ALMANAC

RELIABLE SMART SECURE
INTERNET OF THINGS FOR SMART CITIES

(FP7 609081)

D3.1.3 System Architecture Analysis & Design Specification 3

Submission Date 31st December 2015 – Version 1

Published by the ALMANAC Consortium

Dissemination Level: Public

Project co-funded by the European Commission within the 7th Framework Programme
Objective ICT-2013.1.4: A reliable, smart and secure Internet of Things for Smart Cities

Document control page

Document file: D3.1.3 System Architecture Analysis and Design Specification 3 - v1.0
Document version: 1.0
Document owner: Dario Bonino (ISMB)

Work package: WP3 – Smart City Platform Architecture
Task: All WP3 tasks
Deliverable type: R
Document status: ☒ approved by the document owner for internal review
☒ approved for submission to the EC

Document history:

Version	Author(s)	Date	Summary of changes made
0.1	Dario Bonino	2015-11-12	Initial structure, partially derived from D3.1.1 and D3.1.2
0.2	Dario Bonino	2015-11-16	Structure amendment according to suggestions from Marco Jahn
0.3	Dario Bonino, Raphael Ahrens	2015-11-17	Added content hints for the security perspective, added few comments
0.4	Dario Bonino	2015-11-17	Added comments / amendments from partner feedback on the ToC
0.5	Dario Bonino	2015-11-19	Added contributions to Section 3
0.6	Dario Bonino	2015-11-24	Added Methodology sections and High-level Architecture
0.7	Dario Bonino, Angel Carvajal Soto	2015-11-30	Integrated first federation contribution with major comments
0.8	Dario Bonino, Raphael Ahrens	2015-12-01	Integrated contribution on the security framework, added introduction to the Information View
0.9	Mathias Axling, Matts Ahlsen, Jaroslav Pullman	2015-12-01	Update of Functional, Information and Deployment Views. Update of Federation Perspective, added Scalability Perspective. Amended / revised Semantic Representation Framework
0.10	Dario Bonino, Angel Carvajal Soto	2015-12-08	Integrated partners contributions, integrated latest version of the federation section, added scalability parts referred to the SCRAL, revised sections from 1 to 6
0.11	Dario Bonino, Marco Jahn	2015-12-15	Integrated contributions on Section 6, added Data Flow section from field to platform.
0.11	Mathias Axling, Matts Ahlsen	2015-12-17	Elaboration of Scalability Perspective and Deployment View
0.11	Jaroslav Pullman	2015-12-17	SRF scalability
0.11	Raphael Ahrens, Marco Jahn	2015-12-18	Security perspective update
0.11	Alexandre Alapetite	2015-12-22	Data sharing section
0.11	Alexandre Alapetite, Angel Carvajal Soto	2015-12-22	Federation perspective v2.0
0.12	Dario Bonino	2015-12-22	Minor amendments and integration of partners' contributions.

1.0	Dario Bonino	2015-12-23	Ready for review
1.1	Dario Bonino	2016-01-05	Integrated internal review suggestions from TIL
1.2	Dario Bonino	2016-01-07	Integrated internal review suggestions from ALEX, ready for submission

Internal review history:

Reviewed by	Date	Summary of comments
Daniele Buosi	2016-01-03	Typo correction and some explanation suggested.
Thomas Gilbert	2015-12-30	Approved with minor comments

Index:

List of Figures

1. Executive summary

This deliverable is the third and last of a series of public reports, and provides the final release of the Smart City Platform (SCP) architecture designed and developed in the ALMANAC project. Such a version will be used as reference in the 3rd year activities and can be assumed stable enough for supporting project evolution beyond the EU funding time span.

Data included in the report provides the rationale behind the final platform architecture design and features an in depth description of all the aspects involved. Particular emphasis is placed on security and federation, describing in detail all the steps carried throughout the project lifespan to design and account for security issues involved by a Smart City Platform, from threat analysis to final design solutions.

Results summarized in the document stem from the third refinement of a mixed design approach merging bottom-up contributions from partners mainly in the 1st year, and top-down, requirements-driven design mainly adopted in the 2nd year of the project.

Particular attention on the documentation process and quality is applied, and well-known, proven standards are adopted wherever possible. More specifically, the entire architecture design documentation process, summarized in this deliverable, is performed in accordance with the ISO/IEC/IEEE 42010 "System and software engineering – Architecture description" (ISO/IEC/IEEE 42010, 2011), which superseded the IEEE 1471 "Recommended Practice for Architectural Description for Software Intensive Systems" (IEEE 1471, 2000).

Although the main purpose of the deliverable is conveying up-to-date information on the ALMANAC architecture analysis and design, it is deemed important to highlight how such design choices and guidelines have been taken into account in the actual platform development carried throughout the project. For this reason, the reader should refer to the upcoming internal deliverables ID 5.2.2, ID5.3.2, ID5.4.2, as well as to already released internal deliverables ID6.2.2 and ID6.3.2, to get an up-to-date view on implementation details regarding activities carried in WP5 and WP6 and, corresponding component prototypes.

1.1 Document structure

The remainder of the deliverable is organized as follows:

- *Section 2 – Terminology*, defines the domain-specific terminology used throughout the deliverable.
- *Section 3 - Introduction*, introduces the third ALMANAC deliverable on the Smart City Platform architecture identifying the purpose, scope and context of the document and, provides the technological background needed to better understand the presented perspectives and design choices;
- *Section 4 – Methodology*, illustrates the adopted design methodology, starting from a brief introduction of iterative and agile design patterns and providing more detail on the adopted documentation-driven approach;
- *Section 5 – High Level Architecture*, provides a 30'000 feet view over the ALMANAC Smart City Platform architecture identifying the relevant component groups / layers, tracing back the history of the platform design and providing a short introduction to specific views on the architecture, as dictated by the IEEE 42010 standard;
- *Section 6 – Functional view*, provides a traditional, yet crucial view on platform components, with particular focus on the respective roles and duties, with the aim of establishing a strong technical understanding of the platform architecture. This enables on one side a better comprehension of underlying issues, and assumptions, and on the other hand provides a strong basis upon which building possible evolutions of the ALMANAC ecosystem;
- *Section 7 – Deployment view*, describes the lessons learned and, the technical solutions designed as part of the platform architecture, under the lenses of final deployment in the real world. This includes considerations on portability, adaptability, modularity and replicability;

- *Section 8 – Information view*, offers an orthogonal view on the platform identifying typical data flows, and volumes, and discusses related issues, expected cardinality and frequencies, etc. The rationale behind such a view is to make the reader aware of the various kinds of information flows managed by the platform, at different levels and to relate such information to the relative stakeholders, whenever possible;
- *Section 9 – Security perspective*, describes the architecture under a security analysis standpoint. A full-stack security threat analysis following the STRIDE methodology is presented, covering all platform components and identifying possible attack vectors and underlying motivations. State-of-the art (or well-known, and widely recognized) solutions are identified and proposed for the threats emerging from the first analysis activity and, finally, specific design choices and trade-offs related to the actual platform design are presented;
- *Section 10 – Federation perspective*, describes the platform focusing on inter-platform communication, both from a technical and from an administrative/legal perspective;
- *Section 11 – Scalability Perspective*, presents the methodology and processes adopted to ensure the platform scalability under increasing workload;

2. Terminology

In this section the domain-specific terminology used in the remainder of the document is introduced in a glossary-like form where each term is associated with a long description identifying the term scope, context and usage. Currently adopted terms are reported in Table 1.

Table 1. Glossary

ALMANAC Platform Instance (PI)	A deployment of the ALMANAC platform. Depending on the choice of deployment this may comprise only a subset of the platform components. E.g. in some cases it may be sufficient to run an instance of SCRAL while in other cases only the Virtualization Layer and the Cloud-based APIs may be needed.
ALMANAC Smart City Platform	The ALMANAC Smart City Platform (or simply platform) comprises a set of software components, guidelines, constraints, best practices, etc. that allow the development of Internet of Things applications for smart cities.
Capillary Network	Capillary Networks are flexible and autonomous communication networks normally used to locally collect information from sensors and actuators in the smart city. Examples of capillary networks include short-range networks based on Wireless M-Bus or DLMS-Cosem e.g. for utility metering (gas, water, electricity), collection of waste management data, pollution and traffic control sensors, smart lighting sensor, heating control sensors, etc.
Cloud-based API	Cloud-based APIs are a set of services that provide access to the ALMANAC platform for developers of smart city applications. These services can be accessed over the network through REST interfaces or Web Socket channels.
ETSI M2M	A standard defining M2M communication and platforms provided by ETSI. The basic concept of the standard is the Store and Share of data coming from smart devices. Data is stored and then shared with the Apps by the M2M Platform with standard APIs (httpREST based).
Federation	Federation describes the inter-operation between different ALMANAC platform instances (and external actors) through a shared communication infrastructure, see Section 10.
IoT-ARM	The IoT-A Architectural Reference Model (IoT-A ARM) provides a collection of generic architectural concepts and constructs considered applicable to IoT system architectures. The IoT-A ARM does not say how to build IoT systems, it is a tool box of concepts, models and recommendations for the domain of IoT systems and their architectures.
Machine-2-Machine (M2M)	M2M describes the ability of two devices to communicate with each other without human intervention through wired or wireless network and often through a M2M Platform. M2M communication is a prerequisite for the Internet of Things.
Federation user	A user or a client application that wants to interact with the services of a federation.
Federation service	A service that is inside a PI of the federation and implements parts of the Cloud APIs.
Trusted Area	The network that connects the single components of the PI and which should not be accessible from the untrusted area. Since the trusted area doesn't need to communicate with the outside, special measurements can be used to block the access to the trusted area. In addition since the components inside the trusted area are in a fixed amount, it is simpler to establish a trust relationship between these components.

Untrusted Area	The untrusted area is an easily accessible network to which federated members have access to. This makes it less secure, since the number of participants can constantly change and special measures must be taken to secure the components that interact with the untrusted Area. The only components that need to interact with the untrusted area are the Virtualization Layer Core, the Smart City Resource Adaptation Layer, the Federated Identity Manager, and LinkSmart. Every other component resides in the trusted area.
Scalability	The ability of a system to increase the maximum workload it can handle by expanding its quantity of consumed resources (Lehrig et al., 2015). The ability of a system either to handle increases in load without impact on performance or for the available resources to be readily increased" (Wilder, 2012).
Subsystem	A logic group of software modules, or components, which share a common rationale or a common purpose.

3. Introduction

Activities regarding the Smart City Platform architecture design in ALMANAC are depicted in the D3.1.x series of deliverables, of which this is the latest and final version. The general purpose of these deliverables is to provide an up-to-date vision on the platform architecture design, from different perspectives and viewpoints.

3.1 Purpose, context and scope of this deliverable

This deliverable describes the last revision of the software architecture underlying the ALMANAC platform at the beginning of the 3rd, and last, year of the project. Within the ALMANAC work package structure, Work Package 3 (Smart City Platform Architecture) is responsible for specifying the system architecture design.

As expected at the third year of the project, most of user requirements have been gathered, validated and successfully integrated in the platform design process (thanks to WP2 – Requirements Engineering and Smart City Business Models). Iterative refinements are still possible[1], and they are likely to continue occurring even after the project lifespan. For such a reason, while this deliverable shall be deemed as final for what concerns the project activities, it is indeed still evolving and possible revisions/updates might be submitted before the end of the project or published on the web after EC funding expiration.

This document aims at conveying the most updated description of the ALMANAC Smart City Platform architecture with sufficient details to support effective platform analysis, understanding and replication. The architecture description strictly adheres to the IEEE 42010 standard and defines several viewpoints, targeted at specific concerns, identifying reference stakeholder groups and adopting dedicated representation models.

While previous versions, were mainly aiming at the definition of a sound, scalable and repeatable platform architecture; capable of responding to the various requirements gathered throughout the project, this version focuses more on aspects making the platform exploitable in the real world. This implies:

a) A stronger emphasis on security design, which started at the beginning of the project;

b) More attention to federated platform deployment, which is investigated more in deep, as the related issues and design solutions are crucial for the actual exploitation of the ALMANAC platform;

c) Increased scalability concerns that are addressed with a more detailed architectural perspective devoted to scalability. This latter perspective reports both the adopted analysis methodology and the solutions that will be deployed in the project.

3.2 Background

The ALMANAC Smart City Platform (SCP) collects, aggregates, and analyses real-time, or near real-time, data from resources (appliances, sensors and actuators, smart meters, etc.) deployed to implement Smart Cities. Such resources are interconnected through an independent, pervasive, data communication network, named Capillary Network. Like the capillary system, which delivers nutrients to the far periphery of a body, the Capillary Network provides connection to the multitude of devices deployed in a smart city. The ALMANAC project aims at achieving this degree of pervasiveness by defining short range radio networks providing local Machine-to-Machine (M2M) connectivity to smart things, thus enabling their active involvement in the Smart City processes.

The Smart City Platform, described in D3.1.x deliverables, provides support to decision processes in both day-to-day and long term city management and, implements intelligent control of the devices, either locally or through M2M communication.

[1] T2.3 – Evolutionary requirements refinement anding on M31 (March 2016)

The technological work in connection with the development of the ALMANAC Smart City platform has been highly influenced by requirements generated in the City of Turin. The city path to become "Smart City" started several years (late 2011) ago, when the City Council took the decision to take part in the initiative of the European Commission "Covenant of Mayors" and – as one of the first Italian cities – engaged itself to elaborate an Action Plan for Energy in order to reduce its CO2 emissions more than 20% by 2020.

Three specific applications (waste management, water supply and citizens' engagement) have been selected for proof-of-concept implementation and evaluation of the ALMANAC ecosystem, and, in particular, of the ALMANAC Smart City Platform. These applications are deemed to be representative of a large number of applications, deployable in a smart city. Given the challenging objectives of the project, during these 3 years, we aimed at achieving a set of cross-domain application use cases which encompass a large amount of heterogeneous devices and generate large amounts of data. This on one side ensures design and testing conditions comparable with the expected settings in real-world smart cities, on the other hand it permits an earlier reach-out to the city, and in the end, to the market, thanks to pilot experimentation and networking activities carried by the consortium, for which the selected applications act as igniters.

4. Methodology

The analysis and design of the ALMANAC architecture follows a hybrid, iterative approach (see Figure 1) mixing bottom-up technology harvesting and harmonization (Year 1) and top-down refinement driven by both requirements elicitation processes, set-up in the project, and by lessons learned in the first phase (Year 2). The last iteration of the analysis and design activity (Year 3) has a dual purpose. First, it aims at improving the architecture definition, clarifying component roles and interactions not yet specified (few, minor changes). Second, it prepares the platform for real-world exploitation focusing much more on security measures, data flow identification, scalability and federated deployment.

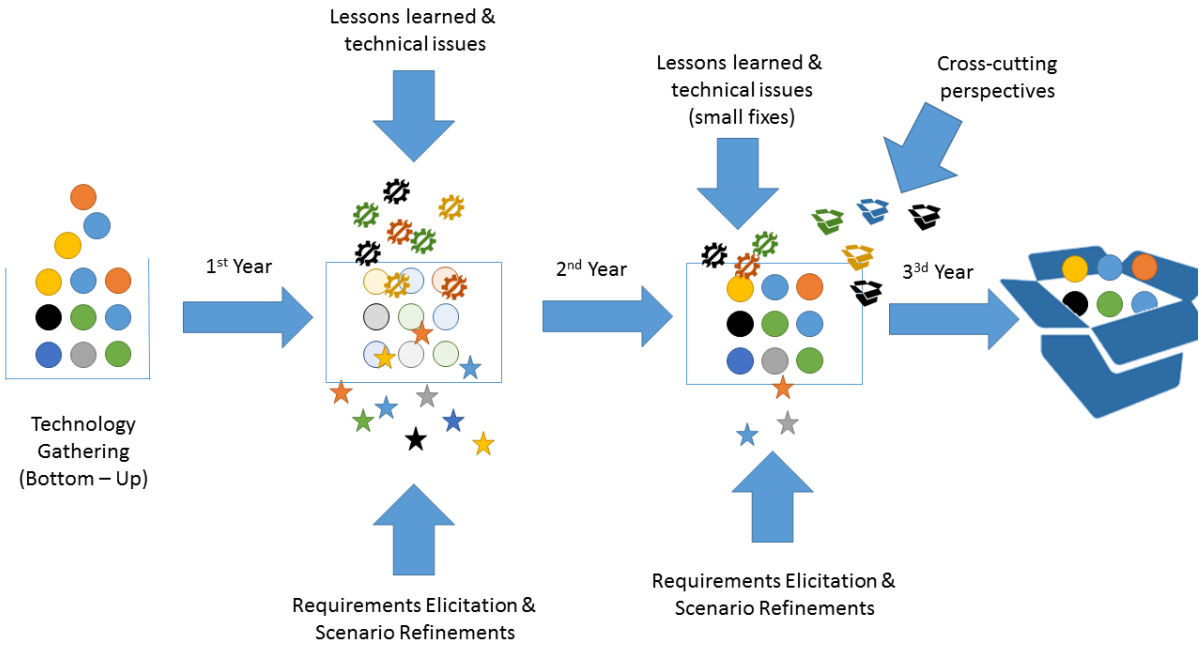


Figure 1. Architecture Analysis and Design Methodology.

According to this rationale, while views referring to the functional aspects of the platform show mainly little adjustments, major contributions can be found in deployment views and in cross-cutting perspectives.

4.1 Methodology implementation

During the first year of the project, the architecture definition process was based on 2 principles. First, partners' brought-in technologies and software were considered as possible building blocks for the ALMANAC platform. Among such technologies, several components were results of previously funded[2] activities. As an example, the LinkSmart middleware was considered as possible communication backbone for inter-platform communication (federation). Second, a strong effort was allocated on the definition of clear boundaries between application-specific implementations and common platform services. These efforts lead to (a) a well-defined specification of the ALMANAC architectural components, and of their respective roles; (b) a hybrid specification approach mixing bottom-up technology gathering with top-down scenario thinking and elicitation.

Architectural activities during the second year of the project focused on architecture revision based on several aspects:

- Lessons learned from the implementation of the prototype applications, and required functionalities, and from performance evidence for all platform components;

[2] Mainly previous projects funded by the European Commission

- Security by design and privacy requirements of the ALMANAC platform, with respect to the envisioned adoption and possible misuses/attacks.
- Scalability requirements of the ALMANAC platform
- Federation between ALMANAC platform instances

These concerns impacted the ALMANAC architecture specification in a top-down manner and lead to better definition of concepts such as Trust, Role-based access control, P2P connection between platforms through improvements of the LinkSmart platform (namely LinkSmart Global Connect). Moreover, changes were pushed on the initial event delivery mechanism, which needed to effectively tackle event streams with frequency and cardinality much higher than those envisioned in the first release of the specification. The platform resource catalogue was part of this refinement effort as evidence gathered during the first year highlighted the need to handle an order of magnitude more sensors than those typically addressed by the initially contributed technology.

The third revision of the architecture builds upon results and outcomes from the 2nd year of the ALMANAC project, and in particular exploits the experience and feedback gathered from many real-world demos carried during dissemination activities (WP9) and, in the end of the year, piloting activities. This results into set of minor updates to the overall platform architecture, as reported in Section 5 and 6. Moreover, according to the adopted hybrid approach, while the foundations of the architecture are settling down and becoming increasingly solid and stable, cross cutting concerns analyzed from the earliest activities gain major focus and attract major efforts in the design and specification process. This, for example, results in a much more precise and structured approach to security issues, corresponding to the completion of the platform threat analysis and security design (already tackled in the first and second revisions of the platform architecture) activities. Moreover, lessons learned from the second year for what concerns portability, scalability and performance are distilled in updated and more impacting perspectives on the ALMANAC architecture, which thus gains increasing readiness to real-world, commercial, exploitation.

4.2 Reference standards

The process used for describing the architecture in this document is based on ISO/IEC/IEEE 42010 "System and software engineering – Architecture description" (ISO/IEC/IEEE 42010, 2011), which superseded the IEEE 1471 "Recommended Practice for Architectural Description for Software Intensive Systems" (IEEE 1471, 2000). This standard establishes a methodology for the architectural description of software systems.

According to the IEEE 42010 specification, an architectural description should identify the stakeholders of the system-of-interest that have, or express, concerns which are fundamental for the architecture design (i.e., that can significantly impact the corresponding design choices). Concerns, in particular, have to consider: (a) the purpose of the system, (b) the suitability of the architecture for achieving system's goals, (c) the feasibility, (d) the implied risks, (e) the maintainability and (f) evolvability of the system.

To better frame significant concerns with respect to architectural design choices, the IEEE 42010 standard introduces the notion of *viewpoint*. According to such a standard, "*a viewpoint frames a concern* when the viewpoint gives the architect the means to express that concern". For example, the formalism of Gantt charts frames concerns about project activities, schedule and dependencies, whereas other notations, such as UML use case diagrams, would not be helpful for modeling these concerns.

Viewpoints are therefore crucial to convey an information-rich view of a software architecture design and can be described as: collections of patterns, templates and conventions for constructing one type of view. Again, according to the standard, this supports "architects in using the right tool for the job when modeling the architecture".

A typical example of viewpoint is the functional one, which contains all functions that a system should perform, the responsibilities and interfaces of the functional elements and the relationships between them. These functions can be described, for example, by using UML diagrams.

Rozanski and Woods (Rozanski - Woods, 2005) propose a set of well-known, widely accepted, architectural viewpoints that can be exploited when describing a software architecture. For each viewpoint exactly one view shall be provided that includes[3]:

- Identifying and supplementary information as specified by the organization and/or project;
- Identification of its governing viewpoint;
- One or more architecture models that address all of the concerns framed by its governing viewpoint; and cover the whole system from that viewpoint. Each model:
    - Includes version identification as specified by the organization or project;
    - Identifies its governing model kind and adheres to the conventions of that model kind;
    - May be a part of more than one architecture view.
- A record of any known issues within a view with respect to its governing viewpoint. (Such as unresolved issues, known conflicts, etc.)

4.2.1 Relevant viewpoints

In the initial version of the ALMANAC architecture design process, the technical partners decided that the three most important viewpoints to be considered and described are the functional viewpoint, the deployment viewpoint and the information viewpoint. As better described in the following, such viewpoints have 3 corresponding views, respectively named Functional, Deployment and Information views. More in detail:

- The Functional viewpoint (whose view is reported in Section 6) describes the functional elements needed to meet the key requirements of the architecture. It presents proposals in a descriptive way and UML diagrams will assist in the understanding of the proposal. It describes responsibilities, interfaces, and interactions between the functional elements.
- The Deployment viewpoint (whose view is reported in Section 7) describes how and where the system will be deployed and what dependencies exist, considering for example hardware requirements and physical constraints. If there are technology compatibility issues, these can be addressed in this viewpoint as well.
- The Information viewpoint (whose view is reported in Section 8) describes the data models, the data flow and the data distribution over the described platform. Moreover, it also defines which data will be stored, and where, and, which data will be manipulated, and where.

Additional architectural perspectives will be used to address quality properties and cross-cutting concerns. One of the most representative examples of such a cross-cutting perspective is about security: threat analysis, data security and related functional elements are clearly spread over the above three different views and impact the architecture design at several, different abstraction levels. Similarly, scalability issues impact many parts of the ALMANAC architecture thus deserving a dedicated perspective.

4.2.2 Views and Perspectives

Given the above-defined relevant viewpoints, it is almost straightforward defining the views to be reported and documented in the overall ALMANAC architecture description (i.e., in this deliverable). They encompass: the Functional View, the Deployment View and the Information view. Cross-cutting concerns, on the other hand, include Security, Federation and Scalability.

The remaining sections of the deliverable are organized accordingly. In particular, *Section 6 – Functional view*, provides a view on platform components, with particular focus on the respective roles

[3]This is a small excerpt from the ISO/IEC/IEEE 42010 specification.

and duties whereas the *Section 7 – Deployment view*, analyzes the lessons learned and the technical solutions designed as part of the platform architecture with the precise purpose of facilitating the deployment in the real world, including aspects related to portability, adaptability, modularity and replicability. S*ection 8 – Information view* offers an orthogonal view on the platform identifying typical data flows, and volumes, and discusses related issues, expected cardinality and frequencies. *Section 9 – Security perspective*, describes the architecture under a security analysis standpoint. A full-stack threat analysis is performed and results are complemented by a set of recipes adopted to tackle and limit identified security issues. *Section 10 – Federation perspective*, describes the platform focusing on inter-platform communication, both from a technical and from an administrative/legal standpoint, whereas *Section 11 – Scalability Perspective*, presents the methodology and processes adopted to ensure the platform scalability under increasing workload.

5. High-level Architecture

Before diving into specific views on the ALMANAC architecture it might be worth getting an overall picture of the underlying aspects / concepts, also with respect to the temporal evolution of the design specifications. This section, therefore, provides a short summary of the design evolution through the past years of the project and an high-level overview on the latest platform logic architecture, including covered concepts / aspects.

5.1 Previous versions

5.1.1 Year 1

The first phase of the architecture definition process was intended to collect and categorize the technologies and software components that the individual partners of the ALMANAC project brought in. This bootstrapping ensured that partners' expertise had been quickly identified and exploited at the best. It also helped identifying gaps in the architecture that needed to be filled to achieve the ALMANAC vision of a smart city platform.

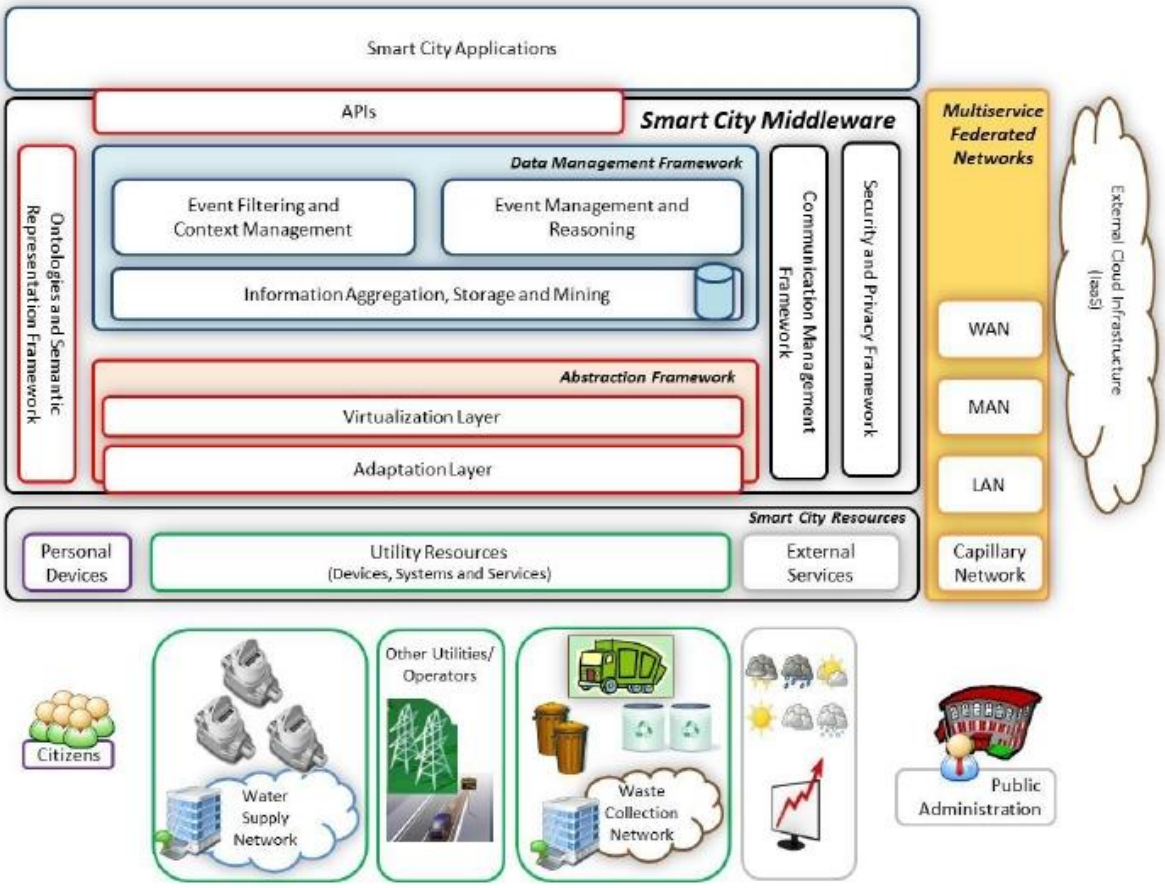


Figure 2. Initial vision of the SCP architecture (high-level).

Existing component descriptions were analyzed with respect to their position in the abstract architectural vision (reported in Figure 2) to check whether, and how well, required functionalities were covered. Missing components and modules covering advanced, yet feasible solutions to smart city issues were also included. The result of this process lead to the initial architecture definition.

In a synergic work with the other project work packages, the architecture specification took advantage of the scenarios developed in D2.1 (ALMANAC WP2, 2013), and instantiated them using the components already identified. This allowed crosschecking services provided by each component, and sketching the interactions between them. Similarly, initial discussions on application scenarios fostered

better API design, with a quite strong focus on open solutions, which is one of the relevant ALMANAC outcomes.

5.1.2 Year 2

The first revision of the architecture, carried at the beginning of the second year of the project, was based on the aspects emerging from both lessons learned in the first prototype implementations and from additional requirements gathered throughout the first year of the project. The main driving forces included:

1. Lessons learned from the implementation of the prototype applications and corresponding functionalities, together with preliminary performance analysis of involved platform components;
2. Security and privacy requirements of the ALMANAC platform;
3. Scalability requirements of the ALMANAC platform;
4. Federation between ALMANAC platform instances;

The main outcomes of the revision process, fueled by the above forces, impacted both single components and the whole platform architecture. In more detail, the design process lead to the changes described in the following.

Replacement of Event Manager

Addressing points (1) and (3), the event propagation was subject to major updates. While the communication paradigm didn't change, the protocols and components used did. The Event Manager (EM) was replaced by a MQTT Broker. With this change unnecessary overhead in Event Management due to the use of HTTP/SOAP Web Services was reduced. MQTT is, in fact, a protocol designed to deliver telemetry data with the smallest possible overhead. This includes, among other features, the possibility to apply "run-and-forget" delivery and byte encoding of data payloads, which ensures a very low performance impact for each single transmission. Adopting an MQTT-based communication, and, an MQTT broker, in the ALMANAC platform allowed for a much more lightweight and efficient event management. Furthermore, MQTT is a well-known standard in the IoT community.

Trust in ALMANAC

Considering points (2) and (3), a distinction was made between trusted communication between components within an ALMANAC instance and untrusted communication between ALMANAC instances. In the first case, initial security was limited to Transport Layer Security (TLS). In the second case, ALMANAC started to implement security and access control at the "external" borders of such instances (role-based access control, in particular). Furthermore, the capillary networks (represented by ETSI M2M) and external services and applications were placed in the untrusted zone to ensure worst-case security.

Role-based Access Control

Handling points (1) and (4), role-based access control of resources is managed by the Federated Identity Manager via policies. These policies control access through distributed Policy Enforcement Points (PEP) inside a platform instance. PEPs shall be located in the Abstraction Layer (SCRAL), the Virtualization Layer (Virtualization Layer Core, VLC) and the LinkSmart Network Manager. Interaction between platform instances have been managed by policies enabling trusted communication within federated ALMANAC networks.

LinkSmart Global Connect P2P

Considering aspects (3) and (4), the architecture allows communication between instances of the ALMANAC platform. This is handled by LinkSmart Global Connect, which implements peer-to-peer (P2P) transport communication, allowing federation of ALMANAC instances.

Resource Catalogue

To address points (2), (3) and (4), and as a lesson from the year 1 prototype applications, a directory of ALMANAC resources was required by several components. First, the security components needed a directory to address and verify their components, thus addressing point (2). Second, a platform instance must be able to manage and track new resources, as in point (3). Finally, the platform needs to find resources in a federated network of ALMAMAC instances, thus addressing point (4). For all these activities a directory of resources is needed, which has been implemented in the Resource Catalogue.

The continuous refinement process in WP3 led to a first update of the logical architecture (Year 2) as depicted in Figure 3. Architecture revision after the 1st year of the project. This updated version comprises the following subsystems of the ALMANAC platform:

- Abstraction Layer to abstract from physical smart city resources to virtual IoT resources
- Networking to enable communication within and across instances of the ALMANAC platform
- Security and Privacy Framework enabling trust-based communication and policy management
- Data Management Framework for fusing, storing and distributing data
- Semantic Representation Framework for modelling and management of semantic knowledge
- Virtualization Layer, exposing the APIs for searching, lookup, and addressing of smart city resources and services
- Cloud-based APIs providing to developers access to ALMANAC instances

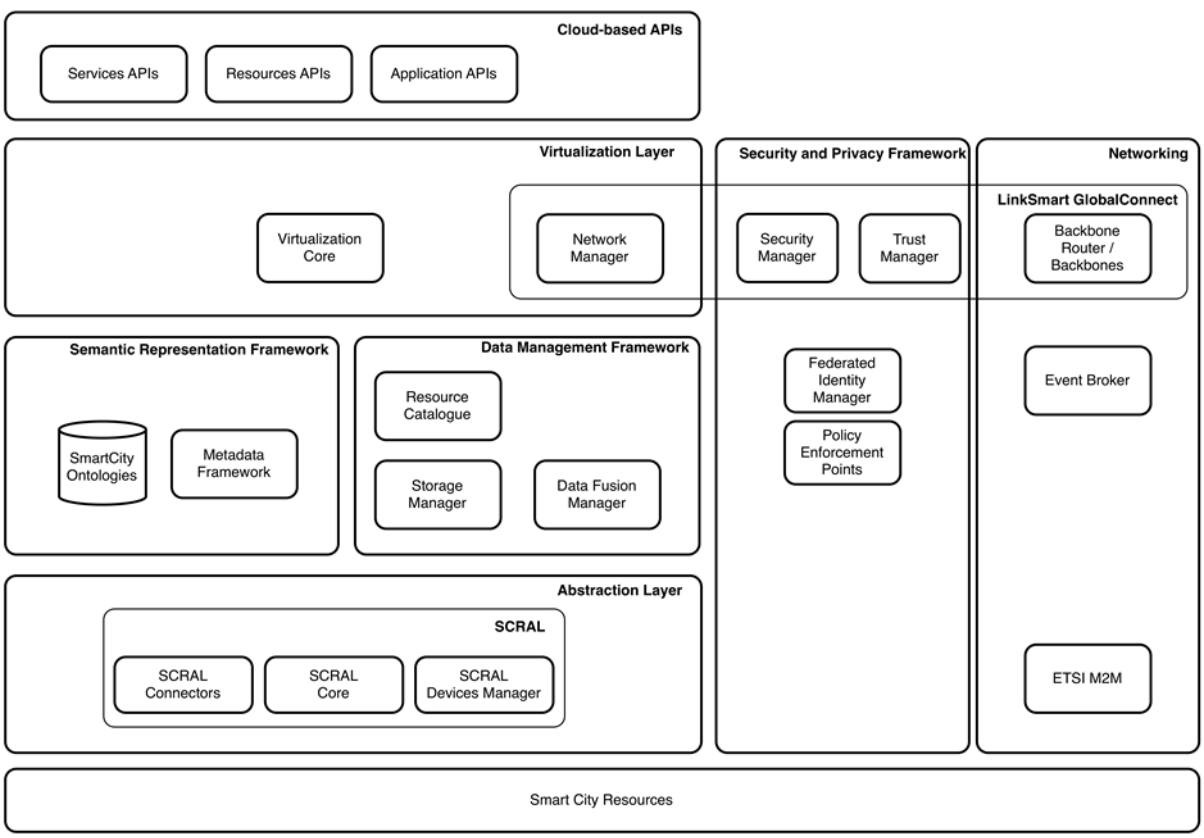


Figure 3. Architecture revision after the 1st year of the project.

