ALMANAC

RELIABLE SMART SECURE
INTERNET OF THINGS FOR SMART CITIES

(FP7 609081)

D3.1.1 System Architecture Analysis & Design Specification 1

Date 28th February 2014 – Version [1.0]

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 Thingsj for Smart Cities

Document control page

Document file:	D3.1.1 System Architecture Analysis and Design Specification 1
Document version:	[1.0]
Document owner:	Mark Vinkovits (Fraunhofer FIT)
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	Mark Vinkovits (FIT)	2013-12-03	Initial ToC
0.2	Mark Vinkovits (FIT)	2014-01-30	Contributions chap. 3 and 9.
0.3	Alexandre Alapetite (ALEX), Giampiero Bono (ISMB), Matts Ahlsén (CNET), Mathias Axling (CNET), Riccardo Tomasi (ISMB)	2014-02-07	Contributions chap. 4, 5, 6, 7, 9 and annexes
0.4	Mark Vinkovits (FIT)	2014-02-10	Integrated contributions
0.5	Matthias Axling (CNET)	2014-02-12	Extended contributions in ch 6,8,9.
0.55	Riccardo Tomasi (ISMB), Giampiero Bono (ISMB)	2014-02-14	Added annex 3
0.6	Mathias Axling, Matts Ahlsén (CNET)	2014-02-17	Extension of ch 6,8,9 and annex.
0.7	Mark Vinkovits (FIT)	2014-02-17	Contribution to ch. 2,4. Integration of input to annex 3.
0.8	Mark Vinkovits (FIT)	2014-02-18	Final review and approved for internal review
1.0	Mark Vinkovits (FIT)	2014-02-25	Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Summary of comments
Jesper Thestrup	2014-02-24	Accepted with minor comments
Maurizio Spirito	2013-02-25	Accepted with minor comments

Index:

Table of Figures

1. Executive summary

This deliverable describes a first version of the architecture used within ALMANAC. There will be an updated version of the architecture description in deliverable D3.1.2 due in month 18 and a final version in deliverable D3.1.3 due in month 28.

First, the methodology used to achieve and document the architecture is presented. The architecture is the result of a bottom-up phase, where individual partner technologies had been in focus, and a top-down phase, where applications and platform services had been defined. The definition has also been guided by the requirements collected in WP2. The documentation of the architecture is based on IEEE 1471 "Recommended Practice for Architectural Description for Software-Intensive Systems" (IEEE 1471, 2000). It implies a process which builds on a set of relevant architecture viewpoints.

In the functional view the components, their functionality, and their interactions are described. The main components are:

- Data Management Layer: the component that enables caching and querying of collected smart city data,
- Virtualization Layer: the component that abstracts from specific resources to virtual entities, easily accessible by smart city applications,
- Smart City Resource Adaptation Layer: the component that uniforms access across a number of protocols, enabling interoperability when accessing devices,
- Policy Management Framework: the component that protects the privacy of stakeholders in a transparent way across the platform.

The deployment view describes how and where the system will be deployed and what dependencies exist, considering hardware requirements and physical constraints.

The information view describes the data models and the data flow as well as the distribution. The viewpoint also defines how the distinction between fresh and latest values is achieved.

Finally, a number of use cases are described which will be relevant for the ALMANAC project. The purpose of these use cases is to clarify how the ALMANAC platform will work and which components are relevant for the different tasks.

2. Introduction

This chapter outlines the purpose, background, and context of this deliverable as well as the structure of the remaining document.

2.1 Purpose, context and scope of this deliverable

This deliverable defines the initial architecture for the ALMANAC platform. The requirements for the architecture can change during the course of a project and some aspects of the architecture need verification during development, and therefore the architecture described here cannot be considered to be final or complete. There will be an updated version of the architecture description in deliverable D3.1.2, due in month 18, and a final version in deliverable D3.1.3, due in month 28.

Within the ALMANAC work package structure, Work Package 3 (Smart City Platform Architecture) is responsible for specifying the system architecture design. Having completed the previous steps in WP2 (Requirements Engineering and Smart City Business Models), i.e. an initial set of requirements, this deliverable defines the system architecture, preparing for prototypal implementation to be carried out by the technical work packages.

The architectural description includes aspects related to the identification of the major system components, how they should interact and how their external interfaces should be defined.

Chapters are presented as follows. In chapter 3 the methodology for documenting the architecture will be introduced. Chapters 4 to 8 contain the different views of the architecture; these are then instantiated for specific technical use cases in chapter 9. The document also possesses three additional annexes that report activities that supported the architecture definition process.

2.2 Background

The ALMANAC Smart City Platform (SCP) collects, aggregates, and analyses real-time or near real-time data from appliances, sensors and actuators, smart meters, etc. deployed to implement Smart City processes via an independent, pervasive data communication network. ALMANAC aims at achieving pervasiveness by defining a short range capillary radio network providing local Machine-to-Machine (M2M) connectivity to smart things and enabling their active involvement in Smart City processes. The SCP allows decision support and implements intelligent control of the devices through the capillary networks with a M2M management platforms, as well as management of local installations.

The technological work in connection with the development of the ALMANAC Smart City platform will be highly influenced by requirements generated in the City of Turin. Its path to become "Smart City" started 2 years 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 CO_2 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 in the ALMANAC platform. These applications are deemed to be sufficiently representative for a large number of applications, as will be visible from the use case descriptions. Given the challenging objectives, we have aimed for a set of 1) cross application domain use cases that 2) consist of a large amount of heterogeneous devices and 3) generate large amounts of data.

3. Methodology

In this chapter we present the methodology and individual steps that were taken to achieve the architecture that is presented in later chapters. The architecture definition process was selected based on two principles: first, each partner brought in technologies and software that were to be considered for the architecture, additionally the ALMANAC platform was to be built based on the LinkSmart middleware, which had also guided us with its pre-existent structure; second, since the aim of the ALMANAC project is to provide an environment that hosts a variety of applications running on top of it, we needed a way to regularly remind us to distinction between specific application implementations and common platform services. The manifestations of these principles are the bottom-up approach that initiated our architecture definition process, while the scenario thinking and use case driven approaches can rather be seen as top-down thinking.

This chapter is structured according to these two phases. In section 3.1 we present how we have gathered and combined individual technologies and components the partners brought into the project; additionally, we added solutions that were foreseen and will be developed within the ALMANAC project. The result of this phase was a more detailed view on the initial ALMANAC architecture as presented in Figure 1. Section 3.2 afterwards deals with the top-down phase. This phase was driven through the selection and extension of one of the scenarios presented in D2.1 (ALMANAC WP2, 2013). During this phase we held a number of workshops that lead us step by step to the state that is presented in this deliverable.

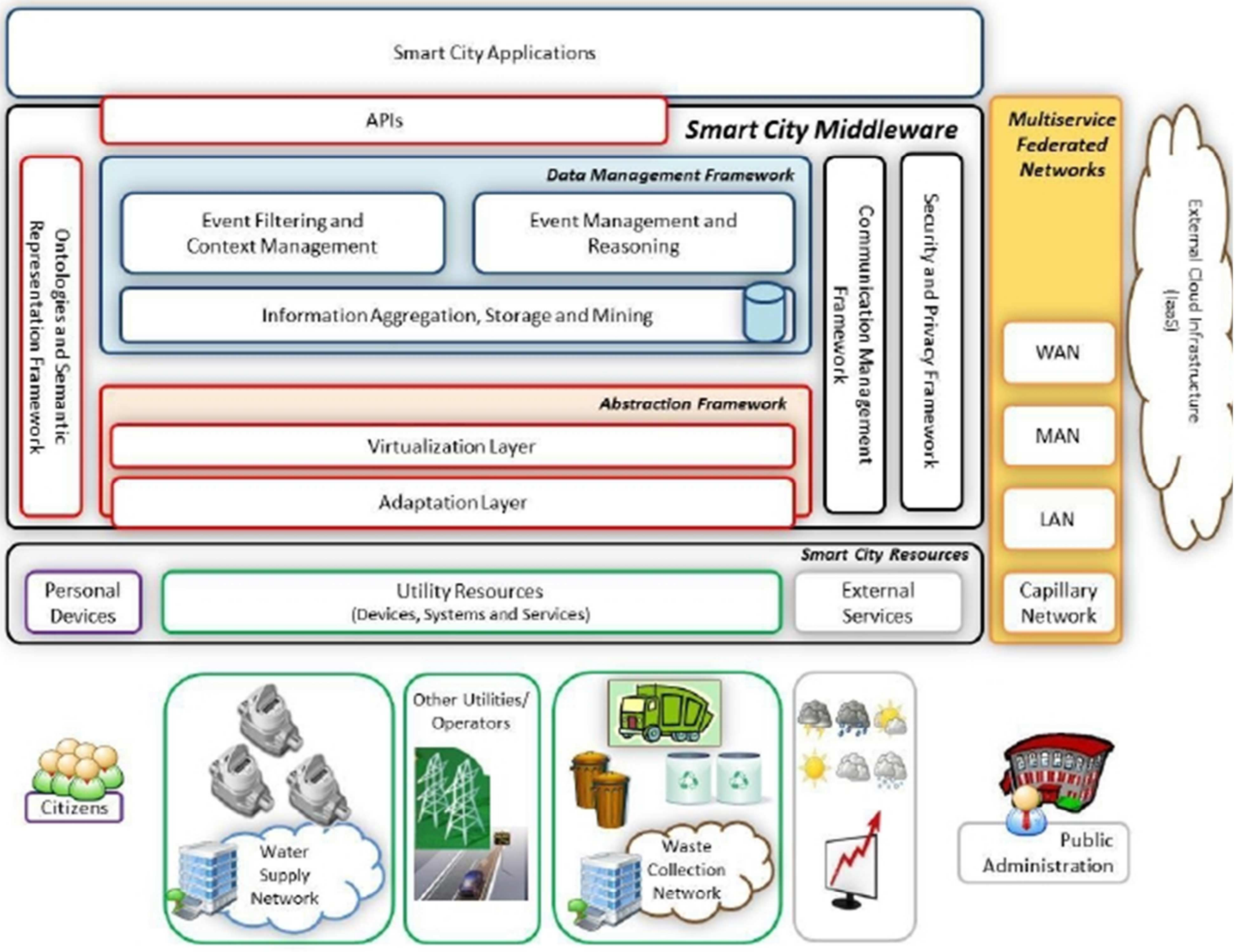

Smart City Applications
APIs
Smart City Middleware
Data Management Framework
Event Filtering and Context Management
Event Management and Reasoning
Information Aggregation, Storage and Mining
Abstraction Framework
Virtualization Layer
Adaptation Layer
Ontologies and Semantic Representation Framework
Communication Management Framework
Security and Privacy Framework
Multiservice Federated Networks
WAN
MAN
LAN
Capillary Network
External Cloud Infrastructure (IaaS)
Smart City Resources
Personal Devices
Utility Resources (Devices, Systems and Services)
External Services
Citizens
Water Supply Network
Other Utilities/ Operators
Waste Collection Network
Public Administration

Figure 1 Initial ALMANAC architecture as presented in DoW

3.1 Bottom-up process

The first phase of the architecture definition process was intended to collect and categorize the technologies and software components the individual partners of the ALMANAC project brought in with them; through this, we wished to achieve that partner expertise got quickly identified and used as best as possible. It also helped us identify gaps in the architecture that needed to be filled to achieve the smart city platform, as envisioned by ALMANAC.

As first step of this phase we have defined a template: it required partners to provide short descriptions of components, place them into the initial architecture of Figure 1, define services and dependencies within the platform, and think of services that would be open or even accessibly through the cloud-based API. Altogether nine component descriptions have been acquired this way; however, as they were partially on different abstraction levels, they could not be mapped one to one into the architecture. The collected component descriptions can be found in annex 1.

As next step we have taken the provided component descriptions and identified their position in the initial architecture. We did so to see how well required functionality can already be covered, and to be able to identify gaps that still needed to be addressed. We also added some components that seemed to be realizable based on the provided component descriptions. The result of this process can be seen in Figure 2. Looking at the distribution of components in Figure 2, it is visible that the layers and frameworks, as specified initially, are very ambiguous and are of different nature; therefore some of the layers are populated with multiple components, whereas other segments of the architecture figure are barely covered. Having identified this issue, we targeted to clarify the role of each layer during the following phases.

