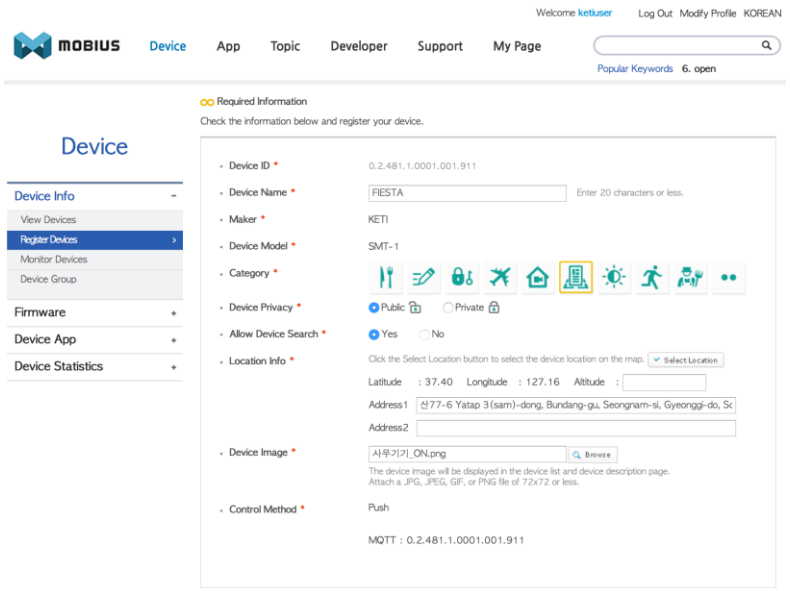
After acknowledging the device ID and passcode, the device is authenticated.



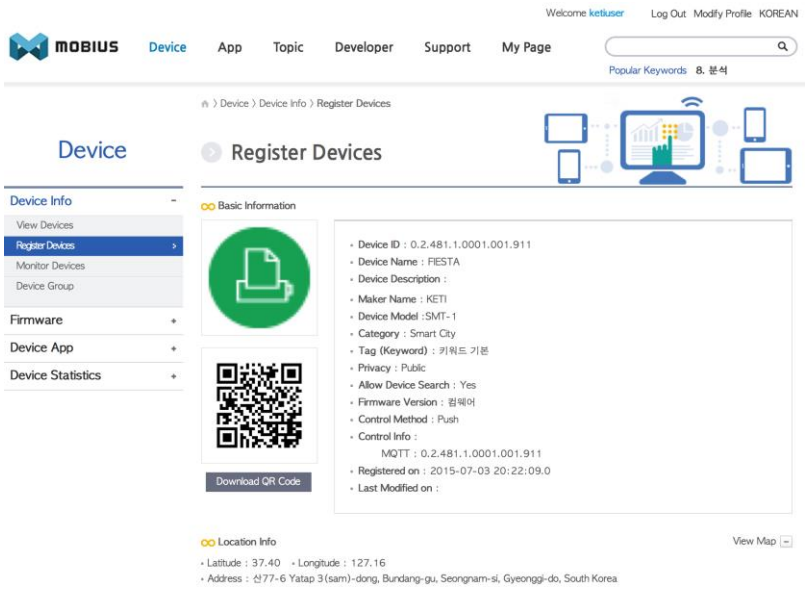
2. Device location:
   In this page, the device location is configured in two ways: map-based location and address searching.



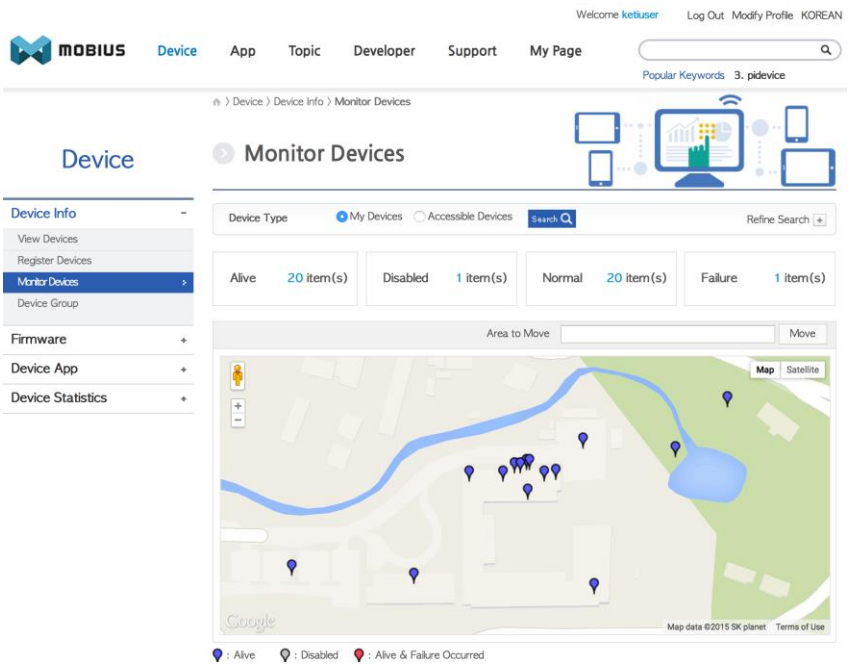
3. Device information:
   Additional information about the device is input including name, category, image, etc.



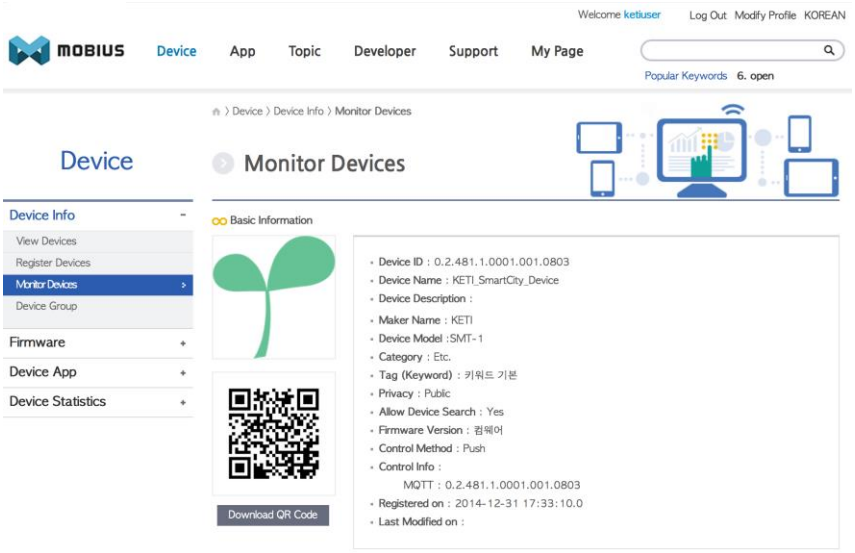
4. Device registration complete:
   Registration is complete, and registered information is shown.



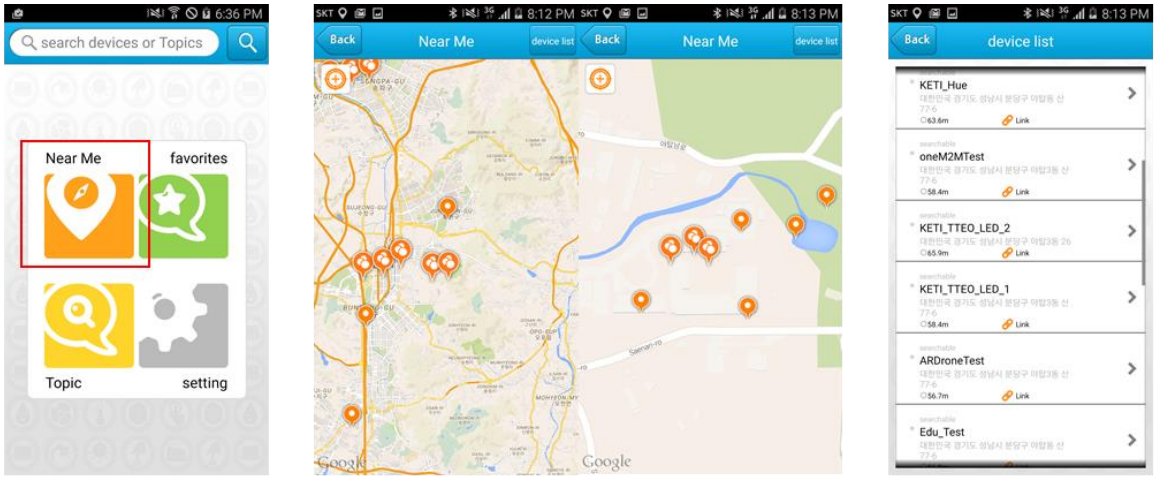
- Device monitoring page
  1. Browsing maps:
     This page shows the registered devices around the place of interest. While browsing the map, users can choose an IoT device of interest and click to see more detailed information.



  2. Device status monitoring:
     Once clicking a specific IoT device, the web page shows the detailed information about the device including device ID, name, maker, category, etc.

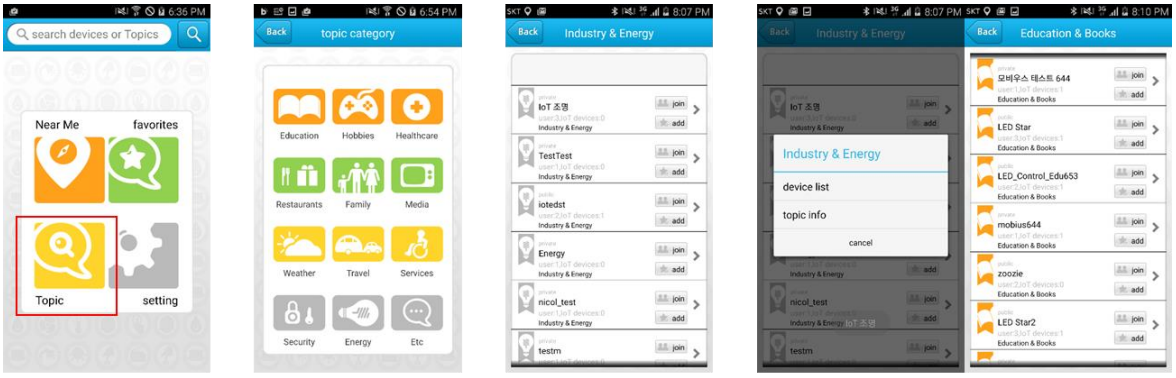


- IoT Browser: based on the indexing- and searching-related APIs offered by the Mobius, the IoT Browser allows users to explore IoT devices registered in the Mobius on their smartphones through a map-based interface.
  - Discovery services: to discover IoT devices registered in the Mobius, the IoT Browser provides map-based discovery and topic-based discovery.
  - Map-based discovery is started by touching 'Near Me' button in the main menu of the IoT Browser (a). It allows the user to browse the place of interest (b). When the user clicks a balloon icon, it shows the list of IoT devices registered at the selected location (c).
  -