5.2 Current design

The third revision of the architecture, defined at the beginning of the third year of the ALMANAC project accounts for insights gained throughout pilot and demo activities deployed in the second year of the project. Moreover it integrates additional hints and refined requirements gathered through WP2

and WP8 concurrent activities. Differently from the previous architecture revision, no major changes in the overall logic architecture have been defined, whereas few, minor, and specifically targeted amendments have been accomplished to better prepare the platform to actual real-world exploitation.

Among the modifications, it is worth citing:

1) a better re-organization of cloud-based APIs, integrating the corresponding specification work carried in WP7;
2) a better definition of provisioning services and application-level data access points, e.g. through additional API-level components;
3) some major improvement in modules orchestration and coordination at the Virtualization Layer;
4) increased support of diverse capillary and sensor level technologies including: planning tools for capillary deployment, different WSN pilots, third party services, etc.
5) improved metadata handling and definition
6) security integration at all platform layers

As can easily be noticed all updates have a low impact on the logic architecture of the ALMANAC platform, thus confirming the suitability of the second year revisions and the reached maturity grade. However, almost all amendments have significant impact on the exploitability of the platform in the real world, starting from the much more deep understanding and addressing of security issues, and going up to scalability-related policies and solutions tackled in this architecture specification update.

Figure 4 depicts the logic view of the ALMANAC architecture at the time of writing, with main changes highlighted in orange. Violet modules belong to the LinkSmart Global Connect and together with all other platform components in black, have been revised and amended, but with a minor impact.

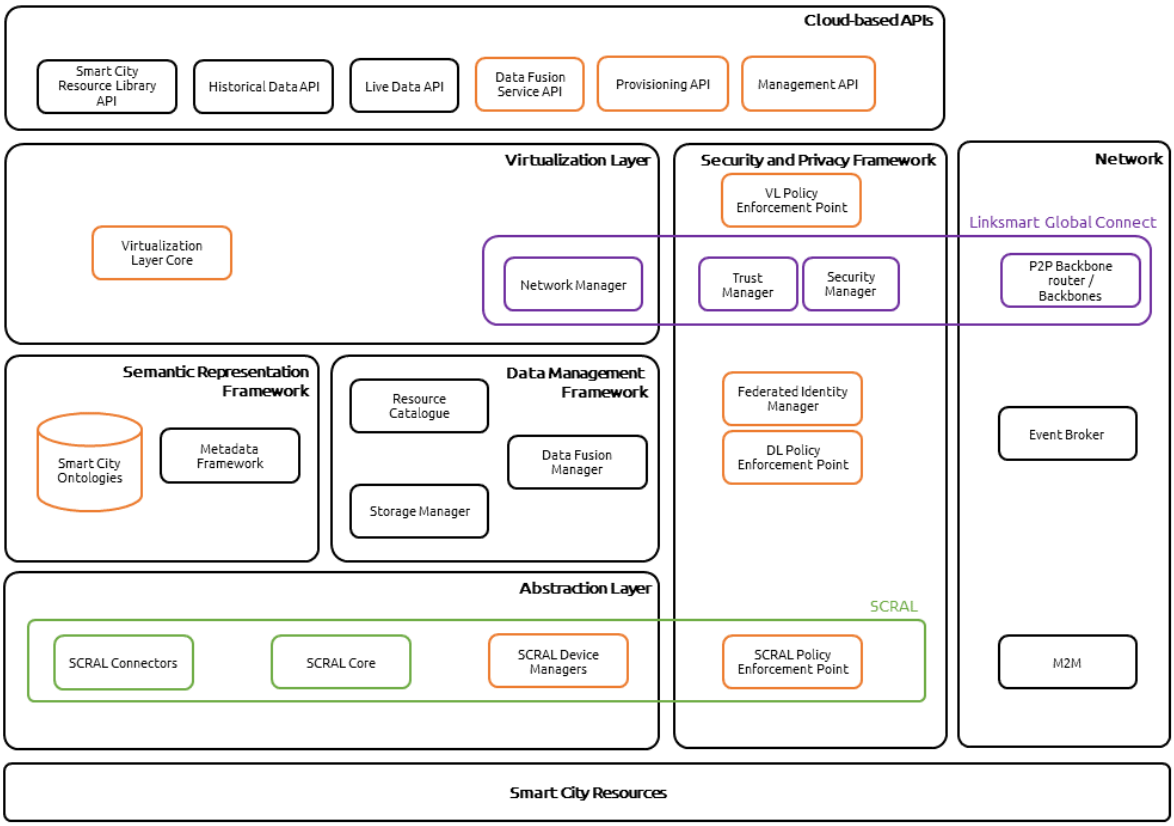


Figure 4. ALMANAC logic architecture overview.

The resulting architecture is almost perfectly matching the previous one, with components better specified and identified. It can be described along two main axis, depicted by vertical and horizontal divisions in Figure 4: the functional axis and the subsystem axis. While the functional axis, which is rendered as a stacked set of platform modules, deserves extensive description and is addressed by a dedicate architectural view, the subsystem axis (mainly horizontal) can be exploited to gain a quick understanding of the main platform functions. Subsystems composing the platform, in particular, encompass, from top to bottom:

- The Cloud-based APIs which have been updated to address feedbacks and suggestions from WP5, WP6 and WP7 activities. This level encompasses all modules acting as service access point for external, possibly third party, applications exploiting the ALMANAC platform. They mostly include REST endpoints mapping to the offered platform functionalities;
- The Virtualization Layer, which gained a much clearer role in orchestrating the platform modules to fulfill Cloud APIs requests, both within a single platform instance and among different instances (federation). Moreover it supports search, lookup, and addressing of smart city resources and services as in the previous architecture release;
- The Semantic Representation Framework, which encompasses updated smart city models based on ontologies as well as a revised and improved metadata management framework enabling effective handling of machine understandable metadata on smart city resources;
- The Data Management Framework, grouping all data-management modules that handle effective storage, elaboration and fusion of live data, and that offer directory services for resources registered in the platform.
- The Abstraction Layer, which provides technology independent, uniform access to physical resources and that maps both live and off-line metadata into shared, standard representations that can easily be handled by upper platform modules.
- The Security Framework, which crosses all layers and provides services and utilities to secure communication between platform modules (or components), to check access permissions and verify information disclosure on the basis of a policy mechanism. The framework encompasses several enforcement endpoints, where application/user permissions are checked and accept/deny actions applied, and one federated identity manager handling roles, permissions and supporting end-to-end control on transferred data, both inside and outside of the platform (see Section 9 for gaining deeper insights on this framework).
- The Networking subsystem, supporting effective communication between platform modules, among different platforms (federation) and between the platform and the outside world (e.g., through M2M communication).

Following chapters will provide deeper details on several platform aspects, in form of relevant views associated to viewpoints identified in Section 4.2.1, thus contributing to a better understanding of the platform architecture specification described herein.

6. Functional View

This chapter gives an overview of the different components of the ALMANAC platform, including their functionality, interfaces, and interactions.

6.1 Overview of subsystems

The ALMANAC platform is intended to be a middleware hosting multiple forms of applications; this function is also reflected in the list of its components. The platform provides 5 main services to applications:

- Interoperability over devices: enabled by the Smart City Resources Adaptation Layer (SCRAL), applications can access any kind of devices, whichever proprietary protocol they may speak, over a uniform, web-service based, interface. In addition to this service, the SCRAL exposes available metadata, and semantic information for connected devices and streams, and makes them first class citizens in the platform, e.g., by enabling the Virtualization Layer to access such an information when needed.
- Virtualization of services: enabled by the Virtualization Layer, the applications relying on the middleware do not have to know where the services or devices they consume are placed, or whether they actually exist. The Virtualization Layer provides service look-up mechanisms that bridge physical network boundaries, or can even wrap arbitrary data-sets – like historic measurements or cached values – as consumable services.
- Composition of rules and caching of data. There are multiple scenarios, where applications are not interested in current device values, but would rather like to be informed if specific thresholds are met, or see trends for particular intervals: this capability is provided by the Data Management Framework. The Data Management Framework provides support for storing, retrieving and managing metadata and, live and historical data as well as for filtering, aggregating, fusing and managing events.
- Semantic modeling and management of smart city resources. The Semantic Representation Framework supports the management and querying of highly structured graph-based RDF[4] domain models and metadata. It provides access to developers to the Smart City Ontologies through a range of service interfaces.
- Considering privacy policies of individual providers: while mainly required by providers of services and data sets, applications can also greatly benefit from knowing that they cannot run into the threat of invading privacy of individual services they consume. Service and device providers individually define policies regarding data they provide; these policies are enforced through multiple Security Enforcement Points (SEP) throughout the platform, thereby enabling applications to access only the data and functionalities for whom they have explicit access rights.

6.2 Smart City Resources Adaptation Layer

The Smart City Resource Adaptation Layer (SCRAL) provides a REST-based, uniform and transparent access to physical devices, capillary networks, systems and services for monitoring (and actuation) in a Smart City context. Specific device functionalities are uniformed, abstracted and mapped to a well-known set of functions and primitives complying with (device) models handled in the semantic framework of the ALMANAC platform. Moreover, due to its nature of interface between the ALMANAC platform and the real world, the SCRAL offers primitives for applying access-control, data-validation and role-based policies on field-level data sources (see Section 9). While typical SCRAL instances are distributed near to physical devices, meaning that more than one SCRAL instance is usually adopted in a single ALMANAC Platform Instance (PI), at least one instance is typically available in a PI. In all

[4] http://www.w3.org/TR/rdf11-primer/

deployments the SCRAL is part of the external attack surface of the platform, and must therefore provide mechanisms to enforce identity verification, access rights, perform data validation and support needed provisioning primitives.

The SCRAL internal architecture (see Figure 5) is organized in three layers, respectively named API, Core and Field-Access. The topmost layer exploits the SCRAL Connector component which exposes REST resources to the upper layers of the ALMANAC platform and the MQTT data source, which feeds the ALMANAC PI broker with both real-time data, purposely uniformed, abstracted and syntactically checked by the SCRAL, and with metadata about connected devices and resources. The core layer hosts core SCRAL components including the SCRAL event-delivery module, the Security Enforcement Point, the Data Validation module and the Metadata Generation component. Finally, the Field-Access layer integrates components (drivers) to wrap and isolate device-specific and technology-specific implementations used to access real physical devices and systems.

6.2.1 SCRAL APIs

The SCRAL APIs[5] are organized in 3 subsets, respectively named control, streaming and metadata, and can either be based on a standard REST transfer (for Request/Response interactions) or on an MQTT streaming protocol (for real-time data flows). Data formats are shared between the different communication channels. Following subsections summarize the 3 different subsets.

Control APIs

A RESTful interface exposing all the sensing and actuating features made available by interfaced devices (either directly or through network-level gateways). In principle, it is reachable through REST by all components of an ALMANAC PI. However, this control endpoint can only be called by the Virtualization Layer, for offering external actuation and querying APIs, no other components in the platform currently need direct access to the SCRAL control endpoint.

This endpoint does not explicitly support semantics (i.e., requests expressed in semantics-enabled languages such as OWL, SPARQL or JSON-LD), however, device representations and functions offered by this API are completely aligned to semantic models and representations defined at the Data Management Framework, thus enabling the platform to seamlessly integrate real-world data with the corresponding semantics-rich descriptions.

[5] Exposed to components part of an ALMANAC platform instance.

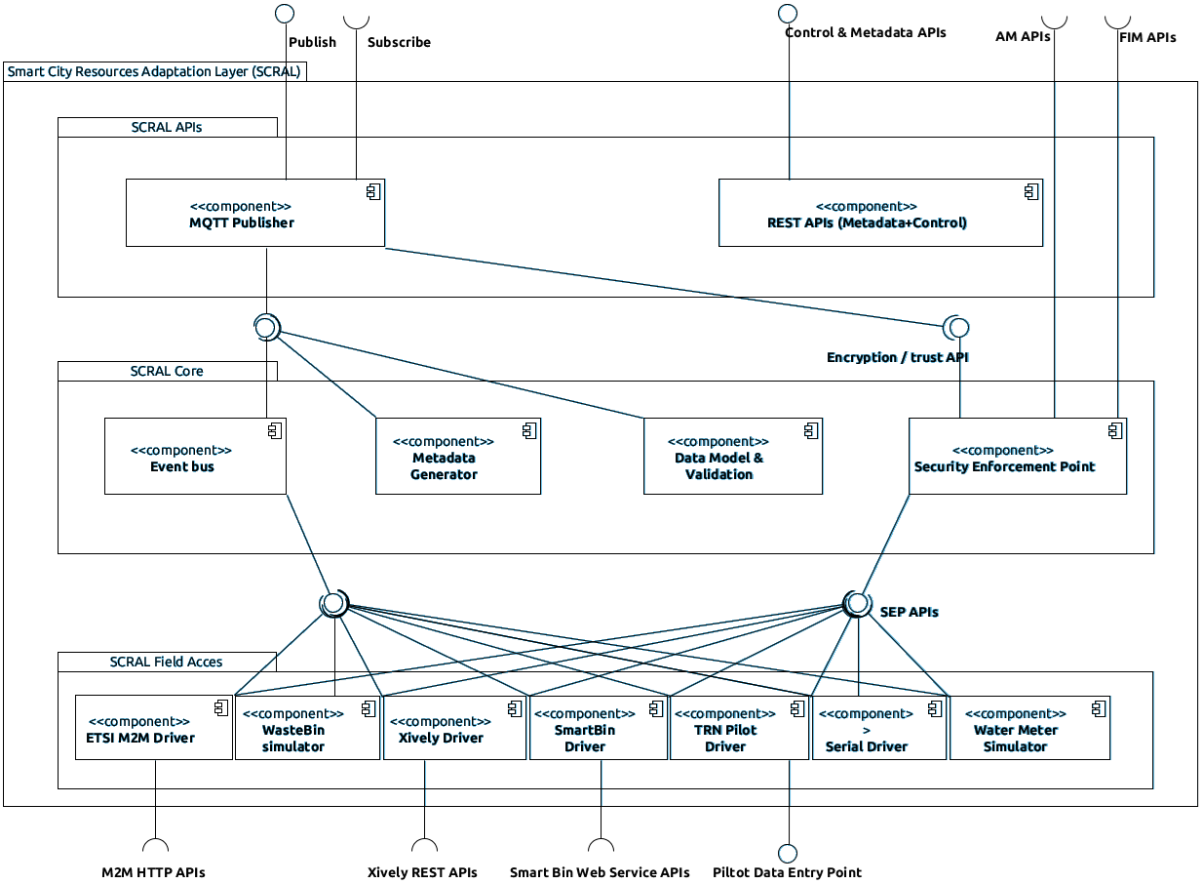


Figure 5: Smart City Resources Adaptation Layer (SCRAL) component diagram.

Streaming APIs

The SCRAL offers to the other components of an ALMANAC PI a constantly updated flow of measurement values and data, generated in real-time by connected devices and capillary networks. While each interfaced technology may have its own data generation pattern (e.g., polling vs event-based) and timing, the SCRAL converts such a data sampling into an event-based data-delivery model. In this way, measurement values, and in general observations, are conveyed asynchronously over a trusted, platform-specific MQTT data flow using a data format complying with the OGC SensorThings API specification[6] (namely, Observations).

Metadata APIs

Device metadata can either be discovered at the field-access level, or can be declaratively injected into the ALMANAC platform through the available provisioning services. Discovery, creation and management of metadata information about devices connected to capillary networks are managed by the SCRAL and made available to the rest of the platform by means of the SCRAL metadata endpoint. The endpoint exploits both a REST-based http interface and an MQTT-based communication channel. Metadata is typically delivered asynchronously, through MQTT as this better fulfils the dynamics of devices joining / leaving sensing and monitoring networks. However, the same data streamed through MQTT can also be gathered through REST APIs, thus allowing other platform components to request snapshots of metadata information regarding devices and services currently connected to the SCRAL. Formats are those defined in the ALMANAC reference framework and mainly stem from the semantic modelling activities carried during the project activities. In other words, exchanged metadata conforms to schemas and ontologies available in the platform through the Semantic Representation Framework.

[6] http://ogc-iot.github.io/ogc-iot-api/

More specifically, devices and resources are represented as OGC Sensor Things API "Things" and carry metadata information expressed in JSON-LD.

6.2.2 SCRAL Core

The SCRAL Core encompasses 4 main modules which enable the core abstraction and adaptation functionalities offered to the rest of the platform. They are respectively: the SCRAL Event Bus, the Metadata Generator, the Data Model and Validation module and the Security Enforcement Point.

The SCRAL exploits an inner event bus, based on the publish-subscribe pattern. The Event Bus can be seen as the backbone communication channel connecting all SCRAL modules at all layers. On such a bus all events coming from the Field-Access layer, as well as the events generated at the core level, are made available to the SCRAL modules, which can subscribe to the subsets they are interested in. The Metadata Generator is the module responsible for generating metadata associated to device entries generated at the field access level. It listens for new device additions, on the Event Bus, and generates the corresponding metadata by applying introspection to device representation classes defined in the Data Model. Generated metadata is attached to physical device information and becomes a new event flowing on the inner Event Bus, namely in the metadata stream. Such a stream, for example, will be used by the MQTT publisher module at the upper layer, to propagate device metadata to all ALMANAC platform components.

All devices and resources handled by the SCRAL are described through a uniform, technology-independent data model (set of annotated Java classes). Such a model, as well as all the tools and utilities needed to perform data validation and conversions is hosted by the Data Model & Validation module.

Finally, the Security Enforcement Point module provides means to safely operate on real world installation. In particular, it checks identity of connecting devices, by contacting the Federated Identity Manager and verifies associated permissions and roles (through the Access Manager component). Moreover, it provides facilities to secure the transmissions between the SCRAL and the other platform component on the trusted zone (e.g., TLS encryptions and message signing) and to verify the data provenance at the field level trough signature mechanisms, etc. The role played by the SCRAL SEP in the overall picture of ALMANAC security is better described in Section 9.

6.2.3 SCRAL Field access

Modules implementing direct interfaces to real and simulated sensor networks, i.e., device drivers, are located at the SCRAL Field access layer. For each interfaced protocol, a dedicated driver module is provided, able to convert low-level sensor data into the high-level, technology-independent representation handled by the upper SCRAL modules. The "pluggable" nature of drivers allows to scale horizontally to multiple, heterogeneous, sensor networks and to finely tune each SCRAL installation on the basis of the actual abstraction needs. Among all the available drivers, which are continuously being improved and integrated, simulators deserve a special mention. Currently two simulation drivers are included in the SCRAL respectively addressing waste bins and water meters. They offer very simple behaviour simulation with a tuneable throughput / cardinality, which permits to stress the remaining parts of the SCRAL for scalability testing purposes. As an example a single instance of waste bin simulator can easily generate data for over 400k sensors, at the usual rate of one measure every 10 minutes. This, on one side, supports testing the SCRAL component in realistic load conditions, and on the other hand allows simulating sensors not yet connected to the ALMANAC platform, but for which information such as type and location, is available.

6.3 Virtualization Layer

The Virtualization Layer is directly exposed to external applications and their users, with a mission to facilitate the use of features offered by internal ALMANAC components. It enables search, lookup and addressing of services registered to the ALMANAC platform. Through the Virtualization Layer, applications can communicate with the ALMANAC platform; for instance, they can show ALMANAC data to citizens or city authorities. Applications can access the full functionality of connected IoT

resources, including actuation of IoT devices, provided that they are valid users of the platform and that they have sufficient permissions.

The Virtualization Layer is composed of two main components, namely the Virtualization Layer Core (carrying the main logic) and the LinkSmart Network Manager (for communicating with other ALMANAC platform instances). The Virtualization Core communicates with the other ALMANAC components through HTTP REST and MQTT, and with external applications though HTTP(S) and WebSocket.

The Virtualization Layer Core is in charge of several kinds of "virtualizations":

1. Proxying of requests/responses and of events to/from the different ALMANAC internal components, making the ALMANAC instance look like a single (Web) service to the eye of external applications. This activity is subject to access and identity verification to prevent external apps / users to access data they are not entitled to.
2. Routing of requests/responses either to internal components of the same ALMANAC instance, or to another ALMANAC instance when needed (through the LinkSmart Global Connect component), leveraging the federated nature of the ALMANAC platform.

   Requests may require aggregating responses from several ALMANAC instances of the federation. For instance, if a client makes a "federated request" for the Resource Catalogue, the Virtualization Layer will query the local Resource Catalogue instance as well as the other Resource Catalogues instances of the federation, before aggregating the responses into one coherent response to be sent back to the client. To the eyes of the client, such a federated request will make the distributed Resource Catalogues appear like one single but "bigger" Resource Catalogue.
3. Wiring ALMANAC components in such a way that some requests requiring the collaboration of several ALMANAC internal components may be executed by external applications in a single call.

   For instance, a request may require discovering some IoT resources via the Resource Catalogue or the Semantic Framework, querying for historical data from the Storage Manager, and aggregating resulting information before a response could be returned to the external application.
4. Transforming requests/responses and corresponding payload, in selected formats not supported natively by the ALMANAC internal components. This aims at easing the interaction with third-party applications that are not ALMANAC-aware but may exchange data with ALMANAC through standard formats.

The Virtualization Layer may, finally, accept some queries using "real world" terms. These might include a postal addresses or town name. It will then convert the query to an ALMANAC internal format (e.g. using GPS coordinates). Such a functionality requires a third-party database or API such as the Google Maps Geocode API[7].

6.4 Data Management Framework

The Data Management Framework provides components for storing, retrieving and managing metadata, and live and historical data, as well as for filtering, aggregating, fusing and managing events. Figure 6 reports the main framework components.

[7] Example of Google Maps Geocode API request: https://maps.googleapis.com/maps/api/geocode/json?address=Torino

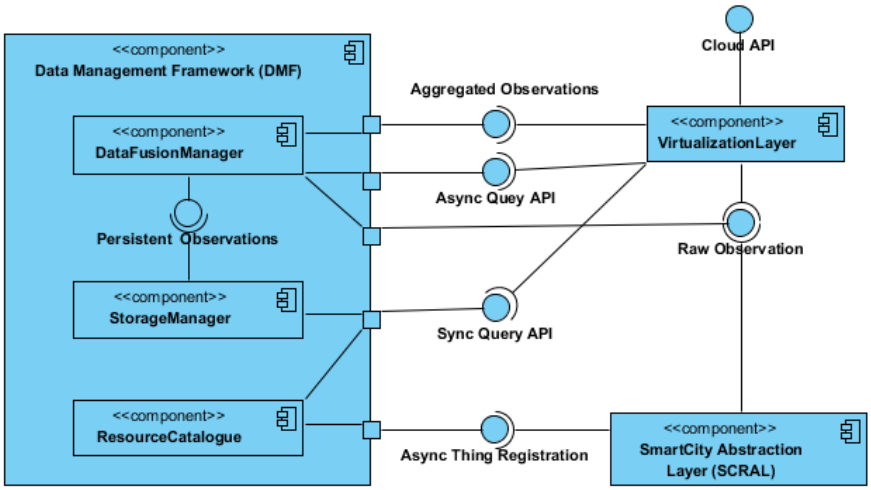


Figure 6. Data Management Framework components and external interfaces.

6.4.1 Data Fusion Language and Data Fusion Manager

The Data Fusion Language (DFL) is a high level, technology independent language for Complex Event Processing (CEP) and Data Fusion operations in the ALMANAC platform. The DFL model and syntax is based on previous results adapted to meet the requirements specific for smart city applications. The DFL design builds on the Pipes and Filter design pattern, providing time windows and stream based processing of events. In the ALMANAC DFL, events are defined as data structures carrying an OGC Sensor Things API compliant payload, which represents an *Observation* instance.[8] The DFL provides the means to exploit multiple event streams. A simplified example in the ALMANAC waste management domain is bad smell detection, which can be implemented by a data fusion process merging information from the current fill-level of monitored waste bins and the current air temperature (see Figure 7 for the corresponding DFL graphical representation).

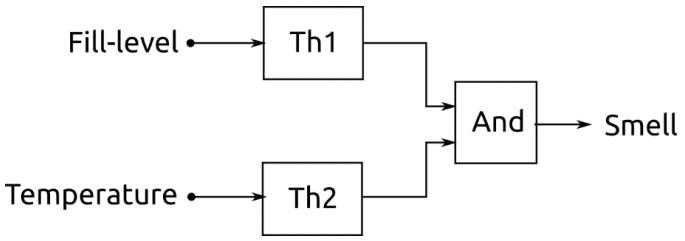


Figure 7: Fusion of multiple event streams using threshold filters

The DFL implementation provides a high-level interface to different underlying Complex Event processing technologies. A REST API is designed for the management of DFL queries, and is being implemented on top of the Data Fusion Manager.

[8] An Observation instance is classified by its event time (e.g., *resultTime* and *phenomenonTime*), *FeatureOfInterest*, *ObservedProperty*, and the procedure used (often a *Sensor*).

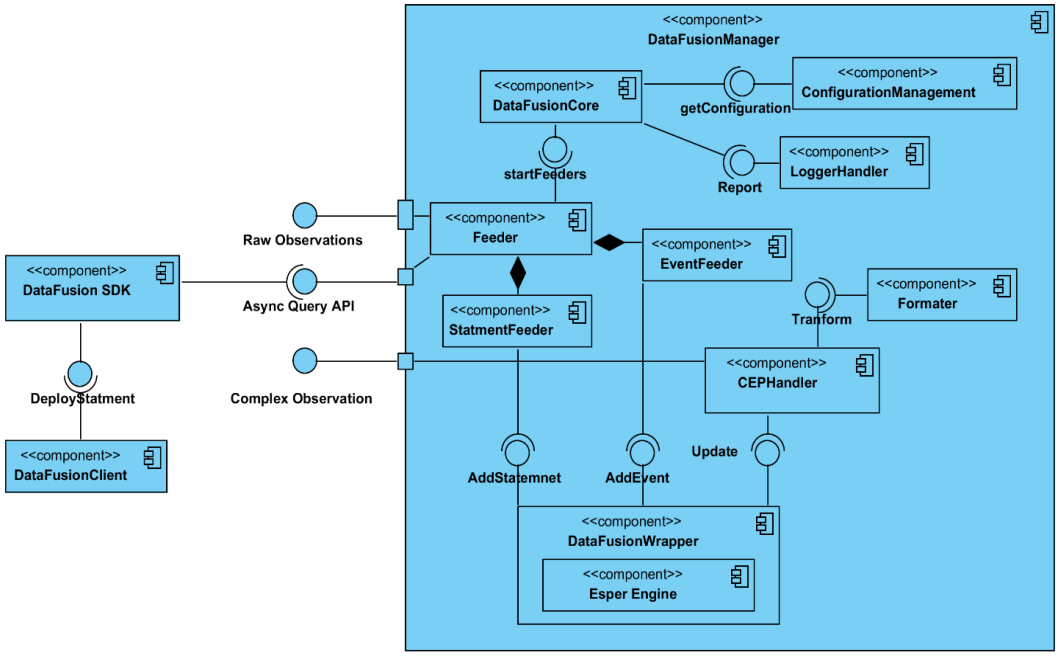


Figure 8: Data Fusion Manager Components

The Data Fusion Manager realizes the execution environment for CEP using the ALMANAC DFL for event handling and data fusion operation definition. The Data Fusion Manager realizes what is commonly referred to as an *event processing agent* in the CEP domain. There are several existing approaches to CEP, both in research and industry, offering different features, advantages and disadvantages. Therefore, the innovation focus for the integration of such engines into the ALMANAC framework, is on providing a proper selection of components with respect to the requirements.

6.4.2 Resource Catalogue

Resources (specifically, Network Resources in the IoT-A jargon) are software components that provide data or are used in the actuation of devices. These resources provide services for applications and end-users, e.g. retrieving and analyzing data about the physical world, invoking actions and so on. An OGC Thing is defined as "an object of the physical world (physical things) or of the information world (virtual things) that is capable of being identified and integrated into communication networks."

The currently available Resources, or Things, exposed by the SCRAL are discovered and managed by the Resource Catalogue, which keeps a cache of Things discovered so far. The Resource Catalogue provides a REST-based interface to select and retrieve data about resources and their services. The IoT Resource Catalogue also provides an integrated way of querying and invoking services on matching IoT Resources.


```
{
    "Thing": [
        {
            "id": "5c7b6daeab2b59fda151da37b25b6527fb41231a111ca3249b6e8a21b2dcf78f",
            "Description": "The DryWasteBin connected to the WasteBinSimulator network.",
            "Metadata": "http://almanac-project.eu/ontologies/smartcity.owl#DryWasteBin",
            "locations": [
                {
                    "Time": "2015-10-22T08:36:11.118Z",
                    "Geometry": {
                        "type": "Point",
                        "coordinates": [
                            "7.65097805",
                            "45.07252262"
                        ]
                    }
                }
            ],
            "Datastreams": [
                {
                    "id": "28d1be5761fa2ad6611628c467a0c0c2027e27220f20b6c591530ad8dbf27839".
```


Figure 9: OGC SensorThings Thing

It has been extended with functionality for querying of resources by location (address or WGS 84 coordinates) and receiving resource data updates in OGC format. With respect to the second year architecture release, querying and data retrieval over large numbers of resources has been optimized.

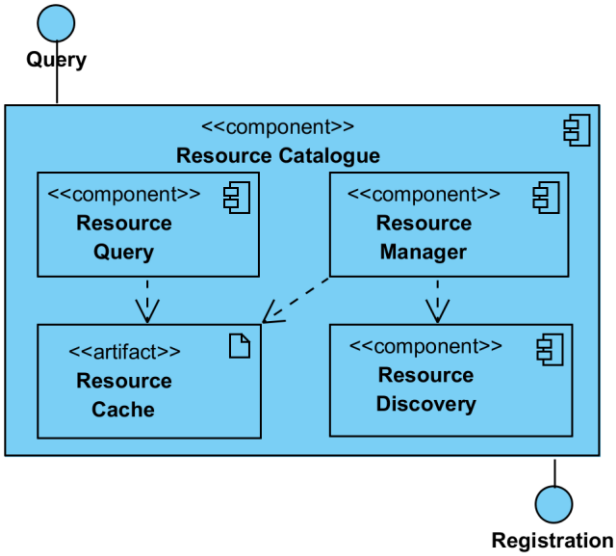


Figure 10: Resource Catalogue Components.

6.4.3 Storage Manager

The storage manager provides persistence for time series data generated by the SCRAL and the Data Fusion Manager. It provides interfaces for storing and querying time-stamped events and measurements. The Storage Manager for a specific ALMANAC PI holds the data generated by the SCRAL(s) and Data Fusion Manager for that platform instance.

The requirements and available resources of ALMANAC PIs may sensibly vary. One platform may have a limited budget and limited storage needs, another might need to store huge amounts of historical data for analysis, and a third may already have and in-house storage solution set up and want to keep the data there. A number of databases exist that are suitable for storing and querying large amounts of time series data [D6.1 A scalable data management architecture for Smart City environments]. The interfaces and query languages for these are not standardized and mostly product specific. The Storage Manager acts as a transparent proxy allowing for integration with different storage solutions based on the requirements and resources of the provider of a specific ALMANAC PI. This product-specific implementation is made by the Storage Adapter.

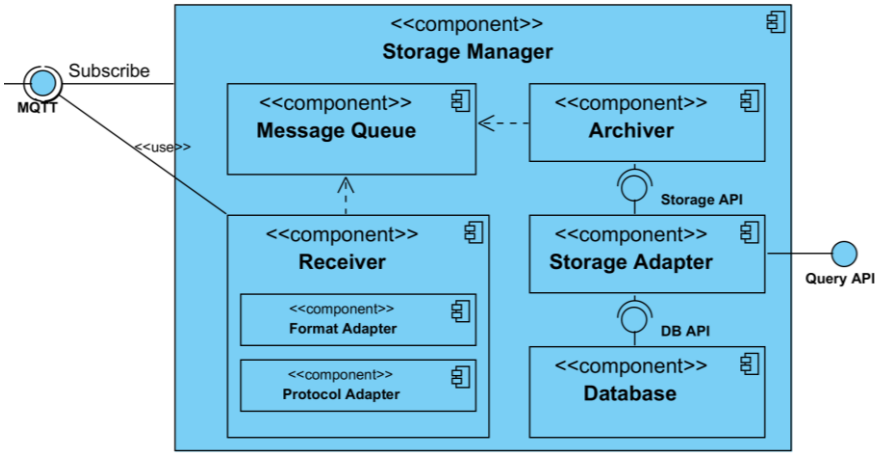


Figure 11: Storage Manager Components.