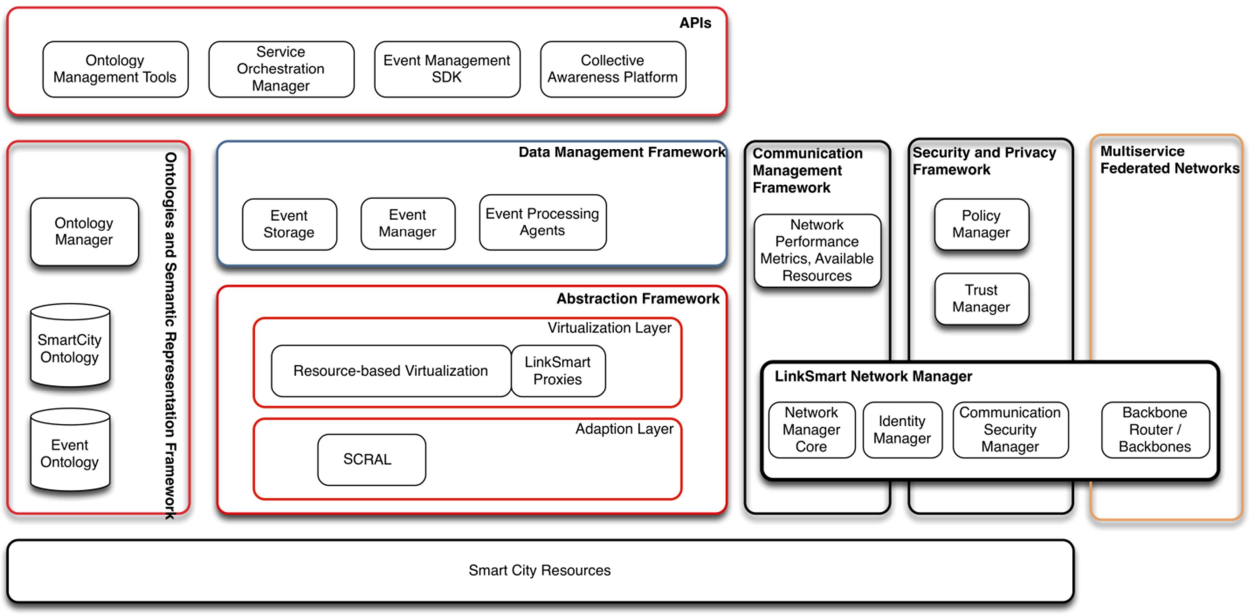

APIs
Ontology Management Tools
Service Orchestration Manager
Event Management SDK
Collective Awareness Platform
Ontologies and Semantic Representation Framework
Ontology Manager
SmartCity Ontology
Event Ontology
Data Management Framework
Event Storage
Event Manager
Event Processing Agents
Abstraction Framework
Virtualization Layer
Resource-based Virtualization
LinkSmart Proxies
Adaption Layer
SCRAL
Communication Management Framework
Network Performance Metrics, Available Resources
Security and Privacy Framework
Policy Manager
Trust Manager
Multiservice Federated Networks
LinkSmart Network Manager
Network Manager Core
Identity Manager
Communication Security Manager
Backbone Router / Backbones
Smart City Resources

Figure 2 Initial component coverage of architecture

3.2 Use case driven/Top-down process

After the bottom-up process, where we identified potential main building blocks of the ALMANAC platform, we needed to better clarify each component's role. To be able to do so, we took advantage of the scenarios developed in D2.1 (ALMANAC WP2, 2013), and instantiated them using the components already identified. This has allowed us to stress the services each component would provide, and sketch the interactions between them. Using this approach, and thinking in terms of applications has also required us to imagine the open APIs, as envisioned by ALMANAC. In the following section we summarize the workshops we held with this approach, present the scenarios we applied and summarize the lessons learnt.

3.2.1 Workshop one

At the first workshop, we used the example of an issue management application for the waste management scenario. We concentrated on the interaction between the application and the cloud-based open APIs of the ALMANAC platform, and looked how the communication between the two entities would look like. We mainly concentrated on the data gathering phase, as that was the most relevant for the upcoming first demonstrator. As an underlying infrastructure, we investigated the possibilities of an M2M platform, and sketched the integration possibilities with the middleware, based on the potential standardized interfaces: NSCL, GSCL, DSCL (Network-, Gateway-, Device Service Capability Layer) (ETSI, 2011). Related to physical deployment, our focus was on the diverse capabilities of a cloud infrastructure compared to lightweight gateways, and on the mechanisms to distribute responsibilities accordingly. Main achievements of the workshop were:

- The GSCL interface has the biggest potential for long-term integration of M2M.
- Storage of historical open data should be located in the cloud, as it has the best possibilities to tackle scalability issues.
- To support scalability of data collection, gateways may be commanded to do pre-aggregation of data before submitting it to the cloud.
- Semantic representation should be divided into following categories: device level, service level, smart city level.
- The Virtualization Layer should be supported with semantic information concerning devices, gathered by the Smart City Resource Adaptation Layer.

3.2.2 Workshop two

At the second workshop we reviewed the results of the previous one, and investigated it from other perspectives: network federation and information ownership. Related to the first point, we found that our view from the previous workshop considered the network too much as a homogeneous black-box, and that therefore, we did not identify components handling the interaction between autonomous networks. While trying to clarify the challenges of federating networks, we found that one of the main driving forces in our process should be the already mentioned question of ownership: who collects, who owns and who can access data produced in the smart city platform. We identified several use-cases of using the ALMANAC system, categorized by ownership of infrastructure, ownership of information and access to information. These questions had a large potential impact on the previously drafted architecture, and we could not answer them at this workshop. As a result we agreed on an additional workshop, with a number of phone conferences in advance as preparation.

3.2.3 Workshop three

We arrived much prepared at the third workshop due to the many phone conferences we held before. Main topics included: elaboration of terms that have been used ambiguously before, clarification of entry points for applications and devices, differentiation of storage approaches used for different use cases. Having established this common ground, we reviewed the previously drafted architecture diagrams and finalized them into the form as presented in the following sections.

3.3 Documentation of architecture

The process used for documenting the architecture in this document is based on IEEE 1471 "Recommended Practice for Architectural Description for Software-Intensive Systems" (IEEE1471, 2000). This standard establishes a methodology for the architectural description of software-intensive systems. One main part of this methodology is the use of viewpoints: collections of patterns, templates and conventions for constructing one type of view. One example is the functional viewpoint (and therefore a functional view) which contains all functions that the system should perform, the responsibilities and interfaces of the functional elements and the relationship between them. These functions can be described using UML diagrams. Moreover, it also describes which stakeholders need to be involved and how to apply their needs in the architecture as stated in the "architectural perspectives" chapter by Rozanski and Woods (Rozanski - Woods, 2005).

In the initial version of the architecture we decided that the three most important viewpoints are the functional viewpoint, the information viewpoint and the deployment viewpoint from which the views of the architectural document are derived.

- Functional viewpoint (section 4): This viewpoint describes the functional elements needed to meet the key requirements of the architecture. It will present proposals in a descriptive way and UML diagrams will assist in the understanding of the proposal. It will describe responsibilities, interfaces, and interactions between the functional elements.
- Deployment viewpoint (section 5): This viewpoint describes how and where the system will be deployed and what dependencies exist, considering for example hardware requirements and physical restraints. If there are technology compatibility issues, these can be addressed in this viewpoint as well.
- Information viewpoint (section 6): The information viewpoint describes the data models and the data flow as well as the distribution. The viewpoint also defines which data will be stored and where. The description of where data will be manipulated is also part of this viewpoint.

To address quality properties and cross-cutting concerns, architectural perspectives will be used. A typical example is security (section 7.1): it should be considered how the data is secured and which functional elements need to be protected. Another perspective that is of concern for ALMANAC is scalability (section 7.2).

4. Functional view

This chapter gives an overview over the different components, their functionality, their interfaces, and their interactions. The functional view addresses concerns of all stakeholders. The overall component diagram of the ALMANAC SCP is outlined in Figure 3.

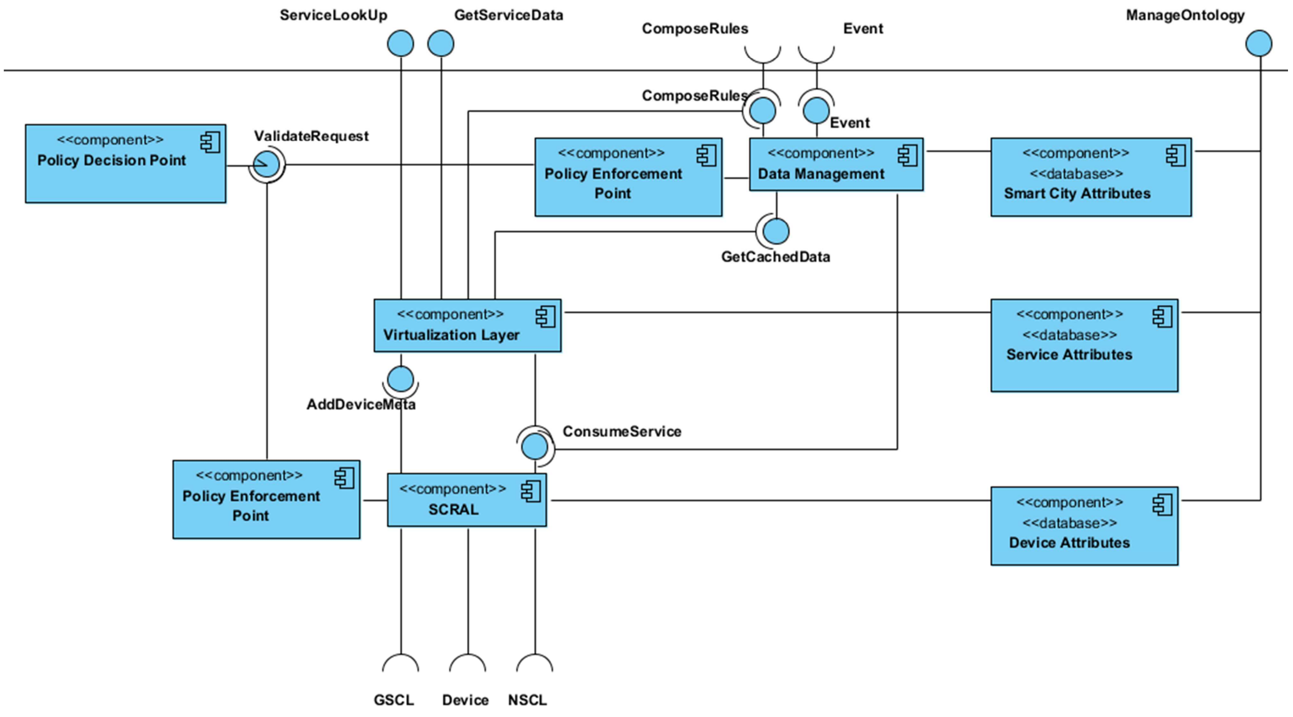

ServiceLookUp
GetServiceData
ComposeRules
Event
ManageOntology
ComposeRules
Event
<<component>> Policy Decision Point
ValidateRequest
<<component>> Policy Enforcement Point
<<component>> Data Management
<<component>> <<database>> Smart City Attributes
GetCachedData
<<component>> Virtualization Layer
<<component>> <<database>> Service Attributes
AddDeviceMeta
ConsumeService
<<component>> Policy Enforcement Point
<<component>> SCRAL
<<component>> <<database>> Device Attributes
GSCL
Device
NSCL

Figure 3 Component diagram of the ALMANAC platform

4.1 Overview of components

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. There are four tasks that can be considered as the main services provided by the platform to the 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. Additionally to this service, the SCRAL exploits any kind of meta-, and semantic information of appearing devices and feeds them into the Virtualization Layer.
- Virtuality 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 entity. The component directly grabs data coming from devices, and by parsing and indexing it, enables later complex querying. The rules or mechanisms that the Data Management entity should execute are either specified directly by applications, or indirectly through the Virtualization Layer.
- 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 Policy Enforcement Points (PEP) throughout the platform, thereby enabling applications to access only the data and functionalities for whom they have explicit access rights.

4.2 List and description of components

This section elaborates in more detail the high level components that have been introduced previously. Each component has its own subsection, where underlying concepts, subcomponents and services are introduced; additionally further dependencies between high level components are analyzed.

4.2.1 Smart City Resources Adaptation Layer (SCRAL)

The Smart City Resources Adaptation Layer (SCRAL) exposes in RESTful fashion any standard or device-specific interface made available by concrete devices, systems and services (i.e. PhysicalDevices) towards all other components of the ALMANAC platform. The SCRAL exposes functionalities made available by devices through a self-described and annotated REST interface. Examples of the exposed features include e.g. devices related to water supply and waste management (at home level and at "utility" level – water distribution network, waste collection network); people – to enable citizens' involvement in the Smart City process (participatory sensing; -third-party services and systems e. g., open data sources, weather forecasting or traffic monitoring services).

The SCRAL externally exposes the following open APIs:

SCRAL API	Description	Examples
SCRAL-EndpointAPI	A RESTful interface exposing all the sensing and actuating features made available by PhysicalDevices. It is generally consumed by the Data Management layer to receive data or directly by applications to control devices. This API does not ensure any kind of semantic interoperability, focusing only on exposing devices features.	*GET /devices/51/channels/3/value* *POST /devices/51/ctrl/5/ON-control*
AddDeviceMeta	A client interface towards the Virtualization layer, used to signal registration or de-registration of concrete devices. It is used to transfer to the Virtualization Layer all kind of meta-information which can only be discovered from the Physical Devices.	*"Device X implements the TemperatureSensor cluster as defined by the ZigBee standard Home Automation 1.1"*
PEPInterface	Policy Enforcement point used to control device-specific security aspect	*"Is user X authorized to read value of water meter?"*

The internal composition of the SCRAL follows a modular, plug-in based approach both towards Physical Devices and towards heterogeneous cloud platforms as described in Figure 4.

