SMARTSANTANDER

SEVENTH FRAMEWORK PROGRAMME

SMARTSANTANDER PROJECT

INFSO-ICT-257992 SmartSantander

D1.1

First Cycle Architecture Specification

Contractual Date of Delivery: 30th April 2011

Actual Date of Delivery: 6th May 2011

Editor(s): ALU-SP

Author(s): See Authors list

Participant(s): ALU-SP, CEA, CTI, EYU, TID, UC, ULANC, UniS, UZL

Work package: WP1

Estimated person months: 21.92

Security: Public

Version: 1.0

Abstract: This deliverable provides the initial SmartSantander architecture compliant with the essential large-scale IoT experimental facility requirements, a subset of the complete set of identified requirements. Hence, it is the starting point for the detailed technical specification, implementation and deployment activities that will be carried out in the corresponding workpackages. It is a living document and it will be updated and upgraded in the different phases and cycles of the project setting the basic guidelines for each development phase.

Keyword list: *Architecture, components, development phases, interfaces, SENSEI, Telco 2.0, WISEBED.*

Authors

Partner	Name	E-mail
UC	Luis Muñoz	luis@tlmat.unican.es
	Luis Sanchez	lsanchez@tlmat.unican.es
	Jose Antonio Galache	jgalache@tlmat.unican.es
	Veronica Gutierrez	veronica@tlmat.unican.es
ALU-SP	Raul Garcia	r.garciap@alcatel-lucent.com
	Pilar Poyato	pilar.poyato_vergara@alcatel-lucent.com
UZL	Sönke Nommensen	nommensen@itm.uni-luebeck.de
	Florian Massel	massel@itm.uni-luebeck.de
CTI	Evangelos Theodoridis	theodori@cti.gr
CEA	Pierre Roux	pierre.roux@cea.fr
	Vincent Berg	vincent.berg@cea.fr
UniS	Alex Gluhak	a.gluhak@surrey.ac.uk
	Michele Nati	m.nati@surrey.ac.uk
EYU	Srdjan Krco Boris Akrapovic	srdjan.krco@ericsson.com boris.akrapovic@ericsson.com
TID	Jose Antonio Jiménez	jajh@tid.es
ULANC	Rajiv Ramdhany	r.ramdhany@lancaster.ac.uk

Table of Contents

List of Figures

List of Tables

Acronyms and Abbreviations

AAA	Authentication, Authorisation and Accounting
ACI	Access Control Interface
ASI	Application support interface
CM	Configuration Management
DoS	Denial of Service
ESI	Experimental Support Interface
FCAPS	Fault, Configuration, Accounting, Performance, Security
FM	Fault Management
HTTP	Hypertext Transfer Protocol
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
MSI	Management support interface
NMS	Network Management System
OLAP	*On-Line Analytical Processing*
OSA/V	OS Abstraction and Virtualisation
OTA	Over-The-Air
OTAP	Over-The-Air Provisioning
PM	Performance Management
RFID	Radio Frequency Identification
SAN	Sensor Area Network

TLS	Transport Layer Security
WP	Work Package
WSN	Wireless Sensor Network

1. INTRODUCTION

The main goal of the project is the creation of a European experimental test facility for the research and experimentation of architectures, key enabling technologies, services and applications for the Internet of Things (IoT) in the context of the smart city.

The project builds on experiences and results of 2 FP7 projects, namely WISEBED and SENSEI, and Telefonica's USN service platform. The WISEBED is a FIRE project that provides the tools and mechanisms for experimentation in wireless sensor networks (individual or federated) in controlled environments. Focus of the SENSEI project was on providing procedures and algorithms for interconnection of various heterogeneous sensor and actuator networks with the end user applications and services via a set of well defined interfaces. The corresponding SENSEI components enabling discovery of sensors and actuators based on their capabilities together with the definition of the related interfaces are taken from this project. The USN platform is facing the end users and provides tools for persistent storage of IoT devices' measurements and a set of service interfaces.

Combining these 3 installations that have addressed the IoT research domain on different levels (WISEBED – wireless sensor networks experimentation research, SENSEI – interconnection of heterogeneous wireless sensor and actuator networks, USN – service platform) with a large number of IoT devices installed around the 4 cities (primarily Santander, but also Guildford, Lubeck and Belgrade) in the real physical environment that brings a range of deployment and maintenance challenges, we aim at providing a unique-in-the-world facility for IoT experimentation and evaluation under realistic operational conditions.

The goal of this document is to define the initial SmartSantander architecture compliant with the essential large-scale IoT experimental facility requirements, a subset of the complete set of identified requirements. Based on these essential requirements, the architectures and functionalities provided by the 3 facilities mentioned above are analyzed and relevant and required components are taken from each. After that, a gap analysis have been performed to identify any missing components as well as to identify which functionalities and interfaces have to be adapted in order to have the components coming from 3 different systems working.

Leveraging the existing WISEBED, SENSEI and USN functionality as much as possible, the first integrated SmartSantander platform will enable the initial deployment of IoT devices in Santander. In the following phases, the number of IoT devices will be increasing and the architecture together with the available functionality of the platform will be expanding based on the already identified technical requirements, but also on the new requirements as the result of the initial deployment experience as well as gathered from the research community through surveys and workshops.

Therefore, this document is the starting point for the detailed technical specification, implementation and deployment activities that will be carried out in WP2 and WP3. This is a living document and it will be updated and upgraded in the different phases and cycles of the project setting the basic guidelines for each development phase.

Section 2, the list of use cases that have been selected for the initial phase of the deployment and a list of desired properties that have been identified as the key properties needed in any IOT experimentation environment are presented.

Section 3 provides an overview of the three deployment phases that have been identified. A general description of each phase is included together with the goals of each phase in terms of the number of devices to be deployed per phase.

Section 4 identifies a list of requirements selected for Phase 0 deployment. The identified requirements are considered to be essential for the envisioned experimental facility.

Section 5 contains the architecture definition of the experimental facility which has been based on the functional decomposition of the overall system according to a logical grouping of the requirements in section 4. The architecture distinguishes four subsystems, namely AAA, experimentation support, application support and testbed management. Several required interfaces are also identified and listed in this section.

Section 6 details the functional mapping of the already existing selected components into the SmartSantander architecture and describes how the components can be extended to fulfil the envisioned SmartSantander functionality. Also a list of new components to be developed is presented to cover the functionality needed by SmartSantander and that is not present already.

Conclusions and references are covered in sections 7 and 8.

2. IOT EXPERIMENTATION NEEDS AND LIMITATIONS OF EXISTING IOT TESTBEDS

2.1. Selected use cases

For the SmartSantander project three groups of users have been considered; experimenters, service providers and city/citizens. These groups are the ones described next:

1. Experimenters**:** group in charge of deploying different experiments over different testbeds and under different contexts, in order to test the correct functioning of the SmartSantander platform, discover new use cases, and detect possible lacks in it.
2. Service providers and application developers: are high-level users using the testbed facility to create higher-level services based on data extracted from the sensor network. These users rely on a generic application services provided by SmartSantander delivering sensor readings (e.g., periodic, query driven, or event based) and on the testbed infrastructure to pick the appropriate internal protocols to best suite the application needs to provide access to sensor data.
3. City and citizens**:** group of users of the SmartSantander facility as well as consumers of services hosted on the facility.

Different use cases have been defined and selected for the different groups of users mentioned before with the goal of achieving a network of wireless sensors from which different testbeds may be extracted, and all of them must meet the group of properties mentioned below.

The use cases selected for the different users in the initial deployment of sensors are the following (between parentheses the properties relative to section 2.2 we aim to achieve):

- Use cases for experimenters:

  1. Re-use of Experiment Configurations (repeatability and replayability): the goal is to save entire testbed configurations from prior experiments, such that precious time-consuming configuration steps can be avoided. Pre-specified configurations can be accessed and loaded by other experimenters to observe results for themselves and compare with their own experiments. Relevant 'repeatability' data from the testbed itself such as statistics on observed radio interference may be gathered over the lifetime of an experiment, to aid experimenters in understanding and contextualizing their results.

     A 'batch processing' facility could help to automate the execution of the same experiment many times, but such that in each successive experiment some parameters are calibrated in order to observe the influence on the obtained results. This could be particularly useful in systems which have 'tuning' parameters, wherein the optimal collection of settings is not known in advance for a particular deployment.

  2. Testbed virtualization (repeatability, replayability, scalability and experimental environment): when an experiment is configured, it is likely that some of the sensing requirements/capabilities are not physically supported on the actual nodes. The system thus provides a facility to virtualize that sensing capability which for the purpose of the experiment can be configured to behave as a normal sensor.

- Use cases for service providers:

  1. OS heterogeneity (heterogeneity): As part of their experiment set up, service providers can specify different Operating Systems to run on various nodes. The system will then combine their experiment software code and generate OS-specific images for flashing particular nodes. In the case of multiple sensor code images for an experiment, the system keeps track of the sensor code image version deployed in each node.

  2. Sensor Code Image Quarantine: Before deploying an experiment to the real testbed, the code image is loaded into dedicated quarantine network or simulator. A test run of the experiment code is performed to create an execution profile, power consumption profile and detect faults such as memory leaks. The quarantine network or simulator must be as realistic as possible in terms of heterogeneity and functionality, compared to the actual deployment network such that an experiment's software is reliably and accurately evaluated. Any malicious experimental code can thus be neutralised before actual deployment.

- Use cases for user communities:

  1. Management of scarce parking resources: This use case provides users with dynamic and real-time information about the parking spaces available in the limited parking zones (called OLA in Santander). This information may be shown in panels located inside the parking zones or via SMS sent to mobile devices. Two different users are distinguished:

     Neighbours of the parking zones have a special identification with which they can park in these areas without paying anything, but a fix quantity per month/year.

     Other users must pay for a ticket whose value changes based on the time the vehicle can be parked in the parking zone (maximum 2 hours). When this time is exceeded, user must obtain a new ticket in order to keep their vehicles parked in the same parking space.

     If the ticket isn't renewed after exceeding the paid time or if the vehicle has been parked without the ticket, drivers are fined with the corresponding quantity.

     Users may receive an alert in their mobile devices before the parking time is exceeded.

  2. Parking Management for People with Disabilities: This use case provides people with disabilities with a service for checking the available parking spaces for people with disabilities as well as, pre-reserving them.

     In case these parking spaces have been occupied by unauthorised vehicles, an alarm is triggered to inform the authorities. The city council can study the statistics they are provided with, in order to plan the number and location of parking spaces for people with disabilities for the future.

  3. Load/unload areas management: The goal of this use case is to keep control of the occupancy of the load and unload areas. This task involves preventing unauthorized vehicles park in areas reserved for the load and unload of goods (RFID tags or similar are used for the identification of authorized vehicles), controlling that the time that one authorized vehicle spend in these areas doesn't exceed an established number of minutes, as well as identifying the need to provide the city with more load/unload areas.

2.2. Desired properties for IoT testbeds

The realisation of adequate facilities for research and experimentation with future Internet of Things based technology solutions motivates several requirements and corresponding challenges. These are mainly driven by several factors:

- The need for increased realism that an experimental facility has to offer to accurately reflect the real world usage conditions of the developed IoT technology
- The increased demand, by the scientific community, for repeatable experimentation and the sharing not only of the results but also of the experiments themselves
- Increased diversity of solutions and mechanisms to be embedded within IoT devices and various levels of the protocol stack

- Increased market pressure to make developed IoT solutions and corresponding technological building blocks more broadly available and re-usable across multiple application domains
- An increased need to understand the potential socio-economic impact of IoT solutions and the inclusion of the human in the experimentation loop
- The need for integration of IoT research and experimentation with other technological developments of a Future Internet and corresponding testbeds.

In the following we provide brief descriptions of the main requirements and resulting challenges.

Heterogeneity

Initial IoT research focused on advancing individual technology-specific building blocks such as radio frequency identification (RFID) and wireless sensor networks (WSN). Many current testbeds target only one such particular technology and often consist of homogeneous IoT devices. As these technologies are maturing, there is an increasing consensus that the IoT will consist of many heterogeneous devices based on different technologies with various capabilities. An appropriate IoT testbed must provide an adequate heterogeneity to tackle the resulting technological challenges [2]. A major challenge for IoT experimental facilities is an effective configuration and execution of experiments across heterogeneous testbed resources and the corresponding management of devices. A further challenge represents the programmability of these heterogeneous devices, which often come with diverse execution environments.

Scale

Another important aspect contributing towards the realism of IoT experimentation is the ability to run experiments at an adequate scale. Many experiments thus far have been limited to smaller-scale testbeds, as hardware, roll-out, and management have been costly. While a few 10s to 100s of nodes may be sufficient for simple experiments, larger scale experiments executed in recent years (e.g., the ExScal experiment [4]) highlighted differences in system behaviour at larger scales. While recently decreasing hardware costs make large-scale testbeds more affordable than ever, a variety of technological challenges must be addressed, in order to make experimentation with 1000s of IoT nodes possible. This includes minimizing human intervention in the operation, plug-and-play configuration of newly deployed IoT resources and fault-management enabling automated recovery of malfunctioning resources. Another example is the development of supporting mechanisms for the experimenter that ease the selection of adequate testbed resources for a particular experiment.

Experimental environment

IoT technologies depend heavily on ambient environmental conditions in which they are deployed. The complex behaviour of observed physical phenomena and varying environmental conditions, together with the difficulties associated with modelling such behaviour are other factors, which motivate the need for

experimentation under realistic operational conditions. Current testbeds are mainly lab-based and provide at best approximate real-world conditions (e.g. by replaying events of previously recorded real-word, small-scale experiments). To support the necessary realism, future IoT testbeds should be deployed in environments that resemble as close as possible the envisioned target deployment environments. Moving testbed deployments out of the lab and into the wild poses challenges, such as realizing out-of-band management and control planes, as many of the currently wired solutions will have to be replaced by wireless ones. Further challenges are due to the increased overhead of the maintenance of testbed equipment in such deployments. Ease of access is often a trade off against the threat of physical tempering, deliberate damage to or the theft of equipment or other malicious behaviour that may cause interference to experimentation.

Mobility

A key role of the IoT is to provide information on real-world entities and events or the interaction capabilities that influence them. Many such entities may move around in a real world environment, thus making the IoT devices attached to them mobile. Handling such mobility and the associated system dynamics is thus a key requirement for future IoT solutions and hence adequate support for it must be provided by experimental facilities. A particular challenge is the provisioning of adequate support to control and exploit realistic mobility of both IoT devices and real world entities during experimentation.

Concurrency

With an increased investment in the experimental facilities infrastructure, supporting multiple concurrent users and experiments will become a necessity for an economically viable operation. Larger-scale testbeds will therefore support multiplexing of concurrent experiments. Unlike existing testbeds for Internet experimentation (e.g, PlanetLab that is based on large server clusters), IoT devices are substantially resource-constrained, making virtualization at the hardware-layer very difficult. Such ”virtualization” is more feasible at the testbed level than on the individual node level and will mainly support the multiplexing of experiments in space and time. An inherent challenge is the selection of testbed resources to minimize the interference of concurrently executing experiments (e.g., caused by collocated wireless transmissions, access to the same hardware resources or even actuating the environment and thus influencing other experiment, etc.), while satisfying the requirements of the experiment. This includes the efficient scheduling of experiments to maximize the number of concurrent experiments across geographically isolated testbed deployments or even across overlapping partitions of the same testbed deployment.

Repeatability and Replayability

There is an increasing trend in the scientific community to repeat experiments within and across different testbeds. Recent calls by researchers have gone a step further by highlighting the need for the *replayability* of experiments [5], even across different testbeds. While contributing to scientific rigor, *replayability* provides researchers with the means to quickly build on someone else’s work thus accelerating the scientific progress of

the community. It is clear that IoT experimental facilities benefit from such capabilities, however a variety of technical challenges exist. *Repeatability* is often very difficult even on the same testbed, primarily due to the wireless nature of many IoT experiments. Ever changing ambient conditions make it virtually impossible to obtain the same results at different experimental runs. While a total replication of real-world conditions may be impossible, solutions supporting the monitoring of the radio environment during experimentation and benchmarking of different testbeds could partially overcome the associated issues by providing hints for the interpretation of experimental data. *Replayability* requires agreement on standards for the specification of experiments, collection of traces, and the packaging of experimental results across a variety of testbeds, which is difficult due to the heterogeneity of different IoT testbeds and the diverging interests of testbed operators.

User involvement and impact

Many IoT applications are centred on human users or require their active participation in experimental data collection. This makes experimentation more difficult to control. For example, a user may forget to attach an IoT device (e.g., monitoring physiological parameters) or fail to provide input for data collection when required. In both cases the results may be invalidated. Experimental facilities that involve human users must also offer mechanisms to allow for the evaluation of the social impact and acceptance of IoT solutions and applications. Challenges include the automated detection of situations when unexpected user behavior influences the validity of the collected data and also the provision of efficient multi-modal mechanisms for user feedback collections, e.g. after IoT interventions during experimentation.

Federation

Federation with other IoT testbeds is necessary to achieve scale or to add capabilities for experimentation, which are not locally available. As mentioned previously, important reasons for limited scale and diversity are hardware cost, space constraints, lack of expertise, or required management resources. Federation is a viable solution to create larger and more heterogeneous experimental facilities out of specialized, smaller-scale facilities. From the management perspective, requirements on federation are a common framework for authentication, authorization, accounting, reservation and experiment scheduling. From an experimenter's point of view, the interconnection of facilities is important in order to undertake experiments that can actually exploit this merged infrastructure.

3. AN OVERVIEW OF THE PROJECT DEVELOPMENT PHASES

Aiming at achieving a massive deployment of an Internet of Things infrastructure in a city scenario, requires careful planning in order to cope with the requirements imposed by the different FIRE stakeholders as well as to reduce the impact on the city services and the citizens. Hence, SmartSantander has conceived an iterative approach, with four iterations, for the deployment of the aforementioned infrastructure. Each iteration embraces the typical engineering project roadmap: i) Requirement identification; ii) architecture specification; iii) Implementation; iv) deployment, v) assessment and vi) refinement.

SMARTSANTANDER PROJECT

In order to speed up as much as possible the project roadmap, SmartSantander looks for synergies with three main pillars:

1. WISEBED [WSB]. The experience gained in this FIRE project is used to reduce the time to deployment in those subsystems where commonalities are identified. In particular, the framework provided by the Testbed Runtime Server (see Section 5 for more details) as well as its architectural modules are migrated to the SmartSantander platform.

2. SENSEI [SENSEI]. Resource Directory (RD). A directory in which the specification of the available resources are included. It plays a capital role both from the service and experimentation perspective.

3. USN. An information repository of the data monitored by the IoT devices deployed around the city. The USN plays a key role in terms of end-user service provision.

Aiming at taking advantage from the very beginning from these synergies an iterative process was conceived. Although in the DoW just three phases were initially planned, in a very early stage of the project (month 2, October 2010) we realized that it was necessary to expand this number to four, starting with a Phase 0 lasting till June 2011 (month 10) which is being used by the consortium to get experience with the deployment process, dealing with both technical deployment issues and administrative ones (for example, Call for Tenders logistics). Last but not least, this Phase 0 is also being useful to reveal some of the companies ready to give support to a real a massive IoT deployment. These four phases are depicted in Figure 1.