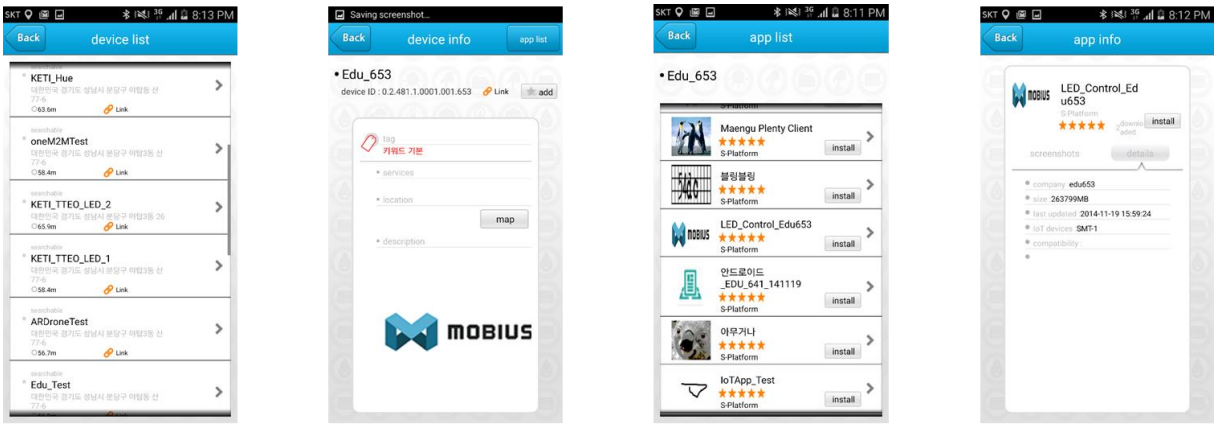
(a) Main menu of IoT Browser (b) Discover IoT devices near user (c) List of IoT Devices

  - Topic-based discovery is started by touching 'Topic' button in the main menu of IoT Browser (a). Users can choose one of 12 topic categories (e.g., education, hobbies, energy) (b). Choosing a topic class pops up a list of topics created by users (c). If the user chooses a specific topic in the user-created topics, the IoT browser shows the list of IoT devices registered under the chosen topic (d).



(a) Main menu (b) Topic categories (c) User-created topics (d) List of relevant IoT devices

- Either map-based or topic-based discovery leads users to the device list page, and users can choose an IoT device of interest (a). Choosing a device in the list shows the detailed information about the device (b). App list clicking brings the user to the page listing relevant smartphone applications (c). Finally, users can choose a specific application and download it on their smartphones to access the device (d).



(a) List of IoT Devices (b) Selected device's information (c) Relevant smartphone apps (c) Install the chosen app

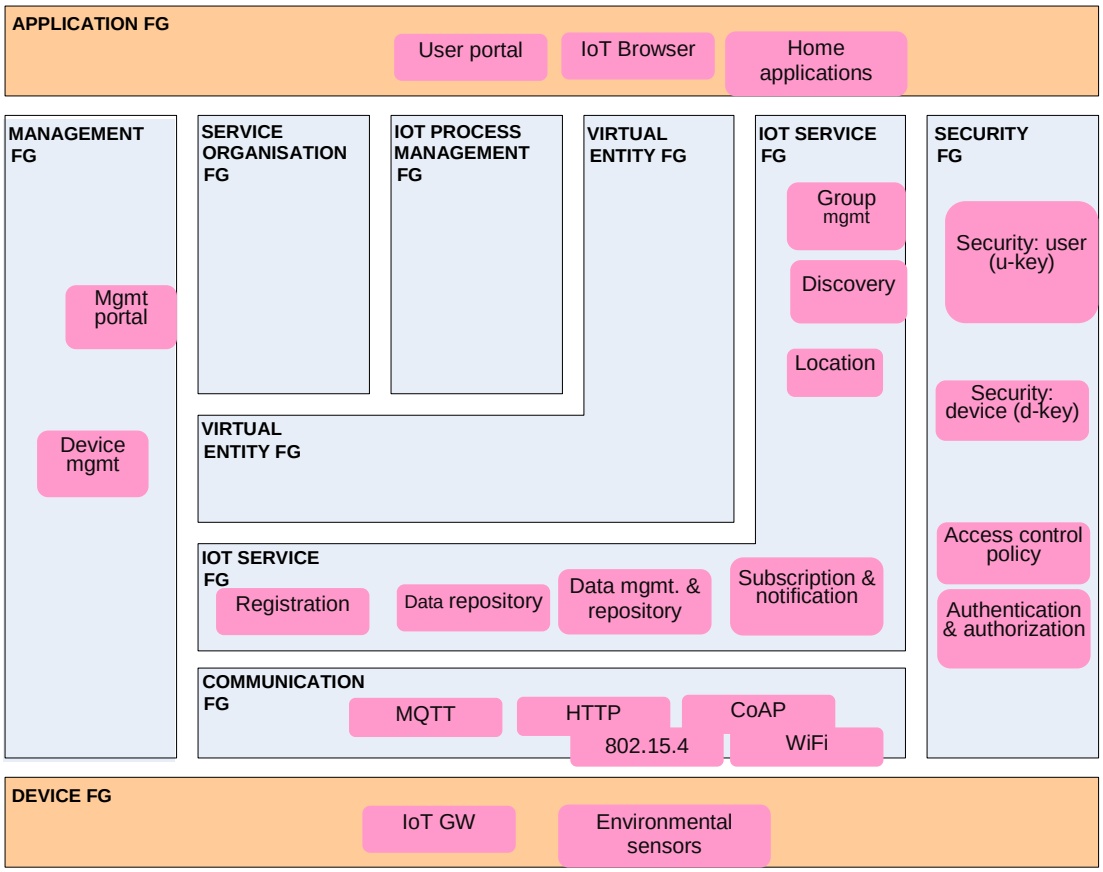


Figure 40. KETI Testbed Functional View

6.3 COM4INNOV testbed Functional View

1) 4G Services
   I. *SERVICE - Access to services via the 4G/LTE air interface by attachment to IMS services APN or by Internet APN*:
   With a 4G UE and the Com4Innov SIM card, a user can be attached to the eNodeB through the Radio Frequencies. Then the eNodeB is connected physically with the core network and the attached UE can use the different services that the C4I 4G/IMS network provides. Moreover, the UE can access the Internet world by IPv4/IPv6 by using the Internet APN and it can access the IMS services by using the IMS APN.

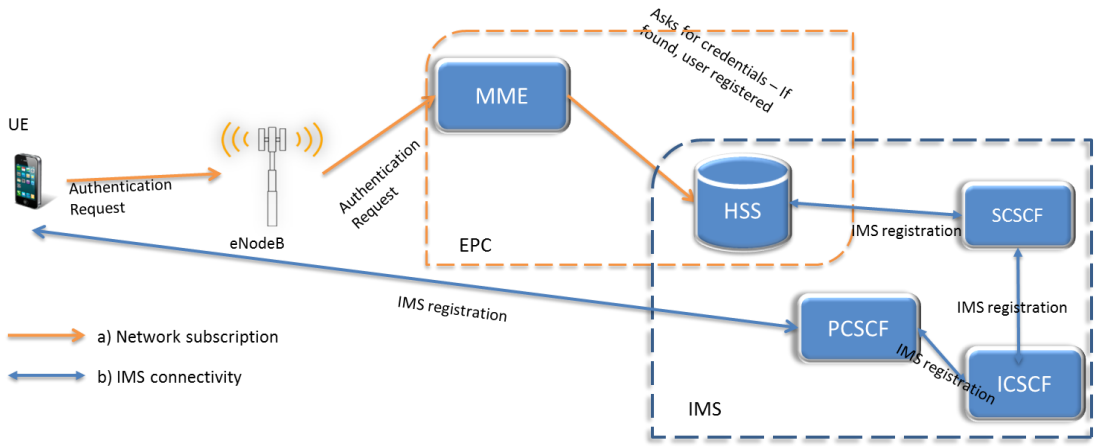


Figure 41. Access to services via 4G/LTE

II. *SERVICE - Access to services via the LTE WIFI Calling interface, including attachment to IMS services APN or attachment to Internet APN*:
By being connected to a Wi-Fi AP and if the UE supports the requirements for a VoWiFi call, the user who is allowed to access the service (or Internet services as well) through the ePDG which acts as a proxy can exchange trusted information and can access safely the services of the IMS network. Then the Wi-Fi AP is connected through Internet with the core network and the attached UE can use the different services that the C4I 4G/IMS network provides. Moreover, the UE can access the Internet world by IPv4/IPv6 by using the Internet APN and it can access the IMS services by using the IMS APN.

In the following figure the way WiFi calling works is shown:

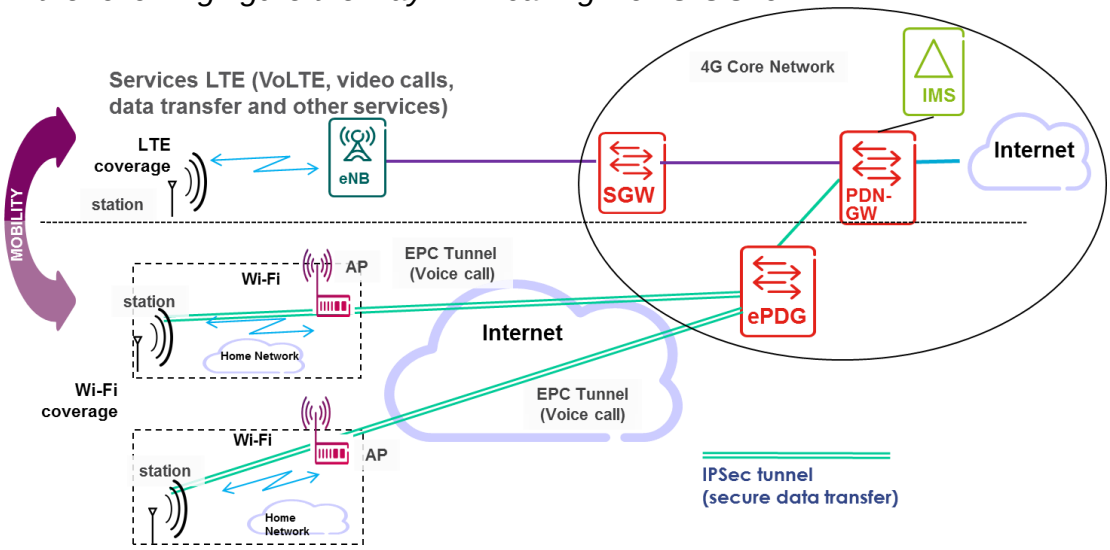


Figure 42. WiFi Calling

III. *Subscriber management*:
By providing the C4I subscriber with the C4I SIM card or SIM Card parameters when virtual SIM concept is needed, he is given access to the C4I core network. Before that, the subscriber has to be registered to the

core network and for that to happen, there are some security mechanisms and protocols that have to be passed. Both parts (SIM card and HSS – acts as database for storing subscriber confidential information) have credentials which have to be mapped in order for the subscriber to be allowed to use the services. For the LTE air interface the security protocol is AKA (Authentication and Key Agreement) and for the IMS is MD5 Digest.