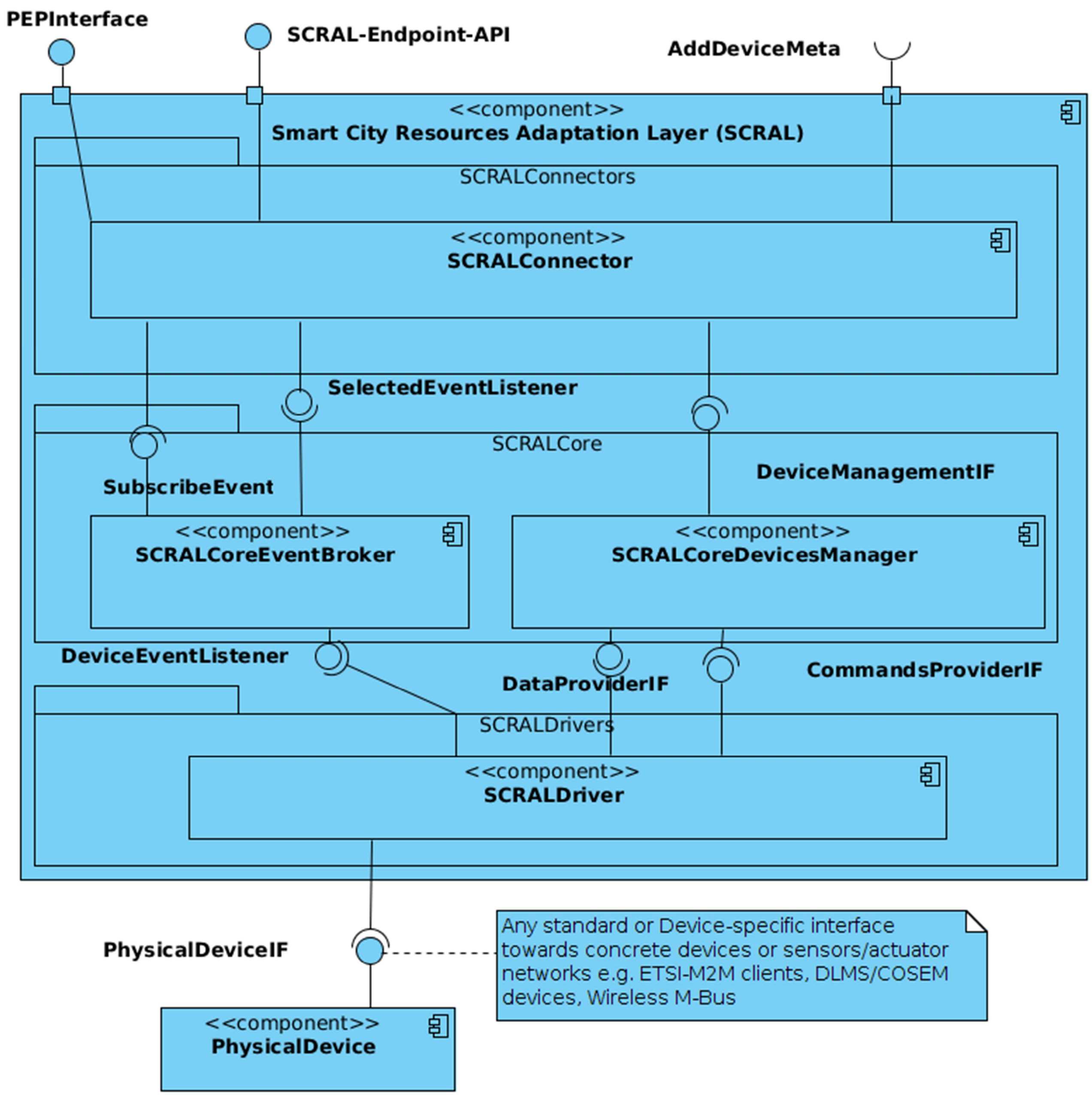


Figure 4 Smart City Resources Adaptation Layer (SCRAL) component diagram

On the south-bound part i.e. the part of the SCRAL in charge of integrating heterogeneous devices, the SCRAL hosts one or more SCRALDriver components. SCRALDrivers are used to wrap and isolate all device- or standard-specific implementations- used to access Physical Devices.

The north-bound part, i.e. the part of the SCRAL in charge of exposing REST resources towards the overall ALMANAC platform, is implemented through the SCRALConnector components. The SCRALConnector follows a REST model, although other connectors might be optionally available (i.e. SOAP connectors or event-driven connectors), derived from previous projects[1]. The SCRALConnector is the main component in charge of exposing the SCRAL Open API.

In the middle part, the SCRAL includes a set of core components which are used to map the south-bound devices features with the north-bound exposed resources[2].

[1] ebbits (http://www.ebbits-project.eu)
[2] All internal interfaces will be specified more in details in the future deliverables from task T5.1.

Normally, the SCRAL is deployed as close as possible to the concrete Physical devices (e.g. on IoTGateways) to enable fast and reliable access. In some configurations it can also be hosted in cloud-like environments e.g. when it is used simply to wrap or reflect complete platforms e.g. the ETSI-M2M NSCL (Network Service Capability Layer).

Features to be specified more in details in future iterations

The SCRAL has also the goal of exploiting context-awareness techniques applied from bi-directional data synchronization, thus ensuring that the information available within the ALMANAC platform is kept up-to-date with information from the physical world at an optimal rate; minimizing network traffic and consumption related to data synchronization logics.

These features will be considered in the future iterations of the architecture.

4.3 Virtualization Layer

The Virtualization Layer is a component within the Abstraction Framework. It is a component that enables search, lookup and addressing of services registered to the ALMANAC platform. Through the Virtualization Layer, applications can communicate with the ALMANAC platform, for example they can show ALMANAC data to e.g. citizens or city authorities. Applications can access the full functionality of ALMANAC connected IoT resources, including actuation of IoT devices. The Virtualization Layer accepts queries using "real world" terms. This could be postal addresses or GPS coordinates, sensor type and exposes functionality. It will convert the query to an ALMANAC internal, URI-based addressing scheme, enabling applications to communicate directly with the IoT devices, or access the devices specifically through the ALMANAC platform. The Virtualization Layer will understand a number of ontologies used by the other components of ALMANAC and corresponding semantic queries. Furthermore, the Virtualization Layer will implement a reasoning algorithm, based on a set of internal rules, to decide whether the data needed to answer the query is required from either:

- A cache of historical data provided by the Data Management
- Fresh data from the devices themselves, by querying them through the Adaptation Layer (in particular the SCRAL, cf. Figure 4).

Clients will be querying the Virtualization Layer using REST to access resources and defines two kinds of requests (cf. Figure 3), namely "Service Data Requests" and "Service Lookup Requests".

Service Lookup Request

The Service Lookup Request offer a way to discover the data sources matching a number of criteria. This service is the point of entry for clients who want to gain knowledge of the underlying network of devices.

Upon reception of a request, the "Service Lookup" performs the following tasks:

1. Translate the "real world" criteria to internal references
2. Use a set of internal rules to decide what ALMANAC component to address, such as the Data Management Framework, specific atomic IoT resources or composite resources.
3. Return a list of matching data sources, with in particular their URI uniquely identifying the source of data in the ALMANAC infrastructure. The response will also contain information about the caching policy of the data i.e. how long this lookup is supposed to be valid, and can thus be cached by the client.

The client application may subsequently decide to query the discovered data sources directly through the Adaptation Layer, or use the Service Data Request described below. The Service Lookup Request is also used behind the scenes by other ALMANAC components.

Service Data Request

The Service Data Requests offer a higher level of abstraction for the client applications. The requests may be used when clients are interested in some data without necessarily having a good understanding of the underlying architecture. In particular, such requests require neither knowledge of the different types of sensors, their behaviour nor their organisation. Clients must pass the resource URI obtained by a Service Lookup Request, to the Service Data Request service. In order to fulfil a request, the Virtualization Layer will:

1. Validate the provided data source URI
2. Call the source of data referenced by the URI with the appropriate parameters, some of them being potentially automatically generated
    a. When addressing the Data Management, it may be needed to ask the Data Management to compose a new rule.
3. Process the data if needed for example transform the format
4. Return the results.

4.4 Data Management Layer

The Data Management Layer provides functionality for

- publishing and subscribing to data streams for components within the ALMANAC Cloud as well as end user applications outside ALMANAC.
- requesting values from resources in the system (sensors and aggregates of sensor data).
- building aggregates of data streams using ontology concepts.
- providing scalable storage for sensors and aggregates of sensor data.

The Storage manager provides distributed, scalable storage for the other components. Publish and subscribe functionality is handled by the Event Manager. The component used to create aggregate data streams is called a *Semantic IoT Resource (SITR*). Calculation of aggregate values occurs on request or when events are recieved from data sources. In the case that the calculation of aggregate values needs to be scheduled, the Activation Manager is used.

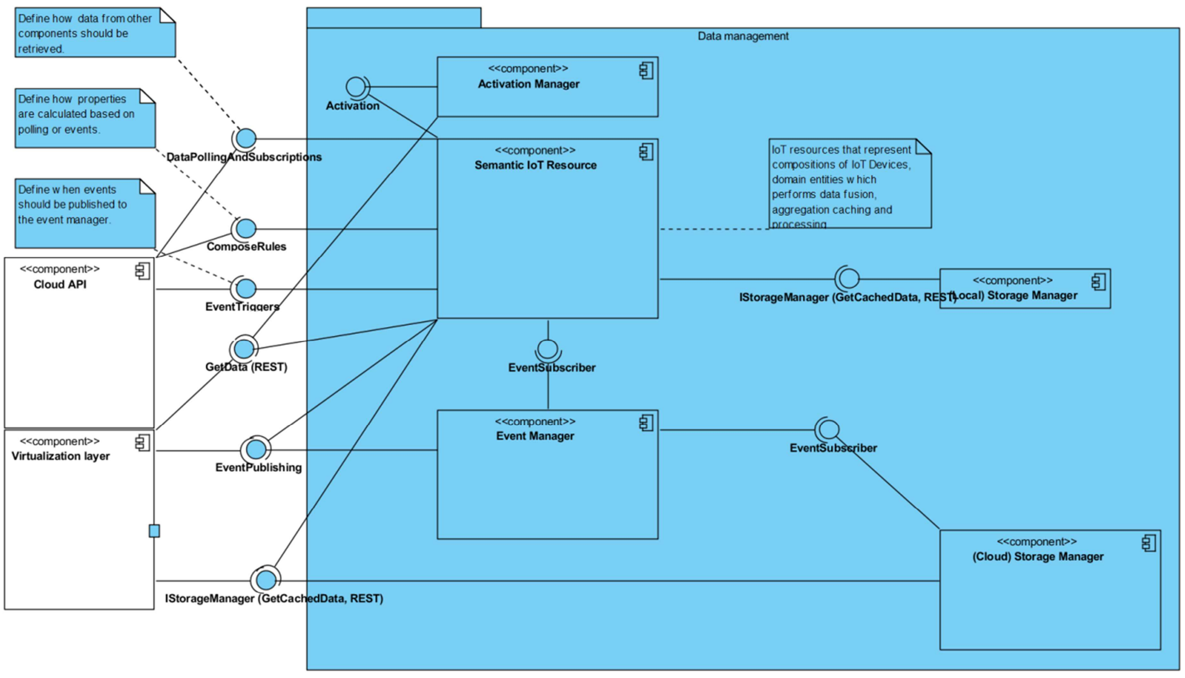

Define how data from other components should be retrieved.
Define how properties are calculated based on polling or events.
Define w hen events should be published to the event manager.
Data management
<<component>> Activation Manager
Activation
<<component>> Semantic IoT Resource
DataPollingAndSubscriptions
ComposeRules
EventTriggers
GetData (REST)
EventPublishing
IStorageManager (GetCachedData, REST)
IoT resources that represent compositions of IoT Devices, domain entities w hich performs data fusion, aggregation caching and processing
IStorageManager (GetCachedData, REST)
<<component>> (Local) Storage Manager
EventSubscriber
<<component>> Event Manager
EventSubscriber
<<component>> (Cloud) Storage Manager
<<component>> Cloud API
<<component>> Virtualization layer

Figure 5 Data Management Framework functionality

4.5 Policy Management Framework

The Policy Management Framework provides policy-driven, access protection for IoT-devices and applications, building on the XACML (eXtensible Access Control Markup Language) standard (Rissanen, 2010). The Policy Decision Point (PDP) is responsible for making the access decisions based on the XACML request it receives from a Policy Enforcement Point (PEP) and the set of policies that have been published to it. By the means of Policy Information Points (PIP) it is possible to handle even more complex authorization requests, as PIPs can help a PDP in resolving attributes that are not included in a request. An illustration of this basic structure can be seen in Figure 6.