The writing, or ingestion, of data in the Storage Manager is separate from the querying functionality. The Receiver component can be used to accept different combinations of protocols and formats, e.g. MQTT or AMQP with OGC SensorThings or SenML[9] payload. The events are put by the Receiver on the Message Queue, which acts as a buffer to smooth high-frequency peaks in the received events. The Archiver component reads data from the queue and uses the Storage Adapter to put the data in the database.


```
{
  "@odata.context":"http://cnet006.cloudapp.net/DmfBeta/$metadata#Observations","value":[
    {
      "Time":"2015-10-
22T10:17:55.403Z","ResultValue":"74","ResultType":null,"ID":"3c763d88fd2abb3b28d10b76affa
81561d167f65ed68675463abe89b9e5effe0635811058754030000","Datastream@odata.context":"http:
//cnet006.cloudapp.net/DmfBeta/$metadata#Observations('3c763d88fd2abb3b28d10b76affa81561d
167f65ed68675463abe89b9e5effe0635811058754030000')/Datastream/$entity","Datastream":{

"Description":null,"ID":"3c763d88fd2abb3b28d10b76affa81561d167f65ed68675463abe89b9e5effe0
"
      }
    }
  ]
}
```


Figure 12: OGC SensorThings Observation

The Storage Manager has been updated according to the decision to use the OGC Sensor Thing standard (reported in D3.1.2). This standard specifies a data model, a message format and an API for accessing and querying IoT data; namely the OGC SensorThings Sensing Profile, based on OData. The Storage Manager Query API now supports queries using the Sensing Profile API and the Sensor Things JSON representation as well as with the former IoTEntity XML format and API (reported in D6.1 M12). The Receiver and Archiver submodules (Figure 11) also accept data in the OCG SensorThings JSON format.

The Query API implements a subset of the OGC SensorThings API Sensing Profile. Complex queries involving type information or other metadata are resolved first using the Semantic Representation Framework (SRF), then the Storage Manager is queried for specific sets of data.

The Storage Adapter MongoDB implementation used in ALMANAC has been updated with a new internal schema to better support the needs of the ALMANAC platform for high volume, low granularity data. Basic archiving functionality to move old data out of the main database instance has been implemented.

[9] SenML IETF Working Draft, http://tools.ietf.org/html/draft-jennings-senml-10

6.5 Semantic Representation Framework

This functional layer is responsible for the management and querying of highly structured graph-based RDF[10] domain models and metadata. It exposes a range of native (Java) and remote (HTTP) service interfaces allowing for various interaction modes either focusing on invocation of ad-hoc, or persistent queries (Remote Procedure Calls, RPC[11]), or the exchange of semantic data resources (REST[12]). In the context of the ALMANAC architecture, SRF logically complements the Storage Manager and the Resource Catalogue components. While the former excels in querying of structure-rich data and its enhancement by means of semantic inference, the second is optimized for bulk storage and querying of large volumes of measurement data, events and the third is specifically suited for storing high numbers of resource instances.

An extensive analysis and specification of the Semantic Representation Framework SRF (originally called "Semantic Representation Layer") was given in deliverable *ID 5.4 Ontologies and Semantic Representation Layer Prototype* (M15). The final version ID 5.4.2 will be published in M30.

The generic re-usable parts of SRF are being developed as part of the open-source LinkSmart[13] Metadata Framework (LDF). The Metadata framework acts as a transparent proxy to any state-of the art RDF store compatible with the SPARQL 1.1 Protocol[14] and considerably augments the standard repository functions and interface coverage, e.g. by support for persistent SPARQL queries and updates, and implementing RESTful management of graph fragments (also known as "semantic resources").

Building upon the LDF infrastructure, the Semantic Representation Framework comprises RDF vocabularies (ontologies), corresponding SPARQL 1.1 query and update statements and custom services.

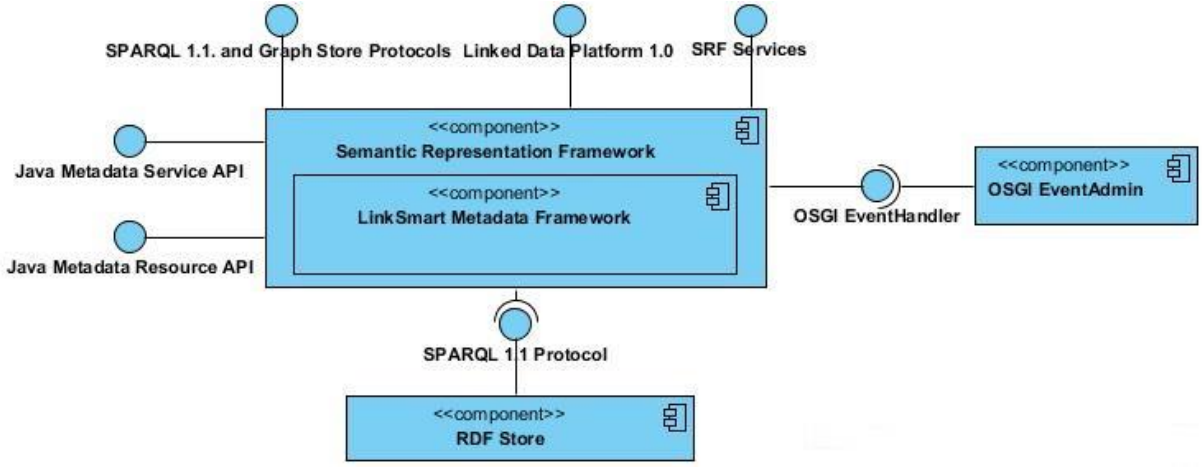


Figure 13: Semantic Representation Framework components and interfaces

Various ALMANAC components leverage SRF in order to implement particular functionality:

- *Virtualization layer* mediates access to internal SRF interfaces and resources as for many other ALMANAC components, thus allowing for location transparency and request federation. It may utilize linkage metadata for resolution of actual requests targets.
- *SCRAL* (Metadata Endpoint) may use SRF to transparently publish, retrieve and query device metadata and service descriptions.

[10] http://www.w3.org/TR/rdf11-primer/
[11] http://en.wikipedia.org/wiki/Remote_procedure_call
[12] http://en.wikipedia.org/wiki/Representational_state_transfer
[13] https://linksmart.eu/
[14] Examples of such graph stores are Apache Fuseki or the Openlink Virtuoso Server.

- *Resource Catalogue* may resolve resource identifiers based on advanced attribute queries.
- The meta-data of network capabilities and resources may support an intelligent management of the capillary network infrastructure.

For the development of Smart City applications, the SRF, described in section 6.5, provides access to the ALMANAC Smart City Ontologies.

6.6 Security and Policy Framework

The Security and Policy Framework goal is to secure components that need to interact with the untrusted area. To serve this purpose, the Security and Policy Framework manages federation users and decides what level of access a single federation user is granted. In addition, the Security and Policy Framework protects the user's information and guarantees that no one can impersonate a federated user. It also allows developers and operators to create a role-based access control system for their federation, which can be adjusted to the specific requirements.

From a technical standpoint, the ALMANAC Security Framework is mainly composed by 2 modules respectively handling access management and enforcing federation rules and agreements. They are referred to as Access Manager and Federated Identity Manager. These core elements are complemented by Security Enforcement Points distributed at the platform boundary, according to the XACML and OpenID Connect reference architectures.

Figure 14 shows the architecture of the Security and Policy Framework consisting of the three components:

1. Federated Identity Manager (FIM)
2. Access Manager (AM)
3. Security Enforcement Point (SEP)

Note that the SEP replaces the Policy Enforcement Point (PEP) from the previous deliverable (D3.1.2). Following subsections better detail the role of each module.

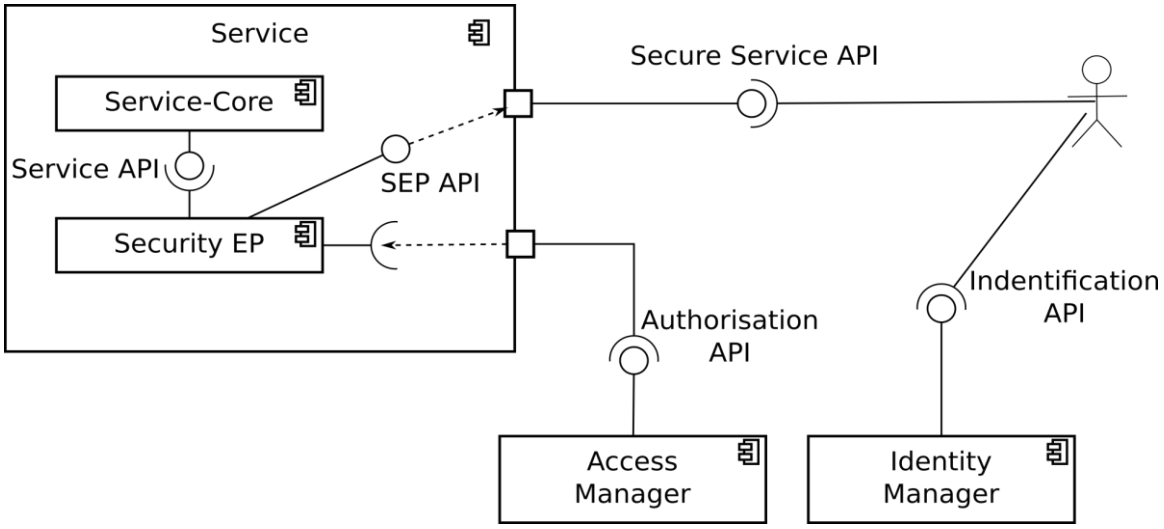


Figure 14. The component diagram of the security and policy framework.

6.6.1 Federated Identity Manager

The FIM has two purposes.

1. Registers the users of the federation
2. Provides a sign-in mechanism for federation users, based on OpenID Connect 1.0.

6.6.2 Access Manager

The Access Manager (AM) provides the following features

- Role-based access control, ensuring that only federation users with the proper authorization can access certain data, based on their role and, the properties of their intended operation.
- Centralized policy management to reduce the complexity associated with managing authorization policies separately across multiple ALMANAC PIs and internally across ALMANAC PI components.

The AM is following the XACML standard.

6.6.3 Security Enforcement Point

The Security Enforcement Point (SEP) is responsible to enforce security requirements of the components that need to exchange information with the untrusted area[15].

The SEP replaces the PEP from the previous deliverable, because the emphasis of the PEP was focused on the authorization. The Confidentiality and the Authentication were implicitly assumed. The SEP makes these assumptions explicit. The components that will make use of the SEP are:

1. The SCRAL - It communicate with the sensor networks
2. The VLC - It communicates with the federation user
3. LinkSmart - It communicates with outside PIs via LinkSmart instances.

All three modules share the requirements that:

- they have to protect the information they handle,
- a federation user has to prove his identity before he can access the service, and
- it must be checked if the federation user has the proper authorization.

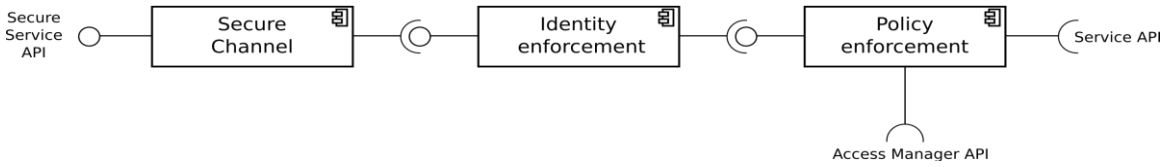


Figure 15. The components diagram of the security enforcement point.

To satisfy these security constraints the SEP itself is composed of three sub components that will enforce the above 3 requirements. Figure 15 shows the components of the SEP and how they are interfaced.

Secure Channel

Establishes a secure channel between the two communicating parties and rejects insecure connections. This is done via TLS 1.2 at the SCRAL and the VLC. The LinkSmart component has its own secure channel implementation.

Identity enforcement

Requires a proof of identification from the federation user and rejects unidentified users. For the VLC and the SCRAL this is done via the OpenId Connect 1.0 protocol and by checking the signature of the FIM that is attached to the JSON Web Token (JWT).

[15] See the Section 9.

Policy enforcement

Connects to the Access Manager (AM) via a secure channel, receives the policy decision and enforces that decision. If any of the checks that one of these components performs fails, the user connection is dropped.

7. Deployment View

This section discusses aspects related to the ALMANAC platform deployment, either at the platform- or at the capillary network level. Subsequent paragraph better detail such aspects tackling deployment of platform instances, of cloud-based solutions and federations. Moreover, issues related to the deployment of capillary networks inside the complex and heterogeneous environments offered by real cities are also considered.

7.1 Aspects of ALMANAC Deployment

7.1.1 Platform Instances (PIs)

The *deployment view* describes how and where the system will be deployed and what dependencies exist. In particular, we describe the deployment of ALMANAC Platform Instances (PI) and their constituent components, and, the integration of the ALMANAC Capillary Network. Two main design choices have influenced the architecture in this respect:

It is assumed that not all components defined in the ALMANAC platform specification must be part of a specific platform instance deployed in the real world. This accounts for different requirements depending on the deployment purpose (private vs public utility installation), context (small city vs utility vs large administrative regions), and so on. For the sake of clarity, under these conditions it is perfectly possible that single, isolated PIs could not include the LinkSmart connection modules, whereas PIs deployed on low-power devices might not include full versions of the Data Fusion Manager and, might avoid local storage taking advantage of other federated, and more powerful platform deployments.

Components participating to a single platform instance do not need to be singleton nor to be physically deployed on the same server, in the same location. Thanks to the DNS-based naming scheme discussed in the previous paragraphs, components can in fact be safely identified, together with the platform instance they belong to, thus moving the loosely coupled architecture of ALMANAC a step further in the current state of the art.

7.1.2 Hosting and Cloud based deployment

Depending on the requirements of the particular ALMANAC PI, the computational nodes where the PI is deployed may be local servers at the organizations responsible for the PI, cloud servers, or a combination of both.

Components in the ALMANAC PI may also be integrated with existing systems. E.g. existing measurement databases may be provided with a Storage Manager database adapter to integrate directly with existing historical data, existing domain models in business systems can be described using the Semantic Resource Framework and sensors (or other data stream sources) can be integrated through the SCRAL.

7.1.3 Federated deployment

ALMANAC defines the concept of federation, i.e., the establishment of loosely coupled networks of Platform Instances in different organizational contexts (e.g., agencies and actors in a city); see Section 10 for more details. The federated deployment implies both technical as well as organizational interoperability supported by open technical standards in combination with business agreements.

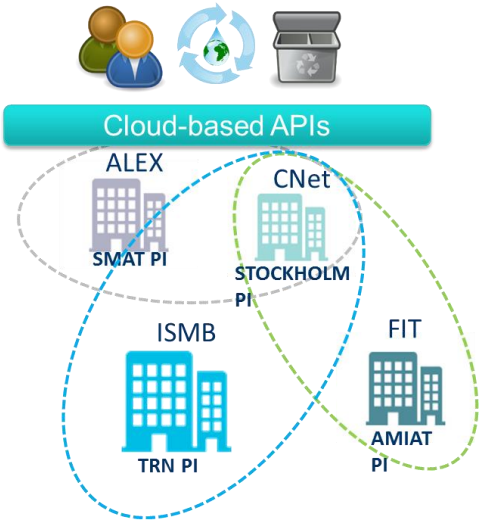


Figure 16. ALMANAC Federated deployment supports complete networks of service providers and consumers sharing large numbers of Smart City resources

The ALMANAC federation concept and its deployment is further elaborated in the Federation Perspective (Section 10).

7.2 Network Infrastructure Deployment

ALMANAC does not foresee the deployment of dedicated networks to support Smart City applications. At this purpose, the ALMANAC network architecture strives to maximize the re-use of pre-existing communication infrastructure both on the capillary and the core network segments, at the same time devising techniques to achieve an adaptable and scalable networking solution for smart cities.

A view of the ALMANAC network is described in Figure 17. The ALMANAC platform is an internet-oriented middleware-based distributed system. It is accessible from any system directly joining the middleware domain (i.e. as a LinkSmart entity) or also by any type of internet-oriented application (e.g. mobile applications, web applications, etc.) through Open Cloud-based APIs.

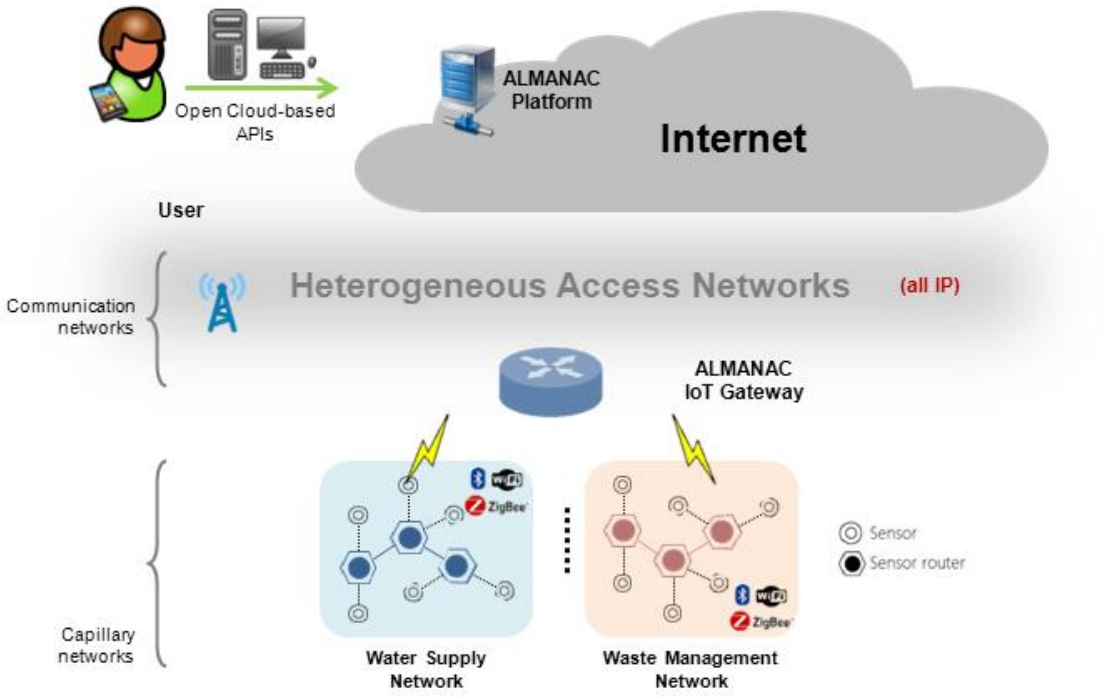


Figure 17: ALMANAC Network View

IoT Gateways are IP-based devices and can be thus directly connected to the internet or, in case this is not possible, can leverage heterogeneous Access networks to reach the ALMANAC platform. Examples of Heterogeneous access network include e.g. 3G mobile networks, public Wi-Fi networks.

From the networking point of view, the ALMANAC IoT Gateway has also the role of inter-connecting several types of local wireless network i.e. capillary networks. Capillary Networks are a flexible and autonomous communication networks normally use to locally collect information from sensors in the smart city. Examples of capillary network include short-range networks based on Wireless M-Bus or DLMS-Cosem e.g. for utility metering (gas, water, electricity), collection of waste management data, pollution and traffic control sensors, smart lighting sensor, heating control sensors, etc.

Capillary Network Standard

The main reason for introducing capillary networks is to avoid excessively extended deployments of traditional infrastructures which may be too expensive and energy consuming when considering millions of metering devices that should be able to work several years without battery changes and that generate a fairly limited data traffic.

The capillary networks provide an infrastructure to collect data from different devices (sensors, meters, etc.) and ensure their collection based on an ETSI M2M compliant service platform16. The basic concept of the standard is the Store and Share of data coming from smart devices. Data is stored and then shared with the Apps by the M2M Platform with standard APIs (http/REST based).

The ETSI M2M compliant service platform represents the first collection point of the data provided by the Capillary Networks and provides to the network applications (i.e., ALMANAC PIs) a set of functionalities to manage and operate the sensor data. The ALMANAC interface to the M2M platform is provided by the SCRAL component.

[16] See WP4 deliverables for full functionality and references

Waste Management and Water supply network examples

In Figure 18, the water supply network and the waste management are brought as examples of concrete capillary networks used in ALMANAC.

The capillary water supply network integrates water leak sensors and flow meter sensors -among others- to monitor the water consumption behaviour of the city and eventually detect possible issues related to this matter. In this capillary network, water metering sensors can communicate via an IoT gateway through different network interfaces such as ZigBee, Bluetooth, 6LoWPAN, Wi-Fi, etc. The sensors in the water supply network are considered to work using the DLMS/COSEM standard.

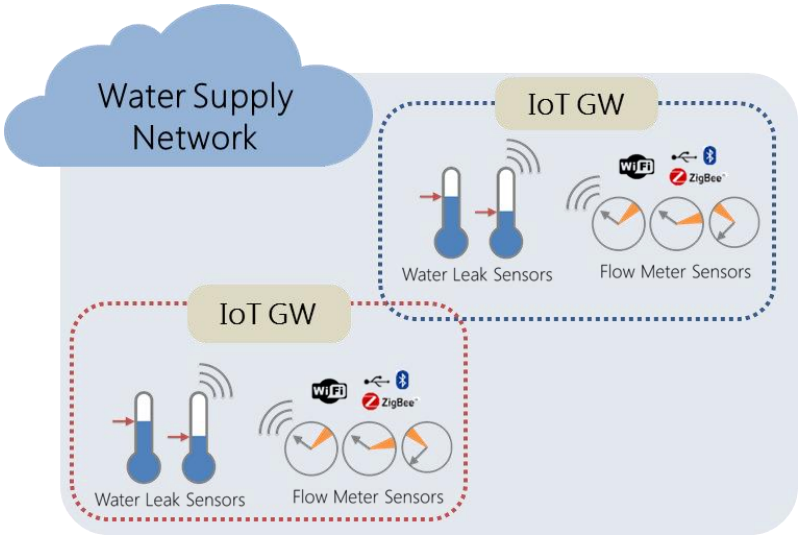


Figure 18: Water Supply Network

In the capillary waste management network, fill level sensors are integrated with meteorological data and geographical location data in order to create virtual sensors, that can provide information useful to predict possible issues related to the waste collection, or to forecast the waste generation quantities in a determined area. Analogously to the water supply network, in this case sensors can communicate to the IoT gateway using different network interfaces.

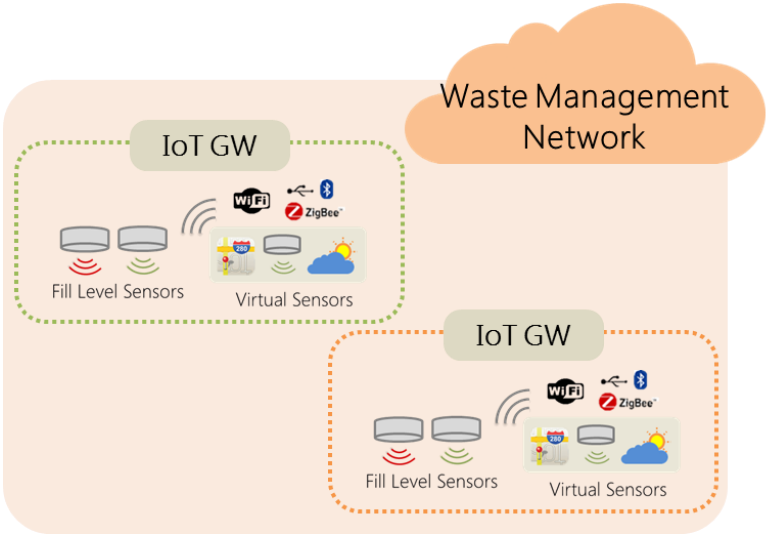


Figure 19: Waste Management Network

8. Information View

Information is a key enabler for all smart city processes, as it allows directing the decision and planning processes on the basis of real evidence. As such, data must be considered as a first class citizen in whatever Smart City Platform. This is particularly true in ALMANAC, since the project aims at fostering actual changes in real-world city processes and policies. Harvesting useful information from data generated by a smart city, however, is not a trivial task. In fact, it has to deal with huge differences in formats, frequencies, cardinality and semantics of information. For example, the same platform is often required to handle real time monitoring of physical quantities, such as temperatures or water flows, as well as more static information such as device descriptions, waste collection schedules, etc. These different types of data streams need different handling, on one hand, and require a layer of interoperability on the other. Such an interoperability can only be achieved at a very high level, and requires continuous efforts to map real data with expressive enough models, e.g., represented by domain ontologies.

This section provides a focused view on data handled by the ALMANAC platform, with a particular attention to high-level, shared models exploited to effectively handle city-generated data streams. First, an overview of currently adopted data models and of the domains they address is presented, helping the reader in framing the context of the section. Then two views on data flows handled by the platform are provided. The former depicts the path followed by field data entering an ALMANAC platform instance, whereas the latter addresses the exchange of information occurring at the Cloud API level, and involving end users of the platform.

8.1 Data types

8.1.1 OGC SensorThings Data Model

The SensorThings Data Model[17] from the OGC[18] has been selected as the generic representation of data in the ALMANAC platform architecture (see Figure 20 for the corresponding data model).

[17] http://ogc-iot.github.io/ogc-iot-api/datamodel.html
[18] http://www.opengeospatial.org/

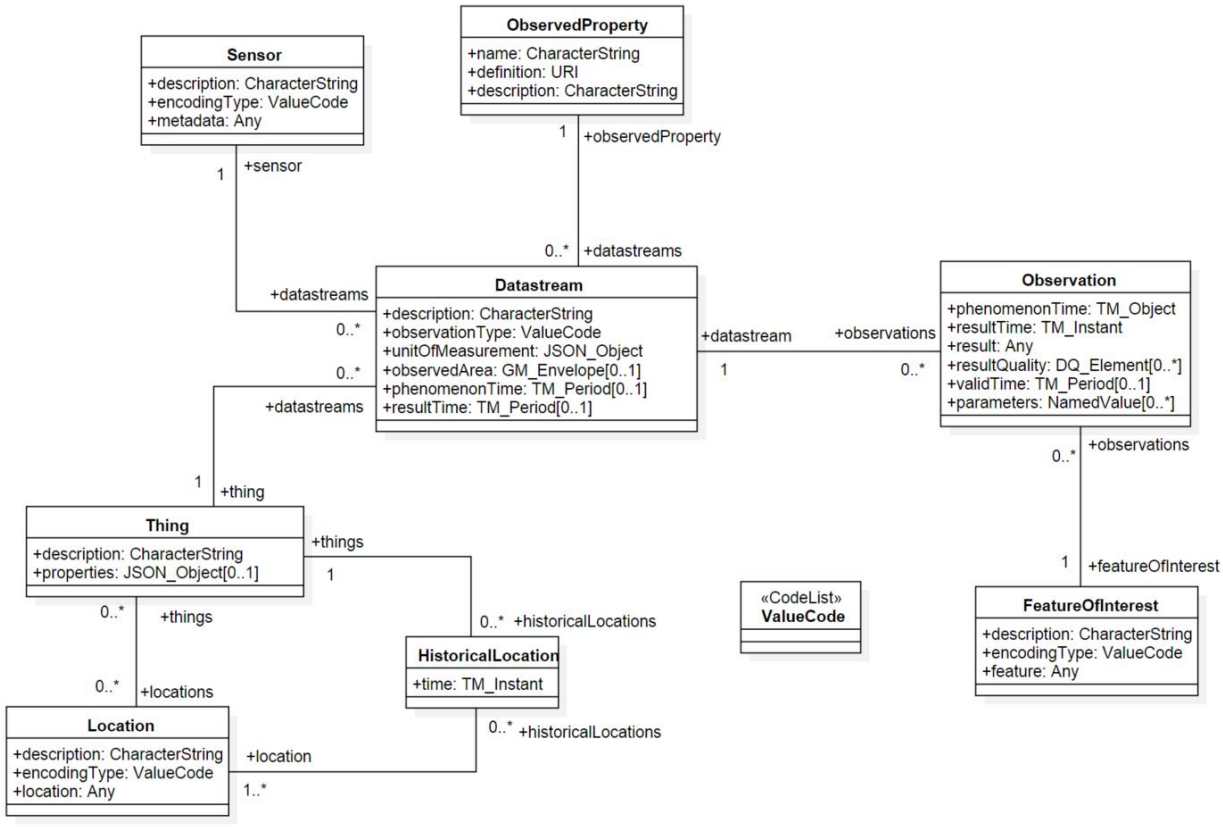


Figure 20. OGC SensorThings Data Model.

The OGC SensorThings API data model consists of the Sensing and Tasking profiles.

The Sensing profile allows IoT devices and applications to CREATE, READ, UPDATE, and DELETE (i.e., HTTP POST, GET, PATCH, and DELETE) IoT data and metadata in a *Thing* service. Managing and retrieving observations and metadata from IoT sensor systems is one of the most common use cases. As a result, the Sensing profile is designed based on the ISO/OGC Observation and Measurement (O&M) model (OGC and ISO 19156:2011).

The key to the model is that an *Observation* is modelled as an act that produces a result whose value is an estimation of a property of the observation target or *FeatureOfInterest*. An Observation instance is classified by its event time (e.g., *resultTime* and *phenomenonTime*), FeatureOfInterest, ObservedProperty, and the procedure used (often corresponding to a *Sensor*). Things are also modeled in the SensorThings API, together with the historical set of their geographical positions

More specifically, in the Sensing profile, a *Thing* has *Location*s and *HistoricalLocation*s. It can also have multiple *Datastream*s associated. A Datastream is a collection of Observations grouped by the same *ObservedProperty* and *Sensor*. An Observation is an event performed by a Sensor that produces a result whose value is an estimate of an ObservedProperty of the FeatureOfInterest.

Following subsections better detail the single data model entries.

Thing

The OGC SensorThings API follows the ITU-T definition, i.e., with regard to the Internet of Things, a thing is an object of the physical world (physical things) or the information world (virtual things) that is capable of being identified and integrated into communication networks (ITU-T Y.2060])

Location

The Location entity locates the Thing or the Things it is associated with. A Thing's Location entity is defined as the last known location of the Thing.

HistoricalLocation

A Thing's HistoricalLocation entity set provides the current (i.e. last known) and previous locations of the Thing with their time.

Datastream

A Datastream groups a collection of Observations and the Observations in a Datastream measure the same ObservedProperty and are produced by the same Sensor

Sensor

A Sensor is an instrument that observes a property or phenomenon with the goal of producing an estimate of the value of the property.

ObservedProperty

An ObservedProperty specifies the phenomenon of an Observation.

Observation

An Observation is an act of measuring or otherwise determining the value of a property (OGC and ISO 19156:2011).

FeatureOfInterest

An Observation results in a value being assigned to a phenomenon. The phenomenon is a property of a feature, the latter being the FeatureOfInterest of the Observation (OGC and ISO 19156:2001). In the context of the Internet of Things, many Observations' FeatureOfInterest can be the Location of the Thing. For example, the FeatureOfInterest of a wifi-connect thermostat can be the Location of the thermostat (i.e. the living room where the thermostat is located in). In the case of remote sensing, the FeatureOfInterest can be the geographical area or volume that is being sensed.

8.1.2 Semantic models used in ALMANAC

This subsection briefly documents semantic models used internally by the Semantic Representation Framework. They are part of the current implementation and are mainly targeted at platform developers and integrators.

The Semantic Representation Framework (via the LinkSmart Metadata Framework) advances the management and retrieval of graph-based semantic data by supporting operations on graph fragments. Such "semantic resources" comprise programmatically-defined excerpts from the overall graph continuum. Subclasses of the `ResourceRequest` hierarchy depicted in Figure 21 express the range of supported operations: `ManagementRequest` covering the life-cycle of resources, `ModificationRequest` pertaining to partial updates on existing resources, `RetrievalRequest` dealing with retrieval of a single or a selection of resources and finally `ProcessingRequest` that cover the processing of resource's content.

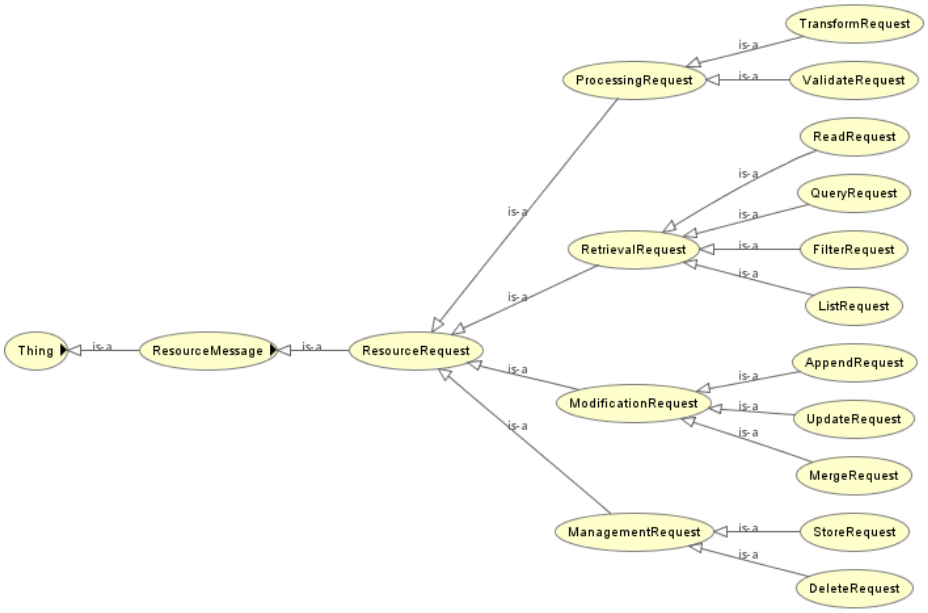


Figure 21. Resource request hierarchy.

Though not limited to, per default the handling of `ResourceRequest` messages is done by invocation of appropriate SPARQL 1.1 query expressions. Such system- or user-provided persistent queries are modelled as instances of the `SparqlQuery` classes depicted in Figure 22. In addition to a parameter specification, description annotation etc. they contain query strings used to handle particular types of requests. `ReadRequest` messages are for example handled via pre-configured `AskQuery` instances, `ReadRequest`s result in a `ConstructQuery` call and a `FilterRequest` is handled by a `SelectQuery` instance.