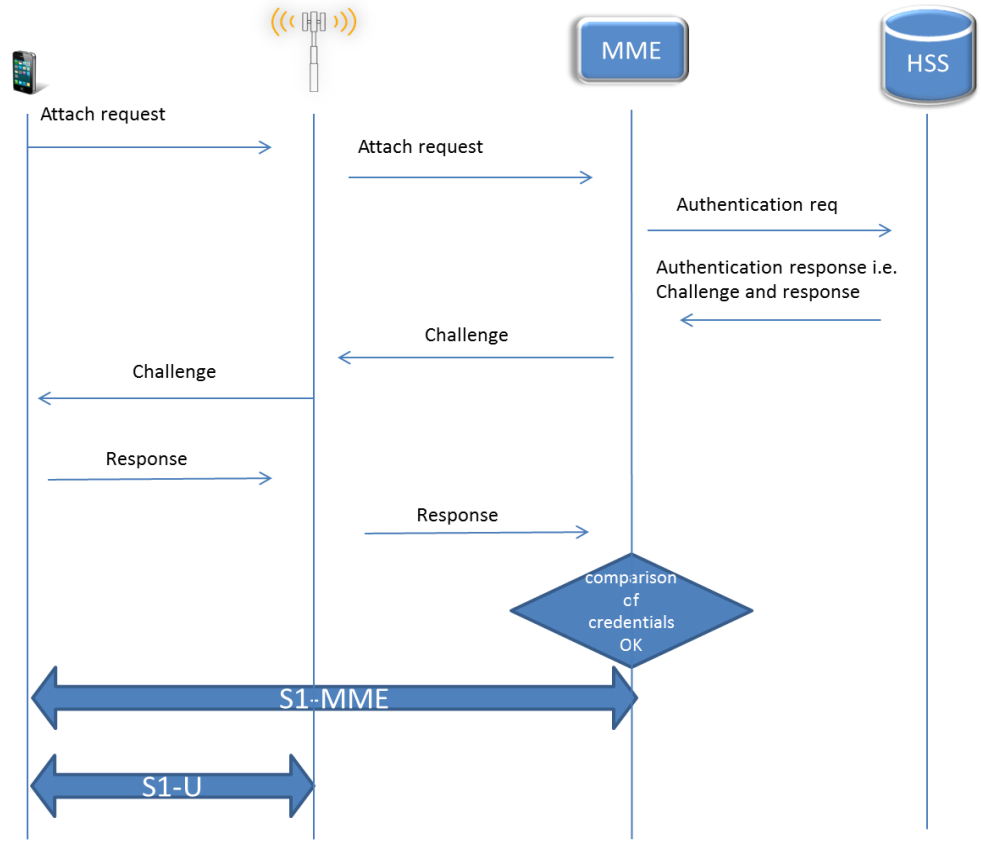


Figure 43. Authentication procedure

IV. TOOLS : *Access to protocol / traces information at different level:*
There is the facility to take traces at different levels during the usage of the C4I network; for instance traces can be taken at the protocol stack at the LTE air interface as well as inside the core network at the different components that the core network consists of (e.g. PCRF, P-CSCF, and MME etc). By that, the messages that are exchanged between the UE and the eNB (LTE air interface) and between the components (inside the core network) can be seen. So, in case a service does not work we are able to identify exactly what is wrong and at which level/component.

V. TOOLS : *Access to network Monitoring information:*
By the management/monitoring tool that is integrated inside the core network and through an alarm system we can be sure at each time if the services work well or an alarm is triggered, which means that a service is down and the network is malfunctioning.

The next figure shows the services described in IV and V:

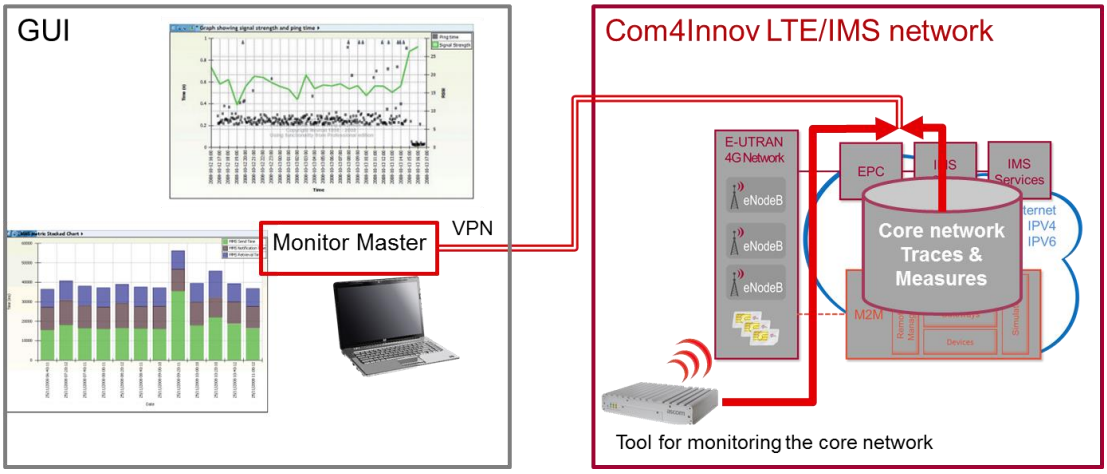


Figure 44. Access to traces and monitoring

VI. TOOLS : *User equipment traffic simulation*:
Com4Innov infrastructure and services provide tools to simulate User Equipment Traffic at the Radio Access Network level. The tools can simulate the maximum number of UEs supported by the eNodeB to load the network with different traffic shape such as HTTP, FTP, UDP, and VoLTE. Specific application simulation might be added on demand. The tool is script driven. It is currently used for mobile application oriented traffic but can be tuned to send M2M / IoT type of traffic. Using this tools one can check how an application is behaving under load.

2) IMS Services
Access Rich Communication Suite of **Services**:

a. *Presence services*:
Once a subscriber is connected to the C4I core network and is ready to use the IMS services, he can show his presence or not (online/offline).This can allow to support an RCS enhanced address book including integrated presence information.

b. *Instant Messaging Services:*
The messaging system of Com4Innov network offers IMS messaging and Short Messaging Gateway services as:

- **RCS One-to-One Chat**
- **Group Chat**
- **File Transfer**
- **Standalone Messaging**
- **SMS over IMS**

c. *VoLTE*:
The IMS architecture connected to the 4G core network gives the possibility to the C4I subscriber to pass VoLTE (Voice over LTE) calls as defined by IR92 in GSMA / RCS standards. This call will be passed over the 4G network. Com4Innov does not support legacy network and there is no fallback to the 3G

network. The only thing that is mandatory for the subscriber is to be well registered to the HSS of the network (by passing the steps that are described in the LTE services section) and have a UE that supports VoLTE calls. VoLTE calls give better QoS, HD voice etc. The audio codec standard that is used is IR.92 and is defined by GSMA.

d. *VoWiFi*:
By being connected to a WiFi AP and if the UE supports the requirements for a VoWiFi call, the user who is allowed to access the service through the ePDG which acts as a proxy can exchange trusted information and can access safely the services of the IMS network. Then the WiFi AP is connected through Internet with the core network and the attached UE can use the different services that the C4I 4G/IMS network provides like having a VoWiFi call. Such solution allows having access to the operator services even when the 4G/LTE network coverage is not sufficient.

e. *ViLTE*:
The IMS architecture connected to the 4G core network gives the possibility to the C4I subscriber to pass ViLTE (Video over LTE and over WIFI) calls. This is defined by IR94 in GSMA/RCS standards. The video codec standard that is used is IR.94 and is defined by GSMA. This combined with VoLTE.

f. *Handover between Wi-Fi and 4G*:
A UE connected to WIFI (with LTE authentication) will be able to get handover to LTE radio network. This can be used when performing test including geographical move and coverage.

In the next figure we show the way the aforementioned services are accessed and used:

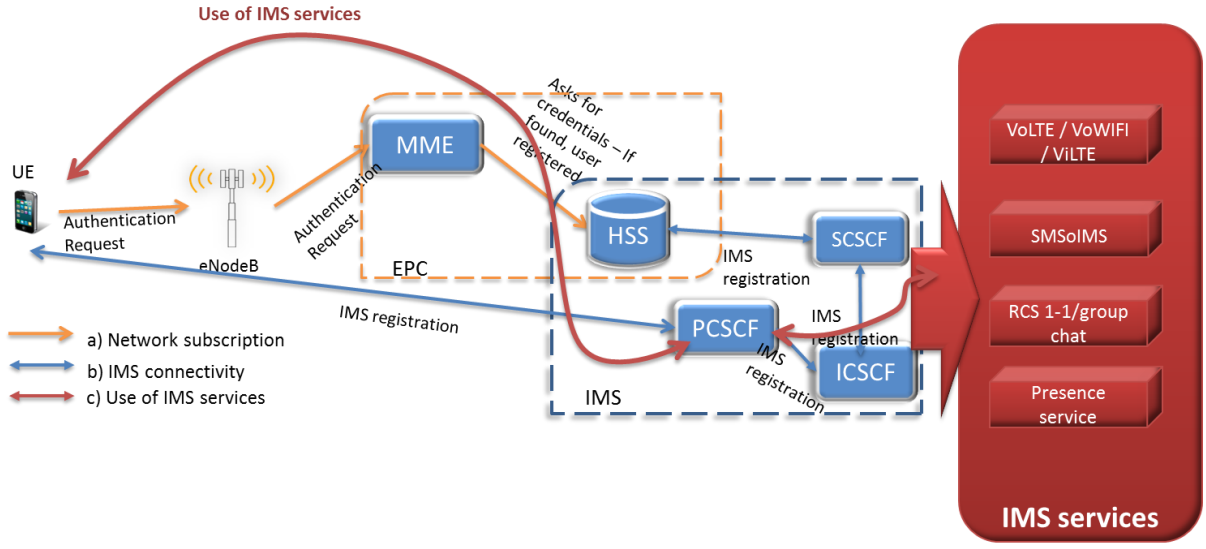


Figure 45. Procedure that is followed in order a subscriber to use the IMS services

3) M2M / IoT

I. TOOLS: *M2M data traffic simulation*: Com4Innov infrastructure includes simulator / robots allowing defining scripts to simulate data traffic from simulated gateway. Each simulated gateway can simulate multiple M2M / IoT devices with different streams of data.