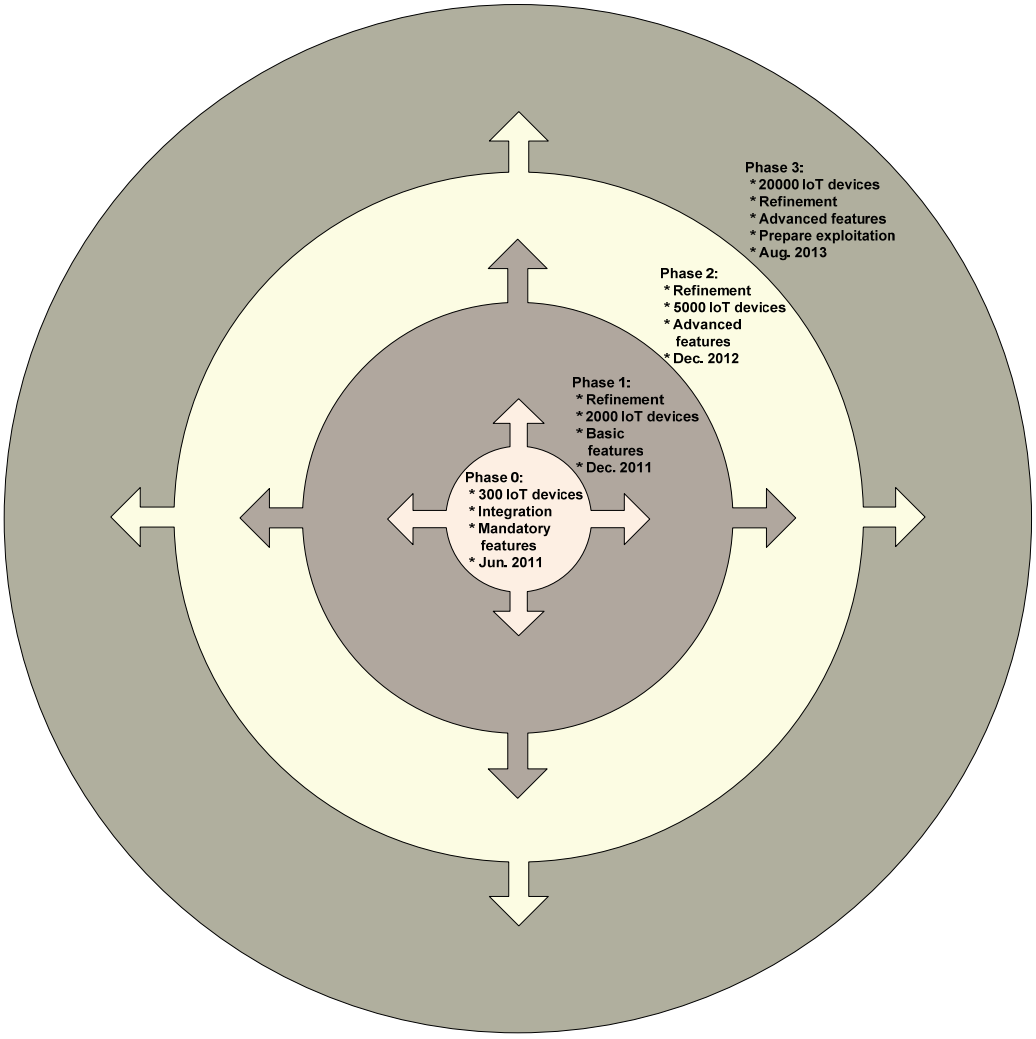

Figure 1. Schema representing the four phases approach agreed in SmartSantander

The following objectives and dates have been agreed in these four phases:

- Phase 0. As it has been said, before launching the Call for Tenders of *Phase 1,* aiming at gaining experience in different issues related to both technological and administrative aspects, this stage was considered to be appropriate before starting with the first big deployment. The amount of IEEE 802.15.4 devices in this reduced pilot was 300 due to the small budget allocated (55,000 €). However, we do agree that this "training" exercise has allowed the consortium to predict and overcome many issues that were unknown (such as technology maturity, service and experimentation dual support, economical trade-offs, deployment considerations, logistics …). The solution provided by the company responsible for equipment provision in the *Phase 0* has proved to be very convenient for supporting both experimentation and service provision. The main idea is integration of two transceivers in the repeaters, one with the IEEE 802.15.4 protocol stack and the other one with this protocol stack and a proprietary routing protocol (so called Digimesh) on top of it. It is expected that

at the end of May 2011 the system will be fully operative and integrated with the Testbed Runtime Server.

- Phase 1, in which around 2000 heterogeneous devices have to be deployed and set up around M15 (December 2011). The goal of this stage is to extend the deployment of the infrastructure, initiated in *Phase 0*, aiming at supporting outdoors parking control service, including monitoring of available places for disable people as well as dedicated load/unload areas. The provision of such a service implies deployment of three types of elements:

    - Sensors under the pavement which include a ferromagnetic sensor as well as the IEEE 802.15.4 protocol stack configured in a very efficient duty cycle in order to maximize the battery duration. These sensors are configured to support an over the air programming (OTAP) operation for no more than four times per day aiming at reducing power consumption. The experimentation supported by these devices is basically related to the service plane (hence involving data aggregation, data mining, etc.).

    - Repeaters placed in public street lamps which allow gathering data from the sensors as well as communicating in a multihop approach with peer devices. These elements are powered by external sources which mean that they do not have any limitations in terms of the battery power. Both experimental and service planes are supported by such repeaters.

    - Gateways placed in a limited number of areas and needed to interconnect clusters of sensors and repeaters to the Internet. In general they are embedded PCs in which the data gathered from sensors and repeaters are stored.

Figure 2 depicts the different elements previously described that have been considered in the Call for Tenders of this Phase 0.

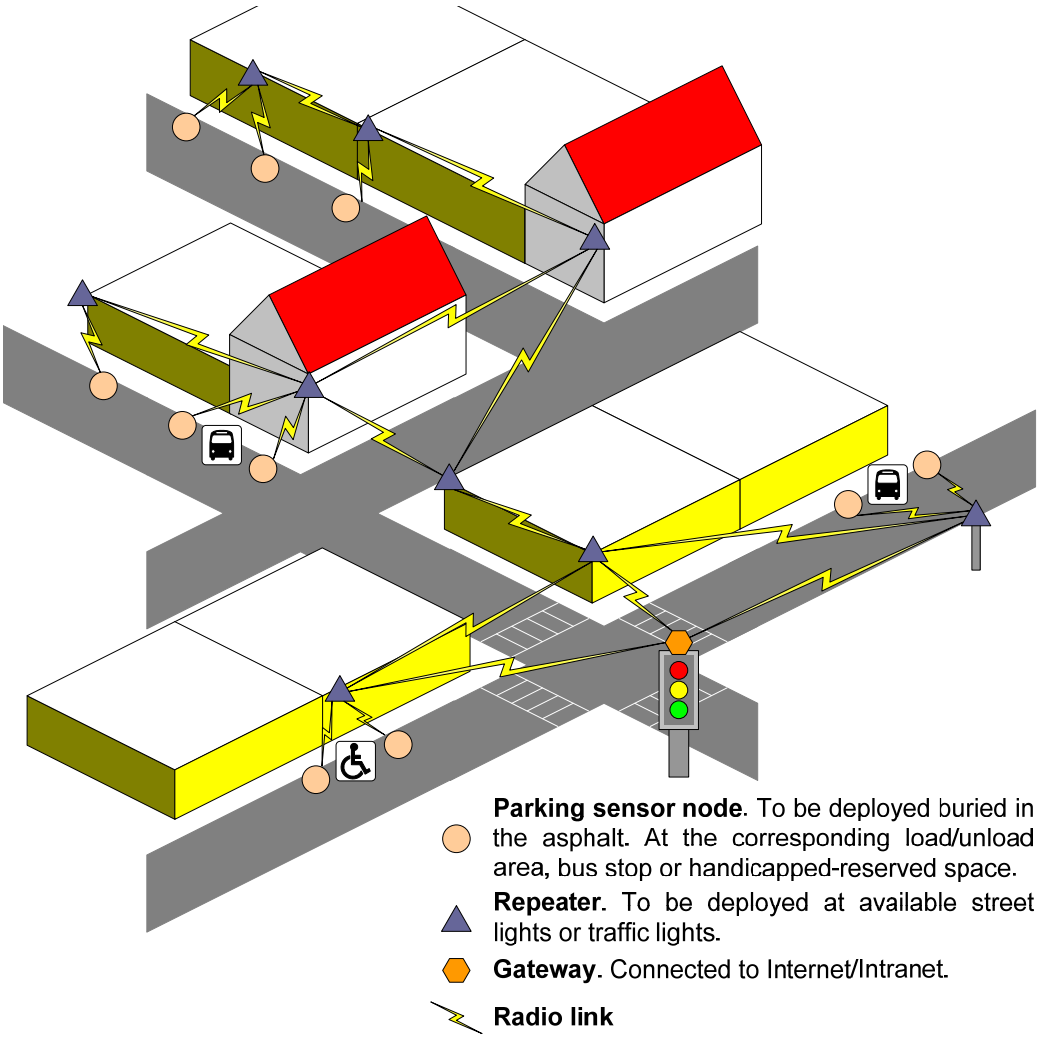

Figure 2. Network elements giving support to both experimentation and service in phase 0

- Phase 2, in which around 5,000 deployed devices will be achieved in M27 (December 2012). Once the Phase 1 is stabilized and the services selected jointly by the Santander City Authorities, citizens and the SmartSantander consortium deployed, the second call for tenders will be launched. In this phase, it is also expected that other sites such Belgrade, Guilford, Lübeck and Melbourne would have initiated their own deployment.

- Phase 3 in which around 20,000 deployed devices will be available at the end of the project (August 2013). Once more, the same procedure followed in the previous three phases will be carried out.

Similar to the previous roadmap, the architectural specification and development work also has been conceived according to an iterative process with the following stages:

- In Phases 0 and 1, as it will be introduced in the following chapters (see Chapter 4), from the list of requirements that have been identified for the SmartSantander experimentation facility, a subset of

them has been identified as being the basic ones that the SmartSantander system must fulfill from its very baseline. Although the first SmartSantander architecture specification (see Chapter 5) has been designed taking into account all the requirements identified, particular care has been taken for addressing this subset. Additionally, another particularity of the SmartSantander system is that it is leveraging components that have been already specified on previous IoT related projects, namely WISEBED, SENSEI and USN. A key aspect of the first iteration in the SmartSantander architecture definition has been firstly the identification of the existing components that would fulfill the requirements and secondly the orchestration of the different functionalities provided by the aforementioned components within an integrated framework.

- In Phase 2, there will be mainly two aspects to be considered from an architectural standpoint. Firstly, refine the architecture specification so that functional and non-functional requirements that have been already identified but were not mandatory to be fulfilled during Phases 0 and 1, are addressed. And secondly, to extend the current SmartSantander system architecture with those aspects that might have been unleashed during the development of functionalities carried out in the first phase. Hence, while the focal point in Phases 0 and 1 is the integration of SmartSantander baseline from previously existing components, in Phase 2, new functionalities not yet covered within the state of the art must be introduced and seamlessly orchestrated into the already existing framework.

- Finally, in Phase 3, subsequent refinement will be carried out based on the experiences gained during previous phase and on the enhancement queries that might have come from the experimenters that had applied to any of the Open Calls that the project had opened during the first two phases.

As it has been explained, there is an intermediate step, which deals with the specification and implementation of these building blocks, between the architectural definition and the deployment of the system. The architecture that is presented in this deliverable is intended to provide the framework and to identify the main interfaces and building blocks that the SmartSantander system should have. However, the same phased approach will be used to avoid a monolithic architecture not able to adapt to the additive developments that will be carried out during the whole project duration.

4. DETAILED REQUIREMENTS FOR SMARTSANTANDER PHASE 0

This section lists functional and non-functional requirements for the SmartSantander test facility based on the use cases defined in WP1-T1.1 and WP4-T4.1 which have been prioritized for phase 0. These requirements form the foundation of the architecture specification work carried out in WP1-T1.3 and the detailed specification of components carried out in WP2. During the project we will also re-evaluate the requirements in the light of new use cases that may emerge from consultations with the research/FIRE community, advisory board and lessons learned from implementation and experiments.

For a complete list of all requirements derived so far please see section 9.1.

Functional Module	ID	Title	Text

Functional Module	ID	Title	Text
AAA	FR001	Authentication	When the researcher provides his personal credentials to the SmartSantander experimental facility, it must authenticate the researcher.
	FR022	User account management.	At any time the SmartSantander experimental facility shall enable the administrator to grant and revoke user access privileges.
	FR030	Session Management	The user must log in just one time in order to be able to access to the SmartSantander facility and begin a session. Once created a valid session the rights may remain until user logout or session timeout is reached. During this period there will be no need to type again user or password to access any of the SmartSantander facilities.
	FR097	Authorization	When a user wants to execute any action the system has to verify that he is authorized to do this action.
Configuration Management	FR002	Experiment configuration	When the researcher uses the SmartSantander experimental facility, the system must provide a mechanism to specify and configure experiments to be carried out on the available testbeds.
	FR005	Experiment deployment	When the researcher uses the SmartSantander experimental facility, the system must allow him to upload sensor node program code, which is then distributed to the reserved set of nodes within the testbeds according to the experiment configuration.
	FR060	Experiment configuration details	To specify his experiment, a FIRE user shall use a web-based client to upload to his working directory and manage the sensor code images he intends to load to the sensor nodes. The system shall allow the FIRE user to specify along with the compiled sensor code image (i.e. the filename and upload directory) additional experiment details such as the image name, a version number, the platform on which the image runs, and a short description about the image the FIRE user shall provide additional details. For platform specification, the user shall be provided with a list of supported sensor node platforms and operating systems.
IoT Node Deployment	NFR0025	Physical Access Restriction	Physical access to the deployed sensor node platform shall not be easy.

Functional Module	ID	Title	Text
	NFR002	OTAP	In order to be FIRE compliant, an important percentage of the nodes, being part of the platform, have to support OTAP.
	NFR004	Maintenance minimization	The Smart Santander experimental facility shall require minimum maintenance (e.g. replacing batteries).
	NFR005	Real Field Conditions Operativeness	The platform must be operative in real field conditions (indoors and outdoors) providing support to both FIRE researchers AND inhabitants.
Reservation	FR010	Testbed reservation	When the researcher starts an experiment on one or more testbeds, the SmartSantander experimental facility must force him to make a reservation for the testbeds he wants to use before he can start the experiments.
Resource Discovery	FR008	Testbed overview	When the researcher uses the SmartSantander experimental facility, the system must provide an overview of the testbeds that are available for experimentation including the nodes' location and capabilities.
Resource Management	FR047	Experiment fallback	The SmartSantander experimentation facility provides to the researcher the ability to reset the entire sensor network testbed to the initially chosen default settings in case of node failures.
	FR054	Remote access	System administrator must be able to remotely access to the gateways in order to perform issues such configuration, update, etc.
	FR129	HW/SW Inventory Database	The system shall support an inventory database where it shall keep track of at least the following: all devices in the network and their position, device vendor, services they are being used for and functionalities/information they can provide, their hardware configuration, release of the software installed on each of them.
Scheduling	FR003	Experiment control	When the researcher uses the SmartSantander experimental facility, the system must provide a mechanism to control (e.g. start, stop) experiments the researcher plans to carry out on the available testbeds.
	FR026	Experiment execution batch mode	Researchers shall be able to run experiments in a batch mode (for example, repeat an experiment several times, change parameters for each execution). They shall receive a handle to

Functional Module	ID	Title	Text
			interact with the experiment when it's running.
User Interface	NFR029	Easy to use user interface	The user must find the user interface easy to use.

Table 1. List of functional and non-functional requirements

5. SMARTSANTANDER ARCHITECTURE SPECIFICATION

5.1. High level overview

The SmartSantander experimental facility has to satisfy a variety of requirements from different stakeholders perspectives, ranging from experimenters, facility providers, service providers to system end-users. The set of functionality can be grouped into four system aspects, leading to the definition of the following subsystems:

- Authentication, Authorisation and Accounting (AAA) Subsystem
- Testbed management subsystem
- Experimental support subsystem
- Application support subsystem

Each of these subsystems exposes its functions via service interfaces, realised by a set of well defined APIs:

- Access Control Interface (ACI)
- Experimental Support Interface (ESI)
- Application support interface (ASI)
- Management support interface (MSI)

The testbed functions will operate across the different logical testbed nodes of different characteristics and capabilities. These are:

- IoT nodes
- Gateway nodes
- Testbed server nodes

Each of the nodes provides different physical characteristics and capabilities tailored to its logical role and will host a subset of the previously defined system functions. Figure 3 provides a high level view of the SmartSantander system architecture, detailing the functional decomposition of the above-described

subsystem. The architecture view shows how those functions are spread across the different types of testbed nodes that we envision the infrastructure will be based on.

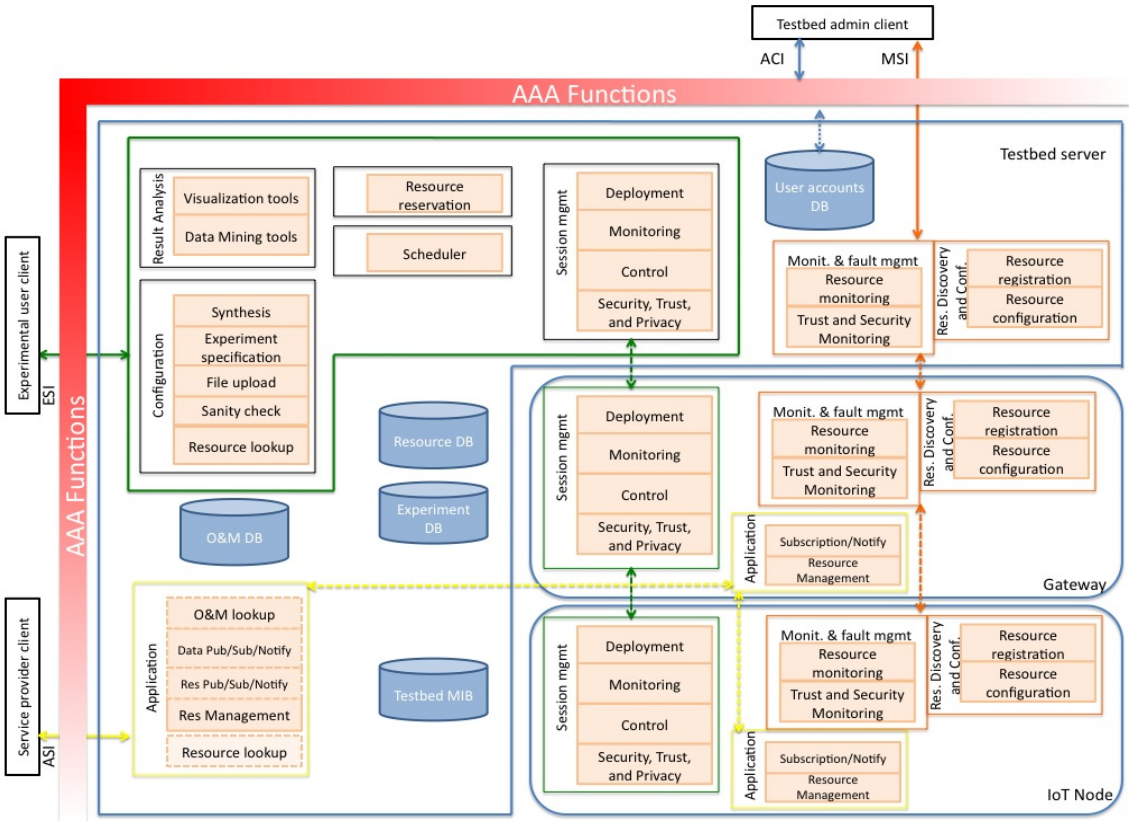

Figure 3. High level view of the SmartSantander system architecture

5.2. Basic Components

5.2.1. Access control and IoT node security

Access control provides the functionality to enforce access rules on the testbed resources in such a way that:

- the person attempting an access is identified and authenticated,
- the role of the person is identified,
- Accessible functions are provided according to the identified role.

Beyond the access control, security should also be provided to physically protect the IoT devices that are exposed in open public places.