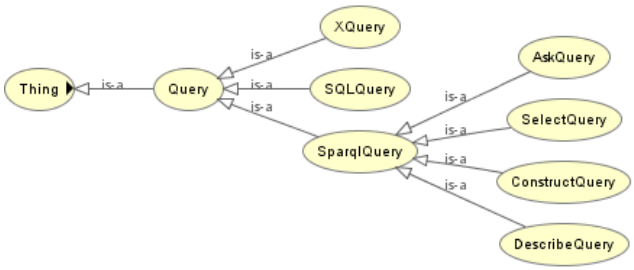


Figure 22. SPARQL 1.1. Query hierarchy

8.1.3 ALMANAC Smart City Ontologies

The Smart City ontologies exploited in the ALMANAC project build on the Linked Open Data design principles, where relevant modelling sources are re-used and connected together to form the basis upon which describing a given knowledge domain. This does not prevent the design of new models, when needed, but mandates a thorough analysis of existing solutions and promotes reuse vs creation of yet-another-model approach. In such a context, the ALMANAC project consortium performed an initial analysis of widely accepted models that can be successfully exploited in the smart city domain. They include among others:

- The Semantic Sensor Network Ontology (SSN[19]);

[19] http://www.w3.org/2005/Incubator/ssn/ssnx/ssn

- The DogOnt[20] Ontology for Intelligent Domotic Environments;
- The SCRIBE[21] IBM Smart City ontologies:
- The models listed in Smart Cities ontology catalogue[22];
- The Places[23] ontology, a light-weight ontology for describing places of geographical interest;
- The Schema.org[24] vocabulary for representing well known geographical properties, e.g., latitude and longitude;
- The GoodRelations[25] ontology for representing any kind of goods, products and services, and the relations involving their offering, exchanges, etc.;
- The MUO[26] unit of measure ontology, representing unit of measures according to the UCUM vocabulary (including International System of Measures values and other relevant units);
- The vcard[27] ontology for representing name and addresses of people and organizations;
- The GeoSPARQL[28] vocabulary and functions, to support representation and querying of geo-spatial data.
- The GeoNames[29] ontology for representing cities and other geographical entities.

A first, preliminary, modelling effort for representing smart city data with respect to waste management issues has been performed, leading to the definition of a light-weight "waste bin" ontology. The ontology focuses on waste-related issues with an open, extensible and scalable approach. According to this design goal, no assumption is performed on the remaining smart city data, and modelling is restricted to the aspects specifically related to waste collection and management, e.g., by representing waste bins, type of disposables generated by the citizenship and so on. All represented concepts leverage existing, well known definitions of properties (and classes / super-classes) with a typical Linked Open Data approach. In such a sense, for example, the city concept is directly linked (`owl:equivalentClass`) to the corresponding Schema.org and Places concepts.

The current ontology version mainly represents waste bins, waste types and the geographical / administrative context in which they are deployed. It is organized along three main hierarchies (*isA* or *partOf*) respectively rooted at the classes: `WasteBin`, `City` and `Waste`.

[20] http://elite.polito.it/ontologies/dogont
[21] http://researcher.watson.ibm.com/researcher/view_group.php?id=2505
[22] http://smartcity.linkeddata.es/
[23] http://vocab.org/places/schema.html
[24] http://schema.org/
[25] http://www.heppnetz.de/projects/goodrelations/
[26] http://idi.fundacionctic.org/muo/
[27] http://www.w3.org/TR/vcard-rdf/
[28] http://www.opengeospatial.org/standards/geosparql
[29] http://www.geonames.org/

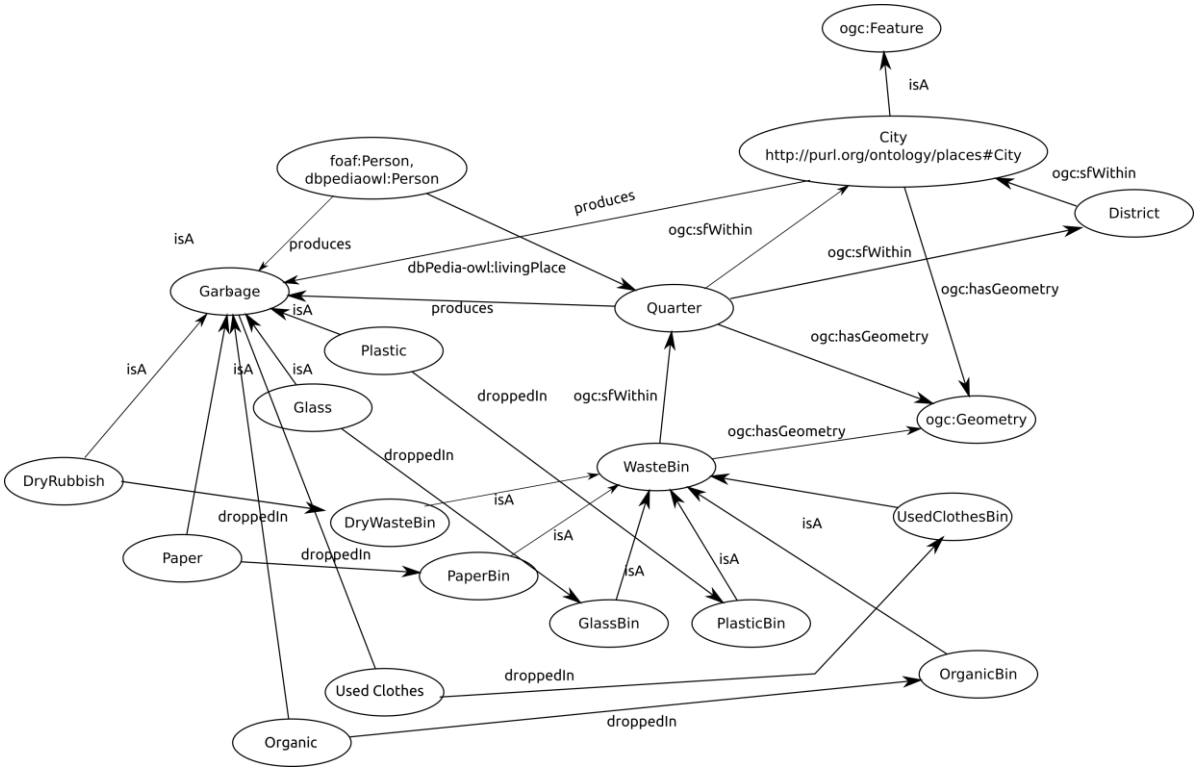


Figure 23: A part of the ALMANAC waste bin ontology.

The former represents different bin types such as bins for gathering organic waste, glass, paper, etc. Six different types of bins are represented including: dry waste, glass and aluminum, organic, paper, plastic and used clothes bins. These types are clearly emerging from a national (Italian, as one of the ALMANAC end users is the Turin's municipality) "standpoint" on waste collection, but it can easily be extended to represent typical waste collection in Europe. The second hierarchy of objects, is organized along a *partOf* containment tree and includes both physical (`City`) and administrative (`District` and `Quarters`) concepts. Since the main concepts represented in this tree are widely used in several knowledge domains and application scenarios, the WasteBin ontology definition is mainly built through equivalence classes, and it only adds geo-spatial information for automatically inferring spatial containment between waste bins and administrative / physical regions. Finally, Waste types are defined to represent the kind of waste collected by a specific bin and to represent, by means of suitable relationships, the waste generation behaviour of quarters and districts.

From a more "technical" perspective, the ontology counts 20 concepts, 6 object properties and 3 data-type properties; it features a SHIQ(D) DL expressivity and it is interconnected with 6 different and widely recognized vocabularies including: the geonames ontology, the places ontology, the GeoSPARQL vocabulary, the Schema.org vocabulary, the VCard and Good Relations and the Unit of Measurement ontologies (MUO). A full instantiation representing the public information available on waste bins deployed in Turin is provided as initial use case, and models around 29k waste bins, one city, 10 districts and 25 quarters.

The above described waste ontology has recently been linked to the latest version of the well-known DogOnt[30] ontology which was recently updated to represent generic IoT devices and entities. Such a link, currently instantiated for the waste bin concept only, establishes a direct relationship between declarative knowledge on waste-bin usage and location modelled in the ALMANAC-developed ontology and the functional descriptions of "smart waste bins" provided by DogOnt. This can, for example, be exploited through code-generation for defining the interfaces exposed by the SCRAL.

[30] http://elite.polito.it/ontologies/dogont.owl, listed on the LOV dataset at http://lov.okfn.org/dataset/lov/vocabs/dogont

8.2 Data flow analysis / presentation

Every ALMANAC Platform Instance (PI) mainly addresses two categories of data streams: data originated in the city, both at the monitoring/sensing level and at the "management" level, and data requested or injected by third party applications exploiting the ALMANAC platform. The two categories have quite different peculiarities and follow distinct, but crossing, paths inside the platform. Besides the solutions adopted to uniformly represent such data, described in the previous paragraphs, it might be interesting to analyze more in detail the path and elaboration undertaken by data flowing inside the platform. This on one hand provides better understanding of the platform inner mechanisms and interactions, and, on the other hand, provides a more informed perspective on commonalities and contact points for information streams stemming from different, heterogeneous sources such as field-level sensors and user-entered data.

8.2.1 Field to platform

The main information source in ALMANAC is the heterogeneous sensing network constituting the Smart City "nervous" system. Like capillaries in living beings, sensors deployed in the city "capture" vital information on city processes and states (e.g., pollution level, traffic, waste levels, etc.) and deliver it to the core elaboration center represented by the ALMANAC platform. In this path, every single bit of information traverses a complex set of modules, each performing a different operation, either on the data itself or on the so-called metadata (i.e., information about delivered data).

Raw values generated by sensors are gathered by a capillary communication network (or directly by the SCRAL) and they undergo several checks and modifications aimed at transforming each measurement point into a meaningful and exploitable bit of information. For each sensor generating data streams, a corresponding metadata description is generated, and kept up-to-date, exploiting standard representations (OGC SensorThings API), open data and semantic models (Smart City ontology). The resulting metadata is attached to the data stream and exploited to better characterize every single measure.

For example, the raw temperature value of 20°C measured by a Do It Yourself (DIY) sensor located in the center of Copenhagen, is enriched (at the SCRAL level) with metadata about the originating sensor. This means that at the SCRAL level the raw temperature value becomes an OGC Sensor Things API observation referred to a well-defined property (e.g., the outside temperature), with a precise generation time stamp, with a standard unit of measure and a geographical positioning in terms of latitude and longitude. Attached to this bit of information, additional (meta-) data permits to identify the owner of the sensor, e.g., a private user, with a medium access-level, and possibly some data quality indicator, e.g., amateur-level sensor vs professional measure system.

At the upper boundary of the SCRAL, data injected in the platform is no longer raw and uncategorized. Instead, it is rich and expressed in a shared, standard, uniform and machine-understandable format, which can easily be handled by higher platform levels where intelligent elaboration takes place (see Figure 24, which summarizes this first step).

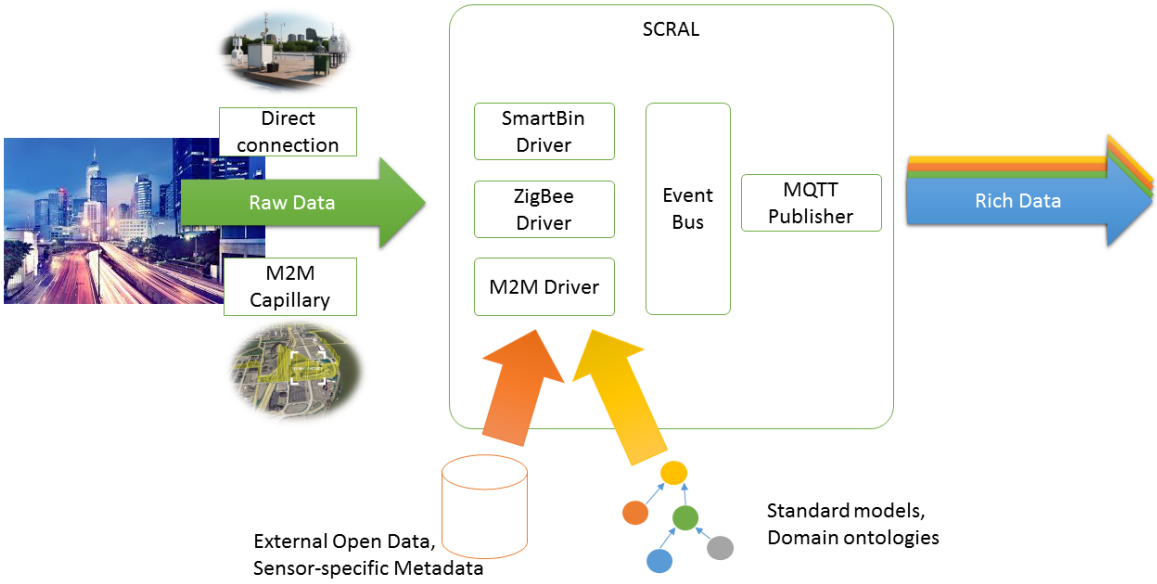


Figure 24. Data Flow from the city to the Almanac PI lower layers.

Enriched data flows into the platform along several parallel paths (see Figure 25). First, data is captured by the platform Storage Manager which provides persistence to gathered data. In parallel the same data flows (selectively) into the Data Fusion manager, which applies complex event processing operators to the incoming data streams and generates new (rich) data, either referred to same originating sensors (e.g., temporal aggregations, statistical properties) or pertaining to completely new sources, i.e., virtual sensors generated by combining (fusing) together different data streams (as in the "bad smell" example reported in Section 6.4.1). Such new data is in turn routed towards the Storage Manager for persistence. While both the Storage Manager and the Data Fusion Manager handle "live" data, i.e., measures plus metadata, the metadata information flow is also routed to the Resource Catalogue component, which is responsible for maintaining a directory of currently connected data sources, together with the corresponding metadata. Such metadata can be further enriched by exploiting the inference mechanisms provided by the Metadata Framework, which hosts domain ontologies exploited by the ALMANAC platform.

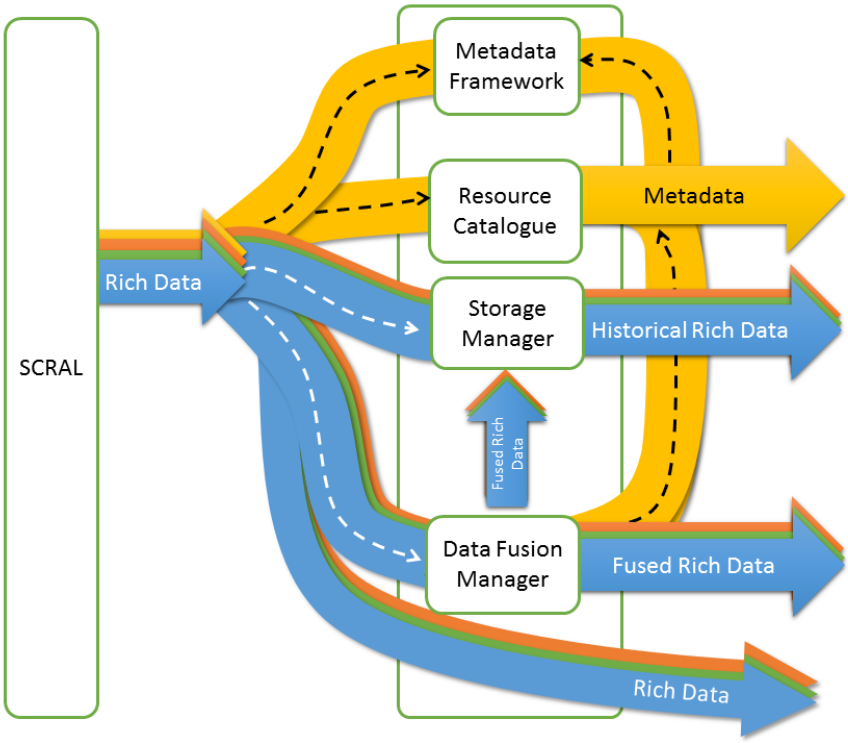


Figure 25. Data Flow, from SCRAL to Data Management

At the highest architectural layer of the ALMANAC platform, rich data, and metadata is made available to end users, i.e., applications using the ALMANAC Cloud APIs and to other federated platforms. Data is either delivered through REST or Web Socket. This layer can be seen as the "endpoint" of the field-to-platform information flow, and the boundary at which the user-to-platform information flow begins. Figure 26 reports the complete field-to-platform data flow diagram.

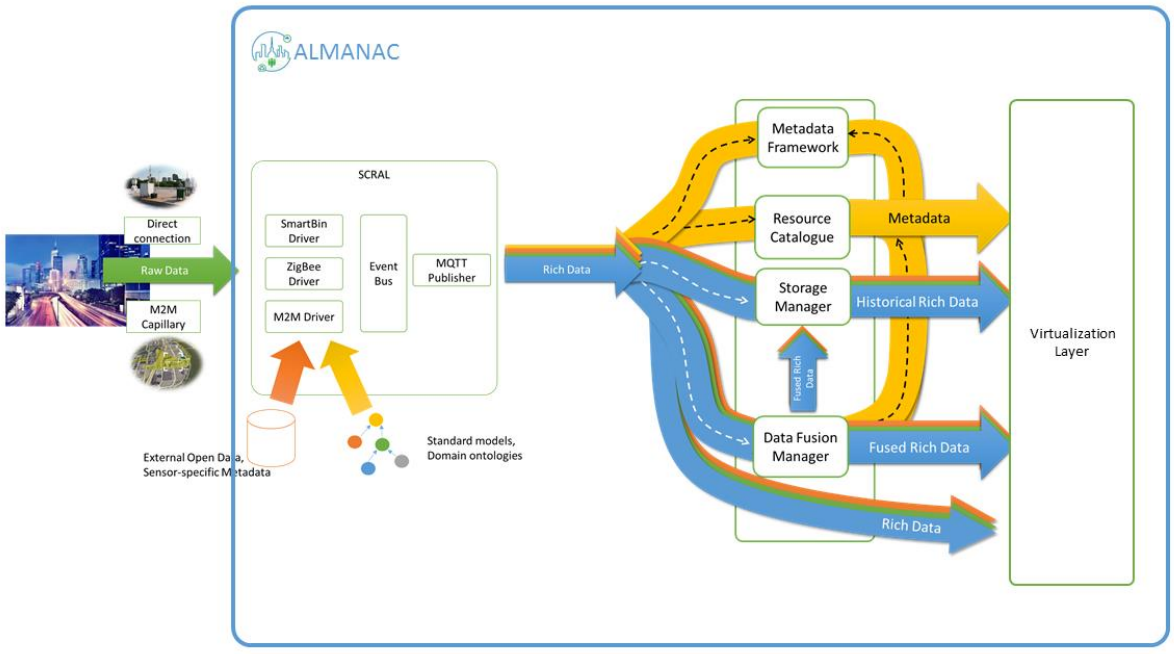


Figure 26. Data Flow diagram from field to platform.

8.3 Data sharing

Data sharing with third parties is a desirable feature to maximize the use of the ALMANAC data. To allow third-party services that are not ALMANAC-aware to receive and consume some ALMANAC data, the Virtualization Layer proposes a few protocol and data conversions to ease the process.

Here are three examples for geographical information, tabular data, and updates.

GeoJSON[31] is a format for encoding a variety of geographic data structures – as well as custom properties. The Virtualization Layer can transform some responses to this format, which is supported by popular third-party services such as Google Maps[32]. Figure 27 is a screenshot of a map generated by the GeoJSONLint[33] third-party service.

Figure 27 - Map produced by a third-party service using the GeoJSON format

CSV: When browsing sources of open data (e.g. open government data from UK[34]), the old plain-text format CSV or TSV is by far the most popular. Thanks to the on-the-fly conversion operated by the Virtualization Layer, it is possible to conveniently import some ALMANAC data into third-party services such as Google Docs Spreadsheet, with a direct HTTP call (e.g. IMPORTDATA[35] API for Google Docs).

RSS is the de-facto standard for receiving updates by HTTP pull. The Virtualization Layer supports ATOM[36] (RFC 4287), which is a newer equivalent of RSS, and can be consumed by a large amount of systems aggregating or reacting upon news and other events. For instance, the fashionable service IFTTT[37] can consume ATOM/RSS, and then trigger a "recipe" such as sending an alert/SMS to a smartphone, switching a lamp on or off, etc.

[31] http://geojson.org
[32] Google Maps API with GeoJSON support https://developers.google.com/maps/documentation/javascript/reference#Data.GeoJsonOptions
[33] http://geojsonlint.com/
[34] http://data.gov.uk/data/search
[35] Google Docs tabular import function https://support.google.com/docs/answer/3093335
[36] ATOM http://tools.ietf.org/html/rfc4287
[37] https://ifttt.com

9. Security Perspective

In the ALMANAC architecture we distinguish between the so-called *trusted area* and the *untrusted area*.

The trusted area is demarcated by the boundaries of a Platform Instance (PI). The platform components that are accessible only from within the trusted area are the Access Manager, the MQTT Broker, the Data Fusion Manager, the Resource Catalogue and the Storage Manager. Each of these components is under direct control of the PI operator.

The untrusted area comprises external networks, e.g. the Internet, and wireless sensor networks. The components that act as a gateway between the untrusted area and the trusted area are the SCRAL, the LinkSmart GlobalConnect (LinkSmart-GC) and the Virtualization Layer Core (VLC). The SCRAL connects the trusted area to the lower level sensor networks. LinkSmart-GC connects the PIs, which are part of a federation over an untrusted network. The VLC connects the federation users to the PI, after checking if the federation user has the required authentication and authorization.

A special role has the Federated Identity Manager (FIM), which provides a login interface for the federation users, so they can authenticate themselves within the federation.

A federation user is an entity (e.g. a user/service) that has been registered at the federation through the FIM and is allowed to connect to whichever PI that is part of the same federation.

9.1 Threat identification

The ALMANAC Platform consists of multiple services deployed in Platform Instances (PIs). Communication between services of different PIs is possible, if they are part of the same federation. When services communicate, they exchange information, either concerning live data or metadata. From a security perspective, the information exchanged by ALMANAC services can be classified as follows:

- *RawStreamData*: Information created by a sensor.
- *FusedStreamData*: Information created through combination/fusion of raw and historical data.
- *HistoricalData*: Information stored for later use.

The following services can be part of an ALMANAC PI and therefore, they will be considered in detail in the threat identification and analysis process described in the following. They encompass:

- Smart City Resource Adaption Layer (SCRAL): For establishing the connection to the sensors/sensor networks. The SCRAL takes in information from the sensors (handling any network-level encryption and/or coding) and forwards it within the PI. The Information from the SCRAL is of type *RawStreamData*.
- Data Fusion Manager (DFM): The DFM fuses *RawStreamData* into *FusedStreamData*. Data fusion rules specify through the Data Fusion Language which and how raw data streams should be fused.
- Data Fusion Language (DFL): The DFL takes instructions from an advanced user/system integrator and translates these for the DFM.
- Storage Manager (SM): The SM accepts *RawStreamData* as well as *FusedStreamData* and stores these as *HistoricalData*.
- Virtualization Layer Core (VLC): Controls the access into the PI and hides (abstracts) the ALMANAC system from the user.
- LinkSmart GlobalConnect (LinkSmart-GC): The LinkSmart-GC is responsible for the interconnection of multiple PIs.
- Resource Catalog (RC): Provides access to abstracted smart city resources.

To outline, and identify, the possible communication flows among ALMANAC services, an information flow analysis was performed.

Information flow diagrams visualize the information flow among platform components and help to clearly identify potential points of attacks and threats. An information flow diagram has been created for each type of data: *RawStreamData*, *FusedStreamData*, and *HistoricalData*.

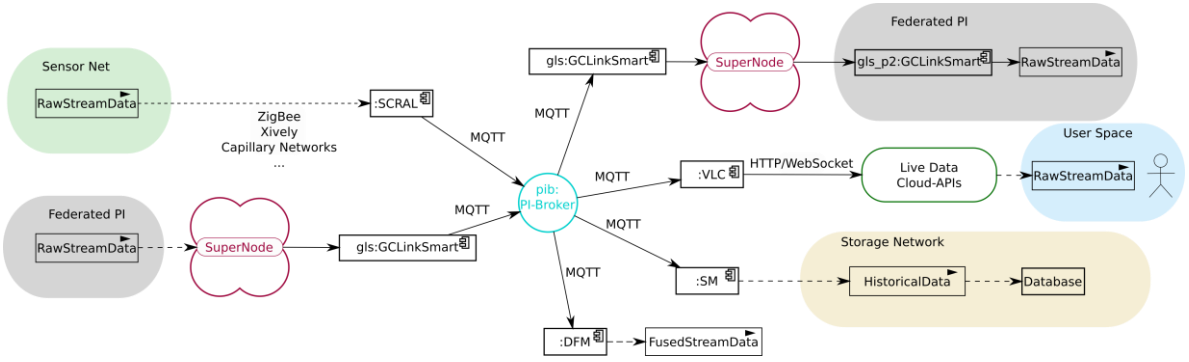


Figure 28 Information Flow Diagram - RawStreamData

Figure 28 shows the flow of *RawStreamData* either created by sensors (SensorNet on the upper left side) and fed into the system by the SCRAL, or by a SCRAL of a federated PI (on the bottom left side) over LinkSmart Global Connect. Such data is distributed via MQTT among other ALMANAC services, depicted as outgoing arrows from the PI-Broker.

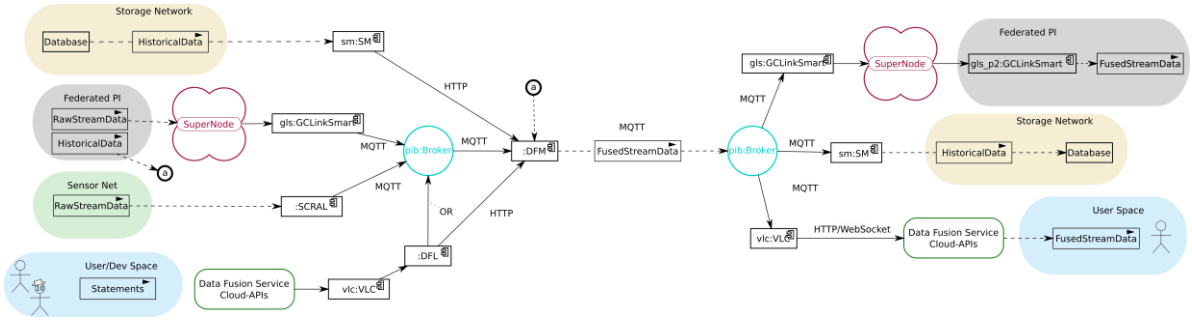


Figure 29 Information Flow Diagram - FusedStreamData

Figure 29 shows the flow of *FusedStreamData* (starting at DFM in the middle of Figure 29 and flowing to the right) which is generated from *RawStreamData* and/or *HistoricalData* by the DFM. The instruction on which information should be fused is created by the DFL.

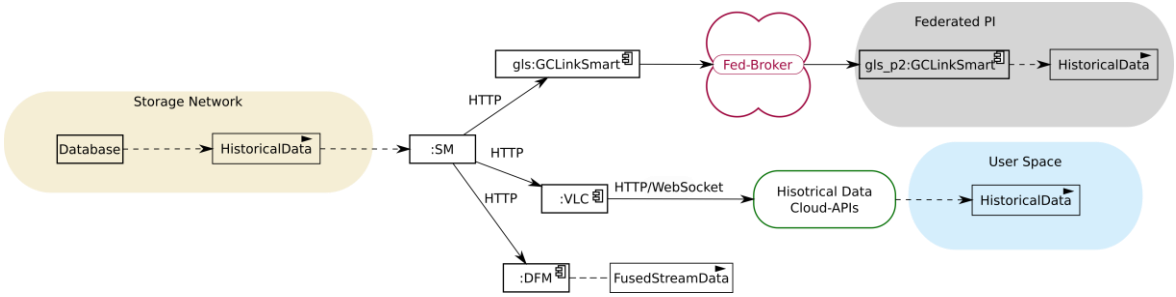


Figure 30 Information Flow Diagram - Historical Data

Figure 30 shows the flow of *HistoricalData*. The *HistoricalData* is stored inside a database system, which should only be queried by the Storage Manager (SM). Another service can ask the SM for *HistoricalData* over HTTP, through the VLC. The SM takes this request and translates it into a database query. The result of the query is forwarded to the requesting service.

9.2 Threat Analysis

This section provides a detailed threat analysis for each ALMANAC service based on the STRIDE Threat Model[38].

9.2.1 Definition

The STRIDE Threat Model provides the means to better analyze the potential threats for a system by grouping threats into 6 categories:

- **Spoofing**. An example of identity spoofing is unauthorized accessing and then using another user's authentication information, such as username and password.
- **Tampering**. Data tampering involves the malicious modification of data. Examples include unauthorized changes made to persistent data as well as the alteration of data as it flows between two computers over an open network, such as the Internet.
- **Repudiation**. Repudiation threats are associated with users who deny having performed an action without other parties being able to prove the contrary. For example, a user performs an illegal operation in a system that lacks the ability to trace the prohibited operations. Nonrepudiation refers to the ability of a system to counter repudiation threats. For example, a user who purchases an item might have to sign for the item upon receipt. The vendor can then use the signed receipt as evidence that the user did receive the package.
- **Information disclosure**. Information disclosure threats involve the exposure of information to individuals who are not supposed to have access to it - for example, the ability of users to read a file that they were not granted access to, or the ability of an intruder to read data in transit between two computers.
- **Denial of service**. Denial of service (DoS) threat is the ability of denying service to valid users, e.g. by making a Web server temporarily unavailable or unusable.
- **Elevation of privileges**. In this type of threat, an unprivileged user gains privileged access and thereby has sufficient access rights to compromise or destroy the entire system. Elevation of privilege threats include those situations in which an attacker has effectively penetrated all system defenses and becomes part of the trusted system itself.

9.2.2 STRIDE threat analysis of ALMANAC

A STRIDE analysis was performed for the main ALMANAC components: SCRAL, Resource Catalogue, Storage Manager, Data Fusion Manager, MQTT Broker, Access Manager, Federated Identity Manager,

[38] https://msdn.microsoft.com/en-us/library/ee823878%28v=cs.20%29.aspx

Virtualization Layer Core (VLC), and LinkSmart GlobalConnect. The following paragraphs provide deeper details on the analysis outcomes, for each considered service.

SCRAL

The SCRAL is responsible for interfacing, and abstracting, sensors and devices (Resources) deployed in the smart city. From a security perspective, a first step in identifying possible threats at this level consists of the identification of data streams "generated" and "captured" by the component, and of the ALMANAC components with which the SCRAL interacts (directly).

Spoofing

It is possible to spoof hosts to which the SCRAL connects to inject Raw Stream Data and/or Metadata inside the platform, e.g. the sensor gateways. Moreover, it is possible to spoof the broker, to gain access to data delivered by the SCRAL (in a Man-in-the-middle type of attack).

Tampering

An attacker can try to change the SCRAL-generated network traffic to inject into other components "scripts and/or data" for gaining access and/or credentials. E.g. insert malicious DFM queries by exploiting the MQTT connection on the trusted zone, or exploiting some "code injection" vulnerability on the Storage Manager, etc. There is the possibility that an attacker could try to manipulate single packets that come from the SCRAL or go into the SCRAL. On the other hand, an attacker might compromise, or spoof, a gateway to "delete" device entries or to make devices appear as offline.

Repudiation

It is possible to access or spoof some low-level network and disconnect some device without being "tracked" or "detected", this is mainly related to the security-level reached by the "connected" network, and is clearly technology dependent.

Information disclosure

Although no authorization / login information is handled by the SCRAL, there might be "sensitive" data streams which are valuable for attackers, e.g. critical sensor data such as traffic light information, proprietary information, etc. These streams can be read by sniffing the network, or by accessing the MQTT Broker.

Denial of service

A SCRAL instance can be overflown with data from the field (e.g., by spoofing a low-level network) with the aim of blocking the SCRAL services (and thus the platform capability to handle live data). It is possible to spoof the broker, to block data delivered by the SCRAL. Moreover, it is possible to "sniff" the SCRAL credentials on MQTT and thus causing its disconnection from the broker. Finally, it is possible to impersonate the SCRAL (see spoofing) and attempt a DDOS attack to other components connected to the broker

Elevation of privilege

An Attacker might use the SCRAL to inject Raw Stream Data and/or Metadata (i.e., generating new devices) inside the platform. Additionally, an attacker can try to inject code in the SCRAL and gain unauthorized access, e.g., by changing the network traffic between the SCRAL and the low level networks (such as the M2M or Smart Bin services)

Resource Catalogue

The Resource Catalogue (RC) provides a REST-based interface to select and retrieve information about Resources in an ALMANAC PI. The RC receives information about the currently connected IoT Resources from the REST-API of the SCRAL, or from the metadata flow over MQTT, and it receives information about the new resources created inside the DFM through its DFL component. Information about resources is queried by the VLC and the LinkSmart services. The VLC needs the RC for replying to requests of federation users. More precisely, the VLC talks directly to the RC of the local PI or, via LinkSmart, to a RC of a federated PI. No other service needs to communicate with the RC.

Communication at the Resource Catalogue is always handled via HTTP with messages carrying payloads expressed in the JSON format.

To summarize, the RC needs to communicate over HTTP with

- SCRAL (REST-API) - sends information about new IoT resources.
- DFM (DFL) - sends information about new fused data.
- VL (VLC & LinkSmart-GC) - query resource information.

Spoofing

It is very simple to spoof either the registration, deregistration, or the querying of resources, since there is no check of the identification of the communication partner.

Tampering

An attacker can tamper with the information stored inside the RC. The attacker can

- register new resources, and
- unregister existing resources.

In addition, the communication between the RC and an authorized communication partner can be tampered with or without the knowledge of the two, since the RC only uses HTTP. If an attacker gets access to the local file system, he is able to change the information that is stored on the RC. Currently it is unclear if an attacker can insert wrongly formatted information.

Repudiation

It is not possible to say who inserted a resource. Even though there should only be the SCRAL and the DFM, as per the architecture design.

Information disclosure

The information inside the RC has minor confidentiality. It is possible to request the information directly from the RC. It is easy to listen on the traffic of the RC.

Denial of service (DoS)

It is possible to insert resources without authorization, hence it is possible to insert an unlimited amount of new resources. Either the RC runs out of hard drive space or it takes too long time to process its queries. This attack wouldn't even be possible to be traced (see repudiation above), since it is not reproducible from where the information was inserted and it is unrelated to the time when the DoS happens.