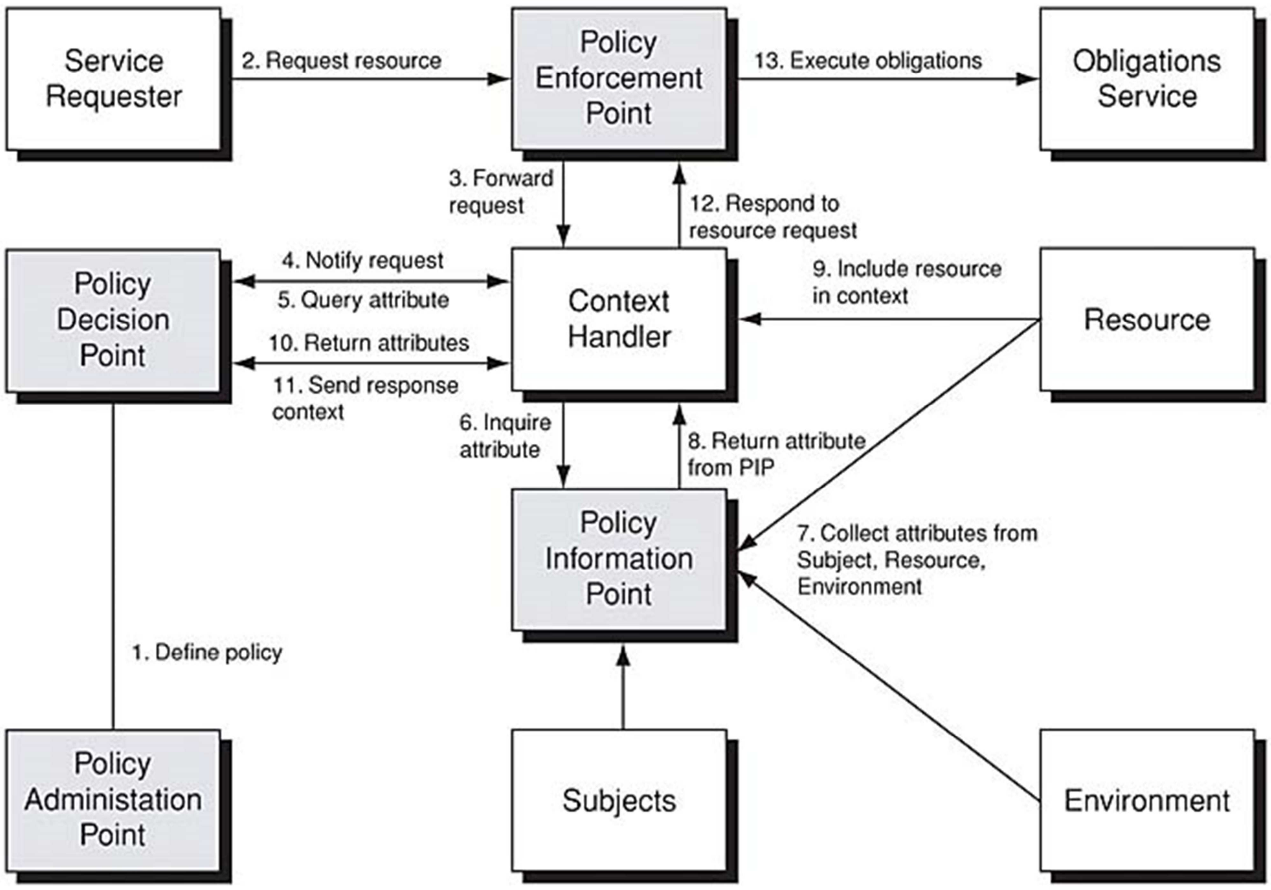


Figure 6 Reference architecture for XACML

The mechanism of PIPs will also be the attachment point for the federated authentication mechanism necessary for the proper privacy protection within the Smart City Platform. Federated authentication is about delegating and conceptualizing the authentication to Identity Providers and can be seen as a superset of Single Sign-On. The authentication itself is performed by a set of trusted set of Identity Providers thereby removing the need for users to directly authenticate against each and every service. As realization engine for the federated authentication, we chose the Security Assertion Markup Language (Madsen et al, 2005). SAML 2.0 is a very mature standard and can very well cooperate with the foreseen XACML authorization framework.

5. Deployment view

This view describes how the ALMANAC platform can be deployed on the field i.e. in concrete Platforms (e.g. servers, Cloud Infrastructure, embedded systems on the field, *PhysicalDevices*) and concrete execution environments (e.g. LinkSmart, etc.).

A simplified deployment configuration is described in Figure 7.

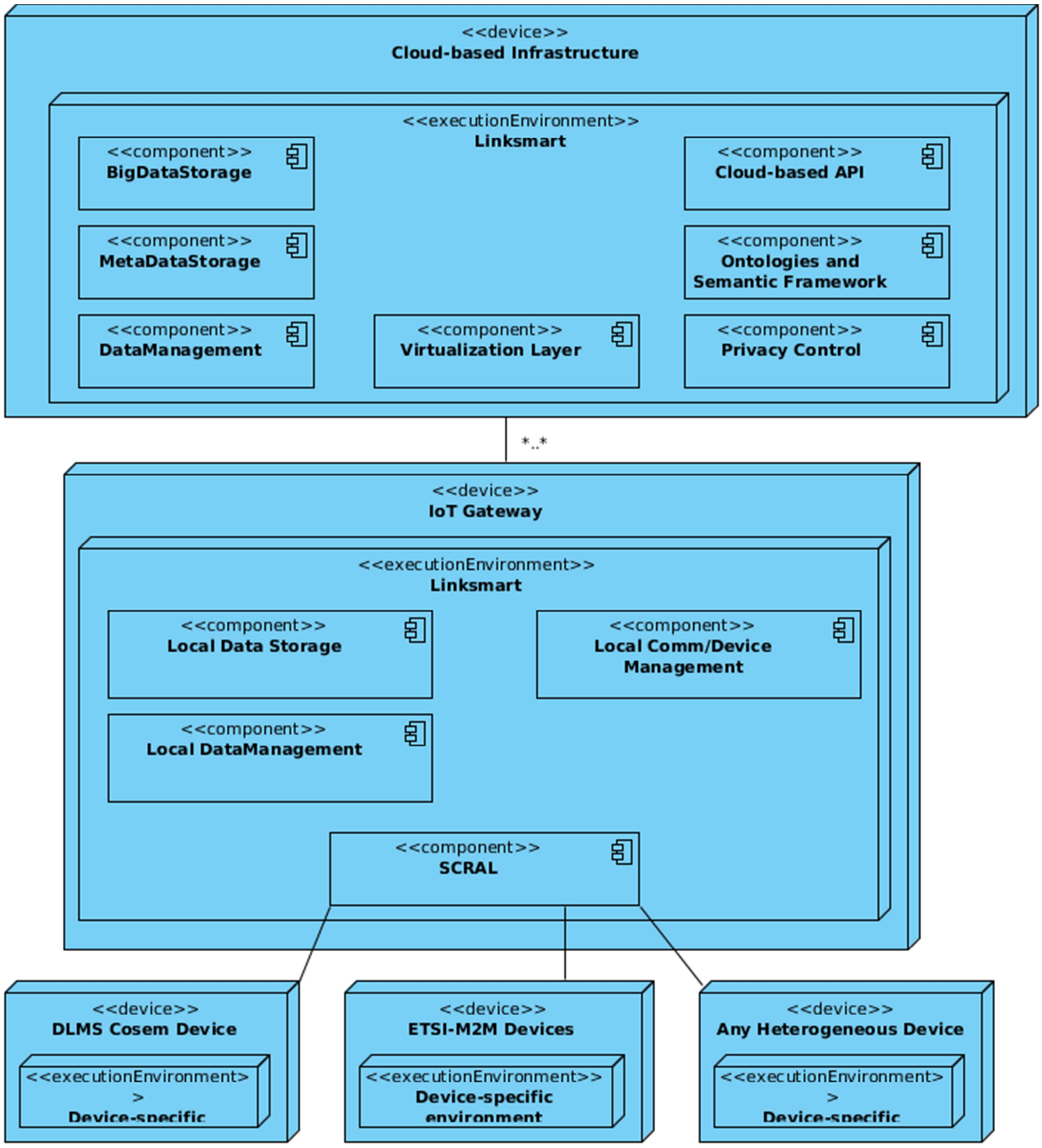


Figure 7 Deployment view, simplified configuration

Most of the functional components handling resources and applications for the Smart City as a whole are hosted in the Cloud-based infrastructure, running within a LinkSmart-based execution environment.

Most of the field components handling direct interaction with Smart City Resources are instead deployed within *IoT Gateways* i.e. low-cost embedded devices able to run LinkSmart. A number of "local" features are at the moment identified on the IoT Gateways e.g. to support local storage or aggregation of data. These features are currently not specified in details, but hey have been foreseen at this stage to support future scalability-critic scenarios. IoT gateways are in a *many-to-many* relationship with the Cloud-based infrastructure, as the same gateway could be inter-connected with several instances of Smart City platforms e.g. hosted by different organizations.

Various features hosted by specific devices (e.g. DLMS/Cosem devices or ETSI-M2M devices) are deployed on-board devices and exposed towards the ALMANAC platform through the SCRAL, running on IoT Gateways.

Figure 8 outlines a slightly different deployment scenario, involving ETSI-M2M gateways.

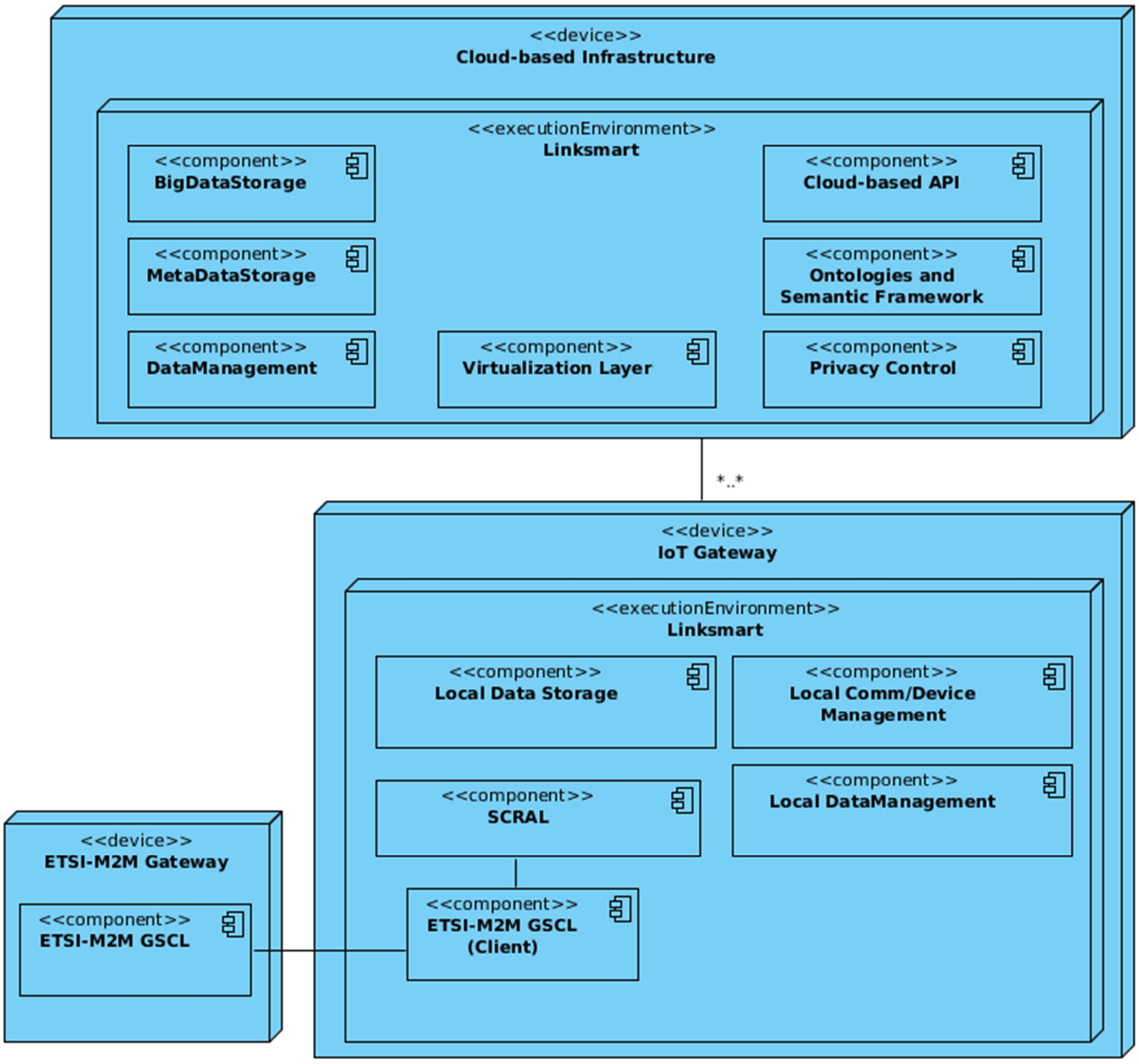


Figure 8 Deployment view, ETSI-M2M Gateway scenario

In case a pre-existing standard gateway exist already on the field, it can be accessed locally by the ALMANAC IoT Gateway through a dedicate SCRAL Driver (i.e. the ETSI-M2M GSCL Client Driver). This case has been considered at this stage for ETSI-M2M gateway, but it could also apply to other technologies where a need to locally inter-connect a standard gateways e.g. OMA Lightweight or Wireless M-Bus occurs.

Figure 9 finally, outlines a deployment scenario where an ALMANAC infrastructure is required to inter-operate with a pre-existing ETSI-M2M infrastructure.