5.2.1.1. Access control

Access control is meant to ensure that only authorized actions are performed on WSN testbeds. Individuals accessing the testbed must be identified and authenticated, and their role must be identified.

The following roles are identified:

- **Testbed administrator:** A person who is allowed to manage a testbed.
- **Experimenter:** A person who performs experiments on testbed(s).
- **Service provider:** A person who manages testbed services.
- **Citizen:** A person who uses testbed services.

When several testbeds are deployed in different cities, it is expected that it will be possible to federate them. Then, experimenters belonging to different research organizations will have the possibility to carry out experiments in any testbed belonging to the federation.

A federation aware AAA architecture will be deployed in order to address possible scalability issues in this context. In particular, experimenter authentication will be delegated to each research organization. Such architecture will leverage trust relationships between testbed providers and research entities.

This federation aware AAA architecture will likely be based on the Shibboleth software package, in order to leverage WISEBED assets. However, alternative solutions will also be considered as candidates, as the web oriented nature of the Shibboleth solution makes it cumbersome in some contexts (e.g. for scripting).

In practice, access to testbed will typically use remote web access.

Access control enforcement assumes that PKI based architecture is deployed. Typically Transport Layer Security (TLS) and HTTP Secure protocols can be leveraged for access control enforcement.

Please note that access control is also a way to protect WSN resources (e.g. databases) against attackers.

5.2.1.2. IoT node security

The SmartSantander testbed will be mainly deployed outdoors, in open environments, and thus countermeasures should be taken in order to provide some level of protection against possible attacks. Furthermore, IoT devices typically communicate between themselves by means of radio communications and are thus susceptible to attacks over the air (OTA), even without physical access to the IoT devices. Both types of attacks should be considered in terms of security. In case of the OTA attacks, the following threats and countermeasures are identified:

Actuator command mangling: An attacker may send forged commands towards actuators, or mangle commands sent toward actuators.

Countermeasures: hash controls or signature control.

Forging of sensor measurements: An attacker may send forged measurements reports or mangle measurement reports sent towards gateways.

Countermeasures: hash controls or signature control.

Sensor eavesdropping (privacy enforcement): An attacker may spy confidential information managed within WSNs.

Countermeasures: encryption.

IoT device compromising through OTA reprogramming attack: An attacker may use OTA reprogramming capabilities in order to mangle or to forge software images sent to IoT devices, and jeopardize trust of the WSN.

Countermeasures: secure OTA reprogramming.

Denial of Service: An attacker may prevent WSNs to provide services by means of:

- Radio jamming attacks,
- Protocol attacks through non legitimate IoT devices or compromised devices.

Countermeasures: each DoS type require specific countermeasure.

The same threats also apply to physical attacks, but in addition new threats should be considered as well, such as:

- IoT device compromising through physical attack,
- Vandalism,
- local modification of environment.

Defense against the physical attacks typically requires specific hardware features for IoT devices, such as secure bootstrapping, possibility to disable JTAG, non writable or non readable registers for key stores, and so on. Ultimately, “Trusted Computing” compatible platforms could be an option; however, they will not be available for the low power consumption IoT devices in the timeframe of the project. Therefore, they will not be considered further.

A special care will be taken of the IoT devices with more restricted resources, for which security built in features may be limited. For example, the choice of cryptographic algorithms may be driven by such hardware constraint (e.g. PKI versus symmetric only cryptography).

It should be noted that not all security threats and countermeasures are relevant from the architecture point of view. Experimenters will have the capability to upload software images to IOT devices, so that some security aspects will be left to authorized experimenters without interference from the testbed platform. However, several important security aspects remain under the architecture foundation scope, such as:

- Secure OTA reprogramming,
- Defence against physical attacks toward IoT devices.

In particular, a fall-back mechanism should be proposed in order to securely reset IoT devices into a valid state that allows radio communication with the devices and proper administration of the WSNs.

Commonly deployed security architectures are not appropriate to the SmartSantander network. The scale and heterogeneity of the network as well as the limitation of available resources in the deployed sensors lead to the consideration of a new framework of IoT security architecture.

To reconcile the security needs of the deployed applications with the constraints of the sensor network, 4 levels of security keys are envisaged:

- The “personal” key
- The “pairwise” key
- The “cluster” key
- The “global” key.

The various levels of the secret keys may use symmetric key cryptography shared between several nodes for the low levels of security, and asymmetric key cryptography to secure the personal link between each node and the server for the high level of security (see Figure 4). The proposed IoT security architecture should provide efficient solution with low “overhead”, low complexity and low memory requirements.

The investigated solutions address the problem of the security deployment strategy, including the authentication of the entities, the pre-distribution of keys, the storage of the secret, the management of the keys at the different levels, and the pertinence of the use of asymmetric or symmetric cryptography ; as well as the maintenance of the security during the run time, including the renewal of the keys, the security protocol during the maintenance of one node, the task of reprogramming the nodes, the revocation of a corrupted node. All these features will contribute to making the security scheme robust to attacks.

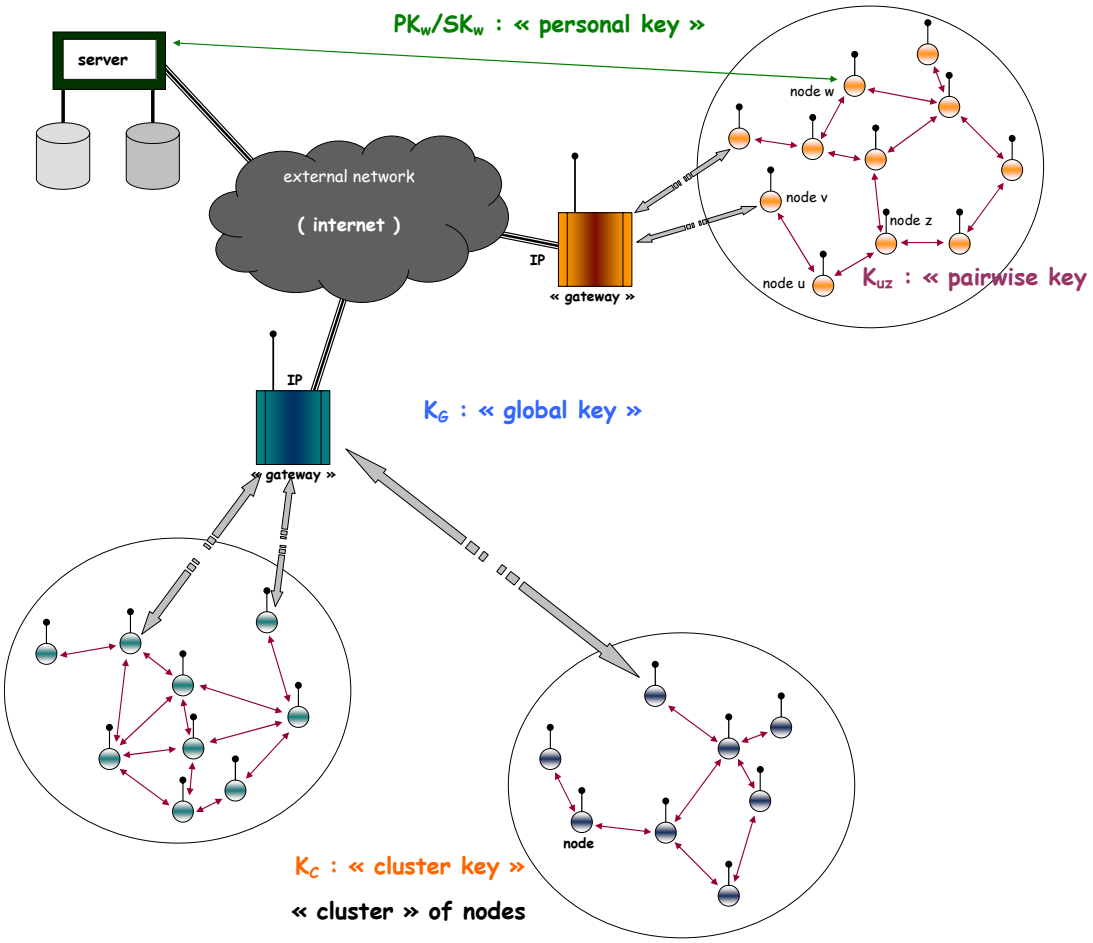

Figure 4. Overview of the IoT security architecture proposed for the SmartSantander WSN

5.2.2. Experimental Support Subsystem

5.2.2.1. Configuration Management

Configuration Management (CM) combines the management and control of all changes to the hardware and software of a system throughout its lifecycle. The main issue addressed by the CM is reproduction of a certain system state. The CM module of the Experimentation Support Subsystem (ESS) addresses this issue for the configuration of resources and experiments.

The CM module is a cross-sectional part of the ESS which itself is part of the testbed server software. Resource and experiment configurations are stored and loaded in specific repositories (Resource DB and Experiment DB).

Configuration with the CM module provides the following four main functions:

- Identification: identification of components making them unique and accessible in some form (resource lookup and discovery)

- Control: controlling the changes of the system during its lifecycle (experiment specification, automatic resource detection)
- Status Accounting: recording and reporting (e.g. sensor data, statistics and errors)
- Audit and review: validating the completeness and consistency among the components (monitoring, defect detection and logging).

Figure 5 shows the functional blocks of the CM module. These are used for the configuration of experiments and resources.

5.2.2.2. Resource Configuration

Resource configuration encompasses the specification and detailed description of the following testbed resources:

- Testbeds
- Gateways
- Sensor Nodes.

Testbeds can be configured as federated testbeds, i.e. testbeds that consist of other testbeds. The testbed configuration summarizes the capabilities of its nodes and the testbed's location. Additionally, one can create any custom topology by leveraging virtual link configurations.

Gateways can be configured for tasks such as data aggregation, detection of node mobility (nodes can be added, removed or updated), and tracking of node defects.

Testbed Nodes can be configured for self-discovery and registration. They can provide general hardware information, sensors, capabilities, and their status (e.g. battery power, etc.).

5.2.2.3. Experiment Configuration

The experiment configuration consists of the specification of the experiment's topological layout (node topology and optionally node virtualization). This strongly depends on prior knowledge about the testbed resources and node connectivity, and requires a valid resource configuration. The experiment specification also consists of the experimental details (software image, target sensor platform), experiment results logging information, and error tracking information (e.g. to which host the sensor readings, experimental results and debug information should be redirected).

Node applications can be uploaded as application image files. When the image is uploaded to the testbed management system, a sanity check is performed first to verify if the uploaded application image file matches the current configuration, answering the question: can this source code be executed on the specified node?

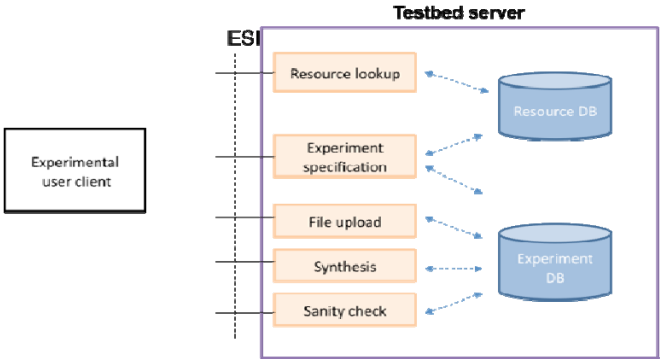

Figure 5. Configuration Management

5.2.2.4. Resource reservation and scheduling

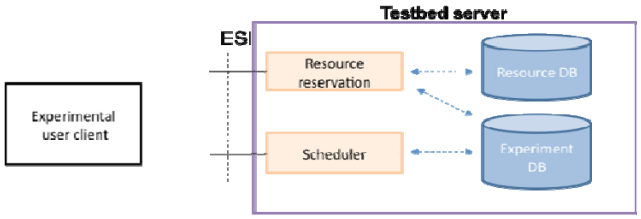

Figure 6. Resource reservation and scheduling functionalities

The Resource reservation and Scheduler modules are in charge of updating the Resource DB and Experiment DB with information required for making a reservation for running a particular experiment.

The Resource reservation module interacts with the Resource DB for changing the status of the available resources from *free* status to *busy* and for associating resources with the corresponding experiments.

When a user wants to run a new experiment, the resources that are not being used in other experiments at present can be easily determined consulting the information available in the Resource DB.

However, busy resources may be reassigned to a different experiment when the priority of the current experiment is lower than the priority of the experiment that is being planned. So, the resources keep the same status, but the owner/experiment changes. In this case, the system shall notify the former owner about the lost resources from his reservation and shall suggest alternative free resources that still meet his experiment requirements.

The resource reservation module also interacts with the experiment DB for filling in all the information about the new experiment. In this DB, relationships between resources, ongoing experiments and proprietor users, in other words, the reservation data is included.

Finally, the scheduler module interacts with the experiment DB for changing or consulting experiments scheduling. When a user makes a reservation, he can modify, in any moment, information about the experiment, such as execution times, experiment start and stop times, necessary resources, experiment priority, sensor node codes and so on.

5.2.2.5. Result Analysis

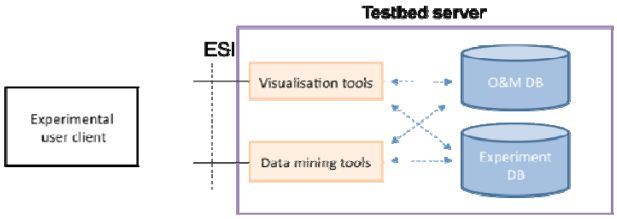

Figure 7. Results analysis tools

The "Data Mining Tools" and "Visualization Tools" modules are in charge of providing functionality concerning analysis and visualization of the data, measurements, network status of the conducted experiments either in real time or offline, after the execution. These two modules act as a middle layer between the "Experimental User Client" and "O&M DB" and "Experiment DB". The "O&M DB" stores the observations (current status of the testbed, IoT nodes status information, gateways status information) and the measurements (collected values from sensors, network statistics (eg. packet loss etc.) concerning a conducted experiment. The "Experiment DB" stores descriptions of the ongoing experiments. All these data can be analyzed or visualized before it's delivered to the Experimental user Client.

The "Visualization Tools" module is responsible to "render" both the observations and measurements or the analysis results on them, through a wide variation of visualization elements like charts (lines, bars, dots etc), pies, surfaces, volumes, scatter, dial graphs and maps. This tool may be capable of depicting the spatial and temporal variation of information delivered and capable of "zooming" (filter in/out) spatial areas or temporal intervals of interest. For example, the researcher should be able to observe the measurements upon a map or to filter data from a specific sub-area for a specific time interval of the experiment. Finally, this module may "playback" the visualization of data of interest in a depicting element through time.

The "Data Mining Tools" module is responsible for providing analysis functionality concerning the observations and measurements of the experiments. The researchers will be capable to select required features from the dataset and to clip some of them in ranges (for example in time or in space) through a predefined set of data mining and statistics methods (like histograms, regression, clustering – group or hierarchical, Nearest Neighbour queries, Decision Trees and machine learning methods). The results of these methods could be dispatched either to the researcher directly or to the "Visualization Tools" module for rendering them in a visualization element. All these functionalities can be exposed by OLAP sub module for online analytical processing.

5.2.2.6. Session management

The Session Management module (hereafter, Session Manager) keeps track of the stateful interactions between each user (FIRE experimenter or application-provider) and the SmartSantander facility. A session is created as soon as a user reclaims his reserved resources i.e. the session manager realizes the topology specified in the user's experiment configuration using virtualized links and mobility emulation if necessary and returns a control endpoint through which the user may manipulate the session. If any of the reserved nodes are not available due to failure, the session manager may re-select an alternative node with similar properties.

As shown in Figure 8, below, the session manager's operations are available via the Experiment Support Interface (ESI). ESI provides operations for the user to deploy execute and monitor his experiment/application. In more details, within a session a user deploys his code on the IoT nodes as per his reservation, and is provided with operations to start/stop and control the execution of his application/experiment. Operations are also provided to reset or re-flash sensor nodes (or reconfigure gateway nodes) within the session, should some nodes fail to start the experiment/application. During the experiment, the Session Manager reports to the Experimental User Client (See Figure 8), as specified by the user in his experiment configuration, sensor readings and/or event notifications, and if necessary debug information. Further, the session manager allows users to inject events artificially or to execute commands/scripts in their reserved network during their experiment execution.

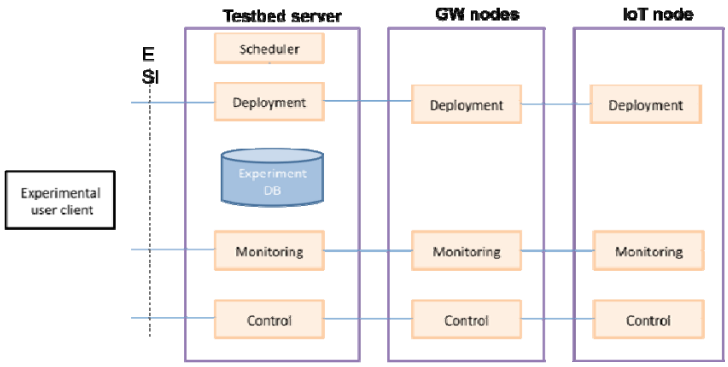

Figure 8. Session Management

5.2.3. Management Support Subsystem

5.2.3.1. Resource discovery and configuration

Resource discovery and configuration (see Figure 9) represent the functionalities that the Management Support System should provide in order to manage the addition of new resources (GW or IoT nodes) to an existing deployment. The objective of these functions is to provide a plug and play configuration experience to the testbed operator (system administrator). In this way, when an already known node but currently disconnected or a brand new node or a new gateway are added to the existing testbed, the only action required from the system administrator should be to physically connect and power-up the new testbed resource. When one of the above events occurs, the topology description of the testbed needs to be updated accordingly as well as characteristics of the GW/IoT node need to be advertised to the system database, in order to be accessible for further experimentation. This requires the implementation on the IoT node of a Resource Registration module that has to cope with its counterpart implemented on the GW to which the node is connected. Once connected, and by means of a discovery protocol, implemented by its Resource Registration module, the new node advertises its offered features (i.e., temperature, humidity, localization,

hardware etc). The GW collects this information and it can both, store them in a local Resource DB (using a given description format), along with a unique identifier for the node (e.g., its MAC address or serial ID) or send them to the testbed server, which will store the information in a global Resource DB. For this reason, as shown in Figure 9, a Resource Registration module interfacing with its counterpart on the GW and with a global Resource DB needs also to be implemented on the testbed server. In the same way of an IoT node, a gateway should be able to advertise its presence and properties and store them in a Resource DB by means of the Resource Registration module running on the testbed server.

After this sequence of actions is executed by the system, the new hardware is registered along with its status and other relevant information within the testbed framework.

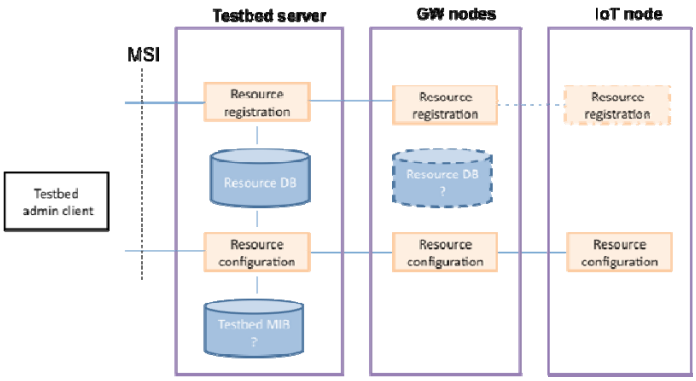

Figure 9. Resource discovery and configuration

Once registered in the resource directory (represented by the Resource DB), the testbed resources become discoverable. This allows the resource users and other testbed functions (e.g testbed reservation or application service) to discover, select and use them. For this reason, the Resource Configuration functionality, by means of which some configuration parameters flow in the opposite direction, from the Testbed server to the GW and IoT nodes needs to be provided. This functionality permits to reach a given GW or nodes and send them the required configuration parameters that can be provided at experiment-time. This information can be related to a required topology/network configuration, other node parameters, security aspects etc. In order to achieve this communication, an interaction with a Testbed MIB, containing among others, also the information needed for reaching a given resource (GW or IoT nodes) is required by the Resource Configuration module implemented on the testbed server.