Elevation of privilege

Vertical: An Attacker might send specially crafted information in a query or a de/registration to exploit the system.

Horizontal: Parties that are able to query the RC are also able to de/register resources. The SCRAL is able to de/register stream resources and query information. The DFM/DFL is able to de/register IoT resources and query information.

Storage Manager

The Storage Manager (SM) receives *RawStreamData* from the SCRAL and *FusedStreamData* from the DFM over the MQTT-Broker and stores it in a database. To do so the SM listens on a specific group of topics on the broker, which are used by the DFM and the SCRAL to publish their streams. The information can later be queried by the DFM, VLC, and LinkSmart. To query the information one of the components needs to send a HTTP/GET request and OData operations. The database used by the SM is currently a MongoDB.

To summarize, the SM needs to communicate with the

- SCRAL over the broker via MQTT
- DFM over the broker via MQTT
- DFM directly via HTTP
- VLC directly via HTTP
- LS directly via HTTP

Spoofing

Since the SM stores every message that is sent to a specific group of topics and, since there is no verification from whom the information came from, if someone has access to the Broker it is possible to act as the SCRAL or the DFM and inject data (malicious or unverified).The only components that should be allowed to query the Storage Manager are the DFM, the VLC, and the LS but, currently, there is no verification to match the requester with one of these three. It is easy for an attacker to access the information directly. The connection between the database and the SM is not secured against spoofing. It is possible to force the SM to reconnect to the database and then act as the database. The same is true for the connection between the broker and the SM.

Tampering

It is possible to connect directly to the database and insert new/fake information or to change and delete the information stored. Someone, who knows the group of topics the SM is listening on, can push new messages to the MQTT broker and so insert new/fake information into the SM.

Repudiation

It is not reproducible, from whom the original message came, since the information is lost at the broker.

Information disclosure

An attacker can listen to the query connections either by sniffing the network or via a man in the middle attack. Also, it is possible to directly connect to the database and read the information stored. Moreover, an attacker can connect to the MQTT broker and simply create his own consumer for the specific group of topics. Finally, an attacker can directly query the SM for information.

Denial of service

To stop the SM from functioning an attacker has two options:

1. DoS applied to the database
2. DoS applied to the SM

To carry out a DoS on the database, an attacker can insert fake information via the methods mentioned in the tampering section. This will increase the amount of storage space used by the database and will create more load for the database and the SM when they process a query. The increase in load might need more investigation on the queries that are typical for the particular PI. A method to apply DoS to the SM is to query the SM for large amounts of data. The method can be refined by spoofing the IP address of the receiver. Of course the two methods can be combined. Additionally, there might be special DoS attacks for the database, depending on the type of database adopted as backend.

Elevation of privilege

Vertical: An attacker might send specially crafted queries or MQTT messages to exploit the SM. Especially NoSQL injects might be a problem, since they might result into code execution or access to internal information

Horizontal: The only one that should be able to insert RawStreamData is the SCRAL, but nothing hinders the DFM to insert fake RawStreamData. Similarly, the only one allowed to insert FusedStreamData is the DFM, but the SCRAL would be able to as well.

Data Fusion Manager

The Data Fusion Manager receives information from the SCRAL and fuses it into a *FusedStreamData*. The *FusedStreamData* is published to the broker, so it can be consumed by the SM and the VL.

The instructions for fusing the *RawStreamData* are given to the DFM as DFL-Statements via a HTTP/REST interface. The DFL-REST interface is implemented by the DFL-Compiler component of the DFM, this component translates the DFL into the internal instructions of the DFM and forwards the instructions to the DFM. The DFL-Statements can be sent to the DFL-REST interface by the platform user (e.g., a third party application) over the VLC.

Spoofing

If an attacker has access to the MQTT broker, he could act as the SCRAL. An attacker could issue a reconnect between the DFM and the broker. In consequence, the attacker would be able to intercept the connection and act as the broker by spoofing the address. Finally, an attacker could act as the VLC and send DFL-Statements into the system.

Tampering

An attacker that is able to connect to the broker can insert information into the fusion processes of the DFM and thereby change the outcome of the *FusedStreamData*. An attacker could try to manipulate the traffic between the SCRAL and the DFM to insert false information. Either between the SCRAL and the broker or between the broker and the DFM. Additionally, an attacker could try to change the outgoing traffic from the DFM that gets sent to the broker. Finally, an attacker could try to manipulate the DFL statements to change the *FusedStreamData*.

Repudiation

Currently there is no logging of user actions, so it is not verifiable which user inserted which DFL.

Information disclosure

An attacker with access to the broker can read any message that was sent to and from the DFM. Moreover, an attacker that can sniff the network of the trusted area, can read the messages that are received and sent from the DFM, which include DFL statements, *RawStreamData* and *FusedStreamData*.

Denial of service

To stop the DFM from functioning an attacker can try to delete all DFL statements. An Attacker could try to insert a large amount of DFL statements to increase the load of the DFM or to let the DFM run out of memory. In addition, an attacker might craft a special DFL statement that requires a large amount of memory or processing power. Finally, an attacker could disconnect the DFM from the broker, for example by acting as the DFM and sending a MQTT *unsubscribe* to the broker.

Elevation of privilege

An attacker might use the DFM to insert *RawStreamData* into the broker. Furthermore, an attacker might use the DFL statements to create a fused stream that directly forwards confidential *RawStreamData*.

MQTT Broker

In a PI, the MQTT broker manages the distribution of messages to possibly multiple receivers. More precisely, the broker distributes the *RawStreamData* messages published by the SCRAL to the DFM, VL and to the SM. It also distributes the *FusedStreamData* messages published by the DFM to the VL and to the SM.

The components communicate with the broker via the MQTT protocol. The MQTT protocol and other broker protocols like STOMP and OpenWire implement the publish-subscribe pattern. This makes it difficult to establish a trust relationship between the sender and the receiver, because:

1. the sender can't restrict who is able to receive his message, and

2. the receiver can't distinguish between senders.

This makes the broker a profitable attack vector.

Spoofing

Only the DFM, SCRAL, VL and the SM need to be connected to the broker. An attacker could try to act as one of these to connect to the broker. An attacker could disconnect a component from the broker, by sending a single *unsubscribe* for all topics.

Tampering

An Attacker could change the traffic that the broker receives and sends on the network level.

Repudiation

A component could send a message in the name of another component.

Information disclosure

An attacker could sniff the network of the broker. Moreover, an attacker can try to listen to topics of other components.

Denial of service

An attacker could flood the broker with subscriptions to increase the load and memory usage of the broker.

Elevation of privilege

A component connected to the broker could send messages to topics other than the ones it is supposed to.

Access Manager

The Access Manager (AM) is an implementation of a XACML policy decision point. It stores the policies of its PI and provides an interface for querying them. PI components can interact with the AM through a HTTP/REST interface.

The AM typically receives queries from the SCRAL and the VL, whereas policies are added by the PI operator.

The AM has the following entry points:

1. The HTTP/REST interface for querying for policy decisions. It should only be accessible to the SCRAL and the VL.
2. Operator interface for inserting policies. It must only be available to the PI operator.

The relevant assets are policies and the decision made by the AM.

Spoofing

An attacker could try to spoof the system by acting as a PI operator.

Tampering

An attacker could try to change the network traffic to influence the outcome of the policy query. This can be done either by changing the query or by changing the result of the query. Moreover, an attacker could try to insert, delete or change policies.

Repudiation

It might be possible for an authorized attacker (a malicious operator) to insert a policy which enables her to query information that is confidential and later remove that policy. If the process of creating and removing policies is not monitored this might lead to misuse.

Information disclosure

Since the policies are not confidential there is no information disclosure threat.

Denial of service

An attacker could try to delete all policies. Additionally, an attacker could send multiple complex queries to increase the load on the AM. Finally, the attacker could try to insert load increasing policies.

Elevation of privilege

An operator can give himself the rights to access all components of the PI.

Federated Identity Manager

The Federated Identity Manager provides Single Sign On for federation users and manages user identification inside a single Platform Instance. It is mainly implemented exploiting available open-source implementations such as KeyCloak[39]. Such a tool is clearly critical in the platform and is subject to the following threats, classified according to the STRIDE methodology.

Spoofing

Cross-site request forgery (CSRF): is a web-based attack whereby HTTP requests are transmitted from a user that the web site trusts or has authenticated (e.g., via HTTP redirects or HTML forms). Any site that uses cookie-based authentication is vulnerable for these types of attacks.

Clickjacking: a malicious site loads the target site in a transparent iFrame overlaid on top of a set of dummy buttons that are carefully constructed to be placed directly under important buttons on the target site. When a user clicks a visible button, he is actually clicking a button (such as an "Authorize" button) on the hidden page. An attacker can steal a user's authentication credentials and access its own resources.

Compromised Access Tokens: an attacker might try to compromise the FIM access tokens to get unauthorized access.

Open redirect: An attacker could use the end-user authorization endpoint and the redirect URI parameter to abuse the authorization server as an open redirector. An open redirector is an endpoint using a parameter to automatically redirect a user agent to the location specified by the parameter value without any validation. An attacker could utilize a user's trust in an authorization server to launch a phishing attack.

Brute Force Attack: A brute force attack happens when an attacker is trying to guess a user's password.

Registration spoofing, i.e., tricking an operator into registering an attacker

Tampering

Typical tampering attacks exploit SQL Injection on the identity (user) database.

Repudiation

Clickjacking: a malicious site loads the target site in a transparent iFrame overlaid on top of a set of dummy buttons that are carefully constructed to be placed directly under important buttons on the target site. When a user clicks a visible button, they are actually clicking a button (such as an "Authorize" button) on the hidden page. An attacker can steal a user's authentication credentials and access their resources. In such a way any action carried on the system is deniable and can be "mapped" onto another, real user (no way to trace back the real attacker)

Compromised Access Tokens: an attacker might try to compromise FIM access tokens to get unauthorized access.

Information disclosure

Password database disclosure: While a human could probably never crack a hashed password, it is very possible that a computer could. The security community suggests around 20,000 hashing

[39] http://keycloak.jboss.org/

iterations to be done to each password. This number grows every year due to increasing computing power (It was 1000 almost 12 years ago).

Denial of Service

An attacker might attempt a DoS by submitting multiple-repeating authentication requests with the wrong credentials. This causes the system to block the user for which attempts are done, thus generating a denial of service for the actual user.

Elevation of Privileges

An attacker might exploit broad or light securirty token scopes to gain higher access privileges.

Virtualization Layer Core

VLC has the role of the main entry point to a PI, i.e., it should not be necessary to expose the other components of the trusted area. This makes the VLC a strategic place to function as a Gateway, which can enforce the policies for the federation user and to reject authorized request.

The VLC has two connections. One to the untrusted area from where requests from the federation users come in and, a second with which the VLC is connected to the trusted area. Over the second connection, the VLC establishes communication to the AM, Broker, DFM, LinkSmart-GC, RC, SCRAL, and SM to serve the request it receives from the federation users.

The federation users send an access token with every request to authenticate themselves to the VLC. They receive this access token from the FIM. To verify that the federation user has the proper authorization to execute the given request, the VLC interacts with the AM Requests will be forwarded into the trusted area only upon verified authorization.

Spoofing

An attacker could try

- one possible access token after another, to guess a valid access token;
- to mimic the VLC so that a federation user is connecting to the attacker instead of the VLC, thus providing the actual access token to the attacker,
- to forge access tokens.

An attacker with access to the trusted area could mimic a component of the trusted network.

Tampering

An attacker can try to manipulate

- the traffic between the VLC and the federation user, or
- the traffic between the VLC and a component of the trusted area.

Repudiation

An attacker could try to manipulate the VLC logs to erase the fact that he was accessing the PI

Information disclosure

An attacker could try to read the communication between the VLC and the federation user, to

- get access to tokens or
- acquire confidential information, e.g. the result of a request.

Possible threats are man in the middle attacks and network sniffing.

Denial of service

To stop the VLC from answering to requests, an attacker could try to delete, or change, all policies in the AM, so that the VLC will reject all requests. Moreover, an attacker may try to overflow the VLC REST service to prevent real-users from actually exploiting services provided by an ALMANAC PI.

Elevation of privilege

Since the VLC is accessible over the untrusted network, it is easier for an attacker to send a request with malicious content to the VLC, in the hope that the content will get executed.

LinkSmart Global Connect

The functionality of LinkSmart Global Connect (LinkSmart-GC) is to provide a tunneling service that enables transparent communication of applications and services beyond the boundaries of a private network. It helps to connect remote local environments over the Internet, which can be discovered dynamically. In ALMANAC it is used to interconnect multiple PIs, where LinkSmart-GC provides HTTP and MQTT tunneling.

A single LinkSmart-GC instance, when started, is connecting to its Supernode, from where it gets a list of other LinkSmart-GC instances with which it can establish a connection.

From the PI, the VLC can register services at the local LinkSmart instance, so they can be accessed from other federated PIs. The registration happens via a HTTP/REST interface.

Spoofing

An attacker might

- act as a legit MQTT resource and send messages over a MQTT tunnel.
- try to subscribe to a remote MQTT topic.
- try to act as a legit HTTP tunnel user to send request over the HTTP tunnel to an HTTP service on the other side.
- try to act as a legit HTTP service that was registered.
- mimic the Supernode when LinkSmart-GC instance boots.
- might try to act as a LinkSmart-GC instance of a federated PI.

Tampering

An attacker could tamper with the

- MQTT message sent between the Broker and the LinkSmart-GC instance,
- HTTP request and responses which are communicated between the HTTP services and the LinkSmart-GC instance,
- HTTP/REST-API registrations and deregistration,
- The message exchanged between the LinkSmart-GC instances, and
- The message exchanged between the LinkSmart instance and the Supernode.

Repudiation

When someone is registering or deregistering a new tunnel it might not be traceable who did it. An Attacker might change the logs files to erase trace of the fact that

- a LinkSmart instance has connected;
- a HTTP or MQTT service was registered or deregistered;
- a connection to a service was requested.

Information disclosure

An attacker might try to read information from

- MQTT messages;
- HTTP tunnel requests and responses;
- connections between two LinkSmart-GC instances.

Denial of service

To run a denial of service attack on the LinkSmart-GC instance of a PI, the attacker can try to

- deregister all tunnels;
- replace all tunnels to reachable destinations;
- attack the Supernode so it is unreachable before the LinkSmart-GC instance could connect to it;
- tamper with the policies of the AM so all request from the outside are denied.

Elevation of privilege

An attacker could replace/remove a tunnel, which was not created by him.

9.3 Counter measures by ALMANAC

To counter the possible elevation of privilege attacks, it is necessary to verify the input data that is coming into a component and, to give the component the least possible privileges. To assure that such an action is difficult for an attacker, it is necessary to test these two requirements by doing code reviews and appropriate penetrations tests. These two methods can be used to reduce the possible attack vectors for privilege escalations. It is obvious that code reviews can only be done with software owned by consortium partners. For third party software it is only possible to either install an updated version, or to switch to a different implementation, when an issue is disclosed.

9.3.1 The trusted area

After a quick review of the threat model, it is evident that the components inside the trusted area need to have a strong form of authentication, confidentiality, and integrity. Since most of the spoofing, tampering and information disclosure threats arise out of the potential damage an attacker can do as soon as he can get access to the trusted area network.

To mitigate these threats, it was mentioned in the second revision of the architecture that TLS (Transport Layer Security) should be used between the components inside the trusted area, this includes the broker. For this third revision this statement should now be explained in more detail:

Whenever TLS is mentioned in this document, it means the current TLS standard version 1.2. TLS 1.2 is defined in RFC 5741 and was later updated with the RFC 6176 and RFC 7465. With RFC 6176 the downgrading of TLS connections to SSL 2.0 is prohibited. With RFC 7465 the use of RC4 encryption algorithm was prohibited. Further, the use of weak algorithms like MD5, SHA-1, and 3DES should be avoided and it should also not be possible to downgrade the version of TLS from 1.2 to 1.1, 1.0 or even SSL 3.0. Since such a list can never be complete and the minimal required key length are increasing every year, it is advisable to consider appropriate literature before an implementation for example from the NIST (NIST, 2012), the BSI (BSI, 2015), or the ANSSI (ANSSI, 2014).

One fact that is easily forgotten is that the client authentication in all TLS versions is optional and needs to be enabled explicitly. By not authenticating the client an attacker might be able to spoof his identity. As a consequence, client authentication is a requirement for the components inside the trusted area. This means that a component that is connecting to another component also needs its certificate to authenticate itself. This opens up the next discussion on certificate creation and distribution.

Each component inside the trusted area needs a certificate to identify itself to other components. Since the relationship between the single components of the trusted area is static, it is possible to pre-share the certificates between the components. A static relationship means that one component only has to

trust a specific set of components, e.g. the RC only needs to trust the SCRAL, DFM, VLC, and LS. Additionally, a component has to permit certain operations only to a specific component, e.g. the RC only needs to supply the de/registration of sensors to the SCRAL. Since the certificates can be pre-shared, there is no need for a certificate authority inside the trusted area. All the certificates in the trusted area can therefore be self-signed certificates, which can be created in the installation process of every component.

The pre-sharing must then be done by the operator of a Platform Instance. The certificates should use RSA or ECDSA as a signature algorithm and at least a SHA-2 hash function; again it is advisable to consider appropriate literature before an implementation.

By using TLS with client authentication in combination with pre-shared self-signed certificates, sniffing, spoofing and tampering attacks on the level network are much harder to accomplish or even practically impossible. The important point here is that TLS only protects the transportation of the message between the components, so it only protects the message on a network level. A potential attacker could still try to access one of the components and read, or tamper with, messages.

The next best attack vector is the message broker, since for the distribution of the messages to the subscribers the broker handles the message unencrypted and unsigned. This gives the attacker the possibility to read or tamper the messages and to spoof its identity. To get access to the broker, the attacker would need to directly connect to the broker or take control of one of the components that have a connection to the broker by an elevation of privilege attack. The components with a connection to the broker are the SCRAL, the DFM, the VLC, the SM and LinkSmart-GC. By gaining control over either one of these components, the attacker could get further attack options for every other of the components that are connected to the broker. To mitigate that threat, all components should be hardened against elevation of privilege attacks, with the already mentioned code reviews and penetrations test. To hinder an attack to directly connect to the broker, the broker uses TLS with pre-shared certificates.

Additionally, it would be possible to use a broker implementation that supports access control lists (ACL). With the ACL it is possible to restrict which component can publish or subscribe to which topics and thereby reduce the potential damage a rogue component could do. But this also increases the configuration that needs to be done by the PI-operator.

To further strengthen the trust in the trusted area it is possible to put all components of the trusted area inside a closed network. This can be done by creating a dedicated physical network for the trusted area, or by creating a VPN if a dedicated network is not possible. An attacker would then need to first get access to the network, before he could start further attacks.

9.3.2 Single component attacks

Access Manager

The AM is a beneficial target for an attacker, because if he could change the policies stored on the AM he could get unauthorized access without much effort. Since all of the network level attacks are prevented through the described use of TLS, few attacks from the threat analysis are left.

The first is the spoofing attack by which an attacker tries act as the PI-operator. To do this the attacker would first need to have access to the machine that hosts the AM, second he would need to spoof his identity so the AM would accept him as a PI-operator. The access should be secured by the fact that on dedicated physical network the attacker would need to have physical access; in a VPN, on the other hand, the attacker would need to have the proper authentication to join the VPN.

To act as a PI-operator the attacker would then need to have the authentication credentials of the PI-operator. To mitigate the threat that the PI-operator loses, or unintentionally gives away his credentials, the PI-operator must have proper training in basic security and social engineering countermeasures. When it is not possible for the attacker to spoof his identity, then it would also not be possible to tamper with the policy decisions and the policies stored inside the AM.

A still possible attack would be the repudiation attack in which an attacker, that might be the PI-operator, can change the policies without anyone realizing it. To mitigate such an attack, the AM should log every policy insertion and deletion. In addition, the two-man-rule can be applied, meaning that when one operator changes the policies another operator is informed of this change. In that way, a malicious PI-operator cannot change the policies without the other operators knowledge.

To execute a denial of service attack on the AM the attacker could delete all policies and by doing so make the whole PI unavailable. But to execute this attack the attacker either needs to be a PI-operator or needs to spoof a PI-operator identity.

In the first case, the operator will be held responsible since it will be recognized via the two-man-rule and traced back to his account. If the attacker got hold of PI-operator credentials, the attacker still needs to have access to the trusted area. But even if an attacker accomplishes it to bypass both security measures, there is still the two-man-rule and it can be traced back to the operator credentials.

The two scenarios are hard to distinguish, since both hold the operator responsible. If the attacker didn't leave any trace when he bypassed the security measures of the trusted area, the problem of distinguishing the two scenarios is not solvable by technology and needs to be decided on a case by case basis.

A third denial of service attack could be that the attacker tries to overload the AM with many complex queries. For this attack to succeed the attacker needs access to the trusted network, in addition he needs to have a certificate from, or control over, the SCRAL, the VLC or the LinkSmart-GC component. Then, at least one of the components is creating an unusual load, which will be detected by the performance monitoring and can be traced by the operators.

Furthermore, there is the risk that an operator can use his privilege to give himself more privileges. This threat is also mitigated by applying the two-man-rule.

Data Fusion Manager

Most threats for the DFM are on the network level and through unrestricted access to the broker. They are addressed in first instance by using TLS, as mentioned earlier, and by securing the broker. This leaves only the denial of service, repudiation, and elevation of privilege attacks.

The DFM accepts DFL-statements, which tell the DFM what information resources it should fuse. This can be used in an elevation of privilege attack where an attacker can create new fused streams by sending DFL statements. But if a raw data stream for which a user has no authorization can be fused by the DFM into a fused stream for which the user has an authorization, an attacker could read confidential information. To mitigate this attack, streams need to have an owner attribute and it should not be possible to create a new fused stream without owning the stream or without the proper policies. The ownership of a stream can also be defined with the policies of the AM.

Similarly, the tampering attack, in which a user can manipulate DFL statements, can be mitigated by only allowing the owner of the DFL statement to delete or change it. To further mitigate this DFL elevation of privilege attack, all operations on DFL statements should be logged to make traceable how a stream was created or deleted. This increases the risk of an attacker to be discovered and by that also decreases the time slot in which the attacker can act.

The denial of service attacks mentioned in the threat model are hard to mitigate without limiting the functionality of the DFM. So the best strategy to reduce this threat without limiting the functionality is by giving the right to create DFL statements only to trusted users, which are under a higher supervision than others.

In addition, a close monitoring of the DFM can help to reduce the risk. The security against denial of service attacks then depends strongly on the authentication and authorization mechanisms in ALMANAC.

Resource Catalogue

The threats for the RC are very straightforward, when the RC is only communicating with the VL, the SCRAL and the DFM. Since the SCRAL and the DFM are the only components that create or remove resources and the components of the VL are the only ones to query the resource information from the RC.

If TLS is used then all the spoofing, denial of service and all but one of the tampering attacks are only possible if the attacker has access to the SCRAL, the DFM or the VL.

The risk of information disclosure can be seen as minimal, because the RC only knows that a resource is there and this information has minor confidentiality.

The risk that the SCRAL, DFM, or VL attacks the RC is minimal, but none of the less it might be possible that an attacker gets access to one of these components. To further reduce the risk, the RC can apply the least privilege principal and differentiate between its communication partners and restrict the allowed operation on a per component basis, e.g. the DFM is only allowed to insert resources that are fused streams and it cannot send queries or remove other resources. By doing so, the attacker needs to have access to the SCRAL or the DFM to execute the denials of service attacks in which resources are added to the RC. This strengthens the security of the RC, because the SCRAL and the DFM are not accessible over a public network and the attacker would first need to get access to trusted area or the sensor networks. By applying the least privilege principal, it is also not possible for the VL components to elevate their privilege and insert new resources.

The RC should also log when resources are added or removed and additional monitoring can make attacks visible to the PI-operators.

The only tampering attack that is still possible is done by directly accessing the file system on which the resource information is stored. To mitigate it should only be possible for the processes of the RC to write to files that store the resource information.

In addition the logging information can be used to trace if and when a resource was added by the RC.

Storage Manager

Similar to the DFM the SM might be vulnerable to spoofing attacks on the network level and to spoofing attack over the broker. The threat can be mitigated in the same way as it was done for the DFM by using TLS and by securing the broker.

An additional spoofing attack on the SM is done by acting as the database of the SM, so that it would be possible for an attacker to replace the database with his own. To mitigate this possibility, the database of the SM should be configured to use TLS and a secure authentication mechanism, which the databases should provide. It should be mentioned that the TLS implementation hold up to the requirements we expect for a TLS connection, e.g. client authentication and algorithm restriction.

Since the information on who sends a message into the broker is lost at the broker level, it is not possible to say who inserted information into the SM. This could be solved by requiring signatures on every message before it is send to the broker. However, this would require all components connected to the broker to have a signing and a validation routine for the MQTT message, which needs to be implemented correctly across multiple frameworks. The gain of the signatures would be to know of whom of the four MQTT capable components a message came from. It was decided that the required work to implement the signatures is not worth the gain and instead rely on the broker to not accept message from unknown sources and to restrict the access rights of the connected components.

An already mentioned weak point of the SM is the connection to the database, because in addition to the spoofing threat, the database has also the potential threat of being the target of a denial of service attack and information disclosure. If the SM is storing confidential information than special care must be taken when performing database backups and when a replication server is used to not disclose confidential information.

While the use of TLS makes the network level denial of service, and information disclosure, attacks practically impossible, others are still possible. These will be mitigated by putting the database into a

private network (VPN or a dedicated physical network) with the SM and make the SM function as a gateway to the database. This makes tampering attacks that change the already stored information difficult because the SM doesn't support an operation to later change the information. Also a weak TLS implementation from the database can be compensated by using the private network.

For the denial of service attacks on the SM itself, it should be safe to rely on the requirement that all components first have to establish a TLS connection. This makes the denial of service attack via querying only possible to the VL and the denial of service via inserting only possible to the DFM and SCRAL. But, since the SM should be able to handle large amounts of queries and large amount of inserts the risk of such denial of service attacks should be minimal. To further mitigate the threat, the SM should be monitored, so that unusual behaviour can be discovered.

To reduce the risk of NoSQL injections into the SM to tamper with the information stored in the database, the SM should clean-up and sanitize all input.

9.3.3 The untrusted Area

The untrusted area is everything that is out the control of the PI-operator, meaning that the operator might not be able to simply restrict which connection into the trusted area should be allowed.

There are three components (LinkSmart-GC, VLC, and SCRAL), which are connected to the trusted area and to an untrusted area. These components act as a gateway into the trusted area and must therefore be especially hardened against potential attacks. Each of the gateway components is connecting the trusted area with a different untrusted area.

LinkSmart-GC interconnects multiple federated PIs over a potentially untrusted network. In addition, the configuration of the federation might change and the LinkSmart-GC instances might need to adapt to these changes. But an attacker might also make one LinkSmart-GC instance believe that a change has happened and by doing so get access to the federation.

The SCRAL connects a PI to the low level networks over which the sensors communicate. The low level networks often use wireless communication and often connect widely distributed sensors and through this fact, it might be physically accessible for an attacker. Both of these attributes make the low level network a potential threat for the trusted area.

The VLC provides access to its PI, for all federation users. These federation users might connect through an untrusted network to the VLC. In addition, the configuration of the federation might change and, as a consequence, federation users might lose or gain the authorization to connect to the PI. The VLC needs to be able to deal with untrusted connections that might come from an authorized or an unauthorized entity. However, to verify the authorization, the VLC first needs to allow any connection.

The FIM is in a way special, since it doesn't act as a gateway into the trusted area, but it is in the trusted area. The function of the FIM is to provide a login mechanism for the federation users, where the federation users get access tokens with which they can access a VLC. The advantage of the FIM is that there only need to be a few FIM instances, but there can be many PIs to which the federation user can connect to with its access token. For scalability reasons there might even be multiple VLCs in one PI to which a federation user might connect to. The VLC remains to be a stateless component, which only has to check the validity of the access token for a request, while the FIM holds the state of the federation users and their attributes.

While constructing the threat model it became clear that the gateways to the untrusted areas need to have a way to identify communication partners and to verify their authorization. It is also necessary that the gateways don't disclose information and that tampering can be detected. Since all gateways need to fulfil these requirements the Security Enforcement Point (SEP) was introduced.

The SEP introduces three conditions before one of the gateways accepts a communication from the untrusted areas. The first condition is that there must be a way to communicate of a secure connection, which means that an attacker should not be able to listen to and not to be able to tamper with the communication. The second condition is that the gateway component is able to identify the communication partner. The third is to verify that the identified communication partner has the

required authorization for the attempted action, which can be queried from the AM. If the first condition fails, the communication must be aborted. If the second condition fails, the communication must be aborted, and if the third condition fails, the communication must be aborted. This is necessary because the third condition requires that the second condition is fulfilled and the second condition requires the first condition to be fulfilled.

How the single components implement the SEP differs, since they all have different requirements, which will be described in the following section where the specific counter measures are discussed.

FIM

Before the counter measures for the VLC can be discussed, the security of the FIM should be clarified, since the VLC is strongly depending on the security of the FIM.

The implementation which was chosen for the FIM in ALMANAC is the KeyCloak server (http://keycloak.jboss.org/). The FIM needs to use TLS for all authentication requests, since it implements the OpenID Connect standard for authentication and a requirement for a secure OpenID Connect is that all communication have to use TLS when access tokens are transmitted. The requirement for TLS should be similar to the requirements in the trusted area for TLS. For a FIM operator this means that the FIM should use the highest TLS standard which is possible for the federation user clients.

The use of TLS reduces the risk of compromising access tokens and federation user credentials, since an attacker cannot read the access token directly from the network traffic. To further mitigate the threat of compromised access tokens, the access token should have a small life span of a few minutes, depending on the security needs of the federation. To further hinder the misuse of access tokens, the scope of the access token can be limited, so when an access token gets compromised the token can "only" be used for a specific scope of services. The possibility of forged access tokens can be reduced by the fact that the OpenID Connect standard makes use of JWT, which can be signed by the FIM. This would require an attacker to first find a hash collision or the private key of the FIM.

The CSRF attack is mentioned here since the KeyCloak documentation mentions this threat for its administration interface, but also describe mitigation measures: "The only part of KeyCloak that really falls into CSRF is the user account management pages. To mitigate this, KeyCloak sets a state cookie and also embeds the value of this state cookie within hidden form fields or query parameters in action links. This query or form parameter is checked against the state cookie to verify that the call was made by the user."

The threat of clickjacking can be reduced by enforcing the same origin policy for all iframes.[40,41]

The thread of open redirectors can be mitigated by requiring the services that depend on the access tokens from the FIM to be registered by the FIM operator, so that a malicious service cannot redirect a client to the FIM to login.

Brute force attacks can be mitigated by blocking clients that have tried too many times for some time. However, this might make a denial of service attack possible in which the attacker blocks the access for all federation users. It is further possible to block the connection for an IP address that has failed at too many login attempts, but this could be bypassed by an attacker who has a botnet[42] available. So in addition the federation users should use strong passwords or stronger authentication mechanisms like one-time passwords (OTP).

The "hacking" of the password database would need an attacker to have access to the operating system. To mitigate this threat scenario, the passwords in the database are hashed and it is possible to increase the number of hashing rounds done by KeyCloak. However, the hashing round might be very expensive. The decision on how many hashing rounds should be used is up to the FIM operator and the security needs.

[40] http://www.w3.org/TR/CSP/

[41] http://tools.ietf.org/html/rfc7034

[42] A botnet (also known as a zombie army) is a number of Internet computers that, although their owners are unaware of it, have been set up to forward transmissions (including spam or viruses) to other computers on the Internet.

SQL injections is possible, but currently there are no known SQL injection attacks for KeyCloak.

What should not be forgotten is the possibility of an attacker that is able to get registered as a federation user. Such an attacker cannot be discovered on a technical level and needs to be identified at the registration process. Since the registration is the only way to get officially access to the system, it should be handled with care. For a federation there should be an agreement on how new users are added to the federation and what kind of identity proof is required. This is necessary since the overall security of the registration process is only as secure as the weakest registration authority. How the registration is performed, needs to be decided by the entities that build the federation because it depends on the security requirements and the means of the organizations.

The process with which each entity registers its federation users should be communicated between the entities that build the federation and should be agreed upon by the entities. For more details on how a registration can be performed see NIST SP 800-63-2 chapter 5.

An example of a registration is where the federation user needs to supply a document issued by her government, in person, at a trusted registration authority, which then can vouch for the registered user. A company or stakeholder that is an entity of the federation could be the registration authority for its employees and the identification can be done at the employment interview. When the company wants to register an employee for the federation it has to vouch for the identity of the employee. When the registration of a federation user is completed the user will be provided with the sign-in credentials. With the sign-in credentials the user is able to sign-into the federation. Typical sign-in credentials are user name and password, or certificates and private key.

VLC

The SEP of the VLC uses TLS as the secure channel with exactly the same configuration provided by the FIM.

For identification of connecting users, the VLC takes the access token, which is signed and issued by the FIM according to the OpenID Connect standard. It then checks the signature, the issue, the scope and life time of the access token, which all have to be valid.

For the authorization, the VLC sends a request to the AM. In the request the VLC adds the role of the federation user and other attributes, which might be relevant for the policy decision of the AM. The result of the policy request is than enforced by the VLC. If the VLC gets no reply from the AM the request from the federation member is denied, otherwise an attacker could run a denial of service attack on the AM to elevate his privilege.

The security of the VLC depends on the security of the FIM, because if an attacker is able to forge or compromise an access token the VLC cannot distinguish between a valid and a fake request. But since the FIM signs the access token this might be a tough problem for the attacker. The same is true for an attacker, which tries to guess access tokens.

A possible attack on the VLC could be to create a clone of the VLC front end and try to trick federation users into using the VLC clone. This could be done by registering similar DNS domains or by open redirector attacks. This threat can be mitigated by using service registration as in KeyCloak and OpenID Connect redirect feature. Alternatively, the threat can also be mitigated when the client gets the token directly from the FIM, by configuring the clients to compare the domain of the service they want to send the access token with the scoped services in the token.

The spoofing of components in the trusted area should not be possible thanks to the use of TLS with client authentication, since all components need a valid pre-shared certificate. The same is true for tampering and information disclosure inside the trusted area. Through the use of TLS, the traffic between the federation users and the VLC should, in fact, be secured against tampering and information disclosure.