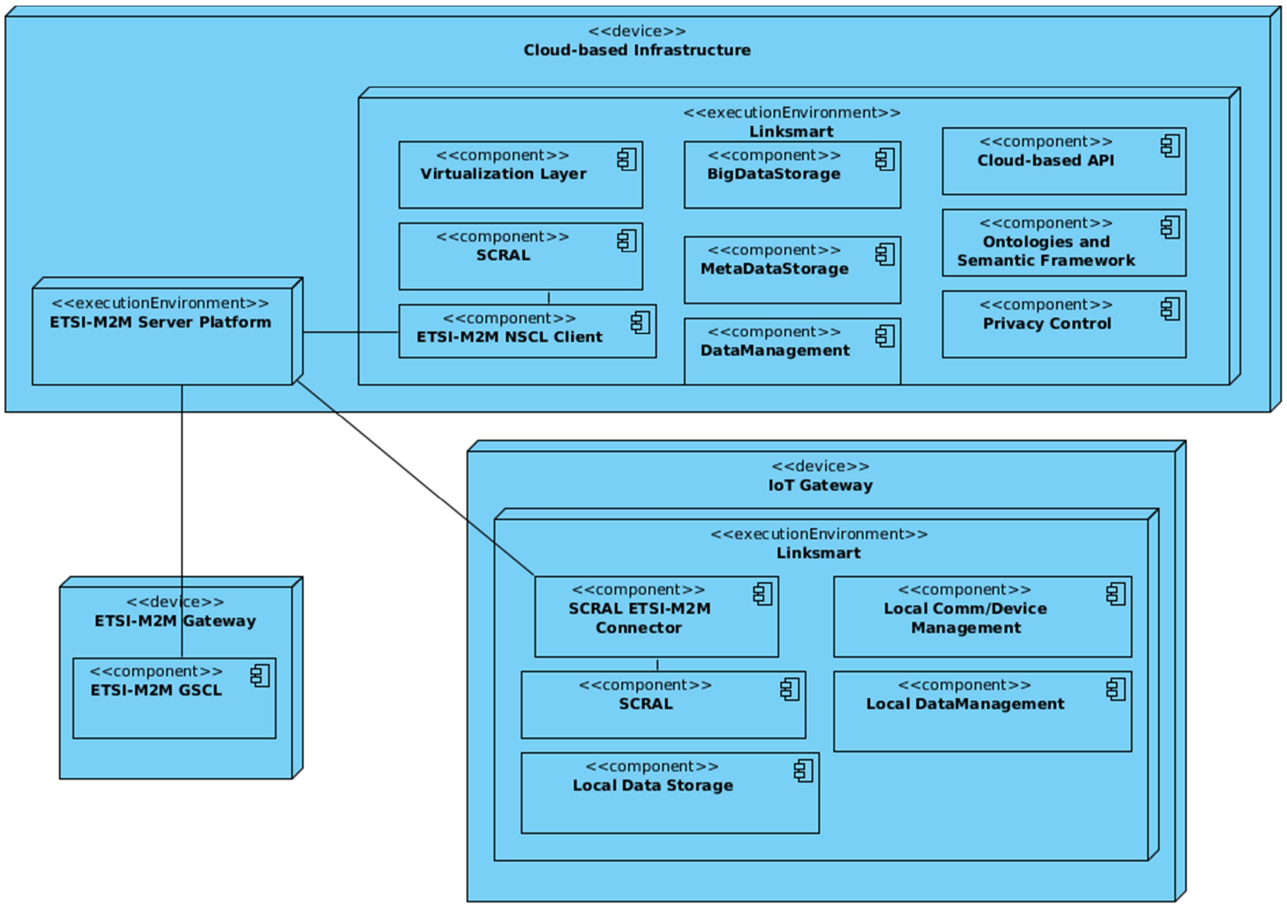

<<device>>
Cloud-based Infrastructure
<<executionEnvironment>>
Linksmart
<<component>> Virtualization Layer
<<component>> BigDataStorage
<<component>> Cloud-based API
<<component>> SCRAL
<<component>> MetaDataStorage
<<component>> Ontologies and Semantic Framework
<<executionEnvironment>> ETSI-M2M Server Platform
<<component>> ETSI-M2M NSCL Client
<<component>> DataManagement
<<component>> Privacy Control
<<device>> IoT Gateway
<<executionEnvironment>> Linksmart
<<component>> SCRAL ETSI-M2M Connector
<<component>> Local Comm/Device Management
<<device>> ETSI-M2M Gateway
<<component>> ETSI-M2M GSCL
<<component>> SCRAL
<<component>> Local DataManagement
<<component>> Local Data Storage

Figure 9 Deployment view, ETSI-M2M Platform scenario

In this case the ALMANAC platform accesses the ETSI-M2M platform through a cloud-hosted SCRAL Driver implementing a full-fledged ETSI-M2M NSCL Client, which in turn exposes all the resources managed by the ETSI-M2M platform. As an extension of this ETSI-M2M-centered scenario, ALMANAC also foresees the case where the IoT gateway can join a standard ETSI-M2M network by implementing a SCRAL ETSI-M2M Connector acting as a standard GSCL.

5.1.1 Network View

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 10. 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.

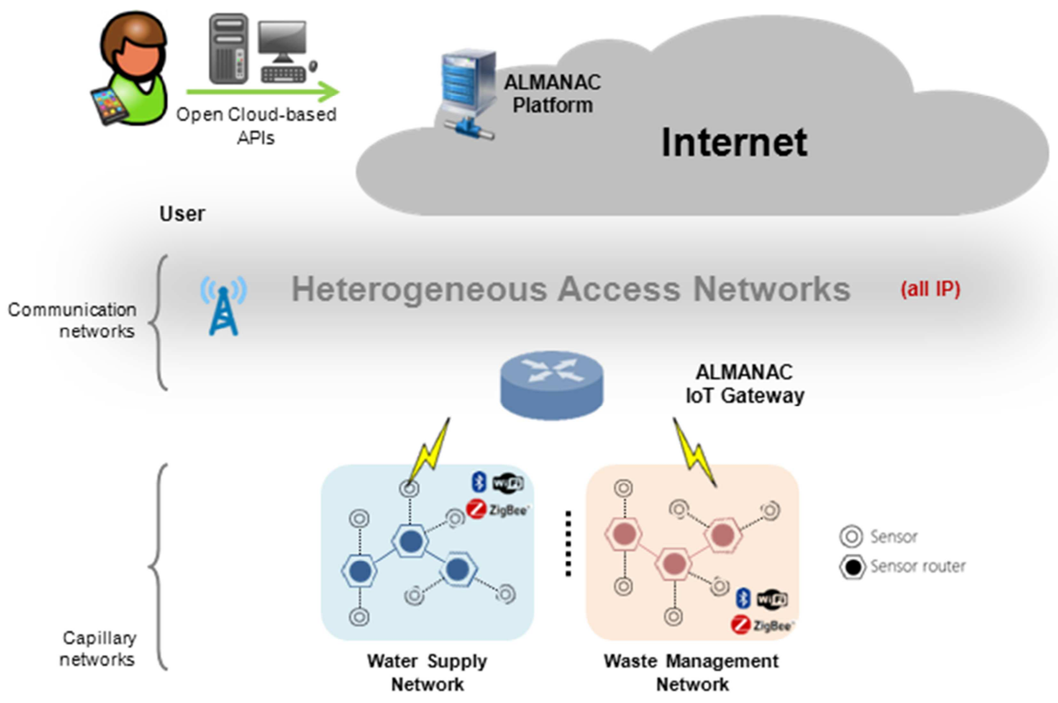


Figure 10 ALMANAC Network View

IoT Gateway are IP-based devices and can be thus directly interconnected 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 and actuators 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.

The main reason for devising 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.

Waste Management and Water supply network examples

In Figure 10, 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, but the design

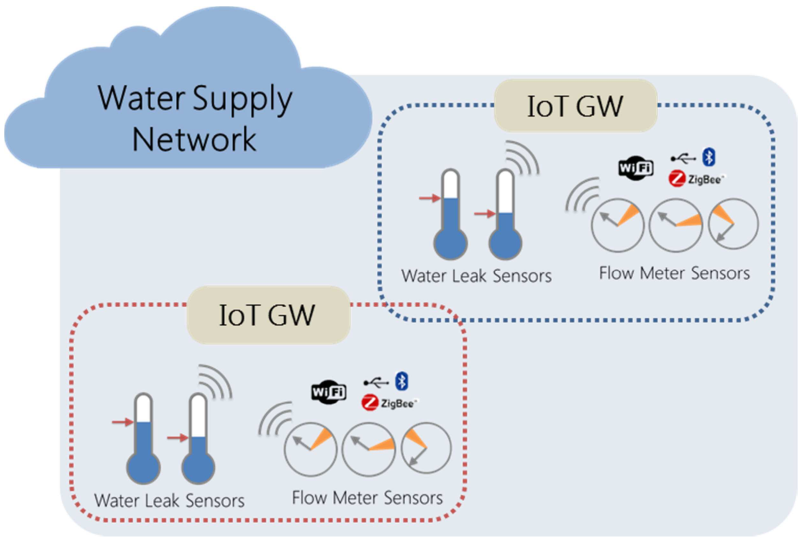


Figure 11 Water Supply Network

In the he capillary waste management network, fill level sensors are integrated with meteorological data and geographical location in order to obtain 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.

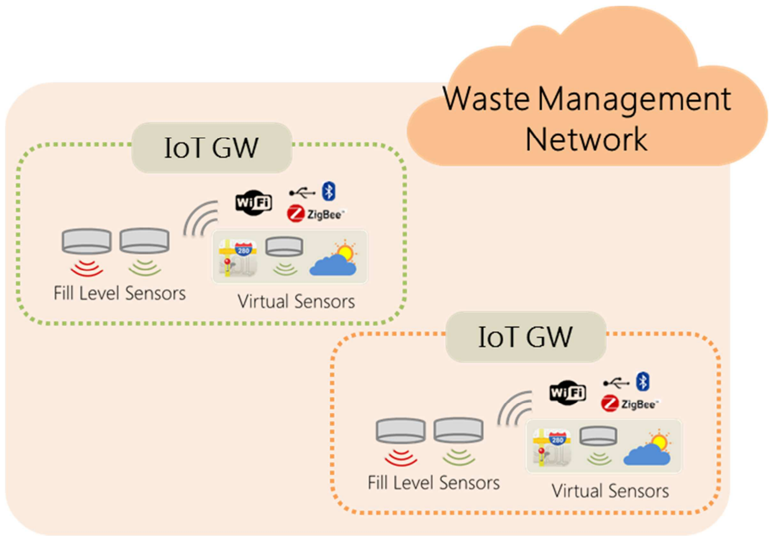


Figure 12 Waste Management Network

5.1.2 Notes on Federation Issues

Federation of multi-service IoT networks is a key issue in ALMANAC. Although specific activities to tackle federation issues will officially start in the second iteration of the project, the consortium has anticipated some work to scope the main rationales and definitions related to federation in the current iteration.

In general, federation is seen in ALMANAC as a "pattern" which allows two independent IoT networks to interoperate and share some capabilities. While interoperability is a pre-requisite for federation, it is important to stress that federation is not limited to the ability of two network to exchange some data, but is always based on some explicit agreement by two parties operating two independent IoT systems.

A known example of federation which has been taken as inspiration is the case of roaming in mobile cellular networks, where a user can authenticate with a remote network by leveraging on a set of known standards and agreements between telecommunication operators in different countries.

In the following, a number of federation related assumptions are reported: they will be considered in the current iteration and then researched more in detail in the second year of activities.

1. In ALMANAC, federation could be considered at different layers e.g.
   - ***at capillary level***: a sensor operated by "*IoT Network A*" might be able to join a third-party capillary network or gateway operated by "*IoT Network B*" thanks to a federated authentication mechanism; this could be useful to seamlessly allow sensors to use the communication infrastructure provided by a third-party operator.
   - **at access-network level**: a gateway operated by "*IoT Network A*" (e.g. the waste company) might be able to use the access network (e.g. Wi-Fi) of a third-party operator (e.g. the municipality or even a private user) thanks to a federated authentication mechanism.
   - **at service level:** an instance of the ALMANAC platform operated by "*IoT Network A*" (e.g. a municipality) might be able to access with a pre-defined granularity to data and events provided by "*IoT Network B*" (e.g. a water company, or another city);
2. There are a set of different services which might be federated e.g.
   - Authentication/authorization/access (network A can authenticate devices or users of network B)
   - Network management capabilities (network A can monitor devices of network B)
   - Databases or Storages (network A and network B can decide to jointly expose a data storage towards a third-party service)
3. There are different reasons to use federation in ALMANAC:
   - to support data and devices from a pre-existing network into ALMANAC (e.g. to seamlessly integrate device and data from the energy monitoring network and jointly offer water and energy management services)
   - to provide services which need data which can only be obtained by multi-service networks operated for other reasons by different utilities
   - to share data and functionalities across platforms operated by different cities (e.g. Torino and Santander) or organizations (e.g. as in enterprise mashups).
4. Since ALMANAC has adopted a broader definition of federation, it does not make sense to define a dedicated component (e.g. a proxy) to cope with federation issues; on the other hand, federation is more seen as a "pattern" which must be followed by all components of interest inside the ALMANAC Platform.

6. Information view

6.1 Overview of information flow

The information view is based on the abstractions of Smart City Resources, in terms of Semantic IoT Resources corresponding to Virtual Entities in the IoT ARM (Domain and Information models) (Bassi et al., 2013).

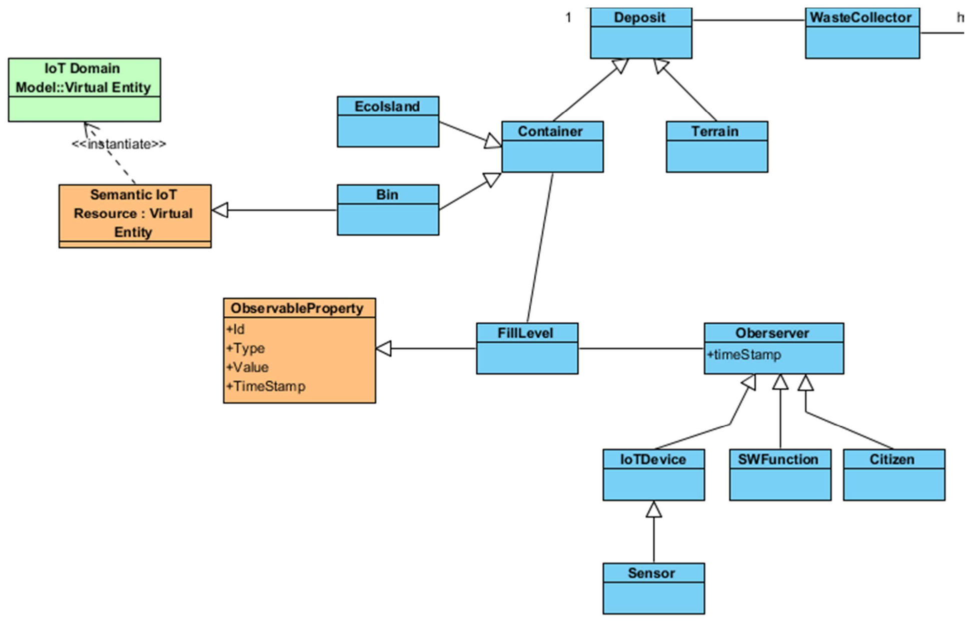


Figure 13 Example application domain model

The application domain model does not reside in the Data Management Layer which uses a resource model which represents resources with properties and data streams of observations of the state of these properties. The metadata for the resources and the properties refer to the concepts of the domain model.