5.2.3.2. Monitoring and fault management

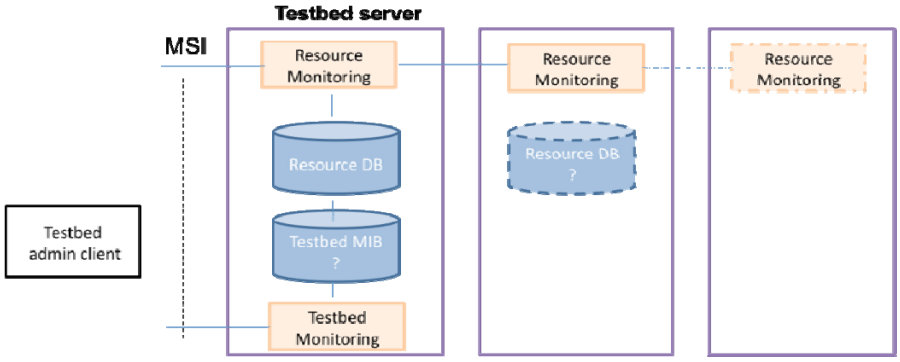

Figure 10. Resource and testbed monitoring functionalities

The resource and testbed monitoring modules are mainly in charge of monitoring resources availability and type, at the three blocks that composed the architecture: Testbed server, Gateway and IoT Node. For each of these blocks, the communication takes place in the following way:

- Testbed server: Once an experiment is running, resource monitoring allows the user to access the resource database in order to monitor the resources currently in use and those available for reservation, in order to carry out an experiment. On the other hand, testbed monitoring module interacts with the Testbed MIB, which retrieves the state of testbed nodes in charge of providing connectivity; such as gateways, routers, repeaters, and functionality offered by testbed to develop the corresponding experiments.
  The Monitoring and fault management subsystem (through resource and testbed monitoring) communicates with the Testbed admin client using the Management Support Interface (MSI), in order to inform about the state and availability of resources, as well as to know the functionality capacity offered by the corresponding testbed.

- Gateway: As entity in charge of a group of IoT nodes, the resource monitoring module offers the capability to access the local Resource database (associated to each GW), in order to monitor all the resources provided by this gateway, thus indicating their main characteristics and their availability degree. In this sense, there is neither testbed monitoring module nor testbed database instance in the gateway, because as part of the testbed, it is not able to offer a global view of the whole testbed.

- IoT Nodes: These nodes are provided with a resource monitoring module instance, so that it accesses the Resource database in order to update the state of the node (resource), indicating its availability and description. The update of this information in the resource database is carried out by the corresponding GW in charge of this IoT node.

5.2.4. Application Support Subsystem

The Application Support System (ASS) is intended to provide the basic functionalities that can facilitate the development of services either for experimentation or final service provisioning.

5.2.4.1. Resource Publish/Subscribe/Notify

This functionality allows users to be notified when either changes to the resource descriptions occurs or when resources matching certain criteria appear or disappear. In order to do that, at the Testbed server there will be a "Resource pub/sub/not" component receiving notifications from IoT nodes and GW nodes, and also checking the Resource DB as shown in the Figure 11.

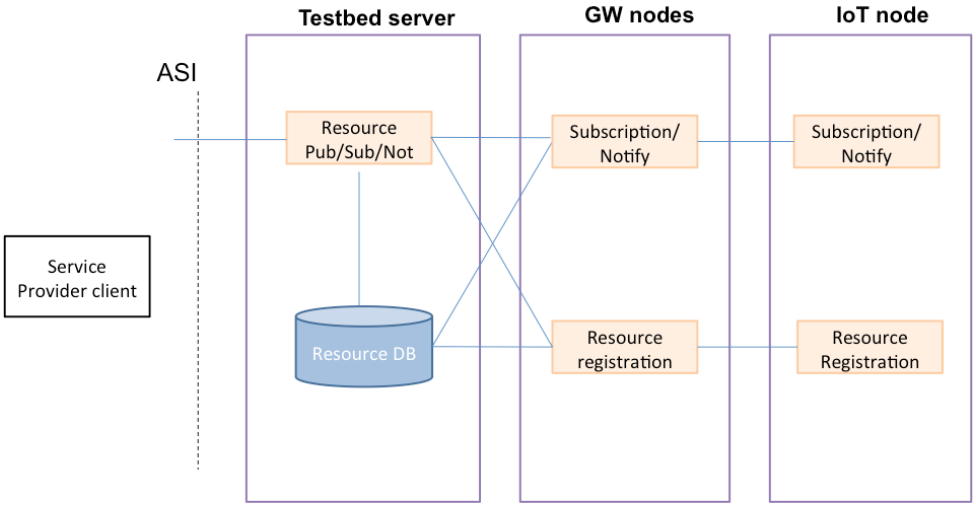

Figure 11. Resource Pub/Sub/Notify functionalities

At the GW nodes and IoT nodes, it is expected that the "Subscription/Notify" and "Resource Registration" modules update the descriptions and register resources respectively when needed.

5.2.4.2. Data Publish/Subscribe/Notify

This functionality will provide publish/subscribe/notify mechanism for applications based on the observations and measurements provided by the resources. The way of working is the following:

- The application will subscribe to the "Data Pub/Sub/Not" component provided by the testbed server expressing filter criteria. This filter can contain simple or complex conditions involving several resources and current or historical measurements.
- The resources in the IoT nodes or IoT Gateways will provide measurements that will be stored in the O&M DB and also sent to the "Data Pub/Sub/Not" module.

– Every time a measurement is provided, it is checked against the filter criteria and if required a notification is sent to the subscribed applications.

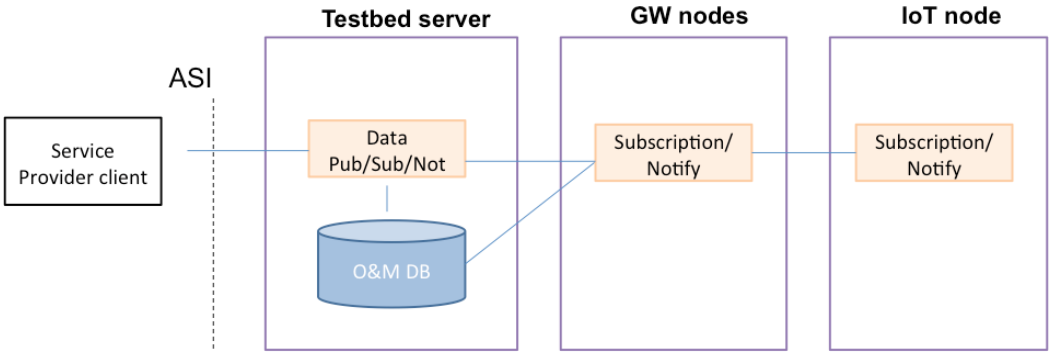

Figure 12. Data Pub/Sub/Notify functionalities

5.2.4.3. O&M Lookup

This allows application to ask for information stored in the O&M DB. This might refer to the current value of an observation or to historical data.

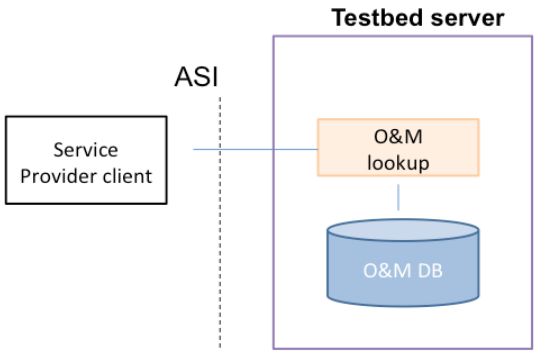

Figure 13. Observation and Measurement lookup

5.2.4.4. Resource Lookup

The Resource lookup function enables applications to search for resources stored in the Resource DB. This allows users to find resources matching certain criteria like location, type of information provided, available functions, IoT node hardware and software capabilities, etc.

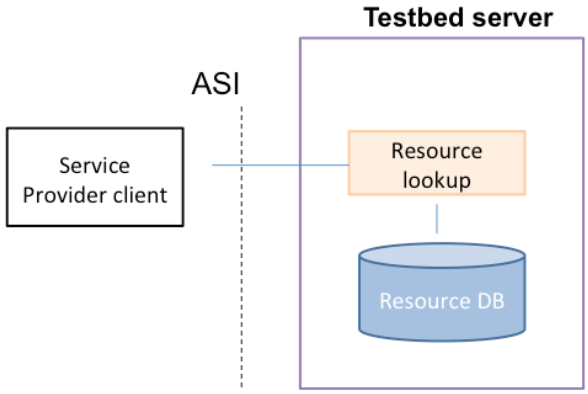

Figure 14. Resource lookup

5.2.4.5. Resource Management

The Resource Management functions allow the Service Provider to add, configure and manage the resources used for services provisioning. When a new resource for service provision is added to the Testbed, it needs first of all to be registered in the Resource DB. A global instance of the Resource DB is stored in the Testbed server and it maintains information related to the resources, their features, the types of supported services, and types of observation and measurement returned, together with the information for accessing them. The Resource Management module located at the testbed server collects the information sent through the Service Provider client and it then stores it in the Resource DB. Moreover, the Resource Management module, through the interaction with its corresponding counterparts located at the GW and IoT nodes, collects and stores in the same Resource DB information related to the resources status, such their availability, their usage degree, etc.

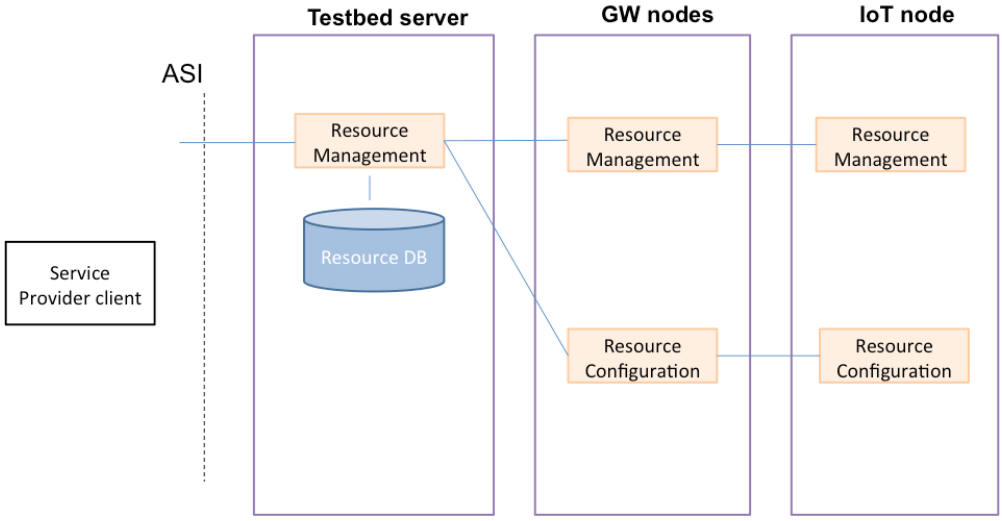

Figure 15. Resource Management

Moreover, when new resources are initially added to the testbed and/or when new service exploiting them are defined, the resources need to be accordingly configured, based on the information stored in the Resource

DB and provided by the Service Provider. This task is accomplished by the Resource Configuration module on the testbed server controlling its counterparts on the GW and IoT nodes, which are in charge to configure the two end points of each resource.

5.3. Required Interfaces

5.3.1. Access Control Interface (ACI)

The *AAA subsystem* comprises functions that control and audit the access to all system functions. The AAA functions are common for all types of the facility users. The AAA subsystem typically represents the first point of contact for users to gain access to other testbed functions. Apart from the management of user accounts, it is the task of the AAA functions to ensure that each user is properly authenticated and receives authorisation to access testbed functions according to its role. The AAA system takes also care that all actions carried out in the system are accounted for according to the agreed usage policies of the testbed.

5.3.2. Experimental Support Interface (ESI)

The *experimental support subsystem* provides all functional features to support the experimental life cycles of testbed users. Many of these functions will be used by experimental testbed users, but also for service providers, as the provision of services can be in essence seen as longer lasting experiments. This includes functions to discovery availability of testbed resources and their characteristics, functions for the configuration of experiments, allocation of testbed resources, control during execution of the experiments and data collection and analysis.

5.3.3. Application support interface (ASI)

The *application support system* provides additional functional features that can be used by application developers and service providers to compose smart services on top of the SmartSantander facility. These are tools for easy discovery of sensor information, effective data management and other support functions that make the development and re-use of sensing and actuation capabilities from the heterogeneous testbed resources easier.

5.3.4. Management support interface (MSI)

The *testbed management subsystem* comprises all functions that are relevant for the seamless operation and management of the testbed. The subsystem mainly caters towards the administrator users and provides many features effectively supporting this role. This includes plug and play configuration of new testbed components, monitoring, performance and fault management.

5.4. Information model

5.4.1. Data stores.

The following data stores will be used:

- Resource database – This database will store descriptions of all resources (IoT nodes, gateways) available at a given moment in the SmartSantander testbed. The resources will be described by XML using appropriate schema (initially, both SENSEI resource descriptions and WISEBED's WiseML descriptions will be supported; in the later phases the SmartSantander description format will be defined). The descriptions will be pushed to the database by the SmartSantander resources whenever they join the testbed and will be kept in the database as long as the resource is available. The database will be searchable by keywords to identify resources required for an experiment or for a service.
- Experiment database – Similar to the Resource DB, this database will store descriptions of the ongoing experiments, reservations and experiments scheduled for deployment. Format of the descriptions will be defined by an appropriate XML schema.
- Testbed MIB – This database will store information about the testbed nodes used to provide connectivity (for example routers or servers) and other functions required for the test bed to function properly.
- User accounts database – In this database information about all registered users will be stored. In addition to the basic information like username and password, information about the user access privileges will be kept as well.
- O&M database – This database will store all the observations and measurements sent by the different IoT nodes, gateways, etc. This will contain live and historic information. The measurements can provide information about the values detected by sensor or the status of all IoT nodes, gateways, etc. Information like the battery status of nodes, applications running by a node. Amount of available memory on each node, the values of counters used to calculate KPIs etc are some of the parameters that could be stored in this database.

5.5. System use cases and interactions

There are four main interfaces in the system, as depicted in Figure 4. The ACI is used prior all interactions, and the other 3 are used depending on the objective of interaction (role of the user or use case). This section will describe the 3 interfaces corresponding to the main use cases (or "conceptual" functions), which are:

- to perform the operations related with the management of the platform, in order to deploy and manage the sensors or IoT nodes integrated or active in the system at each moment (registering/configuration of the devices)
- to receive, store and ultimately process (by end services) the measurements or observations performed by the "low level" sensors present in the system; or to send data or commands from the platform to the IoT nodes (sensors or actuators)

- to configure the execution of an “experiment”, typically involving the management of several physical devices and different “logical” resources in the system

These three functionalities are the main operations that are typically required in order to add sensors/actuators (or other devices such as repeaters, gateways, etc.) to the system, to set up experimental networks, to conduct tests or run any experiment, or to gather and store measurements from sensors (either for experimental or end service purposes).

In order to describe the interaction of involved components of the architecture, from a higher level of abstraction (precise implementation, internals and data/protocol formats are left for other “lower-level” design or implementation documents, such as D2.x deliverables), some “chronological” diagrams and flow charts will be used and explained to describe the main “conceptual” interactions among logical components.

5.5.1. Platform Management

The first group of component interactions that can be highlighted refers to those operations that are necessary for managing the whole experimental facility. As commented before, any “actual” operation or test requires the previous deployment and registration of the devices into the SmartSantander system. Whilst the deployment of the devices is a “physical” (and sometimes involving manual procedures) process, the registration of the device is the first step to include the node in SmartSantander, and typically should occur just after its deployment, being initiated by an Administrator.

5.5.1.1. Adding a device to the facility

Any operation or test requires the previous deployment and registration of the devices into the SmartSantander system. Whilst the deployment of the devices is a physical process involving manual procedures, the registration of the device is the first step to include the node in SmartSantander, and typically should occur just after its deployment, being initiated by an Administrator.

A condensed chart of the operations carried out is presented in Figure 16:

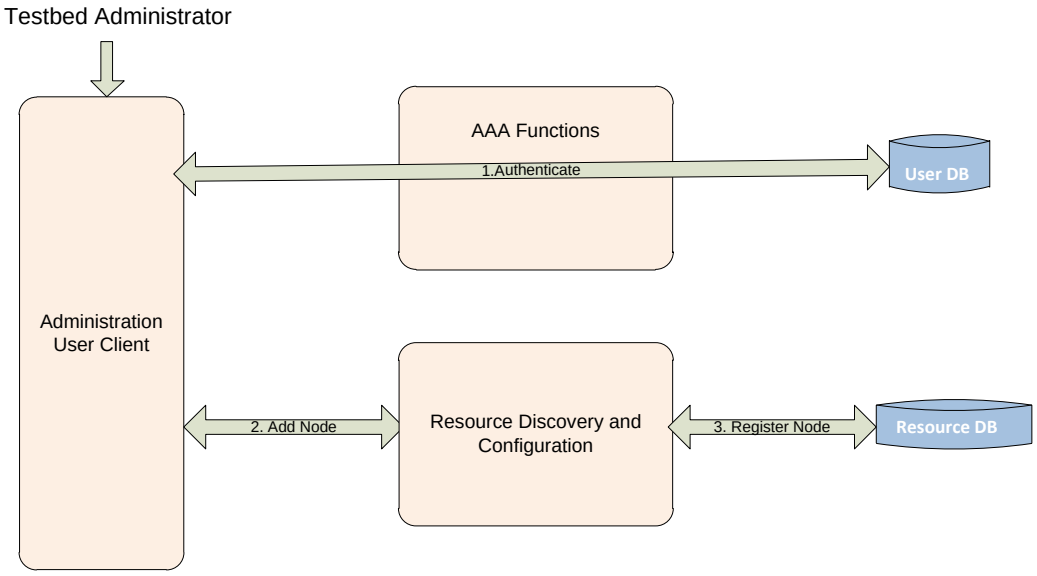

Figure 16. Registration flow diagram

The Administrator triggers the registration process by invoking the mechanism to add a new device to the system. This is done through the Administrator exclusive client, and enabling the administrator to fill in a form or set of fields to describe all the parameters related with the device to register. This set of information is then sent to the Resource Database, which will enable the rest of the architecture to access to this information. As with any other interaction with the SmartSantander system, the Authentication and Authorization steps are the first ones to be carried out.

5.5.1.2. Resource monitoring and configuration

Bearing in mind the large scale of the experimental facility, it is compulsory for the system to have an automatic fault management and system monitoring functionality. Two modes are supported both reacting upon the generation of events that indicate faulty behavior of any of the nodes as well as management and configuration procedures initiated by the Administrator.

Figure 17 below shows these two modes for addressing the resource monitoring and configuration.