The possible denial of service attack on the VLC depends on the attacker to be able to change the policies inside the AM. Here the mitigation strategies against tampering and denial of service attacks from the AM also apply.

LinkSmart-GC

For LinkSmart-GC the SEP looks slightly different from the SEP of the VLC. The secure channel is an own implementation of LinkSmart-GC, which is based on Elliptic Curve Digital Signature Algorithms (ECDSA), Elliptic Curve Diffie-Hellman key Exchange (ECDHE) and AES Galois Counter Mode (AES-GCM)[43]. To use ECDSA all LinkSmart-GC instance need to have a certificate with an ECDSA key pair. The certificate needs to be signed by a federation certificate authority, which can create its own certificates. Through the use of ECDSA certificates the single LinkSmart-GC instances prove to each other that they are part of the federation. Like the VLC the LinkSmart-GC instances request their local AM if a request is authorized.

With AES-GCM the confidentiality and integrity of the message exchanged between LinkSmart-GC instances should be guaranteed, so tampering and information disclosure attacks are mitigated.

For the trusted area connections, LinkSmart-GC uses TLS as described by the trusted area section to mitigate spoofing, tampering and information disclosure attacks.

For HTTP tunnels it might be possible that an attacker might try to mimic an HTTP service from one of the other components in the trusted area. This can be mitigated by certificate pinning, so that a tunnel to a specific service is associated with a specific certificate.

To mitigate the repudiation attacks, LinkSmart-GC needs to have detailed logs on who registered a tunnel and who communicated over a tunnel. The logs should save the information about who, from where and what has been done by the involved actor.

Since the only component that can register tunnels is the VLC, and the VLC and LinkSmart-GC are connected over the trusted area, the GC can restrict the use of registration/unregistration to the trusted certificate of the VLC and to the network of the trusted area. This measure should also mitigate the potential denial of service attack, where the attacker deregisters all tunnels or is able to replace all tunnels. A denial of service attack on LinkSmart-GC depends on the attacker being able to change the policies inside the AM. Here the mitigation strategies against tampering and denial of service attacks from the AM apply.

LinkSmart-GC does not verify if the entity that removes a tunnel is the same entity that has created the tunnel. Since the creation of tunnels can only be done by the VLC, this does not affect the ALMANAC platform. To further strengthen the tunnel registration process and, since more harm can be done by deleting a tunnel, at the time of tunnel creation LinkSmart-GC should create a random token and give it to the creator of the tunnel. Only the owner of the token can than unregister the tunnel.

SCRAL

The SEP of the SCRAL is similar to the SEP of the VLC. For the secure channel, the SCRAL uses TLS, but has to rely on the network-level gateways to ensure that low-end TLS implementations are up-to-date (if available). The identification can either be done via certificates or with the help of the FIMs OpenID Connect tokens. This depends on how the connection between the SCRAL and the low level networks is implemented. If the SCRAL exposes a socket where the gateways connect to, then the OpenID Connect tokens can be used. If the SCRAL polls the gateway for information, the gateway needs to have a certificate that is trusted by the SCRAL. This unfortunately is not always true, e.g., for serial connected network concentrators. In such cases, protection from physical access to the hardware shall be devised whereas security on transferred information shall rely on the low-level network security mechanisms.

Like the VLC and LinkSmart-GC, the SCRAL queries the AM for authorization decisions.

For trusted area connections, the SCRAL uses TLS, as described by the trusted area section, to mitigate spoofing, tampering and information disclosure attacks. If certificates are used for identification, then

[43] https://tools.ietf.org/html/rfc5288

the certificates can be pre-shared, since the number of gateways should be limited and known at deployment time.

Compromising a gateway cannot be prohibited by the SCRAL, but it might be possible to detect unusual behaviour. To help the PI operators in spotting these breaches, the SCRAL should have detailed logs on when the status of a device is changed or when someone tried to.

In addition, the SCRAL can also create owner tokens, to make it difficult for an attacker to remove or replace resources, like it was described for LinkSmart-GC.

Most of the possible denial of service attacks on the SCRAL are from inside the trusted area and by controlling components like the MQTT broker or the AM. The only possible attack from the untrusted low level network is a distributed denial of service attack. This attack needs to be mitigated by the SCRAL with the methods defined in the scalability section.

The SCRAL accepts many different kinds of formats for information, caused by the nature of its application. Due to this fact the SCRAL has the biggest potential for elevation of privilege attacks, which use a weakness in the processing of these formats. The libraries and the code of the SCRAL needs to be reviewed to make sure that the input data is properly verified.

9.4 Registration at the FIM

With the registration, a user, or an application, gets the right to sign-in to the federation. After the registration is done the user or application becomes a recognized user of the federation. Since the registration is the only way to get official access to the system, it should be handled with care.

For a federation there should be an agreement on how new users are added and what kind of identity proof is required. This is necessary since the overall security of the registration process is only as secure as the weakest registration authority. But how the registration is performed needs to be decided by the entities that build the federation. This is caused by the fact that it depends on the security requirements and on the means made available by involved organizations. The process with which each entity registers its federation users should be communicated between the entities that build the federation and should be agreed upon by them. For more details on how a registration can be performed see (NIST, 2013; chapter 5).

An example for registration is when the federation users need to supply a document issued by their government in person at a trusted registration authority, which then can vouch for the registered user. A stakeholder (e.g., a company) that is an entity of the federation could be the registration authority for its employees and the identification can be done at the employment interview. When the stakeholder wants to register an employee for the federation it then vouches that it knows the identity of the employee.

When the registration of a federation user is completed, the user will be provided with the sign-in credentials. Using the sign-in credentials the user is able to sign in to the federation. Typical sign-in credentials are user name and password, or certificates and private key.

9.5 Sign-in at the FIM

The federation users exploit the FIM to sign into the federation. In the sign-in process, the federation users provide their credentials and receive a session token, with which they can authenticate before the services of the federation. The session token is a piece of information that must include:

- the identity of the user
- the attributes of the user that are needed by the AM (roles, ...)
- the issuer of the session token
- the expiration time of the token
- the restriction for what services/PIs the token can be used for.

By using the session token the federation services don't have to provide the capabilities to verify the sign-in credentials of the single user. This reduces the complexity of federation services. The stronger argument for using a session token is the fact that it eliminates the risk that a federated service steals or leaks the sign-in credentials of a user, since only the FIM receives this information. This reduces the risk of losing sign-in credentials.

Additionally, it is possible to restrict what can be done with the session token to a subset of what the user can do with his credentials. Of course the session token still needs to be kept a secret, since an attacker can still use it to cause damage. But, because of the expiration time, the time in which the attacker can act is reduced, compared to losing the sign-in credentials.

To keep the session token a secret, it needs to be transferred securely, either by using TLS or by encrypting the session token directly. For maximum security it is even possible to do both. For the ALMANAC platform TLS 1.2 will be the minimum requirement and session token encryption could be added later on.

10. Federation Perspective

Cities are heterogeneous in several aspects and already possess extensive metering and communication infrastructure. Service owning, managing and liability for such an infrastructure may vary between cities and even within the same city. This situation leads to tight requirements for ICT systems that need to support different, collaborating entities, while at the same time being able to let such entities work independently, autonomously and securely. This heterogeneous set-up requires systems that are able to effectively cope with existing ICT infrastructures and systems as well as with brand new solutions. In such a sense, a smart city platform should, in first instance, be a concrete integration platform. Moreover, as the platform operates in environments with well defined, and strictly intertwined legal frameworks, it should be adaptable to regulations and policies of the city and its structures as they change.

In this context, the ALMANAC Smart City Platform adapts the concept of cloud federation [14], to real city requirements and uses cases. Broadly speaking, the "cloud federation" term refers to the interoperability of different cloud deployments as if they were one, with each deployment keeping firm control of its own subsystems.

Federation is more than a technical solution. It allows different stakeholders to work together depending on the context and needs. Additionally, federation enables stakeholders to attain public policies and regulations shaping the operative boundaries of their respective smart city systems, allowing such systems to evolve following the overall city changes.

10.1 Federation Concepts and Definition

To better understand the federation concepts and involved technological challenges, the terminology regarding federation is developed in this section.

Platform Instance

A Platform Instance (PI) is a sovereign instance representing an organization, institution or, authority in the ALMANAC system. The PI sovereignty is instantiated as the capacity of controlling it's the platform resources, users, and policies, for both inside and outside access to the PI. In other words, a platform instance is a domain where the complete control resides in a single administration and is seen in the ALMANAC system as a single PI.

Federation

A federation is a series of sovereign PIs, which agree to share the same networking overlay (established by LinkSmart GlobalConnect). Their identity is proven by the same identity provider. Entities can join a federation by following a well-defined and shared admission process, which is established by the founding partners of the federation. This ensures that all members of a federation have an identity and are reachable over the Internet. Reachable means that all members must be able to receive federated requests from any other member of the federation.

Federation preserves sovereignty, i.e., it does not enforce particular responses or access obligations to members, which are free to respond to incoming requests according to their own, internal, policies and regulations.

Policy

A policy is a rule which specifies a particular right of a subject, e.g. a user, on a resource or any system object, e.g. a public light. Through policies PIs regulate themselves, inside the PI and between PIs in a federation. The policies are just defined by and only by the operator of the PI. Therefore, there is no centralized control in a federation. For more detail, see security perspective.

Association

An association is an abstract concept, which happens when two or more Entities of one federation agree to work together and they want to share resources. The associations are translated into the

federation via access policies and in some cases via other ALMANAC concepts, e.g. Data-Fusion statements.

Relationships between Federations and Associations

Federations are prerequisites of associations. Two organizations can work together, only if they are in the same federation. This first condition is necessary but not sufficient to establish the collaboration. For two entities to work together they must also establish an association. In some cases, federations are built to establish associations. In such cases, the federation and the association concepts completely overlap. Furthermore, associations are built on collaboration where one or more entities contributes with resources in the association, and other entities consume the resources.

10.2 ALMANAC approach to federation

In ALMANAC, we take the approach of making one-to-one associations, where expected, between each member (platform instance) of the federation. For instance, in Figure 31, the platform instances AMIAT and Turin Municipality are associated because they have defined policies that allow them to exchange data, while AMIAT and SMAT do not have such mutual policies and therefore are not allowing each-other to access their respective data.

In Figure 31, the three platform instances (Turin Municipality, AMIAT, and SMAT) are all part of the same federation, which means that they share a common communication infrastructure based on LinkSmart GlobalConnect (LS-GC). Although all the platform instances of the same federation can talk to each other, it is only when there is a one-to-one association with corresponding policies that real content can be exchanged.

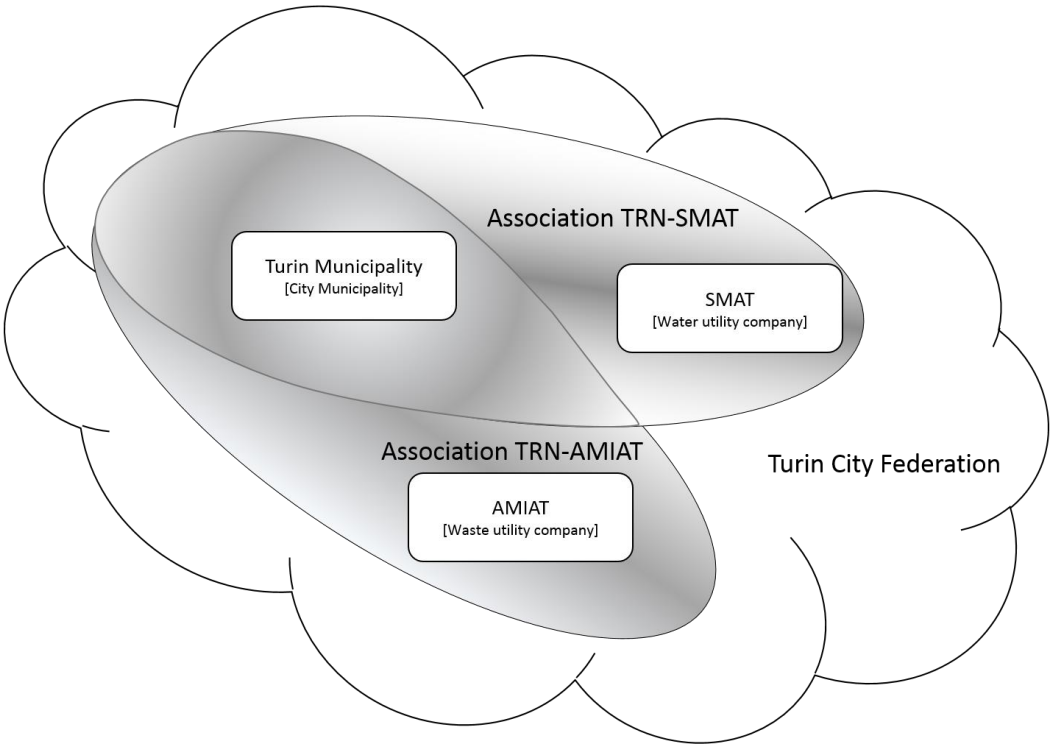


Figure 31: Multiple federations between ALMANAC Platform Instances.

10.3 Federated ALMANAC Deployment

ALMANAC defines a framework in which multiple Platform Instances can participate into associations within federations offering cross-city and cross-nation services. Such associations, typically built around service-exchange agreements are deployed in a federation, on the technical standpoint, by using the same network overlay provided by LinkSmart GlobalConnect. Platform instances exchange

data and distribute tasks according to roles and privileges defined in the association agreement and managed through access control mechanisms provided by the Security Framework. Each Platform Instance can be part of more than one association in a federation or several federations, thus providing the basis for an interconnected deployment of the ALMANAC services across cities.

Routing of requests and data through the different PIs belonging to an association is managed at the Virtualization Layer and exploits a naming/addressing scheme based on standard DNS names and CNAMEs. In the following, the implementation of such federated ALMANAC deployment is described.

The federation is built upon LinkSmart GlobalConnect (LS-GC) providing the federation transport layer also known as overlay and the Federated Identity Manager authenticating the identities of each platform instance in a federation. The instances of LS-GC establish the overlay by connecting all PIs within a Federation, in which each instance of LS-GC interconnects through the LinkSmart SuperNode. To establish a secure connection, each PI provides its identity using the Federated Identity Manager. Resource sharing is built around the associations, which are transformed into policies. Each PI must deploy policies in its corresponding local Access Manager (AM) according to its role in the association. The policies are enforced by LS-GC, for outgoing and incoming federated request. Finally, the Virtualization Layer Core (VLC) handles the request for its corresponding PI transforming them into federation requests if needed. The VLC hides to the requester if a request tries to access a PI resource or a federated resource. For example, when a request arrives in the VLC, it tries to resolve the requested data locally, if it is not possible, it then starts a federated request in LS-GC. If the resource is a federated resource, it could be a known resource or an unknown one to the VLC. In case of a known resource, the VLC accesses it through a federated request using the local instance of LS-GC. If the resource is unknown, then the VLC starts a federated discovery request. In both the previous cases and before the federation request leaves the PI, LS-GC enforces the policies accessing the local AM to determinate if the requester is allowed to operate or discover the external federated object. Both discovery and operation federated request works in a similar way up to this point, with the difference that the discovery request is multicast and the operation request is unicast. The federated request arrives to each end LS-GC involved in the request. The *PI-end* local LS-GC determinates if the request is allowed to operate in the *end-PI* by the *remote-PI*, this is done by accessing the end AM in a similar way as the outgoing request done before. Finally, if the request has been approved, LS-GC redirect the request to the resource or to the VLC for further processing, for an operation federated request or to discovery federated request process, respectively.

10.4 Outlooks for more advanced federation concepts

The current federation definitions and concepts can be used in future scenarios also considering interactions between different federations as well as federation nesting. The currently foreseen options include so-called multi-federation and federation of federations, which are described in the following.

Multi-Federation

It could be the case where a platform instance belongs to one federation, e.g. a city federation such as Bonn, but it wants to belong to another federation, e.g., a corporative federation such as Fraunhofer. Currently, this is supported with some limitations. When a platform instance belongs to two federations, this means that it has two different entities one for each federation. For all practical purpose, there are two different PIs coexisting in the same control domain. There is no way to avoid the duplicity of identities due to the nature of federations. Each federation will have different sets of network overlays and identity provider, creating different sets of communication channels and certificates. Currently, such scenarios are possible, but there are not many support tools to help administrate these cases.

Federation of Federations

A federation could exist inside another federation, e.g. a city build a federation while it already existed in a country-level federation. Such scenarios are possible, but they have limited support. In this case each federation can be encapsulated in a higher hierarchy PI which belongs to the higher federation. In the city example, SMAT and AMIAT belong to the Turin federation, where the whole federation

belongs to the Italy federation. Security and authorization implications for this scenario clearly need further analysis and investigation.

11. Scalability Perspective

11.1 Basic Concepts and Terminology

This section describes scalability issues and the adopted measures in the context of the ALMANAC platform, i.e., the ALMANAC PIs and the components running on these PIs, and, the corresponding deployment of multiple PIs. User applications invoke services exposed by ALMANAC PIs and benefit from the platform scalability in terms of increased usability and performance.

This section mainly focuses on user-centric scalability whereas other application-specific scalability issues are considered out of scope and they are not discussed in this document.

Nodes, Resources and Scalability

A stated before, ALMANAC PI encompasses a deployment of a collection of PI components (such as the VL, RC, SM and SCRAL). Each ALMANAC PI component runs on a computational resource (node) with a certain specified computing and/or storage *capacity*. Examples of nodes include: physical or virtual server machinery, cloud storage services and execution containers in the cloud.

Computational resources are thus constrained by locally hosted deployments or in the case of cloud based provisioning by the corresponding SLAs (Service Level Agreements).

An ALMANAC PI can be configured as spanning over one or more nodes (see Figure 32).

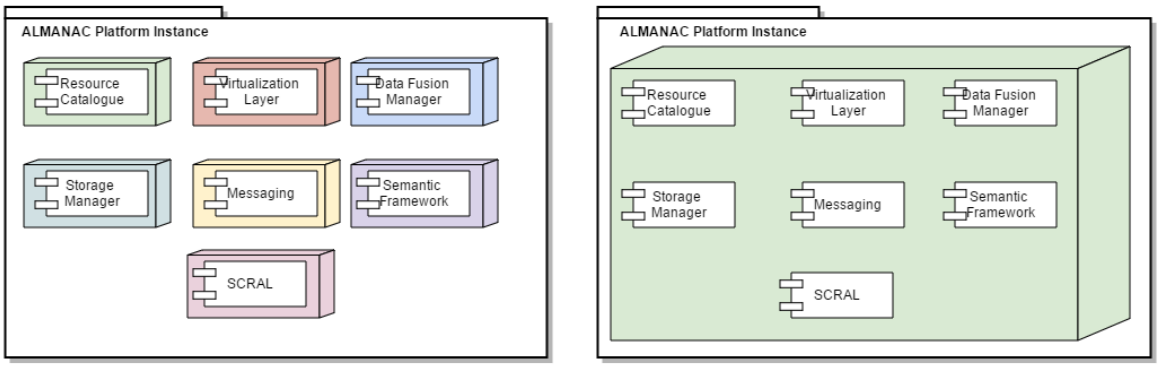


Figure 32: ALMANAC Platform Instance (PI) and nodes.

For the sake of clarity, we would like to differentiate between performance and scalability. By performance we refer to the capability of a system to provide a certain response time *with a given set of nodes and resources*, e.g., to serve a defined number of users or processes a certain amount of data from a server with a certain capacity specification. Although no standard definition is available for these terms (Lehrig et al., 2015), most of the available literature uses a similar definition for performance, e.g. (Wilder, 2012) where it is defined as "… an indication of the responsiveness of a system to execute any action within a given time interval".

Scalability we would like to define in analogous to the definition in (Lehrig et al., 2015) as the ability of a system to increase the maximum workload it can handle by expanding its quantity of consumed resources. Similar definitions are "the ability of a system either to handle increases in load without impact on performance or for the available resources to be readily increased" (Wilder, 2012) or "the capability of a system, network, or process to handle a growing amount of work, or its potential to be enlarged in order to accommodate that growth" (Bondi, 2000).

Scaling is thus about allocating more *resources* for an application, i.e., resource *provisioning*. In this discussion, we assume that the system has been designed so as to use the available resources as efficiently as possible i.e., by maximizing the performance with a given set of resources. Examples of resources needed by an application usually include CPU, memory, disk (capacity and throughput), and network bandwidth. An application or service is said to be scalable if when we increase the resources

in a system, it results in increased performance in a manner proportional to resources added. Resources can be handled in *scalability units,* i.e., groups of resources that could be scaled together.

Vertical/Horizontal scaling

The scaling discussed here concerns the steps that may be taken when the available resources run out and the application does not fulfil its functional or non-functional requirements - the maximum workload of the system with the given resources is reached. We may then scale the system to increase the maximum workload it can handle by expanding its quantity of available resources. We can increase the quantity of consumed resources by increasing the amount of resources within existing nodes, or by adding more nodes.

To *scale up (or scale vertically)* is to increase overall application capacity by increasing the resources within existing nodes. In ALMANAC, e.g., increasing the capacity of the storage server (node) running the Storage Manager in a PI.

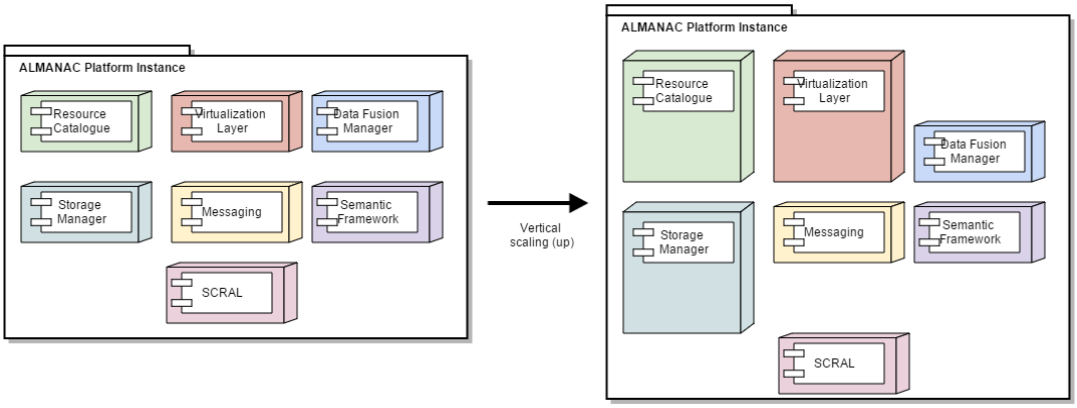


Figure 33: Boost capacity of node, scale up.

Scaling up is usually the simplest and cheapest solution, as is does not require any changes to the design, code details or deployment of the application. While less complex (and sometimes cheaper compared to re-design or code improvements to increase performance) there are limitations to this approach compared to scaling out.

To *scale out (or scale horizontally)* is to increase overall application capacity by adding nodes, e.g., adding an additional Storage Manager node to an ALMANAC PI.

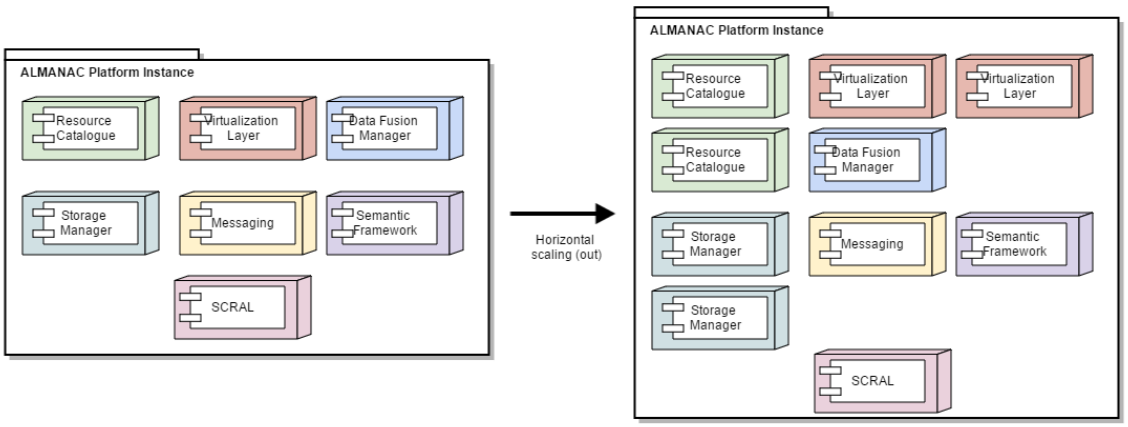


Figure 34: Scale out by adding nodes for PI components.

Scaling out increases the overall application capacity by adding entire new computational nodes. Scaling out tends to be more complex than scaling up, and has more impact on the application architecture. In ALMANAC, scaling out implies the addition of nodes for a PI. We may scale out an ALMANAC PI by adding nodes for specific components (e.g., a Storage Manager) and implement

support for this at the component level. Another option is to add nodes by duplicating the PI and implement support for this at the PI level. In the case of horizontal scaling, the system should also be able to adapt to shrinking demand for resources, to *scale in*. This property is often referred to as *elasticity* (Lehrig et al., 2015).

When all the nodes supporting a specific function are configured identically - same hardware resources, same operating system, same function-specific software - we say these nodes are *homogeneous*[44]. We would add that components executing on different nodes may be homogenous with regards to functionality – all nodes support the same functions – and data or state – all nodes share the same data. This has implications on the design of horizontal scaling.

An *autonomous* node does not know about other nodes of the same type, similarly the same term can also be used for components.

In ALMANAC we will not test scalability by creating a model of the system and performing simulations. The approach we will take is to identify the scalability issues for an ALMANAC PI by analysis of deployed capacity, against application performance requirements, identifying scenarios where the maximum workload may exceed the capability of the PI or components, investigate common design patterns for how these scenarios may be addressed, and, determine how the design of PI components deals with scaling up and out. The validation of these designs will be made by extending the performance tests (ALMANAC MS10, D8.8.2) by exceeding the maximum capacity indicated by the performance tests and note how the system responds. Similar to stress testing, this gives an indication not only how the system will perform but also tries to "break" the system.

11.2 Issue identification and analysis

Analogous to the threat analysis in the security section, here we list a number of scenarios where a high value of the attribute may cause the workload to exceed the maximum that the PI or individual components can handle. Common design patterns to address these problems are described and the design of each component is described from a scalability viewpoint; the possible bottlenecks of each component, the possibility of scaling up or out, and the design implications.

11.2.1 Scenarios for scalability requirements of the Platform

Attributes that may affect workload of PI or components

- Platform internal
    - The number of concurrently reporting resources/datastreams from the SCRAL
    - The number of concurrently reporting DFM generated resources/datastreams
    - The number of generated data instances that should be persistently stored in the system
        - Number of generated resources
        - Number of observations
- Cloud API generated
    - The number of concurrent requests for live data.
    - The number of concurrent requests for Historical data
    - The number of concurrent requests for resources from the Smart City Resource Library API

Performance attributes affected

- Response time
    - Service time - how long it takes to do the work requested.

[44] Wilder, Bill. Cloud Architecture Patterns: Using Microsoft Azure. Sebastapol, CA: O'Reilly Media, Inc., 2012

  - Wait time - how long the request has to wait for requests queued ahead of it before it gets to run.
  - Transmission time – How long it takes to move the request to the computer doing the work and the response back to the requestor.
- Throughput
  - The amount of work accomplished in a given amount of time.
- Resource usage
  - CPU usage
  - Memory usage
  - Storage usage
  - Network usage - data sent and received

11.2.2 Performance and Scalability Design

Some examples of common design patterns used for performance and scalability are summarized here for convenience so that they may be referenced in the component scalability design section. More comprehensive descriptions may be found in e.g. (Wilder, 2012), (Homer et al, 2014) or (Fowler, 2002).

Caching

When certain sets of data are frequently accessed, these may be copied to fast storage located close to the requesting application. E.g. HTTP caching may also be used to avoid unnecessary load on the system by caching data at the HTTP client. The Cloud API layer may also cache data for applications.

Materialized Views

Applications and other components may have need for views on data that is not stored or formatted in a way optimal for the query required to produce this view. The system may then generate prepopulated views over the data in one or more data stores.

Throttling

To avoid that a single application or input source degrades the entire system, the services provided by the system may be temporarily limited. E.g. an application sending a lot of requests may get a "503 Service Unavailable" response telling it to wait, or some functionality of the PI may be prioritized in case of insufficient resources.

Data partitioning

Data stored in the system may be physically divided into separate nodes, so that they are not homogenous with respect to the data they manage. Using horizontal partitioning, the nodes may use the same schema but hold different parts of the data (e.g. different storage manager nodes may hold the data for different periods of time). With vertical partitioning, nodes will hold different parts of the schema, e.g. depending on how frequently the data is updated or accessed. When different parts of the schema are handled by different nodes based on business or usage context, the term functional partitioning is sometimes used.

Load balancing

When the maximum workload of a single component is reached, redundant deployments of the component are created and a load balancing system dynamically distributes workloads.

Queue based load leveling and competing consumers

Instead of passing requests directly on to other components, a message queue can be used to implement the communication channel between the components. The sender component(s) post

requests in the form of messages to the queue, and the consumer component(s) receive messages from the queue and process them, each at its own pace. This way, fluctuations in workload and differences in throughput between different parts of the system can be balanced, and individual components can be scaled out to optimize throughput.

11.3 Local hosting vs cloud hosting

An ALMANAC PI may be hosted on physical or virtual hardware, in an environment owned and operated by a business (e.g. the water utility) or in a cloud environment (e.g. Amazon, Azure).

Depending of the choice of hosting it may be possible to scale up or out automatically. In any case and at the very least, the components need to be able to indicate, when queried or by events, that the capacity limit is being reached. The systems administrator or an auto-scale component may use this information to start provisioning new resources. When automatically scaling out, the component should also be able to be elastic and scale in if there is less demand for resources.

11.4 Platform Instance Scalability Design

The following section describes how the platform components may be affected by the scenarios and how the design of the component makes use of added resources to make the system scalable. Scalability is primarily discussed at PI component level. It is also possible to divide the responsibilities of ALMANAC platform instances to assure that the load of a single PI is manageable and the resource provisioning for each instance is adapted to the expectations for the area of this platform. E.g. instead of creating a PI for Turin City, there could be one for traffic information, one for waste management, one for water and one for electricity.

11.4.1 Virtualization Layer

The Virtualization Layer acts primarily as a front-end service for an ALMANAC platform instance. It is mainly limited by input/output (IO) bandwidth between the client and the Virtualization Layer, as well as between the Virtualization Layer and the other ALMANAC components. If there are many clients, the number of concurrent open connections will be a limitation. This is especially true if the request time to other ALMANAC components increases (e.g. due to more complex or larger requests), in which case it takes longer time to respond to the client, leading to more concurrent connections. The limitation in the number of concurrent connections will also apply when using the WebSocket services, which maintain an open TCP connection with clients.

The Virtualization Layer does not make an intensive use of CPU or memory. Due to its mono-thread nature, the Virtualization Layer scales best on servers with few but fast CPU cores.

With its architecture – which is mainly stateless and without local database – it would be possible to do some load balancing (for instance at TCP level) by adding several Virtualization Layer instances in parallel. If this were to be a solution commonly needed, the Virtualization Layer could be improved by implementing this load balancing directly within it, allowing the instances to be aware of each-other.

11.4.2 SCRAL

The main SCRAL duty is to interface and abstract physical networks, and devices, monitoring city-level data. As such, several aspects shall be tested for performance, including maximum sensor cardinality, maximum sampling frequency, etc. The applied testing methodology aims at identifying the performance limits / bottlenecks of the SCRAL and to relate them to specific features (e.g., cardinality vs. frequency), accounting for relative influences and inter-dependencies. The tested aspects encompass:

1. Capability to interface increasing number of devices (Cardinality)

2. Capability to interface increasing number of networks (Network Cardinality)
    i. multiple instances for the same technology
    ii. different technologies
3. Sampling frequency, given a fixed number of connected devices (Frequency)
4. Memory and CPU workload under the above conditions (Workload)

This set is iteratively improved/amended to better characterize identified issues and bottlenecks. For each aspect, criteria for scalability are defined and proper software tools are adopted to capture relevant figures (e.g., memory occupation, latency, etc.). As a general procedure, reference values (or normal working conditions) are defined for each tested aspect, and the corresponding workload is then increased to reach the component limits.

Ad-hoc set-up for isolating testing and tested components are carefully applied to avoid measuring workloads and figures related to the testing tools rather than to the component under test. Testing is performed both in isolation and in real-world conditions. In the former, the SCRAL runs in a controlled test machine, with no other processes / components running that could hamper / influence the test results. In the latter, the same set of tests are performed in a platform instance running the other components of the ALMANAC platform.

Current evidence show an acceptable capability to handle increasing number of sensors. Preliminary results show in fact that a single SCRAL instance is able to generate up to 400k simulated sensors[45] and the relative data at a sampling rate of 1 measure per sensor every 10 minutes, which corresponds to an event rate of about 600 events per second (400k*1/600s = 667 1/s). This result shall however be interpreted in the light of the maximum number of sensors connected to a single instance, which might vary depending on the technology (e.g., ZigBee allows for a maximum of 255 nodes in a single network), on the physical interfaces of the hardware hosting the SCRAL, etc.

In principle, the SCRAL is able to scale both vertically and horizontally. In the former case, the highly parallel architecture of the SCRAL, where every device driver runs on separate threads, allows to easily exploit increasing numbers of processors and available RAM memory. In the latter, the loosely coupled configuration of ALMANAC PIs allows easy deployment of multiple SCRAL instances, in the same PI. This allows the SCRAL to easily scale in and out, depending on the current platform load conditions.

11.4.3 Data Fusion Manager

The Data Fusion Manager will be tested considering the following aspects:

1. Load tests:
    a. Incrementing load test: Gradually increasing the number of events
    b. Burst load test: Simulating sudden bursts of events
2. Query set tests
    a. Deploying several statements simultaneously
3. Type of data processing
    a. Event routing: Dynamically increment the amount of DFM statements which propagates the event/stream from a topic to another in the same local broker.
    b. Event aggregation: Dynamically increase the amount of DFM events/streams which a statement put together applying an operation.
    c. Event annotation: Dynamically increment the amount of DFM statements, operating in a single stream adding or applying an operation.
    d. Event fusion: Dynamically increase the amount of DFM events/streams from different sources which a statement put together applying an operation.