Resources have a set of properties which have current state, a history and may have a prognosis. Measurements or observations of the state of a sensor or other resources are recorded with the time at which the value represented the state of the property. In addition, a time stamp for when this state observation was made is also recorded. A historical value may be updated with a revised value without replacing the first observation, and forecasted values may be represented in the same data stream as the observations that were actually made.

6.2 Description of information flows

Connecting concepts in the domain ontologies to the device ontologies is done at design time, when a new sensor is introduced in the ALMANAC system, or when a new Semantic IoT Resource is defined. The meanings of properties and their sets of observations (data points) are defined in the ontologies and represented and referred to at run time with the identifiers of the classes and instances in the ontology. Queries and rules use these identifiers to filter data without accessing the ontologies at run-time.

The devices, their properties and the sets of observations for a property and the metadata for them are made available to the ALMANAC system as REST resources. The addressing of the resources will be similar whether a SITR is accessed directly and the value is computed (e.g., GET http://datamanagement.almanac-project.eu/FillLevelSensor/Properties/FillLevel) or this value is fetched from the Storage Manager (e.g. GET http://storagemanager.almanac-project.eu/FillLevelSensor/Properties/FillLevel). Access control for external clients is handled before the request reaches the Data Management Framework.

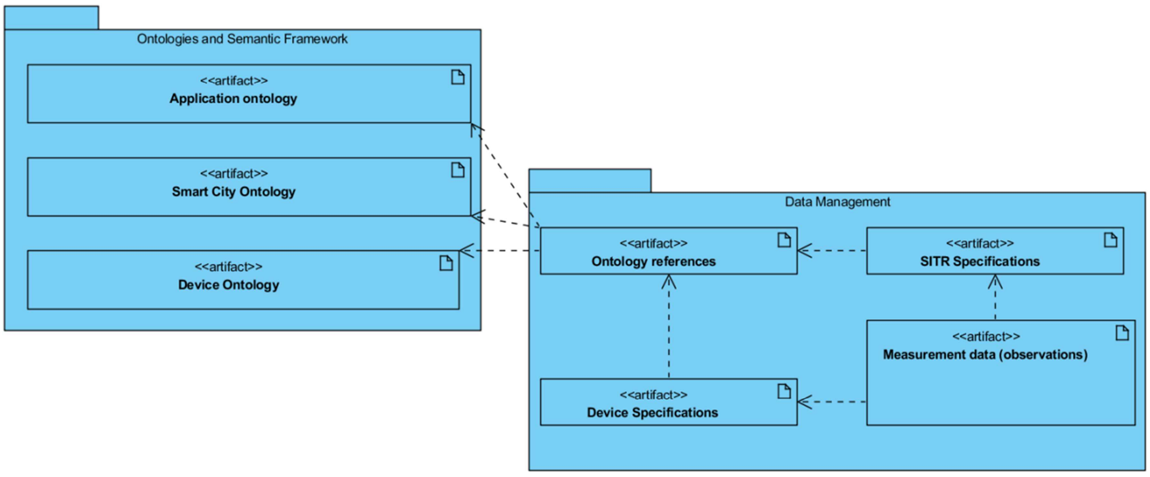


Figure 14 Data storage and domain model

External clients may also subscribe to changes in the properties of SITRs via the Event Manager. Like data requests, these subscriptions must also go through access control.

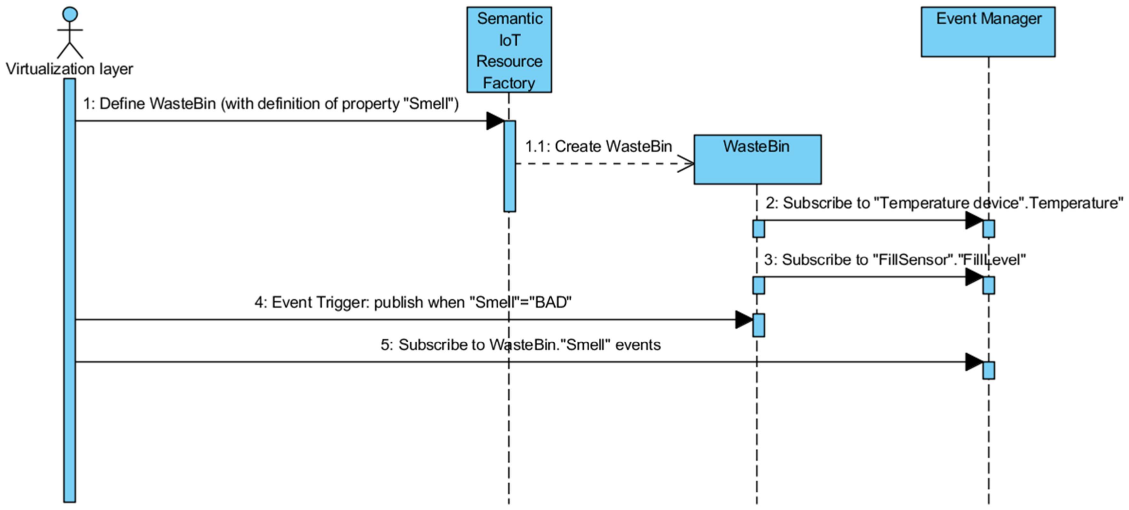

Virtualization layer
Semantic IoT Resource Factory
Event Manager
1: Define WasteBin (with definition of property "Smell")
1.1: Create WasteBin
WasteBin
2: Subscribe to "Temperature device".Temperature"
3: Subscribe to "FillSensor"."FillLevel"
4: Event Trigger: publish when "Smell"="BAD"
5: Subscribe to WasteBin."Smell" events

Figure 15 Defining a Semantic IoT Resource

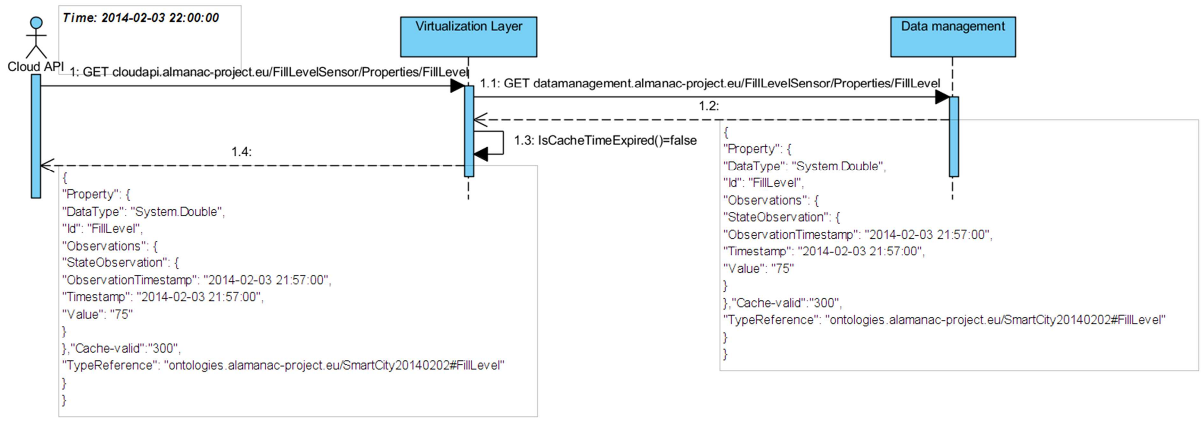
When a new sensor or device is added to the system, metadata such as caching information and value type (e.g. "xmlns:xs=http://www.w3.org/2001/XMLSchema" "xs:string", "xs:int") and references to classes in the ontologies are added to the Data Management Framework. A subscription to the data from the device is set up in the Event Manager so that the device publishes new data at the Event Manager; the Storage Manager is notified by the Event Manager and can store the data.

Data will continuously stream in to the Data Management framework via the Event Manager; directly requesting values from a device will be the exception and not the rule.

Sometimes the Virtualization Layer will query a device directly for data. The Virtualization Layer will not have to update the Data Management Framework with the data, as there should be a subscription set up for this already. The new sensor data will be entered into the Data Management Framework as a side effect of the Virtualization Layer requesting a new value since the Data Management Framework subscribes to property updates from the SCRAL.

6.2.1 Current or latest values

The current value for a property of an IoT Device or Semantic IoT Resource is the latest value for which the "Cache-valid" attribute still applies at the current system time. If the current value requested in the Virtualization Layer, caching instructions can be used to let intermediaries return the latest value stored in the Data Management Framework or request the current value at the device from the SCRAL.


Time: 2014-02-03 22:00:00
Cloud API
Virtualization Layer
Data management
1: GET cloudapi.almanac-project.eu/FillLevelSensor/Properties/FillLevel
1.1: GET datamanagement.almanac-project.eu/FillLevelSensor/Properties/FillLevel
1.2:
1.3: IsCacheTimeExpired()=false
1.4:

```
{
"Property": {
"DataType": "System.Double",
"Id": "FillLevel",
"Observations": {
"StateObservation": {
"ObservationTimestamp": "2014-02-03 21:57:00",
"Timestamp": "2014-02-03 21:57:00",
"Value": "75"
}
},"Cache-valid":"300",
"TypeReference": "ontologies.alamanac-project.eu/SmartCity20140202#FillLevel"
}
}
```

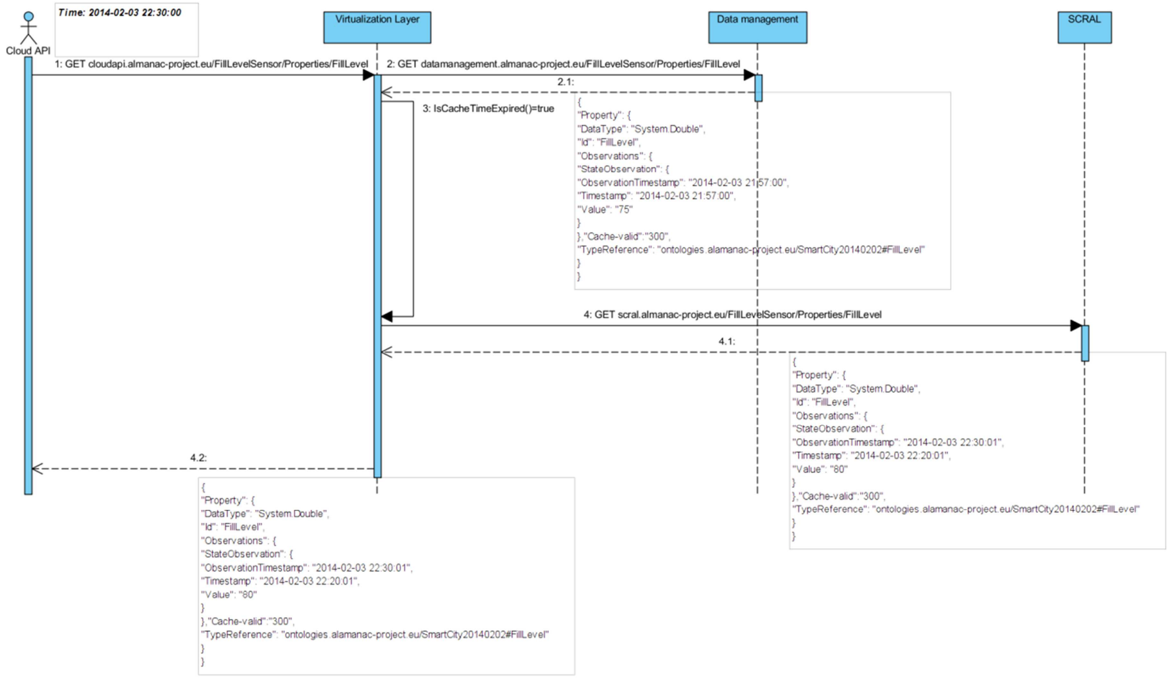
```
{
"Property": {
"DataType": "System.Double",
"Id": "FillLevel",
"Observations": {
"StateObservation": {
"ObservationTimestamp": "2014-02-03 21:57:00",
"Timestamp": "2014-02-03 21:57:00",
"Value": "75"
}
},"Cache-valid":"300",
"TypeReference": "ontologies.alamanac-project.eu/SmartCity20140202#FillLevel"
}
}
```

Figure 16 Requesting the latest, still valid, value for a property

If the Data Management Framework has received a value for this property and the specification of this property states that the value is to be considered still valid, there is no need to query the device directly. It is likely that the Storage Manager will use optimized storage and handling for the latest values for data streams so checking if the cached value can be used should not have a significant impact on performance.

```
{
"Property": {
"DataType": "System.Double",
"Id": "FillLevel",
"Observations": {
"StateObservation": {
"ObservationTimestamp": "2014-02-03 22:30:01",
"Timestamp": "2014-02-03 22:20:01",
"Value": "80"
}
},"Cache-valid":"300",
"TypeReference":"ontologies.almanac-
project.eu/SmartCity20140202#FillLevel"
}
}
```

When the REST resource that is the property of a SITR or Device is requested, the latest stored value will be returned as part of the representation by default. This way, if the value is considered current or not, may be determined by just inspecting the representation and compare the Timestamp plus the Cache-valid timespan to the current system time. The Virtualization Layer and the applications in the call chain before it may themselves decide if the data is "current enough"; this depends on requirements of the end-user application.