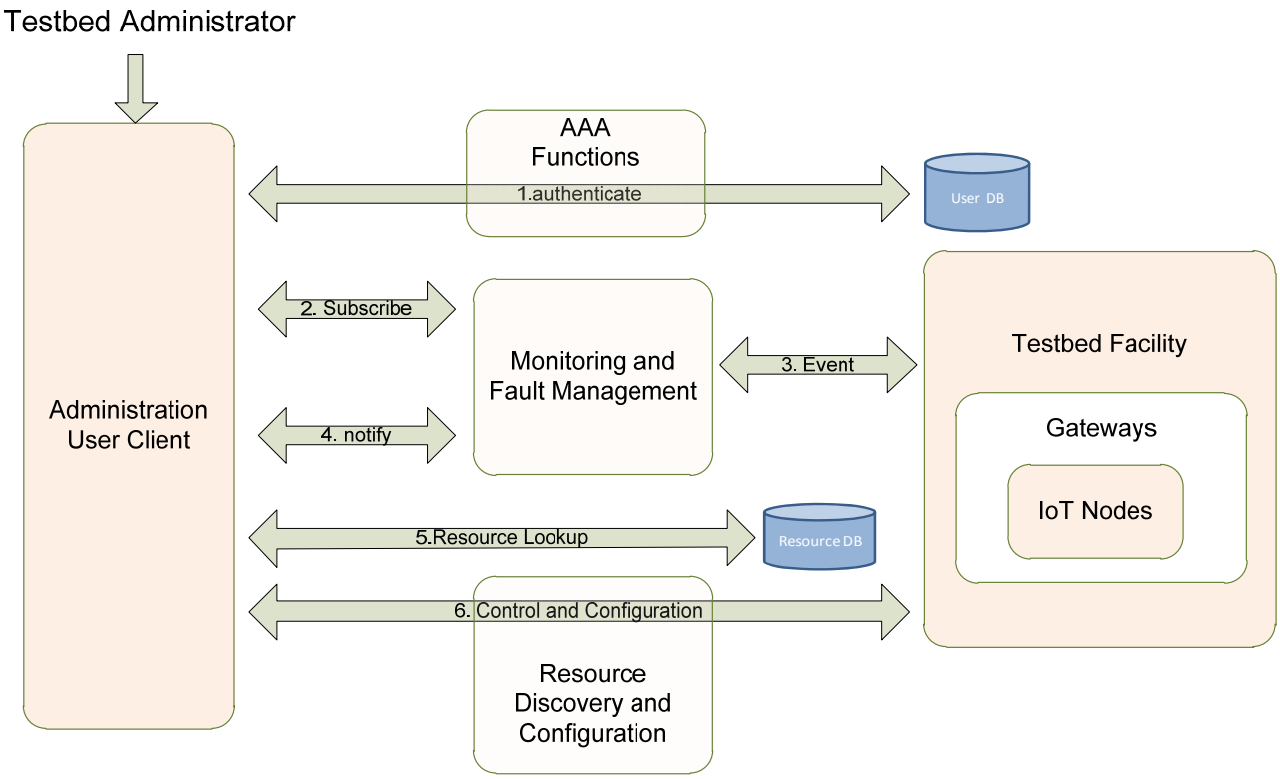

Figure 17. Resource Monitoring and Configuration flow diagram

Similarly to other interactions, the first step is the Authentication and Authorization of the user. Once authenticated, the Administrator will be able to interact with the system both on a subscribe–notify approach as well as on a synchronous manner to directly control and configure the devices on the facility.

As it is shown in Figure 17, the Administrator can subscribe itself through the Administration User Client in order to be aware of any situation for which the system generate an event (e.g. node failure, node battery drain below pre-defined level, etc.). Additionally, it is also possible to interact with the devices directly in order to tune some of its operational parameters. In this latter case, before accessing the devices, the User Client would access the Resource DB to locate the device or devices to be managed and retrieve its current configuration and the way of accessing to them. Afterwards, the SmartSantander system will enable through the Resource Discovery and Configuration block within the Management Support Subsystem the control and configuration of the IoT nodes and Gateways.

5.5.2. Receiving/sending data from/to IoT nodes

Another “elementary operation” for the platform, in order to enable any end-service or test/experiment execution is the retrieval and storage of the parameters or data capture by the IoT nodes (typically sensors), or the submission of data (either response codes, commands to actuators, etc.) to the nodes.

5.5.2.1. Gathering measurements from sensors

The chronological diagram to illustrate the process can be seen in the following figure:

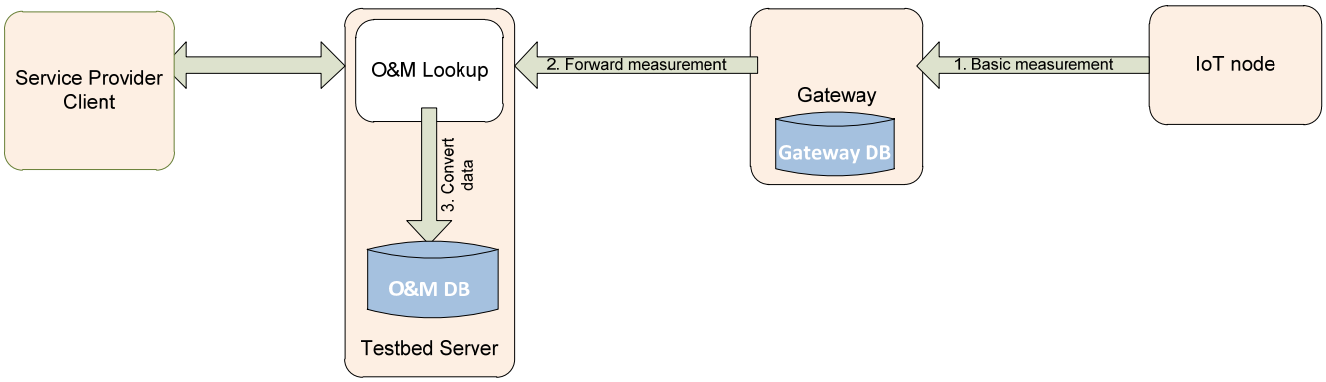

Figure 18. Collection of measurements/observations

Every time an IoT node detects a "data" (or a time-driven event occurs; such a periodic submission of status…), the basic data captured by the sensor is sent to the Gateway nearest to it (probably, through an intermediate Repeater, as shown in Figure 2). The gateway, equipped with some kind of "long-range" IP-based connectivity, will store this data in its internal DB and will forward it to the Portal Server within the Testbed Platform. The Portal Server will include a component in charge of converting the received data (and units) into a format suitable for the USN component also placed within the Testbed Platform and sends the resulting data to the USN component of the Testbed Platform. Once the data from the sensor is stored in the persistent database, it will available for processing, consumption or any operation related with the end service running over SmartSantander's platform.

5.5.2.2. Sending data/commands to nodes

The process of sending any data or commands to the IoT nodes is very similar to the collection of observations/measurements from the nodes, but in the inverse direction.

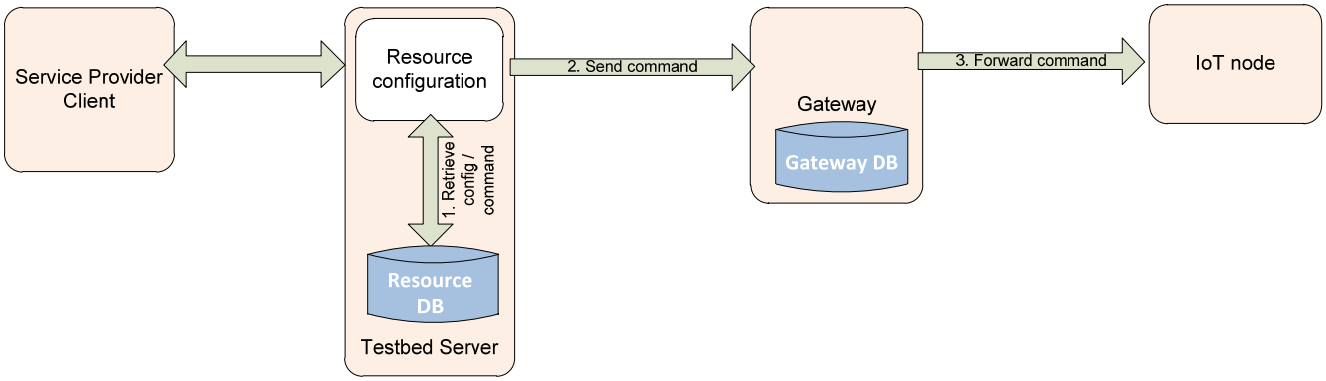

Figure 19. Sending commands to nodes

In this operation, the client requests the submission of any data or command to the IoT node (like in the case of an actuator) through the Testbed Server. In order to complete the operation, it is necessary to retrieve the current configuration of the node from the Resource DB (within the Testbed Server), as well as the commands or instructions (if any) to be sent to the IoT node. Once the required parameters or commands/scripts are ready, are sent to the appropriate Gateway, which will forward the data to the target IoT node.

5.5.3. Configuring an experiment

Several steps and reservation of resources are required for the configuration and programming of an experiment using the SmartSantander platform, as depicted in the diagram:

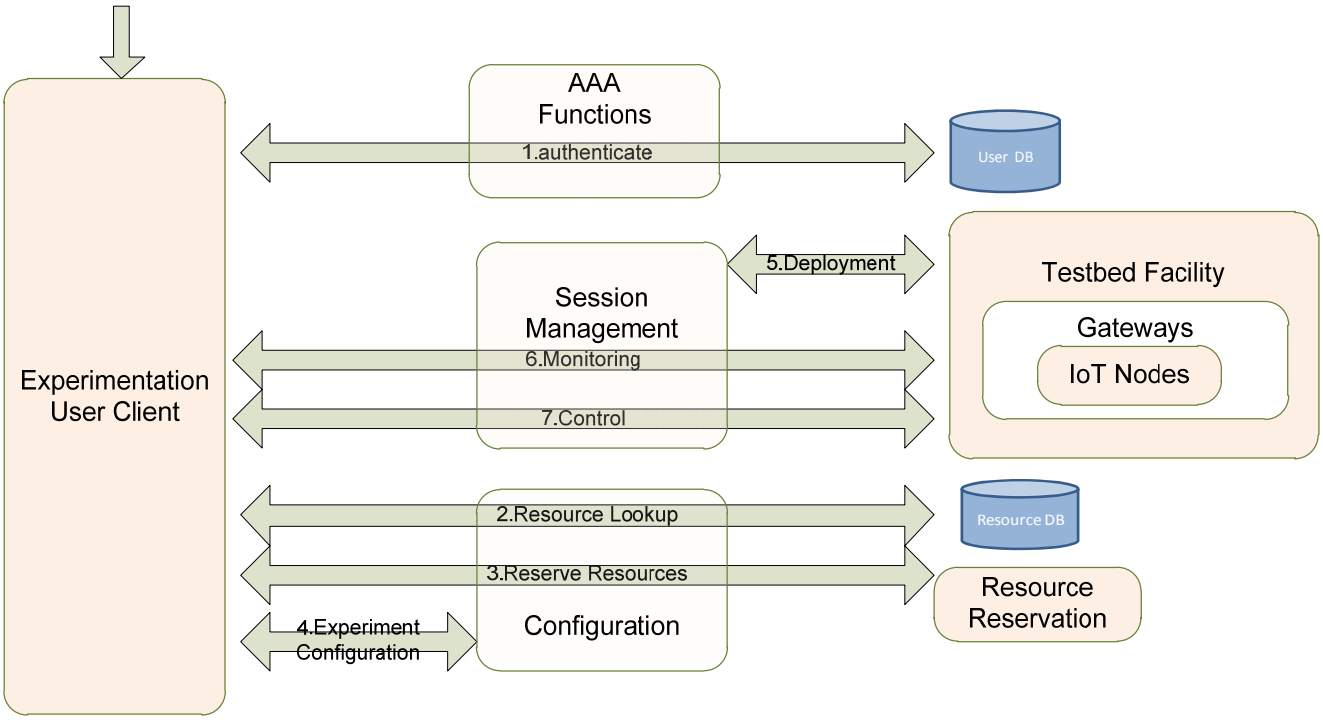

Figure 20. Setting up an experiment

Scientists interested in experimentation, access the SmartSantander facility through the Experimentation User Client. They must be authenticated against the User Database in order to gain rights to access the facility. Then they can configure their experiment: initially is searched for the appropriate resources (selects nodes based on characteristics etc.), then they reserve these nodes (if they are available) for a certain period of time, and set the binaries (code of experiment) that are going to be executed during the experiment and many other parameters. The system schedules the experiment and at the proper time (reservation time) deploys the experiment to the testbed facility and provide access to the experimentation client to monitor and control his experiment.

6. SMARTSANTANDER ARCHITECTURE REALIZATION

6.1. Foundational Components for SmartSantander

6.1.1. WISEBED Components

The WISEBED system is a collection of well-defined APIs and API-clients that allow users to build, deploy and execute repeatable WSN experiments on federated WSN testbeds. These APIs serve to make integration easier, but equally importantly promote modularity and federation of the software infrastructure so that different implementations can be used behind the same APIs and the overall system will still function as intended.

Each testbed is controlled by a **Portal Server** (as shown in Figure 21) which exposes an interface to manage and operate them. The interface consists of the following WISEBED APIs:

- The **Authentication/Authorisation** API, which encompasses secure authorisation, authentication and access control features to protect the testbed from unwarranted access.
- The **Reservation** API, which allows users to reserve the network resources for their experiments.
- The **iWSN** API suite, which includes the **Session Management** and **WSN** APIs, together with the Controller API, allows users/controllers to query the WSN testbed in terms of the sensor capabilities, sensor status and network topology, run the user's experiments on the reserved resources and export experimental results back to the user.

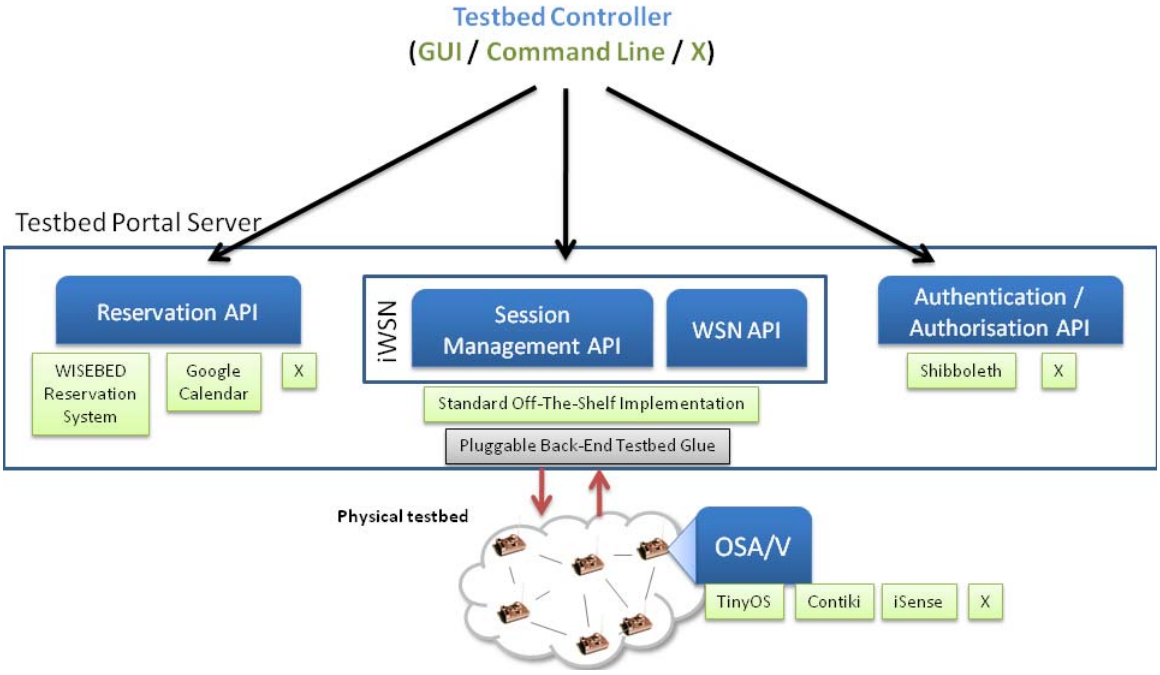

Figure 21. WISEBED APIs

The use of the APIs is illustrated by considering the general flow of control for a user of the WISEBED federation:

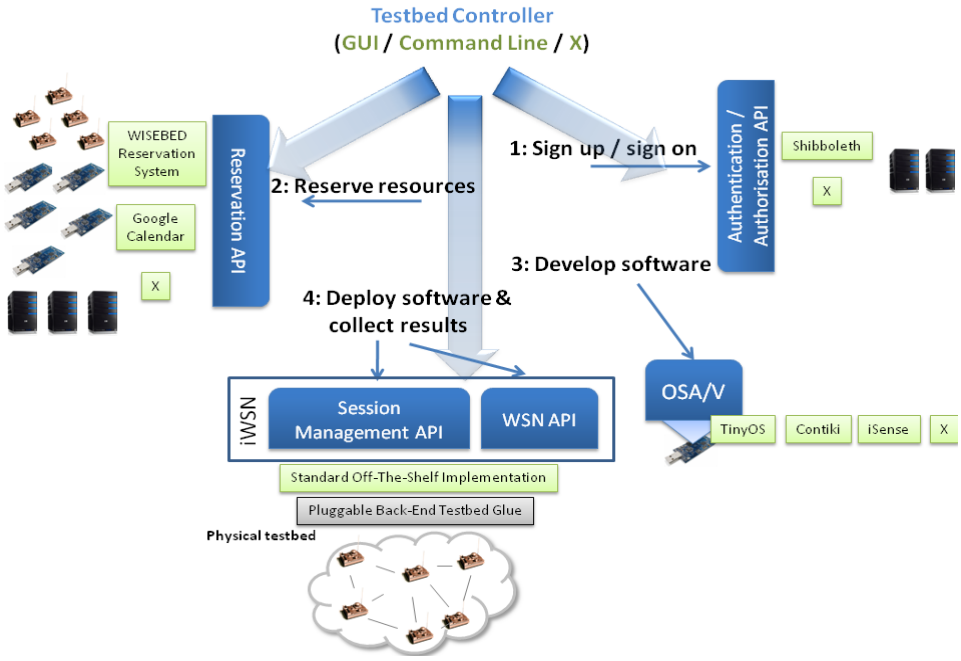

Figure 22. User flow of control in WISEBED

1. Users use a Testbed Controller e.g. a web-based GUI or command-line client to get authorisation to use the federation from and are authenticated by an external security system

2. They then use an external reservation system (again perhaps through a website) which provides that user with access to selected testbed resources at some point in the future and returns to them a 'reservation ID'.

3. Users develop sensor node software for the target sensor platforms. They may use the abstraction of the OSA/V (OS Abstraction and Virtualisation) API to write OS-independent code.

4. When a user's reserved time arrives they – or a system working on their behalf – can use their reservation ID to acquire from the Session Management API implementation, the reference(s) to the testbed service instance(s) (the WSN API instances) at the relevant site(s). They or the controller acting on their behalf, use the WSN API instances to deploy their software images on nodes in their testbed service instance. Experiment progress information and data from sensor nodes are collected and conveyed to the user/controller. The testbed instances are destroyed automatically when their reserved time expires.

A more comprehensive description of the WISEBED suite of APIs can be found in the SmartSantander WP1 Internal Report IR 1.1 available on the project's website. Although, each API engendered a number of implementations across the WISEBED project partners [WISEBED-iWSN-API], only selected implementations were considered as candidate components for realising functions in the SmartSantander System. The selection criteria included the extent the implementations fulfilled their respective APIs as well as the maturity and portability of the implementations. An additional non-negligible criterion for selection of the iWSN implementations is the number of heterogeneous sensor device platforms supported as it has a direct influence on the porting effort required to support sensor device platforms in the SmartSantander infrastructure.

The selected WISEBED API implementations are summarised in Table 2 below.