[45] On an Intel I5 laptop with 8GByte of RAM and 500GByte of HDD

11.4.4 Storage Manager

One scenario is that the number of concurrently reporting DFM generated resources/datastreams from either the SCRAL or the DFM is very high. This may affect the Storage manager in two major ways: it cannot handle throughput of all data events, or it runs out of storage resources.

At the data ingestion end of the Storage Manager, the Receiver may be scaled out to handle a large number of events. The Message Queue subcomponent acts as a buffer to minimize the impact of peaks in events. It also allows the Archiver to work independently of the event reception, and several Archiver processes may be used to write observations from the queue to storage. To further scale out the event handling, several queues partitioned by datastreams may be used with one or more Archivers each, possibly using different data store instances. This will allow the Storage manager to use additional resources efficiently when the scaling out.

Many observations that should be stored may cause the Storage Manager to run out of storage resources or meet limitations in the memory use of the underlying database (e.g. indices stored in memory). Horizontally partitioning, or sharding, the data by datastream will let the Storage manager scale out by adding new storage and computational resources. Both writing and querying data will have to take the sharding into account, e.g. by routing queries by sharding key. Of course, the underying database may also be able to scale out by horizontal partitioning, e.g. using MongoDB sharded clusters. The archiving functionality may also be used to move less frequently queried data out of the database.

Another scenario is that may affect the Storage manager is when the number of concurrent requests for historical data generated by the Cloud APIs are very high. As a result, the Storage manager component may not be able to respond to all queries in reasonable time. The partitioning strategy mentioned above will also address this problem when more resource nodes are added. Alternatively, the same data may be replicated on several nodes and queries can be load balanced between these.

During demonstrations and experimental third-party testing (hackathons) during the ALMANAC project, the most common query has been for the most recent data for each data stream, the "latest observation". Executing this very frequent query by ordering the observations by descending phenomenon time and taking the first observation will put unnecessary load on the data store. Instead, a materialized view or cache containing only the latest observations is kept updated by the Archiver subcomponent, and an OData collection function is supplied to make the queries easier for clients to construct.

Implementing redundant storage and caches of data in Storage manager is facilitated by the fact that the observations in data streams are append-only and non-updatable. All separate copies will be eventually consistent.

11.4.5 Resource Catalogue

Like the Storage manager, a large number of concurrently reporting resources may be handled by scaling out the subcomponents involved in registering updates to the resource cache. Several instances of the Registration service of the Resource Discovery subcomponent may receive events from the MQTT broker. The cache is organized per resource, so there is no need for two instances to access the same resource data for updates. It is also possible to partition the data between Resource manager instances if the stored data becomes too big to store at one node, or too big to query. Queries may then be posted to a set of Resource manager instances that contain all data and the results combined.

To manage a large number of concurrent requests for resources, load balancing can be used. This can be combined with horizontal partitioning if load balancing is done between sets of Resource manager instances that combined have the complete set of resources.

11.4.6 Semantic Framework

The Semantic Framework supplies ALMANAC clients a means to manage, access and query arbitrary (meta)-data structured according to the RDF graph-based data model[46]. It does so by providing a transparent proxy to off-the-shelf semantic databases (SPARQL endpoints) considerably extending the functional range of the standard SPARQL 1.1. Protocol[47] API.

Performance testing and scalability evaluation should therefore consider both layers separately. Being implemented in Java, the framework is accessible to common profiling tools for tracking consumption of computational resources (threads, memory, CPU), for example VisualVM[48]. The framework seeks to increase its performance by optimization techniques like caching, multi-threading and asynchronous processing. Incoming requests are internally turned into lightweight messages. Leveraging a message-based infrastructure the request processing could transparently be scaled across a series of nodes.

Also the underlying 3rd-party semantic databases might introduce a performance bottleneck. Depending on the requirements, systems with an explicit support for performance optimization and clustering should be preferred[49].

11.4.7 Cloud APIs CNET

The Cloud APIs are used by developers to build specialized applications. The needs and usage patters of the applications may vary greatly. Therefore, it may be useful to provide special performance and scaling techniques/capabilities at the Cloud API level. E.g., the Historical Data Cloud API may use a per-application data cache (or even a dedicated storage manager component for the data used by the application) with materialized views of data if the application has a need to intensively query specific data. This will let the application handle a large number of queries without putting load on the Semantic Representation Framework, Resource manager and Storage Manager.

The Cloud APIs should also apply throttling to control consumption of ALMANAC PI resources by applications.

[46] http://www.w3.org/TR/rdf11-concepts/
[47] http://www.w3.org/TR/sparql11-protocol/
[48] https://visualvm.java.net/
[49] http://www.w3.org/wiki/LargeTripleStores

12. References

(Ahlsen et al. 2011) Ahlsen, M., Asanin, S., Kool, P. Rosengren,P and Thestrup, J.(2011): "Service-Oriented Middleware Architecture for Mobile Personal Health Monitoring," MobiHealth 2011, Kos, Greece.

(Al-Akkad et al., 2009) Al-Akkad, A., Pramudianto, F., Jahn, M., Zimmermann, A. (2009): "Middleware for building pervasive systems". International Association for Development of the Information Society (IADIS): International Conference Applied Computing, Rome, pages 1–8.

(ALMANAC WP2, 2013) ALMANAC consortium work package 2 (2013), D2.1 Scenarios for Smart City Applications ALMANAC project deliverable.

(Bassi et al., 2013) A. Bassi, M. Bauer, M. Fiedler, T. Kramp, R. v. Kranenburg, S. Lange, and S. Meissner, Eds. (2013), Enabling Things to Talk - Designing IoT solutions with the IoT Architectural Reference Model. Springer.

(Bondi, 2000) Bondi, A. (2000). "Characteristics of scalability and their impact on performance". Proceedings of the second international workshop on Software and performance - WOSP '00.
p. 195. *doi*:*10.1145/350391.350432*. *ISBN 158113195X.*

(Eikerling et al., 2009) Eikerling, H., Gräfe, G., Röhr, F., Schneider, W. (2009), "Ambient Healthcare- Using the Hydra Embedded Middleware for Implementing an Ambient Disease Management System". In Proceedings of the Second International Conference on Health Informatics, Portugal, pages 82–89.

(Fowler, 2002) Fowler, M., "Patterns of Enterprise Application Architecture", ISBN-10: 0321127420

(Homer et al, 2014) Homer, A., Sharp, J., Brader, L., Narumoto, M., Swanson, T., "Cloud Design Patterns", ISBN-10: 1621140369

(IEEE1471, 2000) IEEE Standard 1471-2000 (2000), IEEE Recommended Practice for Architectural Description of Software-Intensive Systems.

(Jahn et al., 2010) Jahn, M., Jentsch, M., Prause, C. R., Pramudianto, F., Al-Akkad, A., Reiners, R. (2010). „The Energy Aware Smart Home". In 2010 5th International Conference on Future Information Technology, pages 1–8.

(Lehrig et al., 2015) Lehrig, Sebastian; Hendrik Eikerling; Steffen Becker (2015). "Scalability, Elasticity, and Efficiency in Cloud Computing: a Systematic Literature Review of Definitions and Metrics". Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA '15), Montreal, QC, Canada, May 4–7.

(ANSSI, 2014) Mécanismes cryptographiques - Règles et recommandations, Rev. 2.03, ANSSI , 02/2014.

(BSI, 2015) Kryptographische Verfahren: Empfehlungen und Schlüssellängen, TR-02102-1 v2015-1, BSI, 02/2015

(NIST, 2012) Recommendation for Key Management, Special Publication 800-57 Part 1 Rev. 3, NIST, 07/2012

(NIST, 2013) NIST Special Publication 800-63-2, Electronic Authentication Guideline, 08/2013.

13. Annex 1: Supporting infrastructure - LinkSmart

In our previous architecture discussions and illustrations we did not include components specific to LinkSmart. LinkSmart will be the underlying infrastructure for the ALMANAC platform and has multiple effects on the realization and design patterns we will apply. This chapter has the purpose of providing a brief introduction to LinkSmart and a glimpse of the services it provides that are beneficial for ALMANAC.

13.1 Introduction of LinkSmart

The LinkSmart middleware has been developed in the European project Hydra[50]. Hydra was a 4-year integrated project co-funded by the European Commission within the Sixth Framework Programme.

The LinkSmart middleware allows developers to incorporate heterogeneous physical devices into their applications by offering easy-to-use web service interfaces for controlling any type of physical device irrespective of its network technology such as Bluetooth, RF, ZigBee, RFID, Wi-Fi, etc. LinkSmart incorporates means for secure peer-to-peer (P2P) communication, device and service discovery, and respective developer tools.

The choice of a service-oriented architecture (SOA) in the Hydra project turned out to be a viable and successful approach as SOA applies to both the implementation of the middleware managers themselves and for the higher-level device interfaces in the form of software proxies, i.e., devices are also web services in LinkSmart. The SOA implementation works well across platforms as well as network boundaries. The system behind LinkSmart is implemented on two main IDEs, Eclipse and .NET and also provides P2P device interoperability across networks. As the LinkSmart middleware is based on SOA, to which the underlying communication layer is transparent, it includes support for distributed as well as centralized architectures, security and trust, reflective properties and model-driven development of applications.

Several successful applications have been developed to evaluate the LinkSmart middleware in different domains of Ubiquitous Computing including eHealth (Ahlsen et al. 2011) and (Energy-aware) Smart Homes (Al-Akkad et al., 2009), (Eikerling et.al, 2009), (Jahn et al., 2010), (Reiners et al., 2009).

13.2 Architecture of LinkSmart

The software architecture described here is an abstract representation of the software part of the LinkSmart middleware. The architecture is a partitioning scheme, describing components and their interaction with each other. Figure 35 shows the concrete deployment of LinkSmart managers on the specific network components. Each Native Device is connected through the Network Manager. A Gateway further hosts a couple of proxies for closed platform devices (e.g. commercial off the shelf Bluetooth and ZigBee devices).

The gateway must run a Network manager that registers all services belonging to the devices in the LinkSmart network. Moreover, on the gateway a Device Application Catalogue (DAC) can keep track of devices available in the LinkSmart network.

The LinkSmart Application runs on a PC as a dedicated application (i.e. a centralized architecture). The application can access specific services through the network manager if it knows in advance which services it wants to use.

Alternatively, the application can browse the devices on the DAC first, based on specific criteria and access their services.

[50] http://www.LinkSmartmiddleware.eu

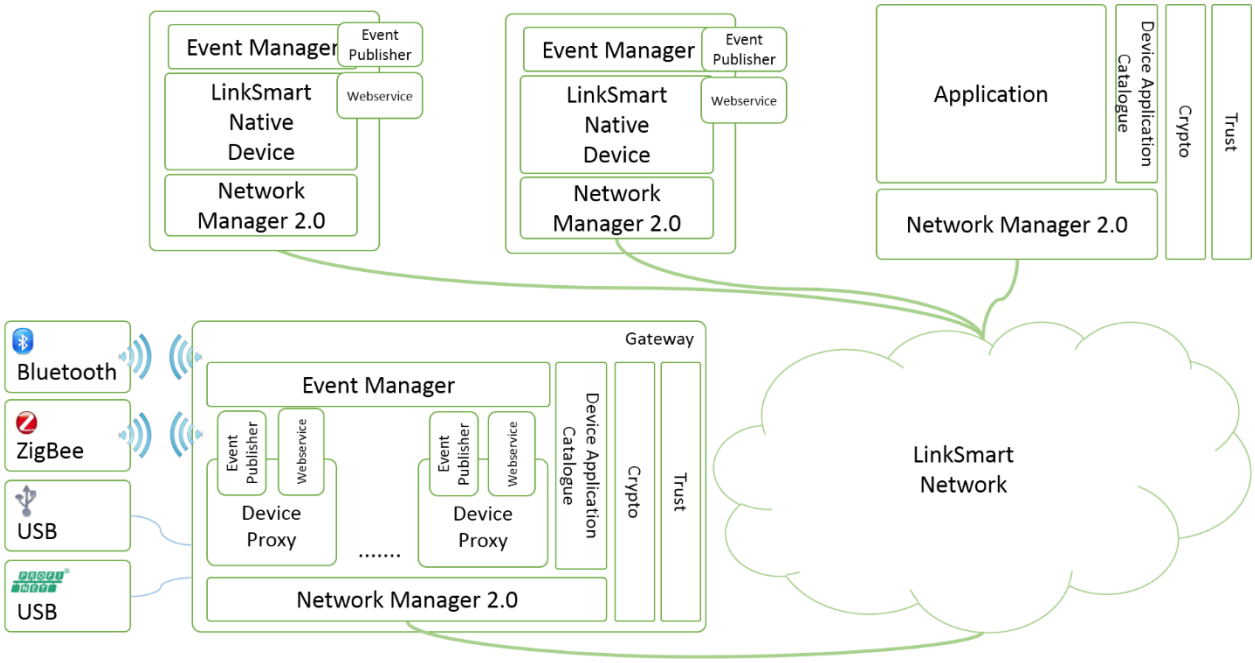


Figure 35: LinkSmart Example Concrete Deployment

The EventManager handles the publishing and the subscribing of events. Applications can subscribe to an EventManager for a topic they are interested in or publish events, for example events containing sensor measurements. An event consists of a topic and key-value pairs.

The TrustManager and the CryptoManager are optional components. Their purpose is to increase security and robustness of the system.

In Figure 36, we see an example of a LinkSmart device network. LinkSmart distinguishes between powerful devices which are capable of running the LinkSmart middleware natively and smaller devices that are too constrained or closed to run the middleware. For the latter, proxies are used and once proxies are in place, all communication is based on the IP protocol.

The figure below illustrates these two device types. On the right, there are devices which can host the LinkSmart middleware and which are able to establish communication with services on the platform. On the left, devices are depicted which cannot operate the LinkSmart middleware, either because they have resource constraints or proprietary interfaces. For these devices proxies are created on a Business Area Network or a Personal Area Network node (in this case a mobile phone). For a service the communication with a proxy is not any different from communication with a LinkSmart enabled device.

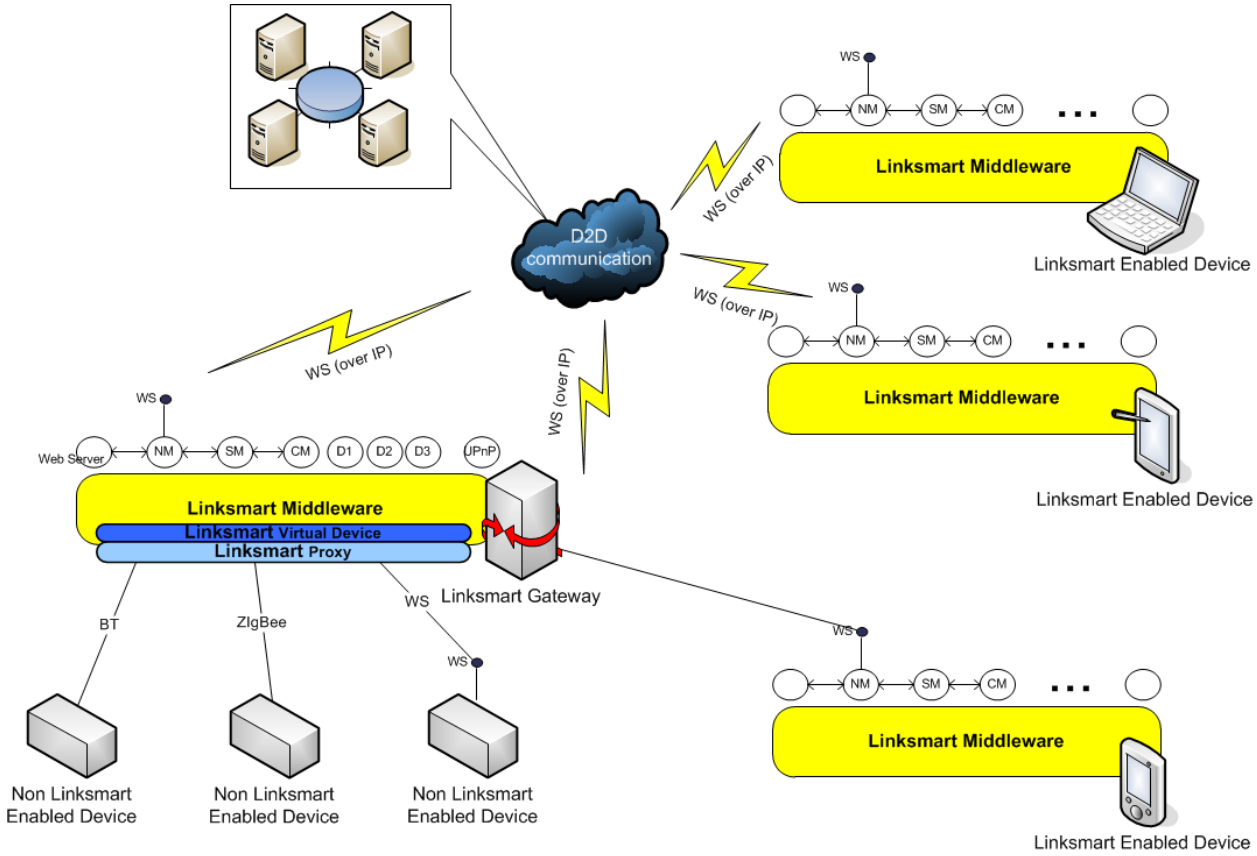


Figure 36: LinkSmart Example Device Network

14. Annex 2: IOT-ARM and ALMANAC

The ALMANAC Smart City Platform (SCP) is based on an Internet of Things (IoT) architecture. For this reason the architecture development in ALMANAC builds on state of the art IoT system architectures and reference architectures. For the latter we will particularly look at the IoT Architectural Reference Model (Bassi et al., 2013).

14.1 Reference architectures

Reference architectures have a long history in IT and telecom systems design. Their main purpose is to act as common guides to the generation of architecture in specific domains.

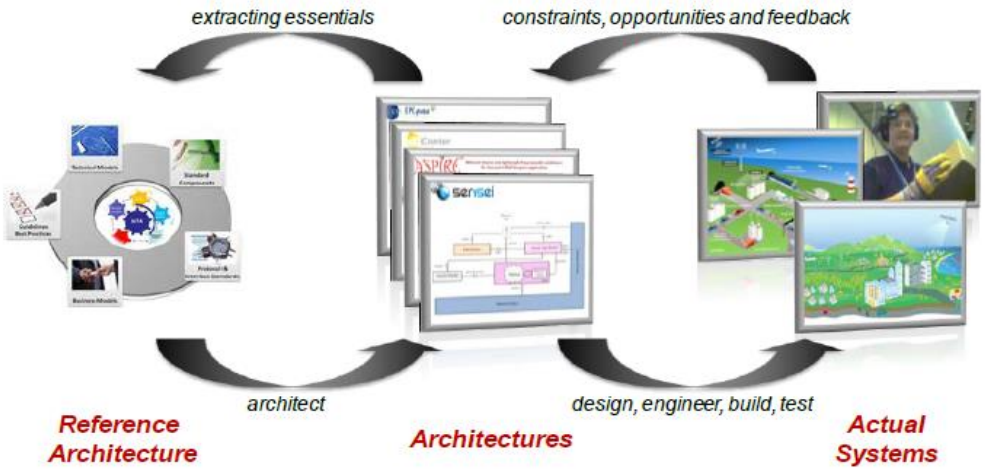


Figure 37: A reference architecture provides the instruments and guidelines for domain specific architectures from which specific system designs are derived.

A Reference architecture serves the following roles and usages,

- A common vocabulary reference for an ICT design domain.
- A structured collection of concepts, models and guidelines for description of domain specific architectures.
- A collector and generalization of good (possibly best) practice in the domain.
- An instrument for comparison, explanation and benchmarking of different designs in the same domain.

There a number of difficulties that face developers and users these frameworks. They tend to become very complex and cumbersome to apply and also to comply with. Some architectures also appear too generic for the intended domain.

14.2 Elements of the IoT ARM

The IoT Architectural Reference Model (IoT ARM) provides a collection of generic architectural concepts and constructs considered applicable to IoT system architectures. The IoT ARM does not say how to build IoT systems, it is a tool box of concepts, models and recommendations for the domain of IoT systems and their architectures. The IoT-A reference model can be used as a baseline to derive new IoT architectures but also as a reference to explain and compare different existing IoT system designs.

The reference model framework was developed by the IoT-A[51] project and addresses IoT in terms of an overall IoT Architectural Reference Model including the subsets:

[51] http://www.iot-a.eu

- Business and stakeholder scenarios
- An IoT Reference Model
- The IoT Reference Architecture

The first two parts define the objectives, context and concepts of the overall architectural framework4.

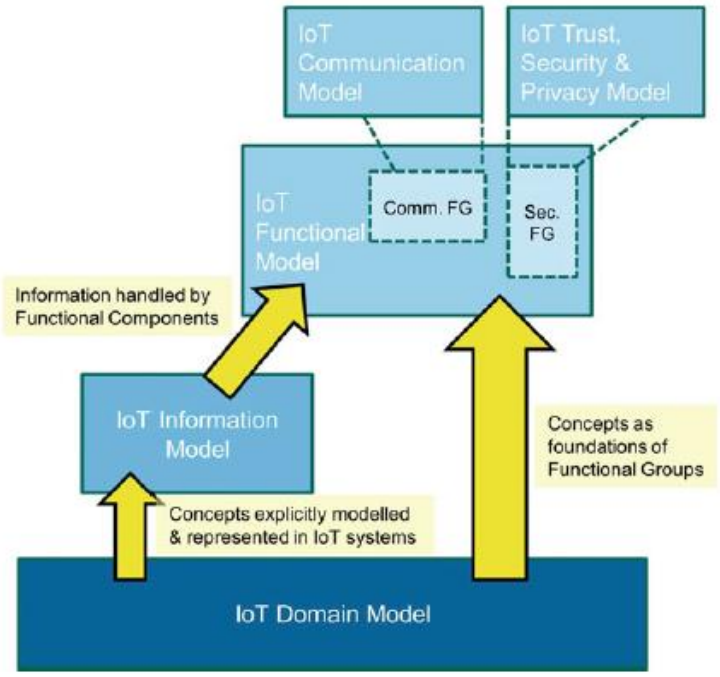


Figure 38: Sub-models of the IoT Reference Model (From (Bassi et al., 2013))

The Reference Architecture is meant as the reference and architectural guideline for building (instantiating) compliant domain specific IoT architectures from which systems can be designed and implemented.

14.2.1 IoT-A domain model

An important part of the Reference Model is the definition of the central IoT domain oriented concepts. The IoT Domain Model names and relates these central concepts in the IoT Reference Model.

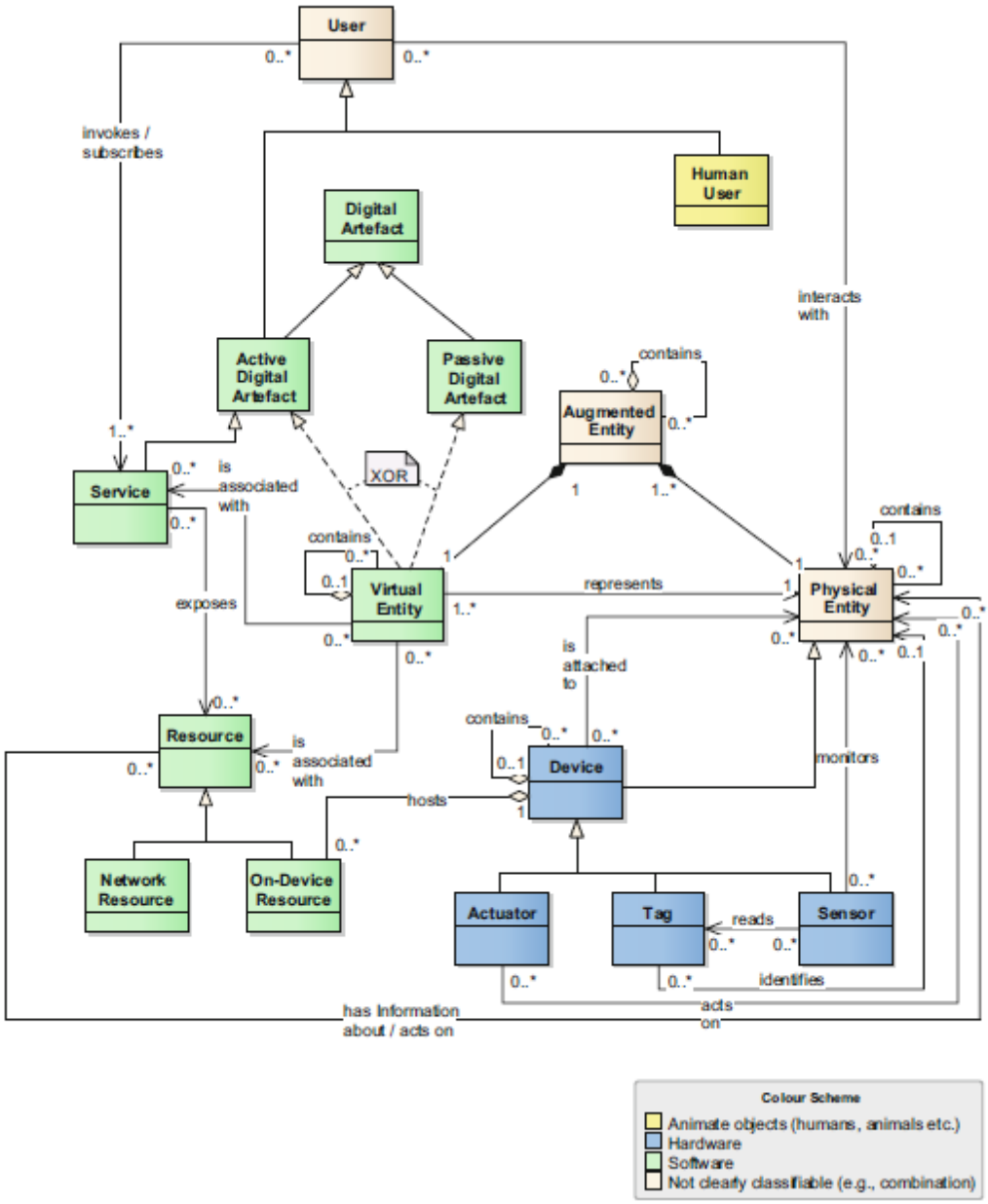


Figure 39: UML version of the IoT-A Domain Model

In the IoT-A domain model, real world physical entities have corresponding digital representations in virtual entities. The physical entities can be subject to monitoring or actuation by means of various IoT devices. The devices can be attached directly to the physical entities, or the physical entities are in the operating range of the devices (e.g., through a wireless net). The software part of the device that provides information on the entity or enables actuation of the device is modelled as a resource. The functionality provided by the resource is exposed by means if services. Services provide well-defined and standardised interfaces, hiding the complexity of accessing variety of heterogeneous resources. The interaction with a physical entity can be accomplished via one or more services associated with the corresponding virtual entity.

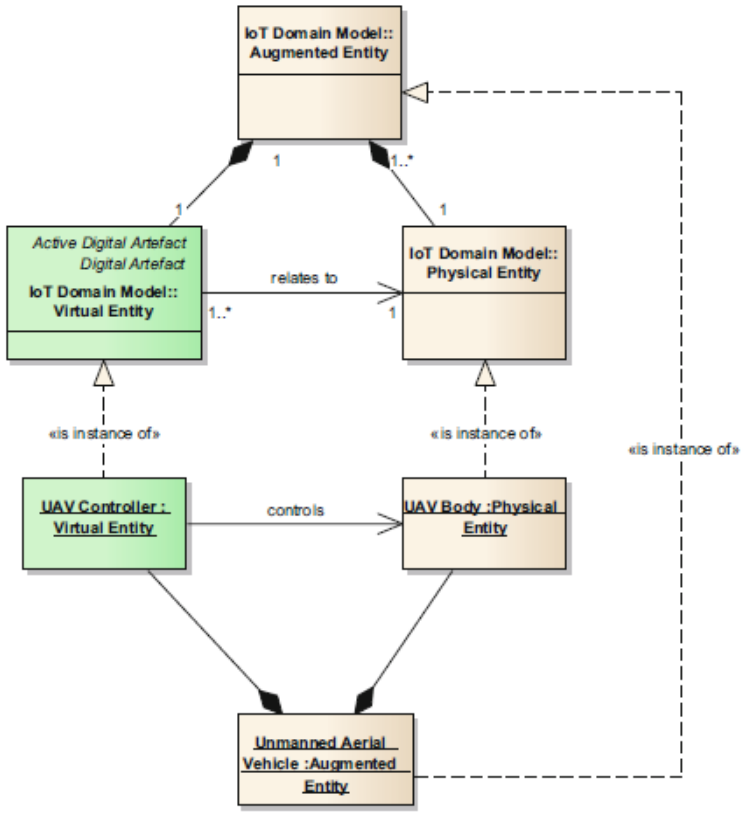
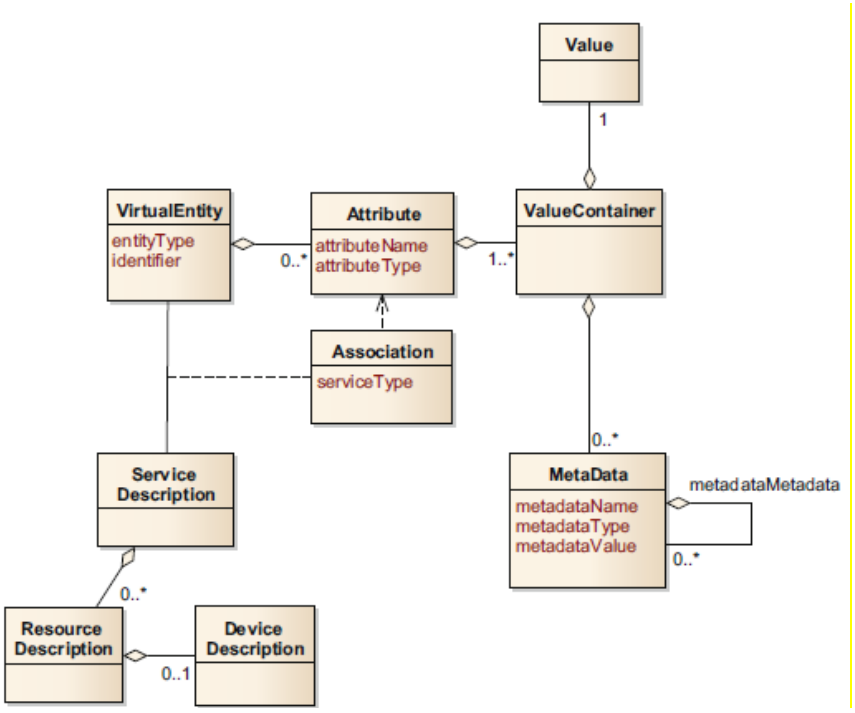


Figure 40: Example modelling using the Domain Model.

14.2.2 Information Model

The structuring of the Virtual Entities from the Domain Model is detailed and modelled in the IoT Information Model. The Information model is intended to meta model those concepts from the Domain Model that should be explicitly represented and managed in an IoT system.



The entityType of a Virtual Entity can possibly refer to an external ontology that can define the set of attributes, and similarly for the attribute and service types.

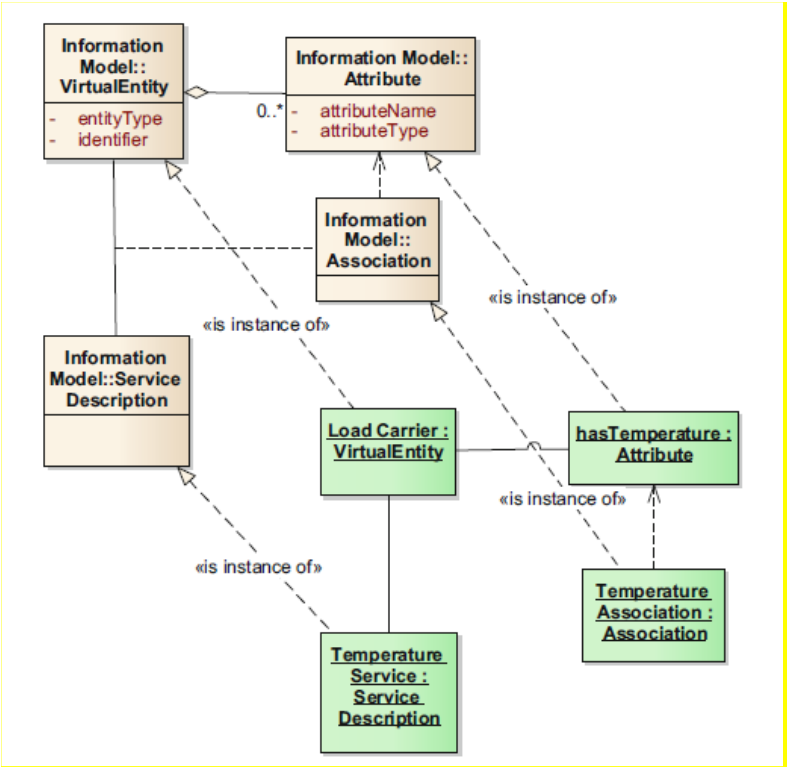


Figure 41: Example instantiation of the Information Model

Other more implementation oriented information models include:

- Entity model: The Entity Model specifies which attributes and features of real word objects are represented by the virtual counterpart. Ontology based on ER/OWL
- Resource model: The Resource Model contains the information that is essential to identify Resources by a unique identifier and to classify Resources by their type, like sensor, actuator, processor or tag. Ontology based on ER/OWL and standard ID system, e.g., EPC/GS1
- Service description model: Services provide access to Resources and are used to access information or to control Physical Entities. Service description framework, e.g., USDL
- Event processing model**:** Describes the objects, rules and agents used to receive, process and dispatch events in an IoT system.

14.2.3 Functional model

The components of the IoT ARM are organized into groups in the Functional Model. This model is then the basis for defining the Functional View in the reference architecture.