II. TOOLS: *M2M Radio sensors simulation*. As for the data traffic simulator, Com4innov provide a tool that can simulate radio traffic, as it would be generated by wireless sensors. It can be used to test other devices / gateways reception capabilities.

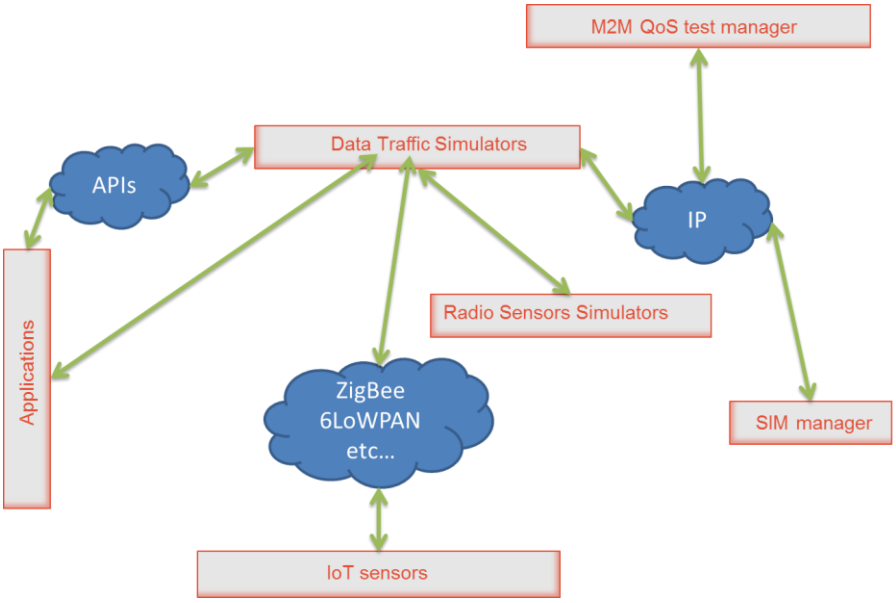


Figure 46. M2M Radio sensor simulation and data traffic simulators

III. *Capability to plug sensors or gateways to the infrastructure*: C4I owns many sensors (temperature, humidity, sound etc) and there is the capability of plugging these sensors to the infrastructure of C4I for better exploitation of the data they provide and monitoring etc.

IV. Access to a set of devices (sensors library) and their management application. List of type of devices (temperature, humidity, rain, pressure…). Here a non-exhaustive list of sensors available on the platform:

- Indoor Temperature
- Temperature and Humidity
- Outdoor Temperature
- High Temperature Thermocouple
- Low Temperature PT1000
- Heat flow
- Solar Radiation (albedo meter / pyrometer)
- Rain Gauge

- Anemometer/wind meter
- Fissurometer
- Gauge and temperature in concrete
- Temperature in Concrete
- Strain gauge
- Noise pollution
- Remote reading meters

V. Access to test environment with virtual machines, cloud environment, Generic Enabler applications such as BigData / IoT Broker / Configuration Manager from FIWARE Lab European project. Com4Innov provides capability to host applications and use FIWARE generic enablers in these different applications domains.

In Figure 47 the mapping to the Functional Components of the Functional Model of the services that the Com4Innov testbed provides is shown. A brief explanation of why these services are chosen to be mapped in these Functional Components follows:

On Security Functional Group: The “Subscriber Management” service that is provided is mapped in Security FG of the Functional Model as it treats security functionalities, authentication and authorization issues of users in order to use the IoT system and its applications. Also it can be mapped to “Management” Functional Group as it treats and manages the way a subscriber/user will be registered into the network in order to use the services.

On Communication Functional Group: The Communication Functional Group is related to the different communication protocols and access methods that are used in an IoT system in order to access the IoT system. Also, it contains the communication protocols for the exchange of data between the different devices that are installed inside the IoT system. Thus, both provided services “4G Access” and “WiFi Access” are two different access methods to the infrastructure.

On IoT Service Functional Group: The IoT Service Functional Group contains IoT Services as well as functionalities for discovery, look-up, and name resolution of IoT Services. So, “M2M data traffic simulation” and “M2M radio sensors simulation” are IoT services (or at least tools that are very close to be IoT services) because they simulate functionalities between devices.

IMS Services (VoLTE/RCS/VoWiFi/ViLTE): The IMS infrastructure provides services and applications that can be used by an authorized external user. Thus, these services are mapped into the “Application Functional Group”.

Network Monitoring Information: This service is mapped to “Management” Functional Group because it operates as surveillance to a big part of services whether they function properly. Also, it can go to “Service Organization” Functional Group.

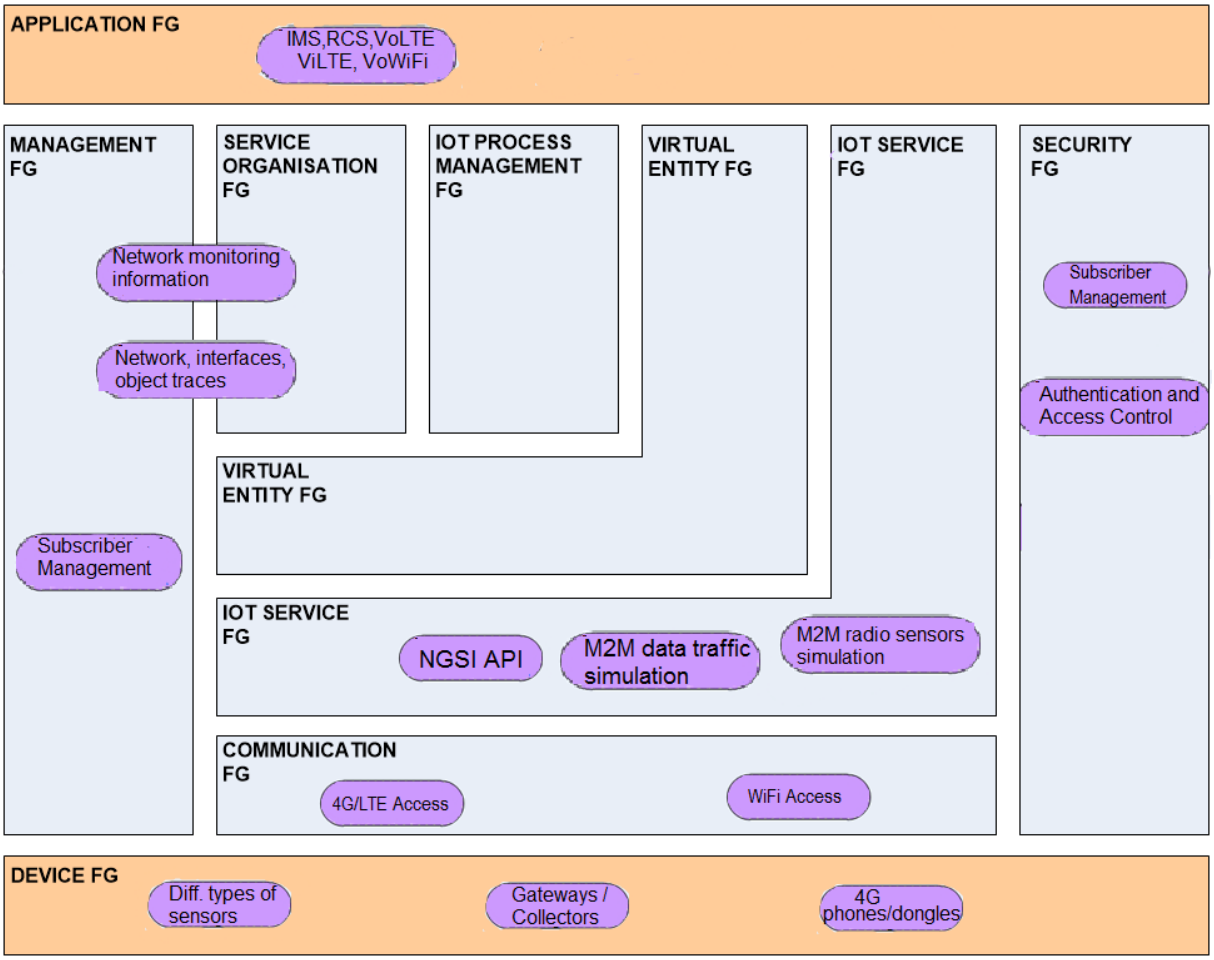


Figure 47. Com4Innov testbed services mapped to Functional View

6.4 SmartSantander testbed Functional View

As it has been introduced in Section 2.2, the SmartSantander testbed operates on an EaaS manner supporting two kinds of IoT experimentation, namely Native Experimentation and Service Experimentation. Within the FIESTA-IoT federation, SmartSantander testbed will focus on the information-centric Service Experimentation. Main functional blocks addressing the experimentation support lifecycle requirements are described below:

- **Resource Directory**

This component stores the descriptions from all the resources available at the SmartSantander testbed. These descriptions are natively specified following SmartSantander proprietary resource model but they will be upgraded to semantically annotated descriptions supporting FIESTA-IoT ontologies. The Resource Directory exports the part of the REST IoT API in charge of managing resources. This way it is possible to register new testbed resources but more importantly, as shown in Figure 48 it is possible to discover available resources. Discovery queries can filter the results based on location (e.g. all devices in this area), resource capabilities (e.g. devices that measure temperature or humidity), metadata available at the resource descriptions (e.g. devices which can measure noise between 0 and 30 dB) or any combination of them (e.g. devices measuring temperature in this area with an accuracy of 1%). It is also possible to subscribe to asynchronous notifications upon appearance of newly available resources matching the conditions set by the experimenter.