WISEBED APIs and Libraries	Component
iWSN API (Session Management, WSN and Controller APIs)	Testbed Runtime (Java; supports iSense, TelosB, Pacemate devices) iWSN backend implementation (C implementation; supports TelosB, ISense devices)
	WISEBED Command Line Client (Scriptable BeanShell client for machine-driven authentication, reservation, and experimentation
AAA API	Generic Java component for machine-driven Shibboleth authentication
	Shibboleth SNAA (SNAA authenticating against a Shibboleth federation)

WISEBED APIs and Libraries	Component
	Shibboleth SNAA (SNAA authenticating against a Shibboleth federation)
	Dummy SNAA (always grants access/authorisation for testing purposes)
	Dummy SNAA (always grants access/authorisation for testing purposes)
WSN Reservation API	In-memory RS (provides volatile in-memory storage for reservations, e.g., for labs courses or temporary installations)
	Google Calendar RS (uses a Google Calendar for reservation storage)
	Database RS (using the Java Persistence API with support for all major databases)
	Dummy RS (no functionality, primarily intended for testing or environments without a need for reservation)
	Command Line Client (to start the different RS servers)
OS Abstraction and Virtualisation	OpenCom Software Development Kit
Algorithms	WISELIB (template-based algorithms for multiple purposes)

Table 2. Selected WISEBED API implementations

6.1.2. SENSEI Components

As per SENSEI terminology, the digital world consists of a) resources which are representations of instruments (sensors, actuators, processing elements), b) Entities of Interest (EoI) which are representations of people, places and things and c) Resource Users which represent the physical people or application software that intends to interact with Resources and EoI. SENSEI offers a rendezvous service that allows a user to locate suitable resources based on desirable properties while allowing resources to advertise these properties. To enable this, it provides information models that describe Raw Data, Observation and Measurement (O&M) Data, and Resources (for example, sensor devices) and a number of components for resource registration, discovery and entity binding.

In the following a brief description of SENSEI framework components that fulfil the requirements of the SmartSantander architecture is provided. A more comprehensive inventory of the SENSEI project can be found in SmartSantander WP1 Internal Report IR1.1 on the project's website: **www.smartsantander.eu**.

Resource Directory

The role of the *Resource Directory* is to make the glue between *Resources* who advertise the operations they offer and potential clients that look for particular functionalities. A discovery mechanism typically offers the two following primitives to its clients:

Publish: *Resources* register themselves and advertise their functionalities that are stored in a directory;

Lookup: *Resource Requestors* can send requests to the discovery mechanism in order to get a list of *Resource Providers* that match their requirements. In this case, the discovery mechanism performs a lookup on available directories by matching incoming requests against the functionalities that *Resource Providers* have advertised and outputs the resulting list of *Resources*.

The principles of a typical discovery mechanism are depicted in Figure 23. .

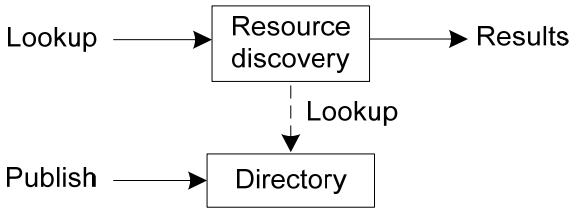

Figure 23. Resource Directory principles

In addition to these functional primitives, the *Resource Directory* mechanism should satisfy a set of security properties to provide adequate guaranties to both resources and users. We group these requirements into two main categories as follows:

- **Compliance with resource security policies**: *Resources* define security policies specifying the set of credentials required to access and update a particular *Resource Description*. These policies are matched against requestors' credentials;
- **Access control:** *Resource Requestors* have to comply with the *Resource Directory* security policies to be granted access to its functionalities.

The Resource Directory component offers the basic functionalities required by a discovery mechanism - that is, registration and lookup - but also includes a notification mechanism to which clients can subscribe to in order to be informed about Resource updates. In addition, access control mechanisms are integrated in the component to enforce security policies specified on the one hand by the Resource Directory administrator to protect the access to the directory and on the other hand by Resource owners so that only authorized requesters can access Resource Descriptions. The Resource Directory is composed of the following modules:

- **Request Handler:** It acts as the interface of the *Resource Directory* for both *Resource Providers* and *Resource Requestors* by means of the lookup and registration handlers. As such, the request handler enforces the access control policies defined by the *Resource Directory* manager for both lookup and registration operations. The security policy enforcement is performed by the *Policy Enforcement Point* (PEP) based on the policies stored in the *Access Policy Repository* (APR);
- **Resource Database:** This is the storage component for *Resource Descriptions*. A *Resource Description* specifies the information required by clients to access the operations offered by a *Resource Provider*;
- **Resource Description**: Each *Resource* available within the SENSEI administrative domain has a corresponding "entry" in the *Resource Directory*;
- **Lookup Manager**: The *Lookup Manager* handles incoming lookup requests issued by authorized clients. A *Resource Lookup Request* consists of a set of tags specifying the functionalities required by the requesting client and of a set of credentials satisfied by the requestor. The lookup process is composed of two steps. The request is first matched against *Resources* tags and a subset of *Resources* functionally matching the request is returned. The security credentials provided by the requestor are then compared to the security policies associated with the functionally matching *Resources*. Finally the set of *Resources* that match both functional and security requirements is returned to the client;
- **Registration manager:** The *Registration Manager* processes registration and update requests issued by authorized *Resource Providers*. Upon registration of a particular *Resource*, the associated *Resource Description* is stored in the *Resource Directory*. The former description should be valid, that is compliant with the *Resource Description* data format. When required, *Resource Descriptions* can be updated by either the *Resource Provider* or any other authorized peers. The security policy required to update a given *Resource Description* is specified in the *Resource Description*. Upon receipt of an update request, credentials provided by the requestor are matched against the latter policy;
- **Notification Manager:** The Notification Manager is in charge of maintaining subscriptions for clients who would like to receive updates when *Resource Descriptions* relevant to their request have been modified, e.g., the addition of a new resource matching the request or the update of a *Resource Description*;
- **Peering Agent:** Several instances of the *Resource Directory* may be deployed over the Internet and this component makes it possible to interconnect these various instances.

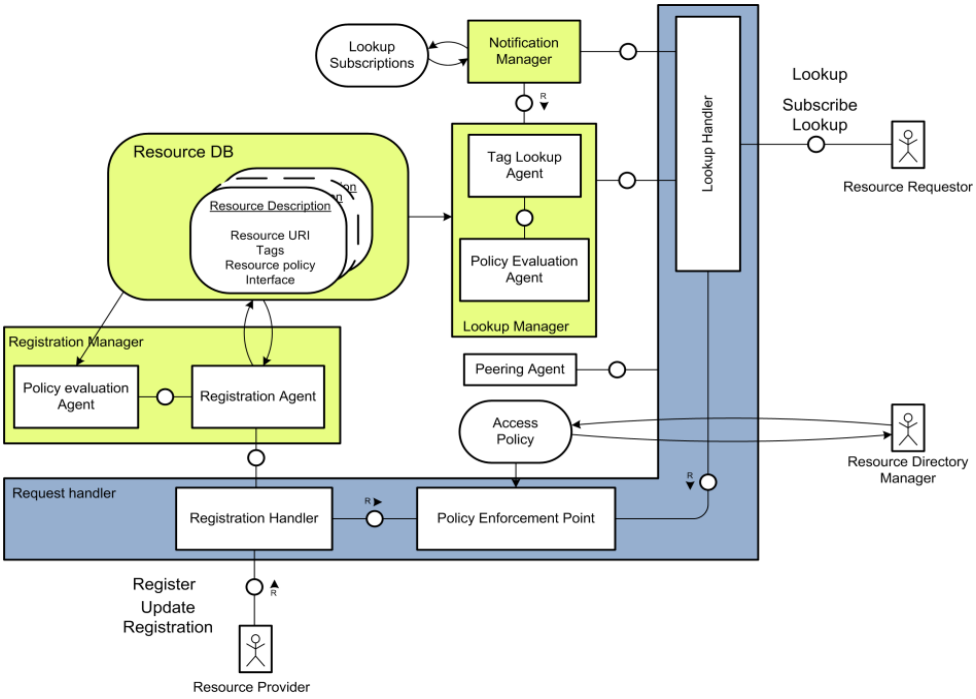

Figure 24. Resource Directory FMC diagram

Entity Directory

The Entity Directory (ED) is the storage place for Entity Bindings that bind an operation of a resource to the attributes of entities of interest. The ED features an interface that enables the management of bindings (storage, update, deletion) and to look up the latter. The ED component consists of a database denoted Mapping Table wherein entity bindings are stored. The database can be modified through the Entity Publication Interface that allows publishers to store new entity bindings, update and delete existing ones. The entity bindings that have been registered can later on be looked up through the Entity Lookup Interface. Finally, an ontology specifying the entity model that defines how classes of entities are linked to each other and their attributes allows handling entities within the lookup processing operation.

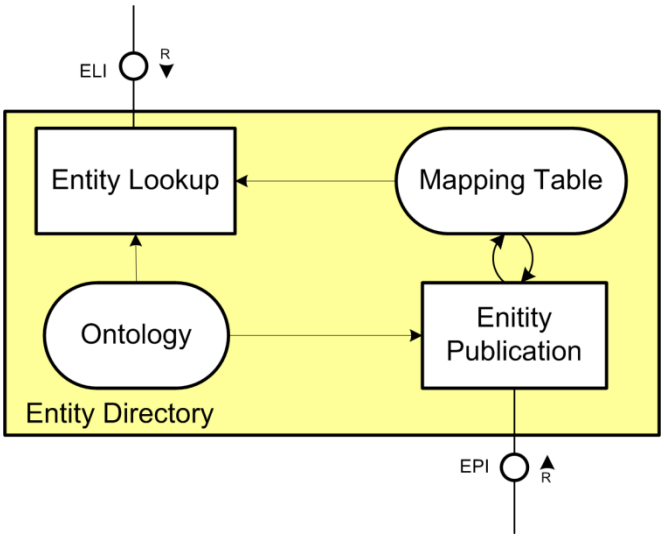

Figure 25. Entity Directory internal architecture

Semantic Query Resolver

The Semantic Query Resolver (SQR) contains the subcomponents Request Analysis, Task Planner, Plan Deployer, and Abstract Plan Data Base. It provides the SQI (Semantic Query Interface) to Resource Users and makes use of Resource Directory's Resource Lookup Interface and Entity Directory's Entity Lookup Interface, EM's Resource Execution Interface as well as Resource Creation Interface Resource Host's can provide.

The SQR acts as a single entry point for Resource Users into the SENSEI system. The SQR provides the Semantic Query Interface (SQI) to Resource Users that offers convenience methods for Resource lookups as well as information and actuation requests. The SQR is able to trigger the creation of processing Resources, which can be available only on demand of Resource Users. If a Resource Template description can be found for a type of Processing Resources, the SQR is able to initiate the creation of an instance of a processing Resource via Resource Creation Interface (RCI).

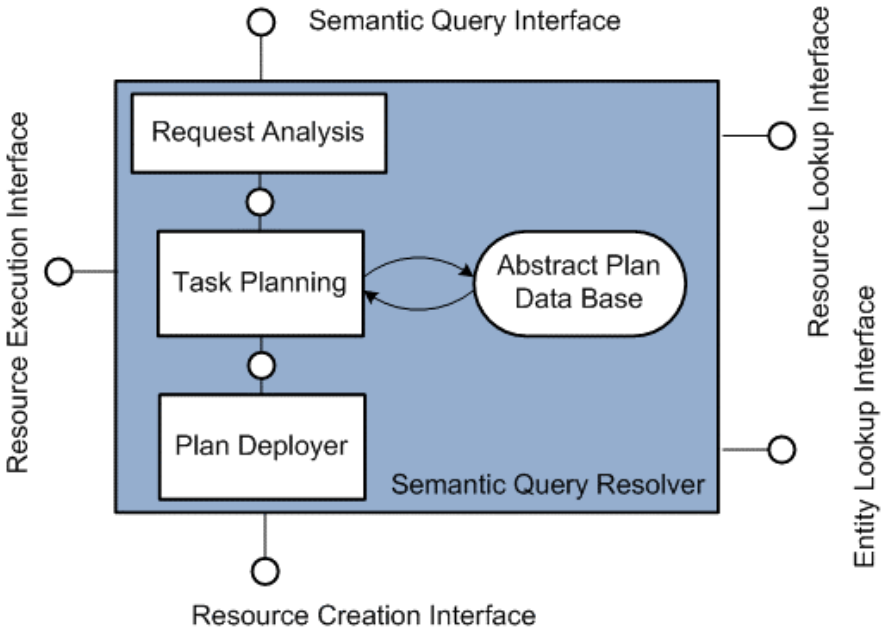

Figure 26. Semantic Query Resolver

The Query Analyser receives the query form the SQI and analyses the parts of the query to determine the intention of the Resource User's request. At first it needs to be distinguished between synchronous one shot requests and asynchronous subscription requests. The SQI offers different operations for either use case. After this analysis, the Resource User's queries are interpreted and need to be translated into a task or a combination of tasks to be executed by the SENSEI components in order to fulfil the RU's request (Task Planning). In the simplest case the Execution Plan consists of one Resource operation only. In the case the Resource User's request cannot be fulfilled by just one Resource the SENSEI Task Planner tries to find combinations of Resources whose interworking would lead to the result the Resource Users is interested in.

The Task Planner would create a composite Resource containing of several atomic Resources, with their respective Resource operations.

The Telco2.0 is composed by two elements, the *USN Platfor*m and the *Open Telefonica Platform*, both required as bases for the SmartSantander platform.

6.1.3. TELCO 2.0 Components

The Telefonica Telco 2.0 Open Platform is composed by two main components

- The Ubiquitous Sensor Network Platform (USN) intended to provide facilities towards the development of services Internet of Things area.
- The Open Telefonica Platform that provides the means to access the capabilities provided by telecom operators, like send SMSs, trigger phone calls, etc.

6.1.3.1. USN Platform

The USN Platform is an end-to-end open platform intended to be used in a broad range of Internet-of-Things application scenarios and services. It provides the following functionalities:

- Sensor Discovery: The platform provides information about the all the registered sensors, allowing efficient look-ups based on the information they provide.
- Observation Storage: A repository where observations or sensors data are stored to allow later retrieval or information extraction.
- Publish-Subscribe-Notify: The platform allows services to subscribe not just to the observations provided by the sensors but also to complex conditions involving also other sensors and previous observations.
- Homogeneous Remote Execution capabilities: This functionality allows executing tasks in the sensor nodes, either to change configuration parameters or to call actuator commands.

The USN platform has two main and interrelated components, the **Sensor Network Gateway Adapter (SNGA)** and the **IoT Enabler**.

IoT-Enabler

IoT-Enabler functionalities can be exposed to applications both through SIP and Web Services interfaces, also supporting synchronous and asynchronous exchanges.

As Figure 27. illustrates, the IoT-Enabler functionalities in the USN Reference Architecture are provided through the following entities:

- The Sensor Description Entity (SDE) is responsible for keeping all the information of the different sensors registered in the USN-Platform. It uses the SensorML language.
- The Observation Storage Entity (OSE) is responsible for storing all the information that the different sensors have provided to the platform. This information is stored using the O&M language.
- The Notification Entity (NE) is the interface with any sensor data consumer that require filtering or information processing. The main functionalities provided by this entity are the subscription (receive the filter that will be applied), the analysis of the filters (analyse the filter condition) and the notification (notify the application when the condition of the filter occurs).
- The Sensor Tasking Entity (STE) allows services to perform requests operations to the sensor network, like for example a request to gather data, without the need to wait for an answer. The service will receive an immediate response and, when the desired data gets available it will receive the corresponding alert. This is mainly used for configuration and for calling actuators.
- The Service Protocol Adapter (SPA) provides protocol adaptation between the Web Services environment and USN-Enabler protocols (SIP and HTTP) to facilitate the development of services in an environment different to the IMS.
- The Catalogue and Location Entity (CLE) is defined as optional in the current Architecture. The main purpose of the entity is to provide resource discovery in a distributed environment. For example, in an architecture where several Sensor Description Entities (SDEs) exist, a client might be interested in a particular sensor. The client should interrogate the CLE to know which particular existing SDEs contain the information needed.

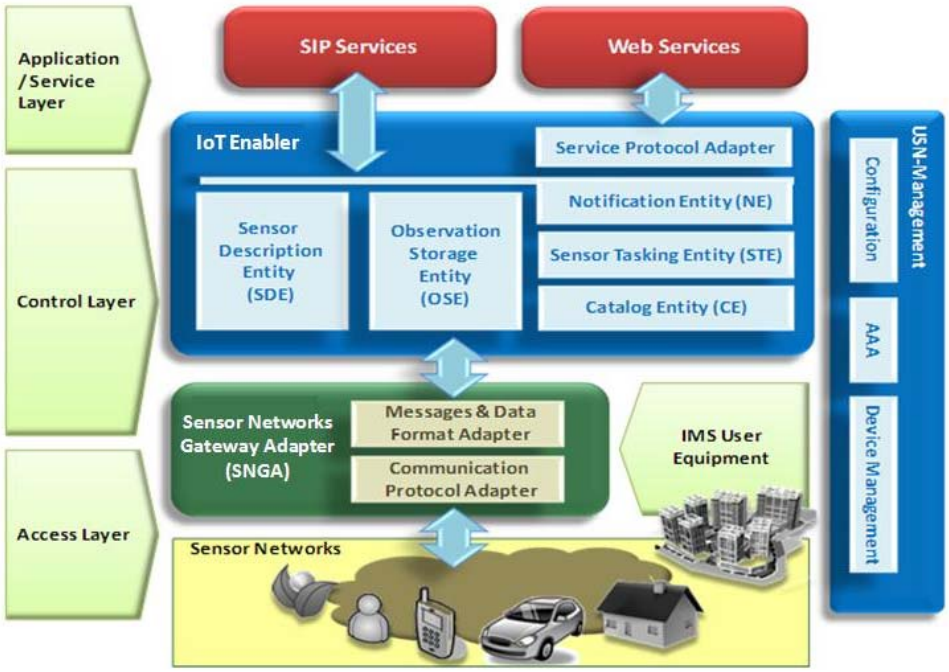

Figure 27. USN Platform Reference Architecture

The basic capability that a client application will demand from an IoT-Enabler is to subscribe to specific sensor information and receive it when available. For this reason, the mechanisms the IoT-Enabler provides are similar to those in standardized enablers like the OMA SIMPLE. However they differ in the type of information they manage.

The IoT-Enabler operations and protocols follow request/response operations encoded in XML format or key-value pairs, allowing transport protocol independence. However, to preserve the semantic content of certain operations in a SIP environment, without processing the XML document, the name of a SIP messages can be used to represent the meaning of the operation it transports. SIP messages carrying information are: PUBLISH, SUBSCRIBE, NOTIFY and MESSAGE.

6.1.3.1.1. Sensor Network Gateway Adapter (SNGA)

The Sensor Network Gateway Adapter (SNGA) represents a logical entity acting as data producer to the IoT-Enabler that implements two main adaptation procedures to integrate physical or logical Sensor and Actuator Networks (SANs):

1. Communication Protocol Adaptation. As a connection point between two networks, SNGA implements the SIP stack protocol for communication with the IMS network and services, and it is also responsible for applying routing procedures to any message send/received from each particular sensor/actuator in the SAN. One of the benefits when using IMS approach will be providing flexible inter-networking facilities between non-IP and IP sensor networks.

2. Sensor Data Format Adaptation. Through this functionality SNGA provides IoT-Enabler with both SensorML (meta-information) and O&M (observation & measurements) data from specific SANs data (i.e. ZigBee). Beyond simple format adaptation to SensorML, SNGA can also enrich sensor data with specific additional meta-information (location, capture conditions, references...) to expose it to context-aware applications through the IoT-Enabler.