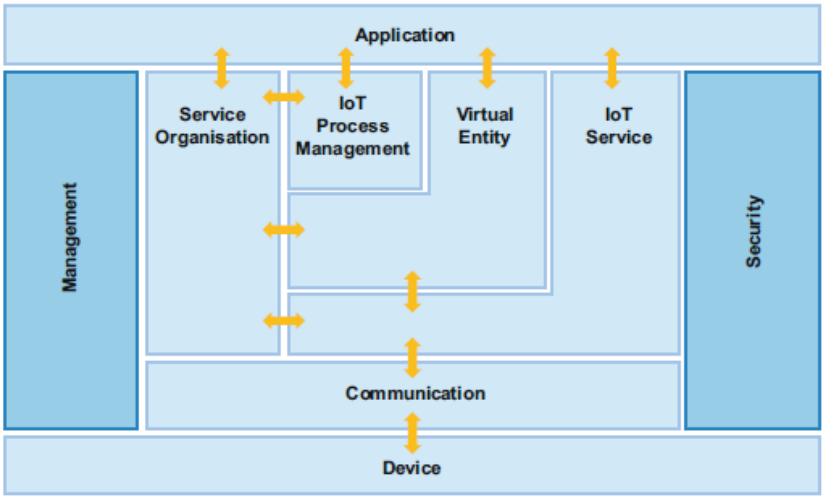


Figure 42: Functional Model

The Application and Device layers are outside the scope of the reference model.

14.2.4 Views in the Reference Architecture

The *reference architecture* defines the Views, View Points relevant for IoT systems architecture design. Following the conventional approach the ARM describes three views, each one with a number of View Points focusing specific aspects of a view,

- An Functional View
- An Information View
- A Deployment and Operation View

The Functional View provides a layered structure of various function groups (e.g., "IoT Service"), with specific functional components (s.a. "IoT Service resolution" and IoT Service"). The "Application" and "Device" function groups are considered out of scope in this reference model.

(Layered) Functional view

Functional Components organized in Function Groups describe the Functional View in the ARM. This is the common two dimensional (almost) layered model of software component abstractions.

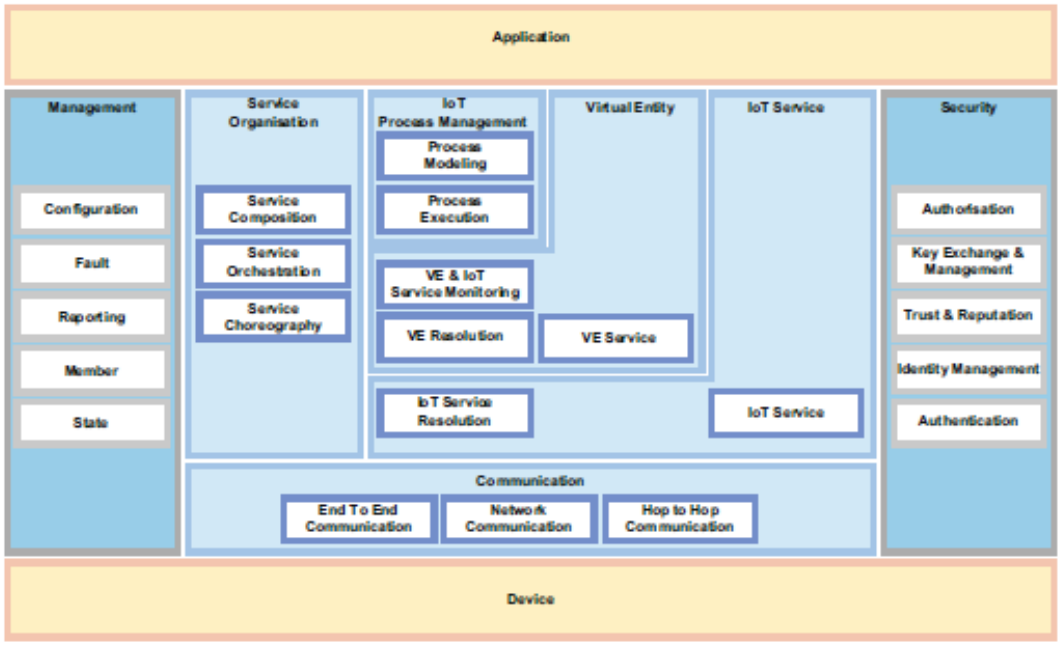


Figure 43: Function Groups

The IoT Service FG contains IoT services as well as functionalities for discovery, look-up, and name resolution of IoT Services. It consists of two Functional Components:

- IoT Service
- IoT Service Resolution

An IoT Service exposes one Resource to make it accessible to other parts of the IoT system. Typically, IoT Services can be used to get information provided by a resource retrieved from a sensor device or from a storage resource connected through a network. An IoT Service can also be used to deliver information to a resource in order to control actuator devices or to configure a resource. Resources can be configurable in non-functional aspects, such as dependability security (e.g. access control), resilience (e.g. availability) and performance (e.g. scalability, timeliness).

……..

The main functions of the IoT Service FC are to (1) return information provided by a resource in a synchronous way, (2) accept information sent to a resource in order to store the information or to configure the resource or to control an actuator device and (3) subscribe to information, i.e. return information provided by a resource in an asynchronous way (Bassi et al., 2013).

The figure below shows the corresponding Function Groups in the ALMANAC Functional View.

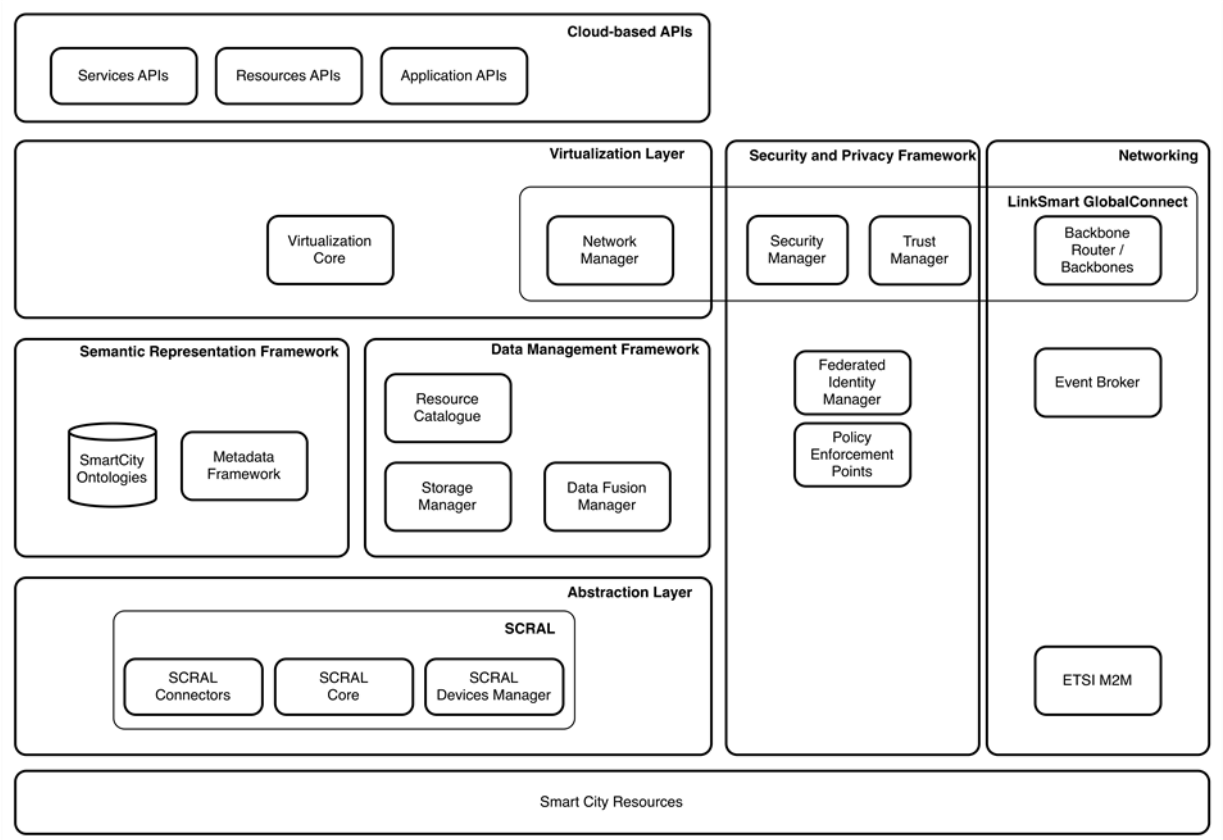


Figure 44: Function Groups in the ALMANAC architecture

Information view

Based on the IoT Information Model, this view gives more details about how the relevant information is to be represented in an IoT system. The concrete representations are not part of this view.

Deployment and operation view

The following viewpoints are elaborated:

- The IoT Domain Model diagram is used as a guideline to describe the specific application domain; to this extent UML diagrams can be used to further detail the interaction among the many elements composing the target application.
- The Functional Model is used as a reference to the system definition; in particular it defines Functional Groups such as IoT Services and Connectivity groups which are fundamental for a correct definition of the system.
- Network connectivity diagrams can be used to plan the connectivity topology to enable the desired networking capability of the target application; at the deployment level, the connectivity diagram will be used to define the hierarchies and the type of the sub-networks composing the complete system network.
- Device Descriptions relating device capabilities to the service and resource requirements of the target system.

14.2.5 Perspectives (architectural qualities) in the IoT ARM

Perspectives represent non-functional requirements on a systems design, which are orthogonal to the Views of the reference architecture.

The IoT ARM identifies the following perspectives as among the most important for IoT-systems:

- Evolution and Interoperability
- Availability and Resilience
- Trust, Security and Privacy and
- Performance and Scalability

15. Annex 3: State of the art library

In this section a state of the art library on the existing IoT devices, systems, services, research applications, commercial applications, standards and Fi-WARE components in the context of Smart Cities in presented.

15.1 Devices

Examples: a waste sensor, a water meter, a mobile phone, ...

Name	Reference	Description	Relevance
Smart Bin	Producer website[52]	A smart bin sensor: it can monitor the fill level and report via GSM.	It could be considered for testing e.g. in a proof-of-concept (Cost and availability to be verified).
Postscapes IoT	Producer website[53]	List of prominent IoT hardware	Helps with technology scouting about IoT hardware blocks
Smart irrigation controllers	Producer website[54]	List of smart irrigation controllers	Automatic control of water use. Could be influenced by current water situation at city level (ALMANAC water use case)
Enevo Smart Waste Sensor	Producer website[55]	Waste container monitoring service	ALMANAC could target this example for the waste management use case
Xtreme RFID	Producer website[56]	Asset tracking for Municipal Solid Waste[57] [58]	Consider compatibility for waste management use case
Water and leak detection sensors	Website[59]	Smarthome: Home automation super store Many off-the-shelf devices can be found in this website	Commercial solutions for water leak and detection, that could be used as an inspiration for the water use case.
(Underground automated) vacuum waste collection systems	Website[60]	Waste deposited in inlets then sucked away to compacting facilities through underground pipes using vacuum conveying technology	AVAC waste collection solution (not really a device, but a complex future solution already implemented)

[52] http://smartbin.com/how-it-works/smartbin-sensors.html
[53] http://postscapes.com/internet-of-things-hardware
[54] http://postscapes.com/smart-irrigation-controllers
[55] http://www.enevo.com/
[56] http://www.xtremerfid.com/applications/municipal-solid-waste
[57] http://www.xtremerfid.com/news/xtreme-rfid-helps-cincinnati-grand-rapids-enhance-recycling-and-waste-collection-operations
[58] http://www.rfidarena.com/2012/9/27/a-push-towards-recycling-with-rfid.aspx
[59] http://www.smarthome.com/_/Sensors/Water_Leak_Detection/_/L/1SD/nav.aspx
[60] http://www.thecitiesoftomorrow.com/solutions/waste/solutions/vacuum-waste-collection-systems

Urbiotica sensors	Producer website[61]	Waste Container Fill Level Sensor, Air Quality Sensor and Wide range of environmental sensors.	A very interesting presentation on the combined used of this sensors for the waste management problem can be seen here[62].

[61] http://www.urbiotica.com/products-and-solutions/
[62] http://www.metropolis.org/sites/default/files/news/urbiotica-2.pdf

Insteon sensors	Producer website[63]	A wide variety of sensors for the following applications: Remote control lighting, Control and status on smartphones and tablets, Remote control heating and air conditioning (HVAC), Scene lighting, Timers, Occupancy sensing, Leak sensing, Humidity sensing and control, Garage door sensing and control, Email and text (SMS) alerts, Access control (e.g. door locks), Audio-video control, Appliance management, Irrigation control, Energy measurement, Energy savings.	Useful for the waste and water use cases, specially: Occupancy sensing, Leak sensing, Humidity sensing and control.
Texas Instruments Sensors	Website[64] Water meter[65]	A variety of sensors, used in Texas smartgrid projects, f.ex. CenterPointEnergy Specific design sheet for water metering	Interesting web site with connections to especially SmartEnergy, but meters seem to have cross-domain architectural features (MeterUIDs etc.). A number of meters documented in different domains
Contiki OS	Website[66]	"The Open Source OS for the Internet of Things", with low footprint, and including advanced routing	OS to consider for sensors, and for interaction with LinkSmart
RIOT OS	Web site[67]	"The friendly Operating System for the Internet of Things", can run on ARM microcontroler boards such as Arduino Due	OS to consider for sensors, and for interaction with LinkSmart
Tiny OS	Website[68]	OS for low-power wireless devices, such as those used in sensor networks	OS to consider for sensors, and for interaction with LinkSmart

[63] http://www.insteon.com/index.html
[64] http://www.ti.com/lsds/ti/apps/smartgrid/metering/overview.page
[65] http://www.ti.com/solution/water_meter
[66] http://www.contiki-os.org
[67] http://www.riot-os.org
[68] http://www.tinyos.net

15.2 Systems

Examples: a waste management system, a water monitoring infrastructure, etc.

Name	Reference	Description	Relevance
IBM Intelligent Water	Website Leafet (PDF)[69]	Water system monitoring system. "The solution uses advanced data management, visualization, correlation and collaboration technologies to transform the vast amounts of disparate data received from various devices (including metering systems), assets, systems and stakeholders into actionable information that can guide executive and operational decisions."	Example/Competitor to water management application. Could also provide some ideas for the water management application.
EmNet (Company)	Website[70] Article[71]	Sewer system monitoring, analysis and control optimization. "EmNet's Real Time Intelligence and Optimization technology helps utilities maximize existing and planned resources to minimize overflows and save money. EmNet combines the use of traditional hydraulic modelling and novel real time information technology to deliver actionable insight."	Example/Competitor to water management application. Could also provide some ideas for the water management application.
Sewage Grid: Drifting Sensors that Monitor the Wastewater Collection System (Scientifc paper)	Sewage grid document[72]	They use wireless floating sensors to detect leakage in waste water systems.	Innovative solution to water system monitoring.

[69] http://public.dhe.ibm.com/common/ssi/ecm/en/gws03010usen/GWS03010USEN.PDF
[70] http://www.emnet.net/index.php/whatwedo
[71] http://txchnologist.com/post/50336951628/going-against-the-flow-green-tech-sensors-and
[72] https://confluence.fit.fraunhofer.de/confluence/login.action%3bjsessionid=CD08ED1A4382C4F9E19C41EF401F233A?os_destination=%2Fnotpermitted.action%3Fversion%3D1%26modificationDate%3D1391184940244%26api%3Dv2

SeWatch - wastewater and sewage wireless monitoring system Company: Telematics Wireless[73]	Website[74]	Wireless monitoring system for sewers reporting discharge or overflow. It includes • Water-level sensors for sewer system manholes. • Remote Terminal Units (RTU) for data capture with built-in wireless communications. • Primary battery or solar-powered wireless relays/nodes, reader/gateway unit for interfacing to a Network. • A monitoring and Controlling Management application running on PC or server, which alerts on screen or via SMS about manhole overflow and spillovers.	Inspiration for the water management application. System example from the industry. Could be helpful to see which kinds of hard- and software they use.
Postscapes IoT City	Website[75]	List of projects using IoT for city applications	Helps with technology scouting about IoT for cities
Enevo ONe Collect	Producer website[76]	Waste monitoring solution	A competing solution to the waste management use case
Hitachi's water infrastructure solution	Producer website[77]	A water infrastructure solution: water treatment system, an information control system and an energy saving system.	The water distribution control system could provide some ideas for a new approach on water management: the electric power load related to distribution is reduced, and pressure distribution is corrected for each zone
Telefonica SmartCity overview	Producer website[78]	Smart cities platform for a better world based on m2m technology	Not much information about the platform itself, but the way they present the use cases is very interesting (see the picture "Discover all of them in our Smart City" more specifically click on the sections about watering management and waste management)

[73] http://www.telematics-wireless.com/index.php?page_id=1
[74] http://www.telematics-wireless.com/index.php?page_id=138
[75] http://postscapes.com/connected-city
[76] http://www.enevo.com/one-collect/
[77] http://www.hitachi.com/products/smartcity/smart-infrastructure/water/solution.html#plink05
[78] https://m2m.telefonica.com/discover-m2m/smart-cities

CenterPointEnergy	Website[79] Security design[80]	The publicly owned non-profit electric transmission infrastructure owner in Texas defining a market for smart metering for competitive business	inspiration; mostly business

15.3 Services

Examples: a Cloud-provisioning service, on-line weather forecast, ...

Name	Reference	Description	Relevance
Xively (formerly Pachube.com)	Producer website[81]	Cloud service to upload and process IoT data	Example of data aggregation service for inspiration. Consider partial compatibility
IFTTT	Producer website[82]	If This Then That: Channels, triggers, actions... E.g. with SmartThings[83]	End-user empowerment
Node-RED	Producer website[84]	A visual tool for wiring the Internet of Things. Built on top of Node.js. Users can create data flows from IoT Devices to e.g. Twitter in a browser-based editor and deploy on small devices such as Raspberry Pi. ´	End-user empowerment
Google App Engine	Producer website[85]	Google's Platform-as-a-Service. Provides storage, load balancing and other services.	Cloud provisioning
Windows Azure	Producer website[86]	Microsofts Platform-as-a-service. Provides storage, load balancing and other services.	Cloud provisioning
Datahub	datahub.io	Publish and consume open data (based on the open source ckan software platform)	Publish data
OpenDataSoft datastore	public.opendatasoft.com	Cloud platform to publish and consume open data (using OpenDataSoft platform)	Publish data

[79] http://www.centerpointenergy.com/cehe/about/
[80] http://www.centerpointenergy.com/staticfiles/CNP/Common/SiteAssets/doc/123062_SmartMeterDataSecurity.pdf
[81] https://xively.com/showcase/
[82] https://ifttt.com/wtf
[83] http://smartthings.com/news/ifttt/
[84] http://nodered.org
[85] https://cloud.google.com/products/app-engine/
[86] http://www.windowsazure.com/en-us/

freeboard.io	freeboard.io	Visualisation of real-time (~1 second delay) IoT data with widgets, open source, works well with dweet.io as data input and with PubNub.com's real-time network	Visualisation, user interfaces
dweet.io	dweet.io	Publish IoT data with a Twitter-like approach. Publish/subscribe	Publish/subscribe
OpenRemote	www.openremote.com	Open source middle-ware	Inspiration for middle-ware
PubNub	www.pubnub.com	Commercial real-time publish/subscribe service	Publish/subscribe
Busit	www.busit.com	Small French competitor of IFTTT. Interesting user interface	End-user empowerment

15.4 Research Applications

Examples: experiences from on-going projects, etc.

Name	Reference	Description	Relevance
EU projects on IOT	EU Project Website[87]	CORDIS search	Live list of projects regarding IoT funded by the European Commission
SmartSantander	EU Project Website[88]	EU project: world city-scale experimental research facility in support of typical applications and services for a smart city	Could help testing some ALMANAC concepts
URB-Grade	EU Project Website[89]	EU project: helps policy-makers to make better decisions in terms of cost and energy efficiency and to increase citizen awareness of how to save energy and derive a greater proportion of energy from renewable sources. See also the related projects http://urb-grade.eu/related-projects/	Citizen awareness, decision making

[87] http://cordis.europa.eu/projects/index.cfm?fuseaction=app.search&TXT=iot
[88] http://www.smartsantander.eu/
[89] http://urb-grade.eu/

IoT.est	EU Project Website[90]	EU project: Internet of Things Environment for Service Creation and Testing	Abstraction, testing, semantic annotations
Mobosens	Website[91]	US research project, which provides citizens with a platform for collecting and sharing environmental data, from stream quality to drinking water safety	End-user involvement. Inspiration for the use-case about water
SmartOpenData	smartopendata.eu	EU project: Linked Open Data infrastructure (including software tools and data) fed by public and freely available data resources	Geospatial Data http://www.w3.org/2014/03/lgd/report
OpenIoT	openiot.eu	EU project: Open Source cloud solution for the Internet of Things	
UrbanWater	urbanwater-ict.eu	EU FP7 project: Almería, Spain, is installing smart meters in order to achieve greater efficiencies in water management http://cordis.europa.eu/news/rcn/122370_en.html	Water scenarios

15.5 Commercial Applications

Examples: experiences from solutions that are already on the market or close to be launched

Name	Reference	Description	Relevance
Smart Dutch Rubbish Bins - M2M-enabled	News press from techweekeurope.co.uk[92]	A large-scale pilot (6000 Smart Bins) in Groningen (NL). Citizen can unlock bins using an RFID badge + sensors are used to monitor the fill level.	It could be used as inspiration for the Smart Waste use case. Smart Bin could be considered for testing (if available on the market)
HydroPoint WeatherTrak	www.hydropoint.com/products/outdoor-solutions/	Measurement, irrigation control based on weather data, equipment for sensors, wired link & wireless comm., central adm server	Competing system
Sensus Intelligent Water management soluions	sensus.com/web/usca/products/water	Metering, Water system-level network solutions,	inspirational

[90] http://ict-iotest.eu/iotest/
[91] http://nanobionics.mntl.illinois.edu/mobosens/
[92] http://www.techweekeurope.co.uk/news/m2m-bins-tell-council-when-they-are-full-92401

Watersave SmartMeter	www.watersave.com.au/smartmeter/commercial	Australian Smart Water Solution	Inspirational – system descriptions. Management, system structure, Metering citizen interface
IntelliH2O smart water meter	www.intelli-h2o.com/products/technology		American – M2M wireless connection and Management solutions to arrive
PCScale Tower Software	Website[93]	Software to map, route & dispatch trucks, track inventory, scale management & invoice for residential and commercial waste companies	inspirational
AccuTrax Mobile	http://www.accu-trax.com/images/accu-trax_pdf_low_res.pdf	Hardware and software solution for Outgoing business, with software to map, route, document, take pictures, maintain customer / citizen adm, etc.	inspirational
RapportFraStedet	rapportfrastedet.dk	A simple cross-domain, open-source issue reporting app developed in Denmark and shared by a larger number of Danish Municipalities. Available on web and in app stores	inspirational
WaspMote SmartWater	http://www.libelium.com/smart-water-sensors-monitor-water-quality-leakages-wastes-in-rivers-lakes-sea/	a Smart Water wireless sensor platform to simplify remote water quality monitoring	inspiration
CenterPointEnergy.com	www.centerpointenergy.com/cehe/about/, security design at http://www.centerpointenergy.com/staticfiles/CNP/Common/SiteAssets/doc/123062_%20SmartMeterDataSecurity.pdf	The publicly owned non-profit electric transmission infrastructure owner in Texas defining a market for smart metering for competitive business	inspiration

[93] http://www.capterra.com/waste-management-software/spotlight/81310/PC%20Scale%20Tower%20Software/PC%20Scale

renoweb	www.renoweb.dk	Commercial web based IT collection system in use in several regions in Denmark. It comprises two products – the *renoweb* server system by Grontmij (http://www.renoweb.dk/, with online hands-on demo, in danish), and an onboard iPad client system for the vehicle (connected by GPRS and SIM card technology on the iPad, and for some vehicles combined with an antenna). The system includes, collections, route handling and ordering of extraordinary collection issues. The RenoWeb is the dominating system in Denmark, in the capitol region used by 80% of the municipalities. Several integrations exist, to municipal systems (who own the citizen data), waste weights in trucks and "bridge weights" in dumpsters, weighing the total collection truckBin recognition is based on either fill level sensors or (mostly) passive tags recognisable from the bridge weights	competing system, inspiration
GTC (Garbage Transport Computer) WEBLog (upload of weight data to web page)	www.tarp.dk/en	*Garbage Transport Computer* (*GTC*) by the Danish company Poul Tarp A/S, at http://www.tarp.dk/Files/waste-collection-191.pdf. The GTC has a specially designed robust onboard system (TOC II) running windows 8 connected to the weighing technology. The system includes the weights for trucks and dumpsters (Bridge weights). Used in the danish market.	Weighing technology (competition / inspiration)
AllSeen Alliance (by the Linux Foundation)	allseenalliance.org/	AllJoyn open source platform (C, C++, many OS and chipsets) for peer discovery and peer to peer connections over a variety of transports. Members include Qualcomm, Microsoft, Cisco, LG, HTC, D-Link, TP-Link, etc.	Competitor to some aspects of LinkSmart. To consider for interoperability.
Open Interconnect Consortium	www.openinterconnect.org	Specifications, standards, open source code base. Members include: Intel, Samsung, Broadcom, Atmel, Dell	Consider for interoperability and security
Thread	www.threadgroup.org	Consortium: ARM, Samsung, Nest (Google)...	IoT connectivity

15.6 Standards

Examples: sections of standards which cover some feature relevant for ALMANAC

Name	Reference	Description	Relevance
OGC Sensor Web Enablement (framework)	OGC Sensor web enablement Group[94]	interoperability interfaces and metadata encodings that enable real time integration of heterogeneous sensor webs into the information infrastructure	Candidate framework for the various system interfaces in the ALMANAC architecture
W3C Semantic Sensor Network	W3C Incubator Group Report[95]	Ontology to describe sensors and sensor networks for use in sensor network and sensor web applications	Candidate standard for the ontology needs
ETSI M2M	ETSI M2M website[96]	ETSI M2M is the European standard for M2M (Machine to Machine) applications	Used in the ALMANAC architecture and developments
OneM2M	One M2M standards website[97]	OneM2M will be the worldwide standard for M2M (Machine to Machine) applications. It is going to include also ETSI. M2M	When available considered in the ALMANAC architecture and developments
CDMI	ISO standards catalogue[98]	Cloud Data Management Interface (CDMI), ISO/IEC 17826:2012, specifies the interface to access cloud storage and to manage the data stored therein.	WP6 cloud based storage management.
OMA M2M enablers	openmobilealliance.org/about-oma/work-program/m2m-enablers/	Overview of OMAs foreseen standardization efforts for LWM2M ("Light-Weight M2M	Device mgmt, SCRAL
EEBus	White paper[99]	Technology for comprehensive networking of devices and load management between power suppliers, grid operators and end users. Abstracts many existing protocols in a consistent IPv6/XML format. EEBus approach are also targeted at eHealth, Ambient Assisted Living and Security.	Platform partially competing with ALMANAC. Could be considered during interoperability testing, to replace some ALMANAC / LinkSmart parts.

[94] http://www.opengeospatial.org/projects/groups/sensorwebdwg
[95] http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/
[96] http://www.etsi.org/technologies-clusters/technologies/m2m
[97] http://www.onem2m.org/
[98] http://www.iso.org/iso/catalogue_detail.htm?csnumber=60617
[99] http://www.eebus.org/fileadmin/Mediapool/2011_06_EEBus_Whitepaper_en.pdf

HYPER/CAT	1248.io/casestudies/hypercat.pdf wiki.1248.io/doku.php?id=hypercat	HyperCat is designed for exposing information about IoT assets over the web. It allows a server to provide a set of resources to a client, each with a set of semantic annotations. Open, lightweight JSON-based hypermedia catalogue format for exposing collections of URIs, each with any number of RDF-like triple statements about it.	To be considered for exposing IoT semantic properties on the Web.
OASIS OData 4.0	OData Version 4.0 Part 1: Protocol[100] OData JSON Format Version 4.0[101]	Standards for an Open, Programmable Web https://www.oasis-open.org/news/pr/oasis-approves-odata-4-0-standards-for-an-open-programmable-web The Open Data Protocol (OData) enables the creation of REST-based data services, which allow resources, identified using Uniform Resource Locators (URLs) and defined in an Entity Data Model (EDM), to be published and edited by Web clients using simple HTTP messages. OData JSON Format extends the core specification by defining representations for OData requests and responses using a JSON format.	To be considered for data interoperability over an HTTP / REST interface
MQTT	mqtt.org	Machine-to-machine (M2M)/"Internet of Things" connectivity protocol, lightweight publish/subscribe messaging transport. Used e.g. by Xively. Clients available in several languages http://eclipse.org/paho/ . Under OASIS standardisation.	Example of popular M2M protocol to be compatible with
XMPP (Extensible Messaging and Presence Protocol) and its extensions	Overview[102] Publish/subscribe extension[103]	Decentralised, real-time communication, routing of XML messages	Consider adding an XMPP gateway to be compatible with third-party services based on it
PubSubHubbub protocol	code.google.com/p/pubsubhubbub/	Server-to-server webhook-based publish/subscribe protocol for any Web accessible resources. Short discussion on XMPP vs. PubSubHubbub http://documentation.superfeedr.com/subscribers.html#whatapitochoose	To be considered for instance to have a distributed network of Storage Managers / Virtualization Layers
JSON-LD (JavaScript Object Notation for Linked Data)	www.w3.org/TR/json-ld/ http://www.w3.org/TR/json-ld-api/	Method of transporting Linked Data using JSON APIs to expend/compact/flatten the data-structure to get different guarantees, transform from/to RDF,	To consider as ALMANAC's main data-interchange format, at least for semantic data

[100] http://docs.oasis-open.org/odata/odata/v4.0/os/part1-protocol/odata-v4.0-os-part1-protocol.html
[101] http://docs.oasis-open.org/odata/odata-json-format/v4.0/os/odata-json-format-v4.0-os.html
[102] http://xmpp.org/about-xmpp/technology-overview/
[103] http://xmpp.org/extensions/xep-0060.html#subscriber-subscribe

WAMP (Web Application Messaging Protocol)	wamp.ws	Based on WebSocket. Remote Procedure Calls + Publish & Subscribe. Good possibility of load-balancing. Interesting implementation: http://autobahn.ws/ (Open-source real-time framework for Web, Mobile & Internet of Things)	To consider for near-realtime communication with many Web clients.

15.7 Reference Architectures

Name	Reference	Description	Relevance
IoT-A reference architecture	Website[104]	Describes the definition process and the initial functional blocks.	WP3
IoT-A terminology	Website[105]	List of concepts and terms used in the IoT reference model	WP3
IoT-A converged architectural reference model	Website[106]	The complete (converged) architectural reference model for the IoT (November 2012)	WP3
IoT-A Solutions for Privacy and Security	Website[107]	Details the security building blocks and their functionality in the IoT-A architecture.	WP3
IoT-A main book reference	DOI:10.1007/978-3-642-40403-0	Describes the IoT projects objectives & results, IoT ARM reference, usage guides and examples.	WP3
ISO/TC/268/SC1 Smart urban infrastructure metrics	Website[108]	ISO work on metrics for SC (also check CEN groups)	
UK Authority/ BSI	Website[109]	UK Smart City Vocabulary	
IoT Design Patterns	Website[110]	Interesting view discussing "Design Patterns" in IoT architecture	WP3
Eclipse SmartHome	eclipse.org/smarthome/ www.openhab.org	Open Source Services, Interfaces and Tools for IoT. Based on Java/OSGi. Targeting smart homes, but seems to overlap with some of the ALMANAC components	WP3
Intel IoT Platform	Website[111]	End-to-end reference architecture, including data collection, gateways, security, cloud storage, data retrieval, data visualisation	WP3

[104] http://www.iot-a.eu/public/public-documents/d1.2/view
[105] http://www.iot-a.eu/public/terminology
[106] http://www.iot-a.eu/public/public-documents/documents-1/1/1/D1.4/at_download/file
[107] http://www.iot-a.eu/public/public-documents/d4.2/view
[108] http://www.iso.org/iso/standards_development/technical_committees/other_bodies/iso_technical_committee.htm?commid=656967
[109] http://www.ukauthority.com/tabid/64/Default.aspx?id=4603#
[110] http://iot-datamodels.blogspot.it/2014/05/design-patterns-for-internet-of-things.html?m=1
[111] http://www.intel.eu/content/www/eu/en/internet-of-things/overview.html

15.8 Fi-WARE components

Examples: sections of generic enablers from FIWARE which are useful for us.

Name	Reference	Description	Relevance
Object Storage	FI-WARE Catalogue[112]	Provides robust, scalable object storage functionality through an open, standardised interface: it exposes a CDMIinterface on top of OpenStack Swift.	WP6 (cloud based) Storage Management
Cosmos - Big Data analysis	FI-WARE Catalogue[113]	Implementation of the Big Data GE, based on Hadoop ecosystem, including support for MapReduce	Test of usability of Hadoop & Map Reduce in ALMANAC data management
Wire Cloud	FI-WARE Catalogue[114] FI-WARE Mashup YouTube link[115]	Wirecloud is an open source, reference implementation of the FIWARE Application Mashup. Creating mash-ups by wiring existing components/widgets. REST	Consider for end-user creation of mash-up UIs to ALMANAC platform services.
Device Management	FI-WARE Catalogue[116]	A REST API for M2M application developers and a device communication API, receiving ETSI M2M events (and other protocols) together with historic info.	WP4 to decide on relevance
REST for OMA NGSI 9/10	Website[117]	FI- ware version of the OMA NGSI 10 interface which is a RESTful API via HTTP. Its purpose is to exchange context information. The three main interaction types are • one-time queries for context information • subscriptions for context information updates (and the corresponding notifications) unsolicited updates (invoked by context providers)	ALMANAC architecture, WP3,4,5,6

[112] http://catalogue.fi-ware.eu/enablers/object-storage-ge-fi-ware-implementation
[113] http://catalogue.fi-ware.eu/enablers/bigdata-analysis-cosmos
[114] http://catalogue.fi-ware.eu/enablers/application-mashup-wirecloud
[115] http://www.youtube.com/watch?v=yzQqstBAUeo#t=1
[116] http://catalogue.fi-ware.eu/enablers/backend-device-management
[117] http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/OMA_NGSI_10