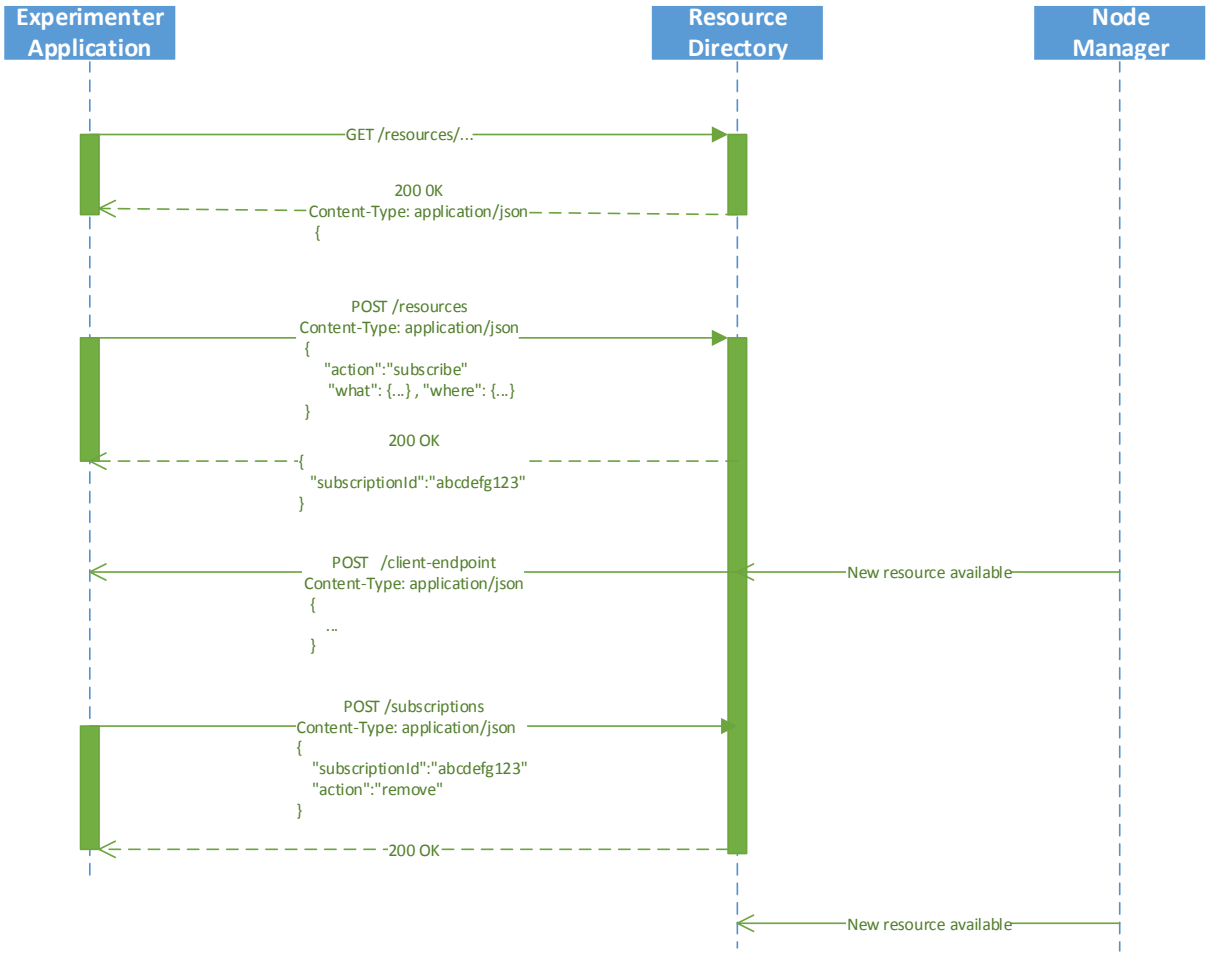


Figure 48. SmartSantander Resource Directory sequence diagram example

- **IoT API**

This is the main functional component at the SmartSantander platform as it encompasses the information-centric services supporting IoT Service Experimentation. The IoT API exports a number of RESTful resources that can be accessed by authorized experimenters to retrieve information generated by the IoT devices deployed within the experimental facility.

  - *IoT API (measurements)*

According to the SmartSantander terminology, a measurement is the minimum piece of information sent by any of the SmartSantander resources. It relates to a specific phenomenon and contains the phenomenon it relates to, the value measured and the unit of measurement in which this value is expressed. In contrast, an observation gathers one or more measurements and adds to it the information about the identity of the resource producing the observation, its location and the time instant in which it has been produced. This way, SmartSantander devices produce observations which contain measurements (e.g. a device with a temperature and humidity sensor will send one observation with a temperature measurement and a humidity measurement).

The IoT API allows accessing individual measurements independently of whether the IoT device had sent it together with other measurements in the same observation (e.g. accessing information on temperature without caring about humidity values).

- *IoT API (observations)*

While in many cases measurements are independent one from the other, even when they have been created at the same time by the same device, there are certain cases in which this is not the case and different measurements within an observation are intimately interrelated. IoT API also addresses these situations by allowing experimenters to access the complete observation. As shown in Figure 49, queries are made exactly the same way as in the measurements case and results can be filtered using conditions for one and/or many of the measurements within the observations.

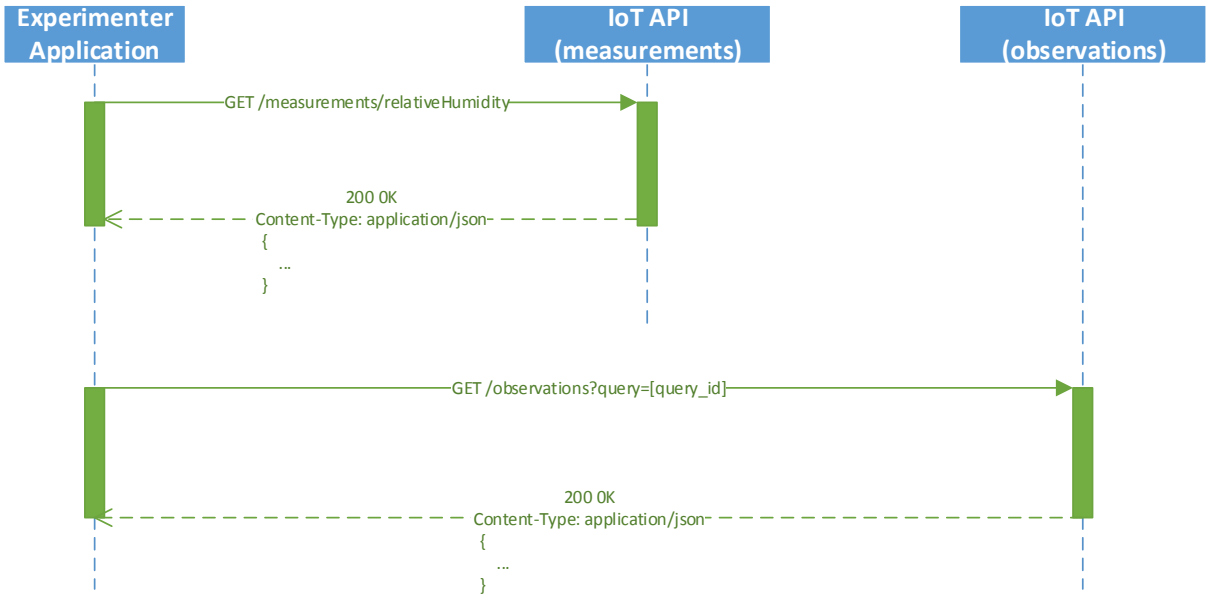


Figure 49. SmartSantander IoT API (measurements and observations) example sequence diagram

- *IoT API (queries)*

The IoT API supports complex ways for filtering and matching the measurements and observations requested. The queries that can be sent to the SmartSantander IoT API can request information based on three criteria, namely *when*, *where* and *what*. This basically allow an experimenter accessing data (measurements or observations) depending on when it has been generated (e.g. measurements generated between 1st July and 15th July), or where it has been produced (e.g. measurements generated 100 meters around these coordinates), or which device or phenomenon is of interest (e.g. temperature measurements or observations reported by resource with id `urn:x-iot:smartsantander:u7jcfa:f3001`). Any combination of the three criteria is, of course, possible.

- *IoT API (subscriptions)*

While queries enable synchronous request-response interaction, IoT API subscriptions enables support for asynchronous publish-subscribe approach, as it is shown in Figure 50. Experimenters might subscribe to measurements or observations of interest (in this case only the *where* and *what* criteria applies) and they will be notified by the platform as soon as such information is generated. Management of subscription is possible so that experimenters can tune the parameters specifying the

information in which they are interested.

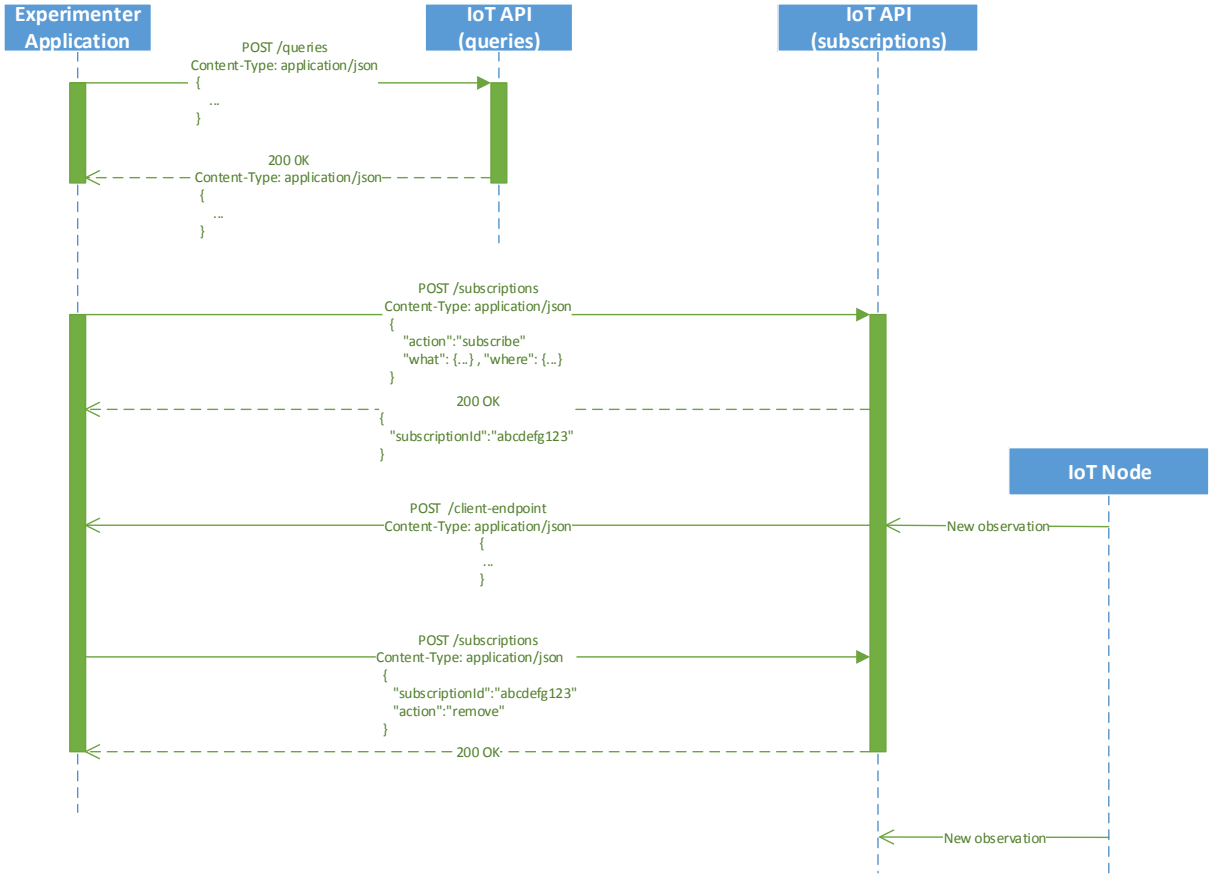


Figure 50. SmartSantander IoT API (queries and subscriptions) example sequence diagram

- *IoT API (users)*

IoT API does not only support the operations on testbed resources and data but also management operations are handled through it. In this sense, registration and management of users and its access rights is carried out using the IoT API RESTful resources.