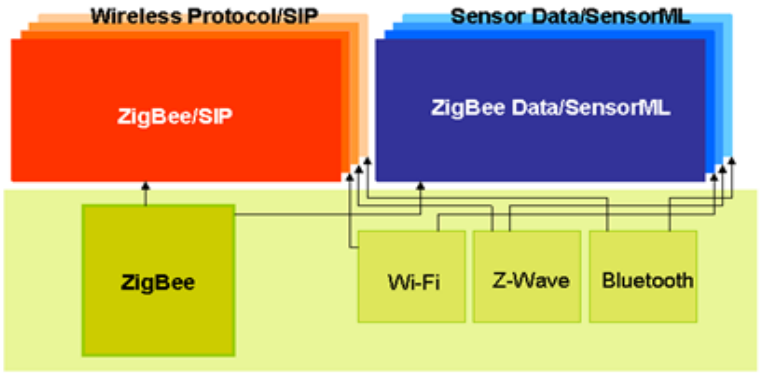

Figure 28. Sensor Network Gateway Adapter (SNGA)

It is by means of these two adaptation processes that SNGA makes the IoT-Enabler independent of the networking and data technologies used in the underlying Sensor networks.

6.1.3.2. The Open Telefonica Platform

The Telefonica Open platform is the element of the Telefonica service architecture responsible to provide access to the capabilities offered by the operator to third parties. These capabilities are offered to registered users through a set of open APIs, defined in REST style. The capabilities currently offered are the following: SMS, MMS, Location, Advertisement, Payment, Click to Call and Call Management and are available in different countries in which Telefonica is present like Spain, UK, Mexico, Argentina, Germany, Colombia, Chile, etc.

6.2. Mapping existing components to key system functions

One of the premises of the SmartSantander project is to not 're-invent' the wheel but to rather leverage existing work performed in the WISEBED, SENSEI and TELCO 2.0 projects. This helps reduce the effort in developing components that will fulfil SmartSantander and allow the project to focus on the practicalities for a city-wide wireless sensor network deployment and on value-added services to WSN users (for example, citizens of Santander). In this respect, selected components from WISEBED, SENSEI and Telco2.0 were analysed in terms of the functions they provide and the underlying information models they conform to. The components were attributed to the functions identified in the different subsystems of the proposed initial SmartSantander architecture requirements.

The attribution of existing components to sub-systems in the SmartSantander architecture is summarised in Figure 29 below.

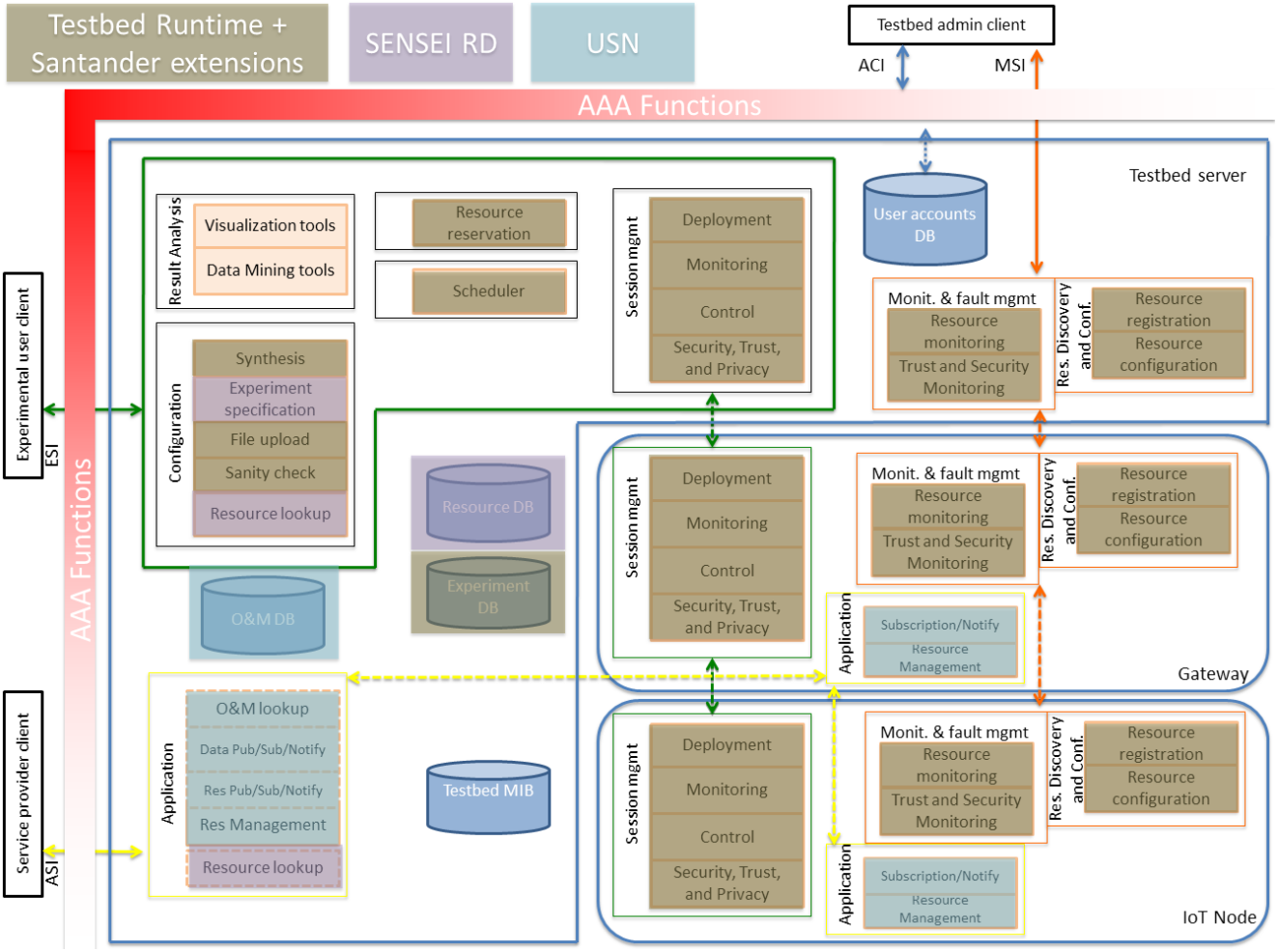

Figure 29. Functionality mapping of WISEBED, SENSEI and USN components on SmartSantander sub-systems

A finer-grained functionality mapping of SmartSantander sub-systems' functions on WISEBED, SENSEI and USN components is summarised in the Sections 6.2.1 - 6.2.3 below. Since the existing components have been developed in different contexts and designed to meet different requirements, they, in some cases, provide only a partial coverage of the functionalities required by the sub-system they are mapped onto. Further, the components may be using wholly different information model to describe resources. In this respect, the components have to be adapted to provide the functionality as to fulfill SmartSantander requirements. Section 6.3 briefly describes the adaptations to the relevant components.

6.2.1. AAA subsystem

System function	Sub-function	WISEBED	SENSEI	Telco2.0
AAA				
	Admin	AA API		
	Authentication	AA API(Shibboleth SNAA)		
	Authorisation	AA API (Shibboleth SNAA)		
	Accounting	Not available		

Table 3. Mapping of existing components to AAA

6.2.2. Experimental Support Subsystem

System function	Sub-function	WISEBED	SENSEI	Telco2.0
Configuration				
	Resource lookup		RD	SDE, CLE
	Specification	WiseML	SQR	
	Synthesis	Not available	Not available	
	Sanity check	Not available	Not available	
	Upload	WSN API different UZL's Testbed Runtime ULANC's iWSN Backend Impl.(S) UZL Bean Shell Client		
	Reset			
Resource reservation		Reservation System API different implementations: UZL's Database RS (S) UBERN's TARWIS (C)		
Scheduling		UZL's Testbed Runtime.		

Session management				
	Execution	WSN AP/Session Management API different implementations: UZL's iWSNBackend Impl. (S) ULANC's iWSNBackend Impl. (S) UBERN's TARWIS(C)		OSE for experimental results
	Deployment	Manual/script based through WSN API again different implementations available		
Result analysis	Visualization Tools			
	Data Mining Tools			

Table 4. Mapping of existing functions to ESS

6.2.3. Management support subsystem

System function	Sub-function	WISEBED	SENSEI	Telco2.0
Resource discovery			RD, ED, RP	SDE, CLE
Resource configuration				
Monitoring and fault management				
	Static	partly through NodeAPI WiseML		
	Dynamic			

Table 5. Mapping of existing functions to MSS

6.2.4. Application support subsystem

System function	Sub-function	WISEBED	SENSEI	Telco2.0
Resource discovery			RD	SDE
Data Retrieval (Observations & Measurements)				OSE
Publish/Subscribe/notify (resource level)			RD	NE
Publish/Subscribe/notify (data level)				NE
Resource Management				STE

Table 6. Mapping of existing components to Application Support System

6.3. Adaptation/Integration of existing components

Exisiting components have been classified by system functions and proposed adaptations have been identified. The Table 7 below summarizes the adaptation and integration of the existing components. A column for the planned phase of the realization has been added as some of the adaptations are not planned for Phase 0 but are considered in the implementation roadmap.

System function	Component	Proposed Adaptation	Planned Realization Phase
AAA	WISEBED AA API	Add accounting. Replace Shibboleth with e.g. OAuth	
Configuration	WISEBED iWSN	Add Multi Hop Over The Air Programming (MOTAP). Enable storage of scripts and sensor node images in some database.	

System function	Component	Proposed Adaptation	Planned Realization Phase
		Add declarative topology specification.	
Resource Reservation	WISEBED Reservation API	Add support for horizontality.	
Scheduling	WISEBED Scripting client	Enable scheduled execution of scripts (batch mode).	
Resource configuration	WISEBED iWSN/Node API	Provide tool support for configuration editing.	
Resource Management		The Telco platform provides means to update parameters defined in the SensorML descriptions of the resources. It has to be checked whether this function is enough as it is or must be extended.	

Table 7. Adaptation and Integration of existing components

6.4. New components development

The set of proposed new components is described in the table here below.

System function	Component	Adaptions proposed	Planned Realization Phase
Configuration	RD	RDF based description format for a testbed resource (see also MSI resource discovery) Replace current resource directory by an RDF store Integrate ED/RD?	
	SQR	Develop SPRQL based testbed resource query/specification language Adapt SQR parser to operate on developed language	
Session Management			
Result analysis			
MSI Resource discovery	RD+RPI+REP	Resource description needs to accommodate HW features and topological characteristics of the testbed. The GW, implementing a REP, can update the resource description via the RPI, as soon as a new resource appears in the testbed. Therefore, a local mechanism for detecting the IoT node property needs to be realized and implemented on the GW and IoT node. Two options could be possible: implicit discovery by means of a local mapping with a pre-existing resource description; explicit discovery protocol with information exchanged between the IoT node and the GW.	
Monitoring and fault management			

System function	Component	Adaptions proposed	Planned Realization Phase
ASIResource discovery	RD+RLI	This can work the same way the Resource Lookup works for the experiments resource discovery. Only a tag indicating that a resource is an application can be required.	
Data Retrieval (Observations & Measurements)		An adaptation on the format in which the measurements are provided must be done to do them in the O&M format	
Publish/Subscribe/notify (data level)		A basic publish/subscribe/notify based on the O&M is currently provided, but it has to be analysedif the current conditions fulfil the needs of the SmartSantander or if it needs to be extended.	
Resource Management		The Telco platform provides means to update parameters defined in the SensorML descriptions of the resources. It has to be checked whether this function is enough as it is or must be extended.	
Data processing		It needs to be first defined in T2.4	
Resource Description		Telco 2.0 is using SensorML as the language to describe the sensors, while the WISEBED is using WiseML to describe them. An adaptation between two formats needs to be defined	

Table 8. Proposed new components

Some of the functionalities already required in the initial architecture description cannot be provided by existing components. This will require implementing some new components. A brief description of these components is provided in the following sub-sections.

Result Analysis

The Result Analysis functionalities need to be provided and implemented. This will require the implementation of the required Visualization Tools and Data Mining Tools and their interaction with the provided information model. A similar interaction can be exploited in order to provide a detailed graphical representation of the available testbed connectivity map. This will allow the users to select the nodes for their experiments based on some required topology features. In order to collect the required connectivity information, new components need to be developed. Process running on each given IoT node are required in order to characterize the packet error rate of their outgoing links, for a given channel condition. The resource description needs to be extended to include this information as well. Tools are finally required in order to graphically visualize this information, allowing adding and removing the nodes to/from the original topology and showing in real-time how the topology can consequently change, based on the collected information. Reasoning components need to be developed for achieving this target, based on the proposed input.

SmartSantander Basic Node Application

Many requirements of the SmartSantander project need some software to be running on the sensor nodes at all times. The most obvious example is the MOTAP functionality. Every application that is flashed to a SmartSantander node has to support multi hop over the air programming, as their backbone connection is not necessarily implemented through a wired connection.

All components that are part of the “IoT node” require adequate software on the nodes which is currently neither available in WISEBED nor in SENSEI or Telco 2.0.

7. CONCLUSIONS

The SmartSantander project aims to develop a large scale experimental research facility for the research and experimentation of architectures, key enabling technologies, services and applications for the Internet of Things (IoT) in the context of smart cities. This report documents the specification of an early reference architecture for the facility aimed for the first development phase of the project. It contains technical requirements and intial design guidelines for the realization of the facility and provides a functional decomposition of the system and an understanding of the interactions across different system components. As such the document is expected to support WP2 and WP3 in making adequate design choices for the specification and implementation activities of the different facility components and provides a reference framework for the initially planned use cases in WP4 to be realized on top of the facility.

One of the key design considerations for the architecture is to make a running facility available to the research community as early and possible, initially at smaller scale and expand it in two further iterative cycles into the

envisioned full blown testbed. This would allow to incrementally accommodate the gained experimentation experience and emerging experiementation needs of the IoT research community.

Therefore it was important to design an initial architecture that is pragmatic enough to be quickly realized within the first year of the project and useful to the research community, but at the same time have a basis that is flexible enough to be evolvable for future experimentation needs and community requirements.

In order to provide a solid basis for experimentation with IoT technologies for broad range of IoT research and services useful for the city of Santander and its citizens a comprehensive set of use cases has been developed from which subsequently functional and non-functional requirements for the facility have been devised. For the rapid intial realization, the project has chosen to base the realization of the initial architecture on R&D results of two FP7 projectsnamely WISEBED and SENSEI, and Telefonica's USN service platform. WISBED components and mechansism provide the architecture with an intial core set of services that are necessary for the operation of the testbed and to support experimentation on top of it. SENSEI components are used to augment these services to increase the usability of the testbed for increased scale by allowing easier discovery of testbed resources and specification of experiments. The USN platform of Telefonica provides an environment and building block to support the creation of innovative services on top the facility. While many of the identified functional components of the architecture will be initially based on an integration of the above mechanisms, the functional decomposition and interface specifications allow for an evolution of the individual building blocks in the next phases of the project, minimising interdependence on each other.

The realization of SmartSantander platform is carried out in 4 phases in an iterative and incremental manner in order to guaranty the smooth and efficient implementation and suffice of the desired requirements: Pilot phase, Phase 1, Phase 2 and Phase 3 with 300, 3000, 5000 , 20,000 devices as optimistic upper bound deployment targets for of each the phases.

The herin presented architectural reference model is intented for the pilot and first phase is by no means the final one and considers as an intial towards the envisioned SmartSantander facility. It is expected that the intial reference architecture will evolve in phase 2 and 3 respectively based on the implementation and deployment experience in WP2/3/4, experimentation experience and feedback gathered through the first open call experimentation later this year by external users and internal experiments across the project partners. The project will also continue to pro-actively gather emerging requirements of the research community and validate existing assumptions based on active engagement through surveys questionaires, workshops and other dissemination events.

8. REFERENCES

[SENSEI-D.3.2] F. Carrez (Editor), "D.3.2 –Reference Architecture". SENSEI, Public Deliverable D.3.2, 2009, http://www.sensei-project.eu/

[WISEBED- Research Academic Computer Technology Institute (CTI), Technische Universitaet

iWSN-API] Braunschweig (TUBS), Lancaster University (ULANC), and University of Luebeck (UZL). iWSN API 2.1.1 (TR-iWSN-API-V2.1.1). Technical report, The WISEBED consortium, 2010. http://www.wisebed.eu.

[WiseML-Description] Stefan Dulman, WiseML Description, WISEBED Technical Report TR-RS-WISEMLDESCRIPTION-1.0http://www.wisebed.eu/images/stories/deliverables/TR/TR-RS-WISEMLDESCRIPTION-1.0.pdf The WISEBED consortium, 2010. http://www.wisebed.eu.

[WiseML-Schema] http://dutigw.st.ewi.tudelft.nl/wiseml/wiseml.rnchttp://dutigw.st.ewi.tudelft.nl/wiseml/wiseml.xsd The WISEBED consortium, 2010. http://www.wisebed.eu.

[WISEBED – SNAA API] Technical report on the Sensor Network Authentication and Authorization API http://www.wisebed.eu/images/stories/deliverables/TR/TR-SNAA-API-V1.pdf The WISEBED consortium, 2010. http://www.wisebed.eu.

[WISEBED-ULANC-iWSN-Backend] Technical report on ULANC iWSN API Backend implementation. http://www.wisebed.eu/images/stories/deliverables/TR/TR-2010-ULANC-iWSN-Backend.pdf

[WISEBED- UZL-iWSN-Backend] Technical report on UZL's iWSN API Backend implementation. http://www.wisebed.eu/images/stories/deliverables/TR/TR-2010-UZL-iWSN-Backend.pdf

[WISEBED- UZL-RS Impl] Technical report on UZL's Reservation System implementation. http://www.wisebed.eu/images/stories/deliverables/TR/TR-2010-UZL-RS-Implementations.pdf

[WISEBED- UZL-SNAA Impl] Technical report on UZL's SNAA implementation. http://www.wisebed.eu/images/stories/deliverables/TR/TR-2010-UZL-SNAA-

Implementations.pdf

[WSB] *Wireless Sensor Network Testbeds - **http://www.wisebed.eu/***

[SENSEI] *SENSEI [Integrating the Physical with the Digital World of the Network of the Future, **http://www.sensei-project.eu/**]*

9. APPENDICES

9.1. Appendix A: Complete List of requirements

9.1.1. Authentication, Authorization and Accounting

ID	Title	Text	Owner	Source	Priority
FR001	Authentication	When the researcher provides his personal credentials to the SmartSantander experimental facility, it must authenticate the researcher.	UZL, CEA	User Account Management (3.1_2)	M9
FR022	User account management	At any time the SmartSantander experimental facility shall enable the administrator to grant and revoke user access privileges.	UZL	User Account Management (3.1_2)	M9
FR030	Session Management	The user must log in just one time in order to be able to access to the SmartSantander facility and begin a session. Once created a valid session the rights may remain until user logout or session timeout is reached. During this period there will be no need to type again user or password to access any of the SmartSantander facilities.	ALU-SP	User Tracking (3.1_1)	M9
FR081	Authorization details	Each service provider should be given rights and limitations in terms of sensor and actuator access, for given testbeds (or set of nodes), at given periods. Means should be provided to prevent service provider to perform	CEA	Threat: RoleUsurpationServiceProvider	M15

		non-authorized operations.			
FR097	Authorization	When a user wants to execute any action the system has to verify that he is authorized to do this action.	CEA	User Account Management (3.1_2)	M9

9.1.2. Application Support System

ID	Title	Text	Owner	Source	Priority
FR099	Telecom operator connectivity	Service providers shall be able to connect telecom operator's infrastructure (e.g. sending SMS, etc.) with the SmartSantander experimental facility	TID	4.4.3_1	M15