Time: 2014-02-03 22:30:00
Cloud API
Virtualization Layer
Data management
SCRAL
1: GET cloudapi.almanac-project.eu/FillLevelSensor/Properties/FillLevel
2: GET datamanagement.almanac-project.eu/FillLevelSensor/Properties/FillLevel
2.1:
3: IsCacheTimeExpired()=true
4: GET scral.almanac-project.eu/FillLevelSensor/Properties/FillLevel
4.1:
4.2:

```
{
"Property": {
"DataType": "System.Double",
"Id": "FillLevel",
"Observations": {
"StateObservation": {
"ObservationTimestamp": "2014-02-03 21:57:00",
"Timestamp": "2014-02-03 21:57:00",
"Value": "75"
}
},"Cache-valid":"300",
"TypeReference": "ontologies.alamanac-project.eu/SmartCity20140202#FillLevel"
}
}
```

```
{
"Property": {
"DataType": "System.Double",
"Id": "FillLevel",
"Observations": {
"StateObservation": {
"ObservationTimestamp": "2014-02-03 22:30:01",
"Timestamp": "2014-02-03 22:20:01",
"Value": "80"
}
},"Cache-valid":"300",
"TypeReference": "ontologies.alamanac-project.eu/SmartCity20140202#FillLevel"
}
}
```

```
{
"Property": {
"DataType": "System.Double",
"Id": "FillLevel",
"Observations": {
"StateObservation": {
"ObservationTimestamp": "2014-02-03 22:30:01",
"Timestamp": "2014-02-03 22:20:01",
"Value": "80"
}
},"Cache-valid":"300",
"TypeReference": "ontologies.alamanac-project.eu/SmartCity20140202#FillLevel"
}
}
```

Figure 17 Requesting the latest, but now invalid, value for a property

The Data Management layer will always return the latest stored value. If the latest value stored is current (i.e., cache time has not expired) or not will not be evaluated with each call to the Data Management Layer. The logic for this is not complex, but continuously evaluating this and possibly accessing the device directly may have an impact on performance and make the system less predictable.

6.2.2 Historical values

Historical values are stored in the Storage Manager. Ad-hoc queries are not handled by the Semantic IoT Resources directly but rather by the Storage Manager.

Sets of data points from devices or SITRs are REST resources that may be filtered by time or other attributes. Some aggregator functions, e.g., main, max and average, may be applied to these. This may be used for simple queries.

```
http://energyportal.cnet.se/DataPortalStorageManager/StorageManager/iotent
ities/SM-7:C9/properties/consumption/observations?take=5&after=2013-12-
03%2022:02:00

http://energyportal.cnet.se/DataPortalStorageManager/StorageManager/iotent
ities/SM-7:C9/properties/consumption/observations/max?take=5&after=2013-
12-03%2022:02:00
```

Aggregates using historical values are created using SITRs. An aggregate value is represented as a property on a STIR. These properties may be calculated when the value is requested, on the reception of an event or on specific intervals.

7. Perspectives

7.1 Security perspective

The security components offered by LinkSmart provide basic functionality to protect communication and entities from malicious activities. The ALMANAC project will build on these components to create its more sophisticated procedures as they require basic security services. We also improve these components as the processes that ALMANAC supports have stricter requirements like performance and accounting. In this subsection we briefly introduce these components; later deliverables will then more specifically specify the security perspective of the architecture.

7.1.1 Crypto Manager

Cryptographic operations are required for protecting the middleware communication from eavesdropping and modification, for authentication of devices and users. These components are merged in the so called Crypto Manager which serves various services like the creation and verification of digital signatures, encryption and decryption and generation and confidential storage of keys. The Crypto Manager will be improved to provide further cryptographic operations, mainly for symmetric key operations as they are better performing. We also add usability features to be accessed through the open APIs, which should help the users better understand the process of protecting their goods.

7.1.2 Trust Manager

The Trust Manager can be used to verify if a token offered by an entity is trustworthy. The decision if an entity is the legitimate owner of a key is based on trust models. The Trust Manager can implement any kind of trust model taking a token as input and returning a trust value as response. Two implementations are already available in LinkSmart, one for PKI and one for Web of Trust.

Development on the Trust Manager aims at improving the cooperation between Network Manager and Trust Manager. Network Manager is only able to use X.509 certificates to protect the communication and can therefore not exploit all the functionality the Trust Manager has to offer.

7.1.3 Communication Security Manager

To protect the message exchange between entities security protocols and cryptographic operations have to be applied. This is a combination of the services offered by Crypto Manager, Trust Manager and a protocol specification. There are different Communication Security Managers which can be linked to the Network Manager. They provide different protocol implementations and fit for other requirements. For example if there is only rare message exchange it is better to use pure asymmetric encryption without sessions but for busy channels the resources used to set up a session are negligible compared to the performance gained by using symmetric security.

7.2 Scalability perspective

The Data Management Framework achieves scalability by

- Reducing message sending by leveraging the LinkSmart publish-subscribe infrastructure (Event Manager) and network transparency (Network Manager), letting the Event Manager partition and delegate the calling of subscribers.
- Delegating aggregation and rule evaluation for sensor data to Semantic IoT Resources, data processing services distributed on different nodes in the ALMANAC cloud.
- Storing data in a distributed database for both sensor data and aggregated data. Short-term data needed for real-time processing may be stored directly on the preprocessing nodes.
- Using caching to replace actual calls to IoT Devices or Semantic IoT Resources. REST calls will use the standard context-expires header or redirection to stored data if the resource instance is not accessible.

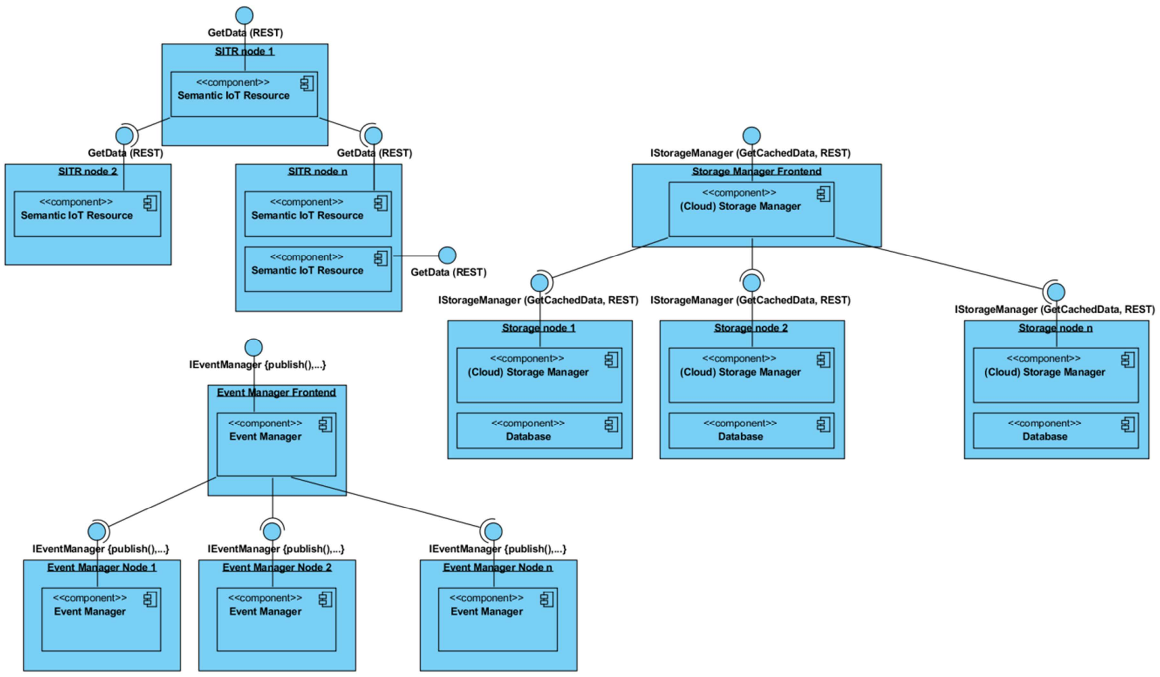
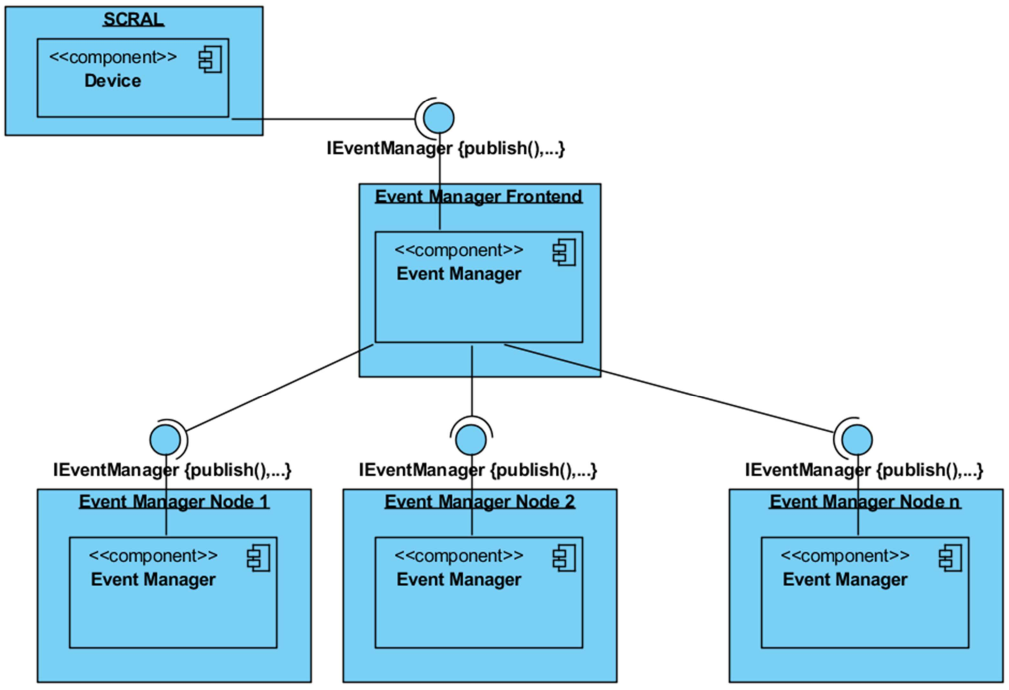


Figure 18 Example of distributed components in the Data Management Framework

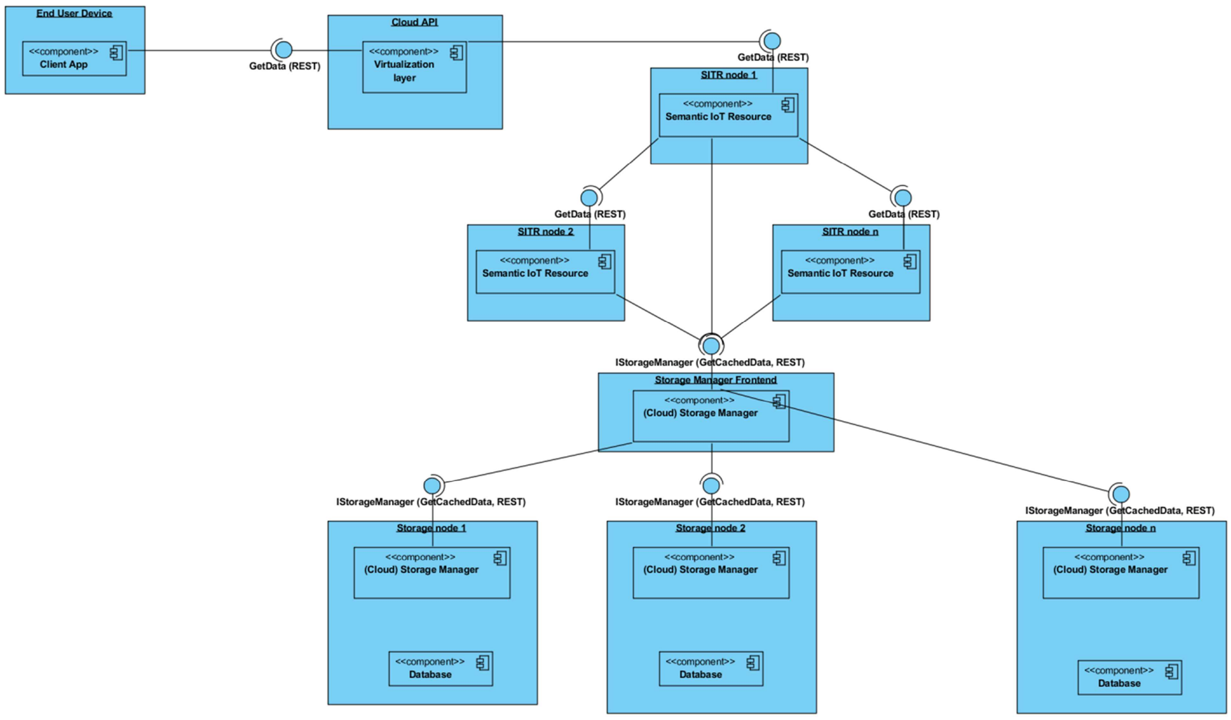
7.2.1 Event Manager

The data from an IoT Sensor or IoT Device may be requested by a large number of applications and Semantic IoT Resources. The data also has to be transferred to long-term storage. To minimize the number of connections, and make the load on the IoT Device smaller in the case of a huge number of subscribers, the LinkSmart Event Manager will be used. It will be extended with the necessary functionality to let several Event Managers co-operate and use load-balancing to service publishers, corresponding to Event Processing Agents (EPAs) in an event processing network, and subscription partitioning for publishing. The Network Manager will provide network transparence and uniform lookup and addressing of IoT Devices and Event managers.