- *IoT API (uom)*

Management of the SmartSantander Units of Measurement taxonomy is also carried out through the respective IoT API RESTful resource.

- *IoT API (phenomenons)*

Management of the SmartSantander Phenomenons taxonomy is also carried out through the respective IoT API RESTful resource.

- **Access control**

As it has been introduced in Section 2.2, IoT API is provided through REST services on top of HTTPS.

Authentication and access control is provided through two different means:

- SmartSantander authenticates and authorizes accesses to its services by using a Certification Authority (CA) certificate, based on the X.509 v3 standard [Coo08].
- Additionally, SmartSantander platform issues API keys that can be used to get access to the platform services.

This module is responsible for intercepting all the queries made to the IoT API and enforcing access control policies. In this sense, it is possible to prevent the access to previously defined resources on a per-user basis. As it has been previously mentioned access rights and policies are manageable through the IoT API.

- **Node Manager**

The platform monitors the status of the deployed infrastructure through a component called Node Manager. This component is always monitoring the testbed and every time it discovers there is no communication, in other words, when a sensor or device has not sent anything after a certain timeout interval, the Node Manager sends the corresponding message to the Resource Directory and the repositories to deactivate the node until it is up and running again. This way, information stored about the testbed resources is continuously kept up to date.

Figure 51 shows the mapping between the SmartSantander testbed components and the ARM FV framework.

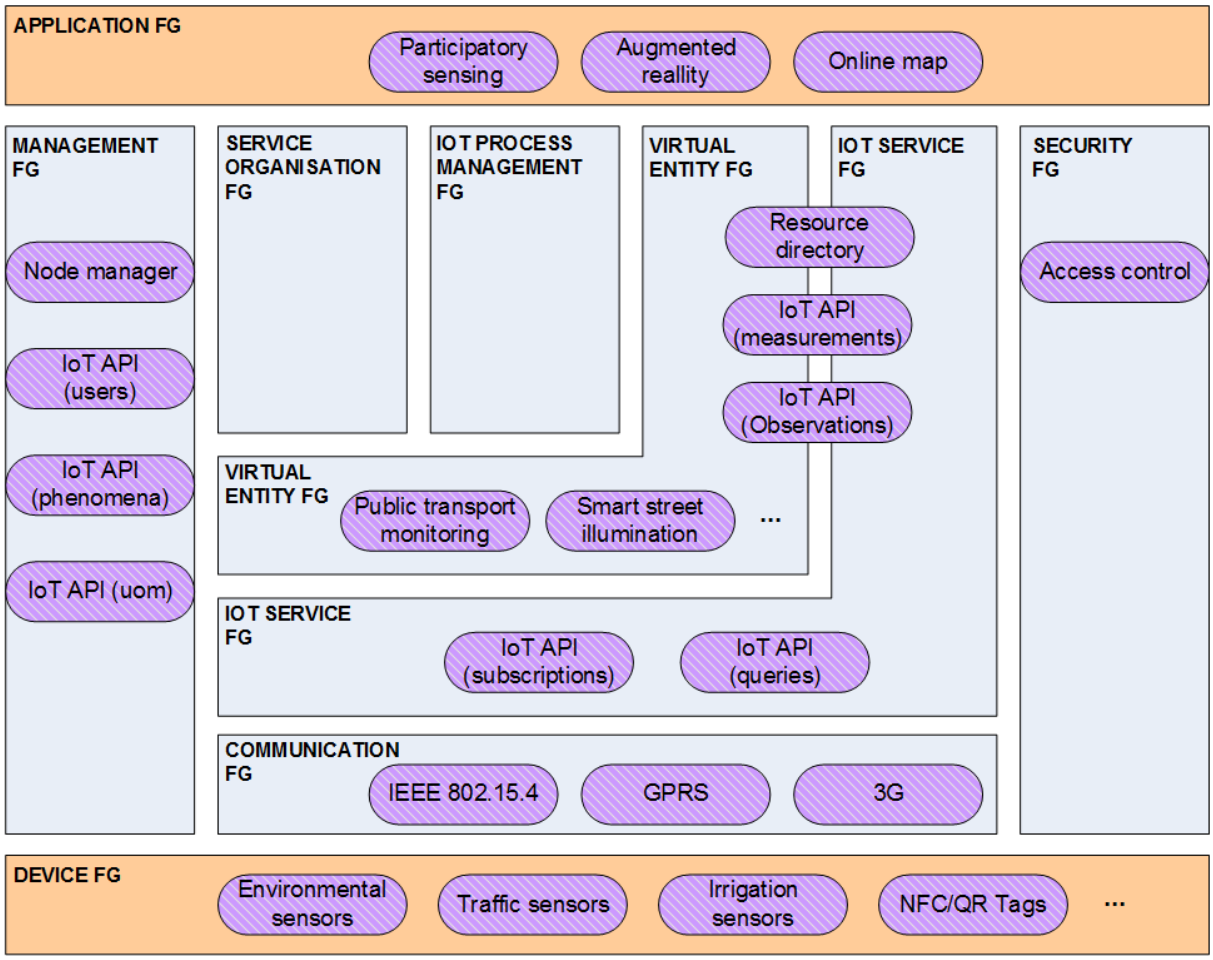


Figure 51. SmartSantander Services mapping to Functional View

6.5 Discussion on the different Functional View

In this section we draw some conclusions concerning the Functional Views generated by the services that each Testbed or Platform provides. In Figure 52 the Functional View for the four testbeds is shown and the services of each one are mapped in the Functional Groups.

From this figure we can see that the Testbeds use different types of APIs, or the same API that can implement different functionalities. For example, Com4Innov uses the NGSI adapter from FIWARE while at the same time UNICAN uses the IoT API which can do different services (e.g. IoT API for subscription, IoT API for queries etc).

Another thing that can be concluded is the variety of different communication protocols that can be used from a user in order to access the platform and benefit from the services that each testbed provides. For example Com4Innov, as it has a 4G/LTE/IMS infrastructure, provides 4G connectivity as well as the user can access the services via the WiFi air interface. On the other side, UNICAN acts as a complementary by using GPRS, 3G or IEEE 802.15.4 for the provision of its services.

Also, by observing the figure, the reader can easily point out that each testbed has its own management services for managing the subscribers, the resources etc. This Functional Group is important because it is of priority for each testbed to have

everything under control in terms, for example of retaining the connectivity or managing the identities of the users that take advantage of the platforms' services and in general to check that everything works as it should.

This Figure acts as an input to the content of the D2.4 "FIESTA Meta-Cloud Architecture and Technical Specifications".