9.1.3. Configuration Management

ID	Title	Text	Owner	Source	Priority
FR002	Experiment configuration	When the researcher uses the SmartSantander experimental facility, the system must provide a mechanism to specify and configure experiments to be carried out on the available testbeds.	ULANC, UniS, UZL	Experiment Deployment (3.6_2, 3,3_7)	M9
FR004	Experiment configuration reuse	When the researcher uses the SmartSantander experimental facility, the system shall provide a mechanism to save experiment configurations and reload them for further experiment iterations.	UZL	Reuse of experiment Configurations (3.3_3, 3.7_1)	M15
FR005	Experiment deployment	When the researcher uses the SmartSantander experimental facility, the system must allow him to upload sensor node program code, which is then distributed to the reserved set of nodes within the testbeds according to the experiment configuration.	UZL	WISEBED requirement	M9
FR027	Application sandbox	When a researcher tries to flash wireless testbed nodes via OTAP the system has to execute a check to ensure that the nodes will still be reachable with the new code.	UC, UZL	Experiment Deployment (3.6_2)	> M15

ID	Title	Text	Owner	Source	Priority
FR060	Experiment configuration details	To specify his experiment, a FIRE user shall use a web-based client to upload to his working directory and manage the sensor code images he intends to load to the sensor nodes. The system shall allow the FIRE user to specify along with the compiled sensor code image (i.e. the filename and upload directory) additional experiment details such as the image name, a version number, the platform on which the image runs, and a short description about the image the FIRE user shall provide additional details. For platform specification, the user shall be provided with a list of supported sensor node platforms and operating systems.	ULANC	3.3_1 (Experiment Specification)	M9
FR061	Node characteristic specification	As part of an experiment specification, the system shall also allow the user to specify the attributes of the sensor nodes (e.g. hardware characteristics, failure patterns and other metrics) he wishes to use for his experiments. If specified by the user, these attributes MAY be used by the system to locate matching nodes. When node reservation clashes occur, these attributes MAY also be used by the system to locate alternative nodes.	ULANC	3.3_1 (Experiment Specification)	M15
FR091	Semantic experiment specification	There needs to be a way for a testbed user to provide a high level declarative specification of the experiment and its requirements (such as specifications and constraints on the type and number of resources, required topology and corresponding link behavior, target environment, required SW components etc.).	UniS	Discovery of suitable testbed resource configurations (3.3_6)	M15

9.1.4. Experimentation Support System

ID	Title	Text	Owner	Source	Priority
FR131	Creating logical topology for the experiment	When a researcher requests nodes for his/her experiment, he/she shall log his requirements using portal. The system shall then identify necessary resources in the network and automatically create VLANs/VPNs that are required to create logical topology for the test.	EYU	3.3_4 Configuration of Network Topology	M15
FR132	Notification mechanism	The system shall provide a generic notification service (e. g. see requirements from FR100 onwards)	TID, ALU-IT	-	> M15

9.1.5. Monitoring and Fault Management

ID	Title	Text	Owner	Source	Priority

FR087	Automatic gateway status detection	CHANGING AVAILABILITY OF GATEWAY DEVICES: The system must be able to determine the change of availability (reconnection, disconnection) of a previously registered gateway device.	UniS	Automatic system detection of gateways devices (3.9_3)	M15
FR112	NMS - polling devices	The NMS solution shall support heartbeat alarms against all Network Elements managed by it, meaning that failure on interface link towards Network Elements or Network Element itself shall give an alarm in the NMS. Polling interval shall be configurable so that traffic load caused by Network Management can be managed.	EYU	4.4.1_1 Network monitoring using an NMS (Network Management System)	M15
FR114	NMS - notification messages	Network Elements shall immediately send notifications to the NMS in case of important system events. The notification will include at least: timestamp, source of the event (address of the source), indicator of the type of the NE, vendor ID and notification description. When specific conditions are met, NMS shall translate these notifications into alarms. Also, optionally it should be possible to configure NMS to acknowledge receipt of the notification. If a receipt is not sent back, the originator will continue to send messages until the notification is acknowledged.	EYU	4.4.1_1 Network monitoring using an NMS (Network Management System)	M15

9.1.6. Management Support System

ID	Title	Text	Owner	Source	Priority
FR051	Administration state	When the system administrator wants to perform any kind of administrative tasks to the SmartSantander testbed, the system must allow him to put the whole testbed into administration state.	UC	Testbed administration state (3.8_2)	M15
FR052	Administration state	When the testbed is into administration state, no reservation can be made effective and the experiments relying on this (part of the) testbed will be delayed until all administration tasks have finished.	UC	Testbed administration state (3.8_2)	M15
FR053	Administration state	The system administrator must have a default image for all the testbed nodes, which supports the (City-related services) and node manageability/administration features, and is the one to be flashed in the nodes for putting them in administration state.	UC	Testbed administration state (3.8_2)	M15

9.1.7. Reservation

ID	Title	Text	Owner	Source	Priority
FR009	Testbed reservation status overview	When the researcher starts an experiment on one or several testbeds, the SmartSantander experimental facility must show him the current reservation status of the available testbeds.	UZL	- (WISEBED requirement)	M15
FR010	Testbed reservation	When the researcher starts an experiment on one or more testbeds, the SmartSantander experimental facility must force him to make a reservation for the testbeds he wants to use before he can start the experiments.	ULANC, UZL	- (WISEBED requirement)	M9
FR061	Node characteristic specification	As part of an experiment specification, the system shall also allow the user to specify the attributes of the sensor nodes (e.g. hardware characteristics, failure patterns and other metrics) he wishes to use for his experiments. If specified by the user, these attributes MAY be used by the system to locate matching nodes. When node reservation clashes occur, these attributes MAY also be used by the system to locate alternative nodes.	ULANC	3.3_1 (Experiment Specification)	M15
FR069	Testbed batch mode	Once a user has entered his system reservation details, the system shall save the reservation specification to the database for later retrieval. The system shall then issue to the user a reservation-key, which he can use to redeem the reserved resources at the specified time-slot and manually launch the deployment and execution processes. If the user selected the automatic scheduling option in his reservation, the Reservation System's scheduler shall use this key to obtain the reserved resources and to execute the reservation specification at the due hour.	ULANC	3.4_1 (Experiment network reservation	M15

9.1.8. Resource Management

ID	Title	Text	Owner	Source	Priority
FR013	Testbed configuration	When an administrator accesses the SmartSantander experimental test facility, the system must allow him to configure the nodes within a testbed and add new sensor nodes along with their positions, their type, and other relevant information about the nodes. He also must be able to update or delete nodes configurations.	UZL	- (WISEBED requirement)	M15

ID	Title	Text	Owner	Source	Priority
FR047	Experiment fallback	The SmartSantander experimentation facility provides to the researcher the ability to reset the entire sensor network testbed to the initially chosen default settings in case of node failures.	TTI, UC	Experiment monitoring and control (3.7_2)	M9
FR054	Remote access	System administrator must be able to remotely access to the gateways in order to perform issues such configuration, update, etc.	UC	Remote access to gateways (3.8_5)	M9
FR083	Uniform Resource Description	A testbed resource requires a standardized representation within the information model of the SmartSantander facility.	UniS	Automatic system detection of sensor nodes and gateways (3.9_2, 3.9_3)	M15
FR129	HW/SW Inventory Database	The system shall support an inventory database where it shall keep track of at least the following: all devices in the network and their position, device vendor, services they are being used for and functionalities/information they can provide, their hardware configuration, release of the software installed on each of them.	EYU	3.11_1 OS Heterogeneity	M9

9.1.9. Resource Discovery

ID	Title	Text	Owner	Source	Priority
FR008	Testbed overview	When the researcher uses the SmartSantander experimental facility, the system must provide an overview of the testbeds that are available for experimentation including the nodes' location and capabilities.	TTI, UZL	3.4_3	M9
FR019	Testbed Configuration Plug & Play	The platform shall support the discovery of the different information providers (e.g. sensors) existing in the system.	TID	4_3.1	M15
FR082	Automatic resource detection	When an administrator adds a new resource to the testbed topology and provides an adequate resource description for that sensor node and establishes physical connectivity with the testbed infrastructure, the SmartSantander experimentation facility shall automatically detect and configure the new resource.	UniS	Automatic system detection of sensor nodes (3.9_2)	M15

ID	Title	Text	Owner	Source	Priority
FR084	Automatic node detection	Upon detection of a newly attached sensor node, a gateway that is part of the testbed infrastructure needs to recognize the node and retrieve an appropriate resource description and subsequently register the resource description with the resource directory of the testbed.	UniS	Automatic system detection of sensor nodes (3.9_2)	M15
FR085	Automatic node status detection	DETECTION OF CHANGING AVAILABILITY OF ATTACHED SENSOR NODES: Upon the detection of a change in availability (re-connection, disconnection) of one or more attached sensor node, a gateway device must update the sensor node status with the resource directory of the testbed	UniS	Automatic system detection of sensor nodes (3.9_2)	M15
FR086	Automatic gateway detection	ADDITION OF A NEW GATEWAY DEVICE: Upon its connection to the testbed infrastructure, a gateway device must register its own capabilities and those of the attached sensor nodes to the resource directory of the testbed.	UniS	Automatic system detection of gateways devices (3.9_3)	M15

9.1.10. Resource Lookup

ID	Title	Text	Owner	Source	Priority
FR092	Semantic discovery of valid testbed resource configurations	When the experiment requirements are provided, the system processes the declarative query and performs semantic matching in order to provide a corresponding mapping with the resources satisfying the requirements.	UniS	Discovery of suitable testbed resource configurations (3.3_6)	M15
FR094	Automatic experiment configuration	When the configuration requirements are provided, the system must reserve adequate testbed resources and create necessary configurations for Hardware and Software components satisfying the provided experimentation required.	UniS	Automatic configuration of an experiment (3.3_7)	M15
FR130	Device booking and software upgrade on request	Researcher shall be able to query inventory database from portal in order to see which devices are available for the experiment and their hardware and software configuration. He/she can then book the resources and request software upgrade if needed.	EYU	3.11_1 OS Heterogeneity	M15

9.1.11. Scheduling

ID	Title	Text	Owner	Source	Priority

ID	Title	Text	Owner	Source	Priority
FR026	Experiment execution batch mode	Researchers shall be able to run experiments in a batch mode (for example repeat an experiment several times, change parameters for each execution). They shall receive a handle to interact with the experiment when it's running.	ULANC, UniS, UZL	Reuse of experiment Configurations (3.3_3, 3.4_4)	M9
FR098	Conflict handling	When two or more users want to access the same resources of the SmartSantander testbed the system shall solve this conflict by allowing concurrent access where possible (servers, gateways, read only) or enforce a serialization (reconfiguration, flashing nodes).	TTI, UZL	WP2-T2.2 Conflict handling	> M15
FR003	Experiment control	When the researcher uses the SmartSantander experimental facility, the system must provide a mechanism to control (e.g. start, stop) experiments the researcher plans to carry out on the available testbeds.	UZL, UniS	Experiment Deployment (3.6_2,3.3_7)	M9

9.1.12. Trustworthiness

ID	Title	Text	Owner	Source	Priority
FR075	Sensor data privacy	The WSN system shall provide means to assign ownership & privacy properties to sensors, and to prevent unauthorized reading of private sensor measurements (through wireless access or public access).	CEA	Threat: SensorEavesDropping	> M15
FR076	Sensor data integrity	The WSN system shall provide means to prevent forging or mangling of sensor measurements (through wireless access or public access).	CEA	Threat: ForgingSensorMeasurements	> M15
FR077	Actuator integrity	The WSN system should provides means to prevent forging or mangling of actuator commands (through wireless access or public access).	CEA	Threat:ActuatorCommandMangling	> M15
FR078	Platform Security	The sensor platform shall provide means to prevent platform reprogramming by unknown or unauthorized users.	CEA	Threat: CompromisingSensorPlatform	> M15

9.1.13. Service Platform

ID	Title	Text	Owner	Source	Priority

SMARTSANTANDER PROJECT

ID	Title	Text	Owner	Source	Priority
FR118	Validation of authorised vehicles for parking spaces for persons with disability	The sensor node network (parking space/vehicle) must be able to validate if the vehicle is authorized to park in parking spaces for persons with disabilities.	AI	WP4_UC1b	M15
FR119	Alerting of Authorities	The sensor node network (parking space/vehicle) must be able to alert the authorities about unauthorized parking (including disabled parking areas), load/unload area and bus stop.	AI	WP4_UC1a/b/d/e	>M15
FR120	Alerting of User	The sensor node network (parking space/vehicle) should be able to visually alert the user if he/she is authorized or unauthorized to park in the parking space.	AI	WP4_UC1b	M15
FR121	Alerting user of exceeded time parking	The sensor node network should be able to alert the user if the time parking is/will exceeded/exceed.	AI	WP4_UC1b	>M15
FR122	Pre-reserve parking space	The sensor node network should allow the user to pre-reserve a free parking space for people with disabilities.	AI	WP4_UC1b	>M15
FR123	Visualization of available/occupied parking spaces, load/unload areas	The sensor node network should allow the user to visualize available as well as occupied parking spaces.	AI	WP4_UC1a/b/d/e	M9
FR124	Change/Cancellation of pre-reservation of parking space	The sensor node network should allow the user to change/cancel a pre-reservation of a parking space for people with disabilities.	AI	WP4_UC1b	>M15
FR136	Information display	The system must be able to provide information on the proximity of the public buses as well as the line they belong to.	UC	WP4_UC6	>M15
FR137	Passengers boarding accounting	The system must be able to gather information on the number of passengers that board a bus in each stop.	UC	WP4_UC6	>M15
FR138	Trip planning support	The system must provide up to date information to the user that helps him/her to dynamically plan his/her trip.	UC	WP4_UC6	>M15

SMARTSANTANDER PROJECT

ID	Title	Text	Owner	Source	Priority
FR139	Bus occupancy estimation	The system should be able to support the estimation of the number of people that is travelling in a bus.	UC	WP4_UC6	>M15
FR140	Detailed route completion time measurement	The system must be able to measure the amount of time that a bus takes to complete its route and dissociate it on a stop-by-stop basis.	UC	WP4_UC6	>M15
FR141	Historical trip time estimation	The system must be able to estimate the time of a trip according to historical measurements	UC	WP4_UC6	>M15
FR142	Traffic conditions estimation	The information of public transport vehicles pace must be available for estimating the traffic conditions at the bus location.	UC	WP4_UC6	>M15
FR146	Correlation between nodes and monitored building	The system must be able to link each given node to the house/public building where the node is deployed in order to associate the consumption to a given building	UniS	WP4_UC4	M15
FR147	Anomalous consumption detection	The system must be able to detect an anomalous consumption in a monitored building (no-one in the building with respect to a still high consumption)	UniS	WP4_UC4	M15
FR148	System event triggering capability	When an anomalous consumption is detected, the system must be able to send an alert to the householder or other interested authorities	UniS	WP4_UC4	M15
FR149	Gateway event triggering capability	When an anomalous consumption is detected (no one in the building but still high consumption), the gateway must be able to send an alert to the householder or other interested authorities	UniS	WP4_UC4	M15
FR134	Timer implementation and management	The node must be able to provide and manage timers, in order to launch the corresponding callback function when these timers expire. When a node detects that a parking , load/unload area site is occupied, then it starts a timer in order to measure how long the site is occupied, sending a message (to the corresponding gateway), when the timer duration (associated with the time reserved by the user) expires.	UC	WP4_UC1a/b/d/e	M9

ID	Title	Text	Owner	Source	Priority
FR135	Tracking of mobile nodes	The system must provide the tracking of mobile nodes, offering a dynamic map with the position of these nodes and also allowing the estimation of future positions through geographic positioning tools. The system must be able to provide the position of buses, taxis, and eventually bicycles, in a real time fashion.	UC	WP4_UC6	M15
FR143	Energy, Gas and Water consumption detection	The nodes must be able to detect the consumption (i.e., energy, gas and/or water) of the given appliance to which they are connected	UniS	WP4_UC4	M15
FR144	Presence detection	The nodes must be able to detect the presence of people in the monitored house/building in order to have a correlation between the observed consumption pattern and respective presence	UniS	WP4_UC4	M15
FR145	Time validation of the data	The system should be able to associate correct time indication to the information collected by each “metering” node	UniS	WP4_UC4	M15

9.2. Non-Functional Requirements

9.2.1. Authentication, Authorization and Accounting

ID	Title	Text	Owner	Source	Priority
NFR027	Common user/service/inventory database	A common user/service database shall be created for user and subscription management that will store information such as which users are registered, which services are available for a user, what has the user chosen for his personal portfolio, which resources are available for the user, etc. *This database shall be populated either through GUI or by bulk import.* The database shall also contain a central inventory of resources (e.g. SIM cards, M2M equipment, etc.).	EYU	4.4.2_1 Service Provisioning for MNO/SP customers who are SmartSantander network	> M15

9.2.2. Configuration Management

ID	Title	Text	Owner	Source	Priority

NFR014	Code Quarantine System	When the researcher wants to run an experiment on the SmartSantander testbed, the code to be loaded in the nodes has to be tested before in a controlled and non-critical part of the testbed.	UC	Sensor Code Image Quarantine (3.6_1)	M15
NFR015	Code Quarantine System	The SmartSantander testbed must have a dedicated number of nodes that are used to test the code associated with the experiments to be run in order to assure as much as possible that it will not produce harmful situations in the overall testbed.	UC	Sensor Code Image Quarantine (3.6_1)	M15
NFR016	Code Quarantine System	The part of the SmartSantander testbed dedicated to the code quarantine must be representative of the overall SmartSantander testbed in terms of heterogeneity, capacity and manageability.	UC	Sensor Code Image Quarantine (3.6_1)	> M15
NFR030	OS diversity in SmartSantander network	In SmartSantander environment there shall be devices that support a diversity of different OSs (such as Contiki, TinyOS, FreeRTOS, etc.) so that all of them can be tested.	EYU	3.11_1 OS Heterogeneity	M15

9.2.3. Experimentation Support System

ID	Title	Text	Owner	Source	Priority
NFR001	Web Interface	When the researcher wants to access the system this shall be possible via a web-based user interface.	UZL	-	M15

9.2.4. IoT Node Deployment

ID	Title	Text	Owner	Source	Priority
NFR0025	Physical Access Restriction	Physical access to the deployed sensor node platform shall not be easy.	CEA	Threat: LocalModificationEnvironment and Vandalism	M9

9.2.5. IoT Node Requirements

ID	Title	Text	Owner	Source	Priority
NFR002	OTAP	In order to be FIRE compliant, an important percentage of the nodes, being part of the platform, have to support OTAP.	UC	-	M9
NFR004	Maintenance minimization	The SmartSantander experimental facility shall require minimum maintenance (e.g. replacing batteries).	EYU	-	M9

ID	Title	Text	Owner	Source	Priority
NFR005	Real Field Conditions Operativeness	The platform must be operative in real field conditions (indoors and outdoors) providing support to both FIRE researchers AND inhabitants.	UC	-	M9
NFR006	Power consumption minimization	The consumption of the deployed sensor nodes has to be low enough in order to have a sustainable experimentation facility.	UC	-	> M15
NFR007	Mobility of nodes	The system shall integrate and support wireless nodes that can be moved over distance.	UC, UZL	3.8_4	> M15
NFR030	OS diversity in SmartSantander network	In SmartSantander environment there shall be devices that support a diversity of different OSs (such as Contiki, TinyOS, FreeRTOS, etc.) so that all of them can be tested.	EYU	3.11_1 OS Heterogeneity	M15

9.2.6. Monitoring and Fault Management

ID	Title	Text	Owner	Source	Priority
NFR017	Automatic error detection	The system must provide automatically detection of node crashes, unplanned resets, failed flash operations or unresponsiveness.	UC	Node Fault Diagnosis and Recovery (3.8_3)	> M15

9.2.7. Resource Lookup

ID	Title	Text	Owner	Source	Priority
NFR031	Use of devices in different test-beds	The users of one test-bed shall have information about the available resources in all federated test-beds so that they can be used for experiments.	EYU	3.3_4 Configuration of Network Topology	M15

9.2.8. User Interface

ID	Title	Text	Owner	Source	Priority
NFR029	Easy to use user interface	The user must find the user interface easy to use.	AI	WP4_UC1b	M15