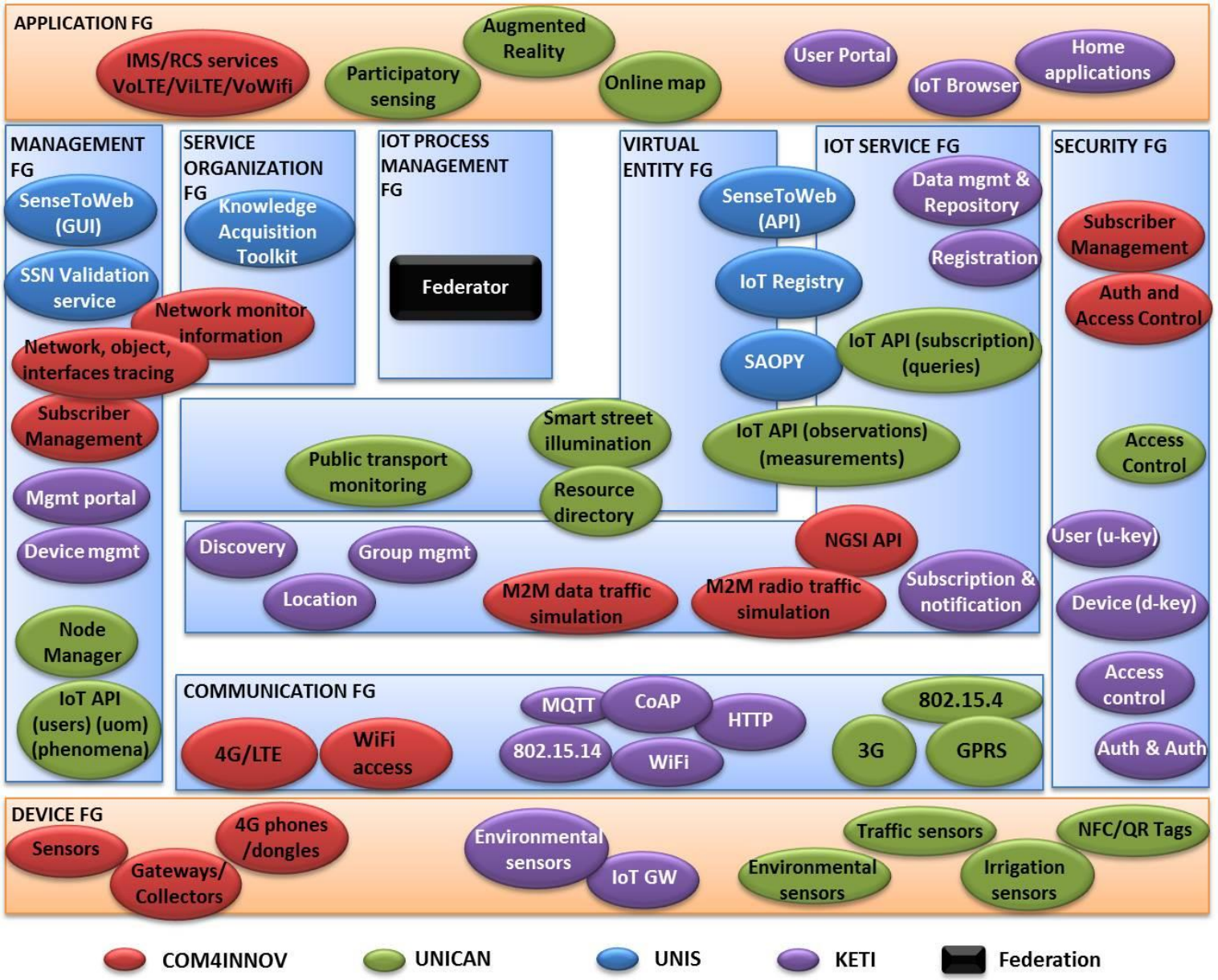


Figure 52. Functional View for all the different Testbeds

7 STATE-OF-THE-ART ON EXPERIMENTATION TOOLS

1. Introduction

FIRE (Future Internet Research and Experimentation) has made significant advances in the development of Experiment as a Service (EaaS) and the creation of tools to directly support the experimenter. Notably, these align with similar initiatives such as GENI[1] in the USA and common globally deployed standards are emerging. The general functions provided by experimentation tools match the established stages of the experiment lifecycle.

- ***Discovery***: the experimenter uses tools to search for testbeds and resources that match their requirements.
- ***Reservation and configuration****:* the experimenter uses tools to request resources to be utilised at a required time, and have the required configuration (e.g. a virtual machine has a specified OS and software installation).
- ***Control***: the experimenter uses tools to directly use testbed resources during the experiment, allowing querying and modification of each resource's state.
- ***Measurement:*** is critical to achieve validation by experimentation; such measurements have the objective of showing an experiment's results to be: reliable; repeatable (when run on the same testbed configuration); reproducible (in a different testbed under the same conditions); and verifiable (when compared against a reference model). The experimenter uses tools to monitor the experiment measures and the testbed behaviour (the experimental conditions).
- ***Reporting***: tools present the experimental results to the experimenter.

The following are FIRE tools developed for supporting the experimenter. This is not a comprehensive list, rather a selection of the mature and useful tools. Further, we only focus on the tool itself, not on the underlying technologies employed to provide experimental resources at the testbed.

2. jFED & FLACK

jFED[2] is a graphical-based tool for experimenters to provision and manage experiments on the Fed4FIRE[3] federation of testbeds. It is developed in Java and acts as a standalone tool (the functionality cannot be executed through a web browser). jFed covers the following of the experiment stages:

- *Discovery*: jFed lists the types of resources needed for experiments (computational nodes, wireless nodes, virtual machines, etc.). These are selected, and then the user can discover which testbeds make these resources available.
- *Reservation and configuration*: The topology editor allows the experimenter to configure the resource and network topology. This generates an RSPEC (a textual and reusable description of the experiment). The experimenter uses

[1] https://www.geni.net/

[2] http://jfed.iminds.be/

[3] http://fed4fire.eu/

jFed to begin the experiment, configuring the selected resources. This is achieved via ssh log-in to the nodes to configure them for the execution of the experiment.

Control, measurement and reporting are then performed by other tools (after the experiment is configured and started.

FLACK[4] is a visual experimentation tool providing similar functionality to jFED that was created as part of the GENI initiative. Flack covers the main functionalities of discovery and resource provisioning over SFA compliant testbeds.

There are also command line tools that provide similar functionality (discovery and provisioning of SFA resources) where a full GUI is not required by the experimenter: OMNI[5] and SFI (part of the planetlab[6] distribution).

Analysis

jFed and FLACK are both robust tools with strong technical readiness and appealing user interfaces; both work well for SFA compliant testbeds[7] and resources. They offer significant value in quickly setting up experiments, and shielding the experimenter from complex configuration tasks.

However, they are tightly coupled to testbeds making their resources available via the SFA testbed federation technologies; there is no possibility to include other advertised resources (e.g. a resource access via a REST endpoint).

3. OMF6

OMF6[8]is a command line tool that runs experiments described in OMF Experiment Description Language (OEDL). It contains a detailed description of the resources involved in an experiment and the sets of actions to perform in order to execute and measure that experiment. OMF6 works with already provisioned resources (and is typically used in combination with other tools such as jFed). The following experiment stages are covered:

- *Control*. Commands in the OEDL script are executed at the resources. An example is the OMF command to change the Wi-Fi channel. Behind the curtains it will determine which wireless driver is used on the resource, and will then select the suitable set of command line commands to execute.
- *Measurement*. The OEDL script executes commands that are instrumented with measurement instrumented software (e.g. using OML[9] packages such as ping-oml the ping tool providing OML measurements).

[4] http://www.protogeni.net/wiki/Flack

[5] http://www.fed4fire.eu/%28http://trac.gpolab.bbn.com/gcf/wiki/Omni

[6] https://www.planet-lab.org

[7] Slice-Based Federation Architecture (July 2010) edited by Larry Peterson, Robert Ricci, Aaron Falk, Jeff Chase

[8] http://omf.mytestbed.net

[9] http://mytestbed.net/projects/oml/wiki

- *Reporting*. OML data is stored in SQL databases for further inspection. The OMF-Web tool[10] can also be utilised to visualise reported data in graph form.

Analysis

OMF6 is largely targeted at networking and resource management experiments; it is again tightly coupled to the SFA vision of experimental testbed configurations. Further, it is not particularly easy to use: a number of command line tools, an experiment language, and a measurement language must be learned and manually configured. Hence, it is targeted at experimenters with strong technical backgrounds.

However, the OML framework does contain useful sub-tools such as the web performance measurement tools (e.g. the httperf-oml2 command line tool) which could potentially be leveraged to investigate QoS and performance management IoT experiments within the FIESTA-IoT domain.

4. EXPERIMONITOR

EXPERIMonitor is a software tool focused on the management of experiment content that allows developers to explore the relationship between QoS and QoE in complex distributed multimedia systems. The tool is specifically designed to support the observation of systems where user-centricity, mobility, ad hoc participation and real-time access to information are critical to success.

- *Measurement*. Data measurements are monitored and collected from constructed and provisioned experiments.
- *Reporting*. EXPERIMonitor offers a dashboard for real-time observation and historical data exploration of data sets. A system and data exploration view is provided. Data exploration is the interface used to derive insights from results data. Data exploration focuses on the participants, activities and interactions with application and services within the system under test.

Analysis

The tool focuses on a particular domain i.e. media content and specific experiment types i.e. QoS and QoE. It is not as generalised as the previous tools, and covers only a subset of the experiment lifecycle. However, it has the potential to be useful for performing QoS and QoE experiment on already provisioned IoT data experiments.

5. IoT-Lab Tools

IoT-LAB[11] provides a very large scale infrastructure facility suitable for testing small wireless sensor devices and heterogeneous communicating objects. Experiments are controlled by REST API that can be accessed by a command line tool or a web portal. The experiment is described in a proprietary JSON format, and REST commands are used to execute experiments.

- *Discovery*: REST commands to view resource descriptions at all sensor testbed sites.

[10] https://github.com/mytestbed/omf_web

[11] https://www.iot-lab.info/tools/

- *Reservation and configuration*: REST commands to submit experiment, start experiment etc. Further REST commands to update firmware of nodes.
- *Control*. REST commands to execute commands on sensor nodes.
- Measurement. Specific measurement tools included in firmware e.g. battery consumption and radio sniffing.
- Reporting. Measurement data stored to file.

Analysis

Simple command line tools are available for the complete experiment lifecycle. However, these are targeted at a specific experimental sub-domain i.e. wireless sensor networking. There is limited reusability in the software tools themselves.

6. Super Stream Collider

Super Stream Collider (SSC)[12] is a platform, which provides a web-based interface and tools for building mashups combining semantically annotated Linked Stream and Linked Data sources into easy to use resources for applications. The system includes drag&drop construction tools along with a visual SPARQL/CQELS editor and visualization tools for novice users while supporting full access and control for expert users at the same time. Tied in with this development platform is a cloud deployment architecture which enables the user to deploy the generated mashups into a cloud, thus supporting both the design and deployment of stream-based web applications in a very simple and intuitive way.

- *Discovery*: No discovery mechanism is integrated, and it has not been applied to discover resources from third party testbeds.
- *Reservation and configuration*: The drag and drop tools support the configuration of data stream from multiple sources. Other actions may not be configurable e.g. Get or a change action.
- *Control*. See before concerning control beyond streaming.
- *Measurement*. The experiment measurement is the data collected from the data streams.
- *Reporting*. The tool provides strong support for visualisation of experimental result.

Analysis

Highly related to FIESTA-IoT, it is however missing the ability to discover and provide more fine-grained control of different resource APIs. It is likely this tool can be combined with other tools for those features i.e. this concentrates on the data outputs and visualisations produced.

7. SmartSantander Service Experimentation Layer

SmartSantander Service Experimentation Layer (SEL)[13] represents the SmartSantander platform for IoT Data Management, embracing the

[12] http://superstreamcollider.org/

[13] https://api.smartsantander.eu/

architecture/framework that defines the building blocks/functional components in charge of harvesting, managing and exposing the underlying IoT data. Concerning the tools that this testbed offers the experimenters to interact with SmartSantander's counterparts, an interface, called IoT-API, enables the access (through a RESTful interface) to the different resources/devices, including as well different types of services (i.e. historical and last values), which can be retrieved in either synchronous (i.e. query-based) or asynchronous (i.e. publish/subscribe mechanisms) manners.

- *Discovery*: The IoT-API allows experimenters to discover the underlying resources deployed over the SmartSantander facility.
- *Reservation and configuration*: As already mentioned, there are two ways (through the IoT-API) to gather the information generated by the different resources of the testbed. On the first hand, experimenters can retrieve either historical or the last values captured by the sensors (synchronous access); moreover, they can subscribe to those services they want and receive the information published by them in an asynchronous way.
- *Control*. Despite the platform allows experimenters to take control on resources and alter their legacy state (e.g. change the frequency of the measurements, reprogram the nodes via Over The Air Programming – OTAP, etc.), the current version of the IoT-API does not contemplate this type of operations yet.
- *Measurement*. Every measurement carried out in SmartSantander is stored in a database (i.e. Data Repository). Thus, different experimenters who point at the same services that expose the data will retrieve identical information.
- *Reporting*. Raw data (JSON) gathered from the IoT-API can be easily parsed and processed. A set of additional tools[14] can be used in order to have a graphical representation of both resources and data.

8. **SmartCampus experimentation tools**

SmartCampus, a user-centric testbed for experimental IoT research, which is deployed in a real world office setting which spans an entire building. The SmartCampus testbed relies on the SmartSantander management framework, developed on top of the WISEBED APIs for the common management plane tasks, such as IoT nodes reservation, reprogramming and collection of out-of-band statistics. Results and traces are collected to a MySQL Experiment DB that stores standard trace format and expose them for further analysis through a well-defined REST interface.

- *Discovery*: currently via the data broker, a json map of the available nodes is provided. An in-house linked data platform used in FIWARE will be employed for the semantic registration and discovery of the IoT nodes.
- *Reservation and configuration*: is done via the WISEBED APIs

[14] http://maps.smartsantander.eu/

- *Control*. Currently there are not actuation devices in the testbed, but it is planned with the new generation of IoT nodes.
- *Measurement*. By default, measurements are made at intervals, and real-time data is stored.
- *Reporting*: results are stored in a MySQL Experiment DB. And exposed via the RESTful "REDUCE" API.

9. Com4Innov's experimentation tools

Com4Innov provides some experimentation tools that can be used by the external user or experimenter under the support and assistance of the operating team of Com4Innov. The experimenter could use Com4Innov's experimentation tools in order to assure that his experiment or application, which runs on the Com4Innov platform, functions as supposed and provides the expected results.

1. Traffic generation – **Mobipass** tool
   - **1.1.** 4G/LTE traffic: Com4Innov infrastructure and services provide tools to simulate User Equipment Traffic at the Radio Access Network level. The tools can simulate the maximum number of UEs supported by the eNodeB to load the network with different traffic shape such as HTTP, FTP, UDP, and VoLTE. Specific application simulation might be added on demand.
   - **1.2.** IoT traffic on LTE: The tool is script driven. It is currently used for mobile application oriented traffic but can be tuned to send M2M / IoT type of traffic (LTE cat1 modules). Using this tool one can check how an application is behaving under load.

2. Protocol Analysis
   - **2.1. TEMS Investigation** tool: TEMS Investigation is used for radio network optimization and maintenance and verification as well as for convenient Protocol and messages analysis. It supports a large spectrum of technologies as LTE-A, LTE, WCDMA etc and it is used for services like VoIP, video telephony, FTP, HTTP, TCP, UDP etc

3. Quality of Service guarantee
   - **3.1. TEMS Investigation** tool: Com4Innov uses the TEMS Investigation tool for troubleshooting, verifying, optimizing and maintaining the 4G/LTE network. This tool can validate the equipment functionality, it can measure and report key performance indicators (KPIs such as MOS – Mean Opinion Score as defined for measuring Quality of Experience) from a user's perspective which leads to the assurance of quality of service (e.g. VoLTE).
   - **3.2. Monitor Master** tool: By this tool that is integrated listening information on the network via probe, and through an notification system we can be sure at each time if the services work well or an notify is triggered, which means that a service is down or malfunctioning. Moreover, the tool allow to understand how the tested application protocol is behaving frame by frame and interacting with the core network.

10. Conclusions

Tool	Resource Domain	Discovery	Reservation	Control	Measurement	Reporting
jFed	Virtualised computational and networking resources	x	x			
FLACK	Virtualised computational and networking resources	x	x			
OMNI	Virtualised computational and networking resources	x	x			
SFI	Virtualised computational and networking resources	x	x			
OMF	Virtualised computational and networking resources			x	x	x
EXPERIMONITOR	Media				x	x
IoT-Lab	Wireless Sensor nodes	x	x	x	x	x
SSC	Linked data streams		x		x	x
SmartSantander SEL IoT-API	Wireless Sensor Nodes	x	x		x	x
FIESTA-IoT GUI (Experiment 2)	Weather/environmental sensors	x	x		x	x
UNIS SmartCampus	IoT nodes	x	x		x	x
Monitor	Monitoring/mana				x	x

Master (C4I)	gement tool					
TEMS Investigation (C4I)	Management/ QoS/QoE assurance tool			x	x	x
Mobipass (C4I)	Simulator of radio traffic (IoT-M2M/LTE)				x	x

The general purpose tools developed in projects such as Fed4FIRE and GENI illustrate how experiments can be controlled an executed. That is: what features should be provided by an experimental tool.

However, they are not well aligned with the IoT experiments in FIESTA-IoT (at present FIESTA-IoT testbeds do not need to be SFA compliant). Similarly, more niche tools demonstrate desirable features and, while these are again not closely aligned with FIESTA-IoT experiments it is possible examples such as SSC can be re-used as part of the FIESTA-IoT toolset.

8 CONCLUSION

This deliverable “Analysis of IoT Platforms and Testbeds” is focused on the Testbeds and IoT Platforms, analysing and describing what they do and how they do it. It also uses the set of test-bed requirements produced in task T2.1 to better understand if each testbed can fulfil the stakeholders’ requirements. This task, then, models the Testbeds and IoT Platforms in functional blocks using the IoT-A ARM model. It gathers the type of information they provide, and how they provide this information so that Task 2.4 can take this into account when developing the FIESTA-IoT Architecture. The outcome of this task will provide a basis for WP3.

In this deliverable the main testbeds and platforms participating in the FIESTA-IoT project are analyzed. The ultimate reason is the interconnection and sharing of data between the testbeds and platforms. The testbeds and platforms that will be interconnected are: SmartSantander, University of Surrey, Com4Innov and KETI. This interconnection will enable the deployment of IoT experiments and allow services within federated utility-based cloud computing environments.

In particular, a very detailed description and analysis of the resources that each of the aforementioned testbeds includes is taken place i.e. the IoT devices and sensors, gateways, APIs. It is very important to the owners of the different platforms to have an exact knowledge of all the resources that could be part of the interconnection of the different testbeds around Europe. Except the material resources written above, there are recorded as well communication protocols, applications as well as security and authentication mechanisms that are used.

Moreover, in the final chapters of this deliverable there is an introduction on the IoT-ARM which is proposed by the University of Surrey in order to model the resources and the different components of the IoT testbeds and platforms. This model will be used in order to translate the available resources of each testbed into a common language. There is a mapping of the gateways, the IoT devices, the APIs etc in a common model, so that the interconnection of the testbeds and platforms becomes easier. Finally, there is the translation of the resources into the IoT-ARM naming.

It can be seen that each of the presented environments provides added value that enriches the overall project whether it is in the data domain or connectivity and capability to run new experiments. Nevertheless, even when platforms and testbeds are interconnected, the large variety of existing and described information demonstrates the need for semantic analysis at the cloud layer in order to facilitate the use of all distributed data and resources.

9 BIBLIOGRAPHY

(Cooper D., 2008) Cooper D., Santesson S., Farrell S. Boeyen S., Housley R., Polk W. (2008). *InternetX.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL)* (Project Management Institute, 2013)*Profile*. IETF RFC 5280, May.

(DigiMesh, 2015) DigiMesh Networking Protocol. (2015). http://www.digi.com/technology/digimesh/.

(Vandenberghe, 2013)Vandenberghe, W., Vermeulen, B., Demeester, P., Willner, A., Papavassiliou, S., Gavras, A., Boniface, M. (2013). *Architecture for the heterogeneous federation of future internet experimentation facilities*, In Future Network and Mobile Summit (FutureNetworkSummit), (pp. 1-11).

(initiative, 2015) FI-WARE initiative (Accessed 2015). https://www.fiware.org/. FIWARE lab, the open innovation Lab (Accessed 2015). .

(Gluhak, 2011) Gluhak, A., et al. (2011). *A survey on facilities for experimental internet of things research*, IEEE Communications Magazine, vol.49, no.11, pp.58-67.

(project) Provisioning of urban / regional smart services and business models enabled by the Future Internet (OUTSMART) project (Accessed 2015). https://www.fi-ppp.eu/projects/outsmart/.

(Tonneau, 2015) Tonneau, A.S., Mitton, N., Vandaele, J. (July 2015) *How to choose an experimentation platform for wireless sensor networks? A survey on static and mobile wireless sensor network experimentation facilities*, Ad Hoc Networks, Volume 30, pp. 115-127.

(Gavras, 2010) Gavras, A. (2010). *Experimentally driven research white paper .* ICT-FIREWORKS .

(Haren, 2009) Haren, V. (2009). *TOGAF Version 9.0.*

(IEEE, 2007) IEEE. (2007). *Guide for Monitoring, Information Exchange, and Control of Distributed Resources Interconnected with Electric Power Systems.* IEEE.

(IEEE, 1990) IEEE. (1990). *IEEE standard glossary of software engineering terminology.* IEEE.

(IoT-A, 2011) IoT-A. (2011, June 16). Retrieved May 29, 2015 from Project Deliverable D1.2 – Initial Architectural Reference Model for IoT: http://www.iot-a.eu/public/public-documents/d1.2/view

(IoT-A, 2013) IoT-A. (2013, July 15). Retrieved May 29, 2015 from Deliverable D1.5 – Final architectural reference model for the IoT v3.0: http://www.iot-a.eu/public/public-documents/d1.5/view

(MyFIRE, 2011) MyFIRE. (2011, May). Retrieved Jully 6, 2015 from D1.2 Taxonomy on common interpretation of testing, testing approaches and test beds models: http://www.my-fire.eu/documents/11433/38630/D1.2+-+taxonomy+on+common+interpretation+of+testing%2c%20testing+approaches+and+test+bed+models?version=1.0

(Project Management Institute, 2013) Project Management Institute. (2013). *A Guide to the Project Management Body of Knowledge* (5th Edition ed.). USA.

(Soukhanov, Ellis, & Severynse, 1992) Soukhanov, A. H., Ellis, K., & Severynse, M. (1992). *The american heritage dictionary of the english language.* Boston, MA: Houghton Mifflin.

(Volere) "Volere Requirement Specification Template" available at www.volere.co.uk (last accessed 31/07/2015)

(Haller *et al*., 2013) "A Domain Model for the Internet of Things", in iTHINGS'2013 proceeding, Beijing, China (see also IEEE eXplore)

(Rozanski&Woods, 2011) N. Rozanski and E. Woods, "Applying Viewpoints and Views to Software Architecture" , Addison Wesley, 2011

(IoT-A UNIs) "IoT-A Unified Requirements", available at http://www.iot-a.eu/public/requirements/copy_of_requirements (last accessed 31/07/2015)