7.2.2 Semantic IoT Resources

Semantic IoT Resources will handle aggregation of data and react to data patterns and raise events. These will represent domain entities or services needed by the end user applications. They may also be connected in a pipes-and-filters style to create more complex sets of aggregated data. BY distributing these on different nodes in the system, adding new nodes if necessary, complex aggregates may be constructed, executed efficiently, and re-used.


End User Device
<<component>> Client App
GetData (REST)
Cloud API
<<component>> Virtualization layer
GetData (REST)
SITR node 1
<<component>> Semantic IoT Resource
GetData (REST)
SITR node 2
<<component>> Semantic IoT Resource
GetData (REST)
SITR node n
<<component>> Semantic IoT Resource
IStorageManager (GetCachedData, REST)
Storage Manager Frontend
<<component>> (Cloud) Storage Manager
IStorageManager (GetCachedData, REST)
Storage node 1
<<component>> (Cloud) Storage Manager
<<component>> Database
IStorageManager (GetCachedData, REST)
Storage node 2
<<component>> (Cloud) Storage Manager
<<component>> Database
IStorageManager (GetCachedData, REST)
Storage node n
<<component>> (Cloud) Storage Manager
<<component>> Database

7.2.3 IoT-cloud enabled Storage Manager

The data generated by sensors and resources will be stored by the Storage Manager service. Data will be distributed by the Storage Manager over the nodes in the ALMANAC system and optimized for the scenarios in ALAMANAC where short-term storage and long term storage are used in different scenarios. Storage will be transparent to both devices and resources producing data and resources and smart city consumers. Data is accessed by REST-based interfaces with URI structure and semantics like the other interfaces in the Data Management Framework, making call forwarding and caching simple.

7.2.4 Caching

Each property of a resource (e.g., a data stream from a temperature sensor or an aggregated value from a SITR) will have an attribute specifying the default amount of time the data stays valid in the cache. Within this time period, the ALMANAC system will not query the device directly for data but instead use the latest value from the Data Management Layer. The cache timespan information can be supplied with the Content-Expires header to let clients and intermediaries cache the information.

If the content-expires header indicates that an intermediate cache may be used, the request need not go any further. If the resource or device is not accessible within a set time span, the latest stored data will be used. Since getting the latest values for a device or resource is a frequently occurring case, the Storage Manager may also use a dedicated database to handle these requests. This way, there will be a minimum wait time to get the latest data from a device or resource.

To let the client specify desired server behavior, the HTTP Cache-Control and Expect headers can be used and possibly extended (this is allowed in the specification[3]) to handle requirements from the

[3] http://tools.ietf.org/html/rfc2616

client, e.g. that the response should be returned within a certain time period, that cached values always should be used or that cached values always never be used.

8. 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.

8.1 Introduction of LinkSmart

The LinkSmart middleware has been developed in the European project Hydra[4]. 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 and (Energy-aware) Smart Homes (Al-Akkad et al., 2009), (Eikerling et.al, 2009), (Jahn et al., 2010), (Reiners et al., 2009).

8.2 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 19 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.

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

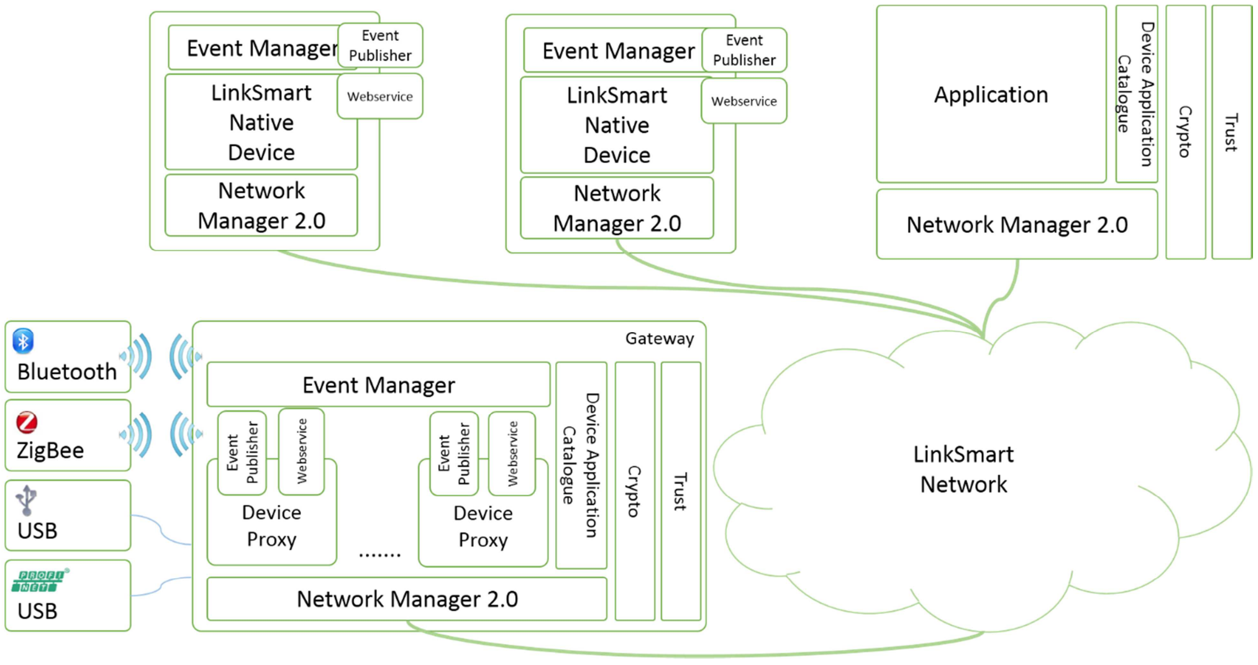


Figure 19 LinkSmart Example Concrete Deployment

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 20, 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.

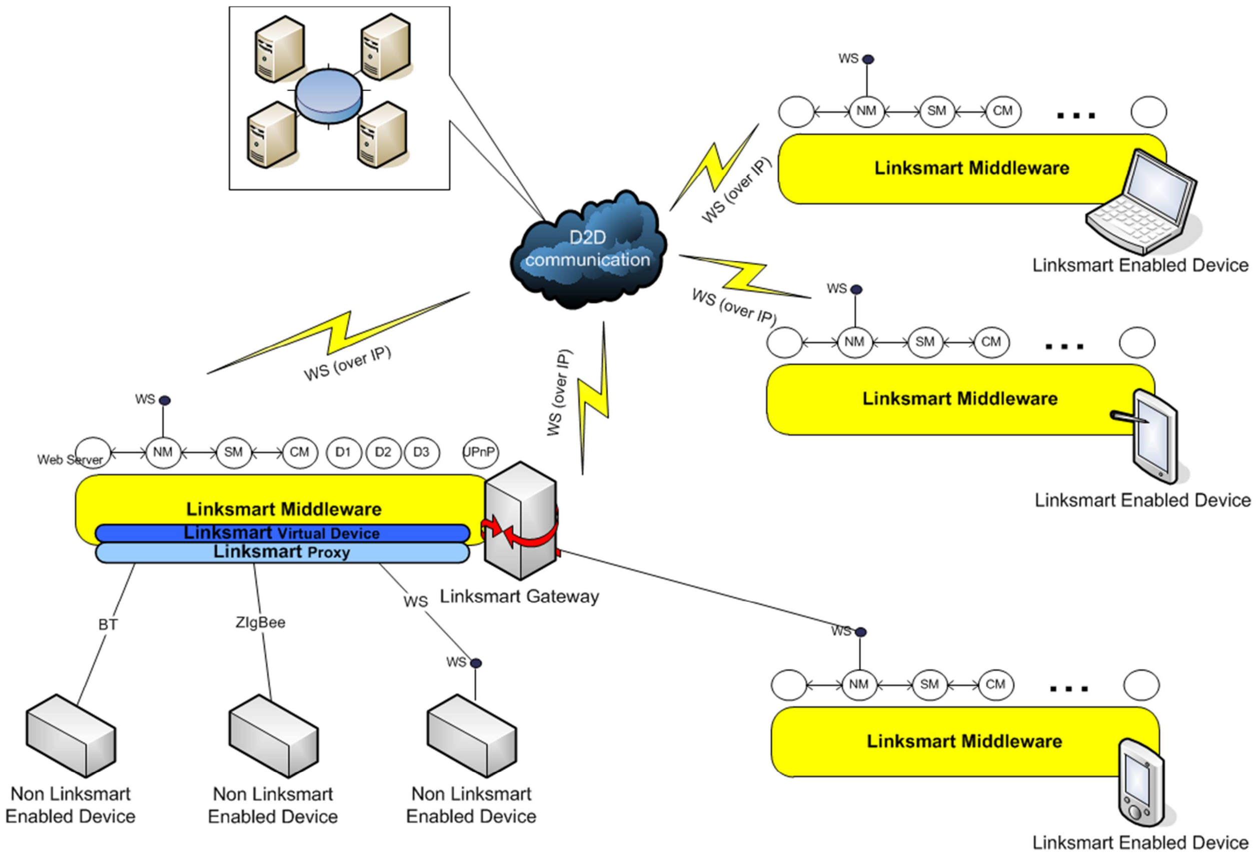


Figure 20 LinkSmart Example Device Network

9. Technical use case instantiation

This section details some selected use cases to clarify how the ALMANAC SCP behaves to deliver some specific features of interest. Technical use cases described in the following can be seen as more generalized and low-level use cases, which are normally employed across several application scenarios (e.g. waste management, water supply, civil engagement).

9.1 Technical use case: Data collection and rule based notification

This scenario describes how aggregation of real-time data flows can be configured to derive additional information and how an end user application may describe the conditions for when it wants to be notified about changes in this information. It also describes the way that the system does rule-based notification events to clients; Smart City Apps or Semantic IoT Resources.

We have identified three categories or roles for ALMANAC developers.

- IoT Device-developers, who develop Virtualization Layer components interfacing with the ALMANAC system.
- Smart city object-developers, who develop virtual entities / Semantic IoT Resources that gather data and aggregates the data into new information.
- Smart city app-developers, who uses the data published from IoT Devices and Semantic IoT Resources to build end user functionality for the smart city.

To explain the use case, we need to provide instantiations for use cases both at design time and run time.

9.1.1 Design-time scenario

The provider of the Smart City defines or uses an application domain model / ontology with the relevant concepts/classes for the application at hand. This model may also be extended by IoT Device-developers.

The supplier of an IoT Device, a fill-level sensor on a waste bin in this case, "plugs" the sensor into the ALMANAC system by using the concepts in the ontology to describe the data reported by the IoT Device. This relates the sensor data to the observable property (e.g. " FillLevel") in the application domain model (with applicable reliability, relations to other properties, etc. also modeled).

The data reported by the SCRAL (IoTDevice) for the property "FillLevel" is now:

1. Possibly annotated with a reference to the type "FillLevel" if this is supported at the IotDevice/Virtualization layer level and
2. Referenced in the instance ontology to have the type "FillLevel".

This is done via the Cloud API and the Virtualization Layer.

The smart city object developer defines a Semantic IoT Resource for "WasteBin" (see Figure 15 Defining a Semantic IoT Resource) and uses the ontology and the Cloud API to find the property type "FillLevel" and the installed IoT Device (sensor) reporting "FillLevel" and then either:

a. let the virtual entity subscribe to data from the specific sensor or
b. let the virtual entity subscribe to data from all IoT devices reporting "FillLevel" in a specific location.

The data from "FillLevel" is stored in the property "FillLevel" of "WasteBin". The developer also provides a rule for how this data should be processed together with data from a weather service to update a new property, "Smell", of "WasteBin". This new property will also be added to the ontology.

The smart city app developer asks the Cloud API for the current value of the "Smell" property for "WasteBin". The value of this property is then calculated. If the smart city app wants to be notified of issues or specific events, a publish-subscribe model is used, and the smart city app subscribes to events for changed smell values.

Run-time scenario

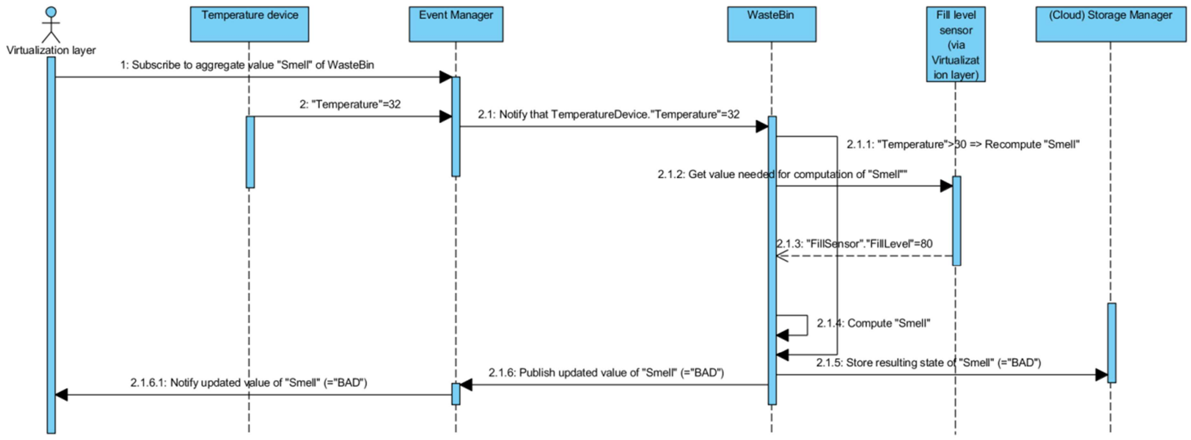

Virtualization layer
Temperature device
Event Manager
WasteBin
Fill level sensor (via Virtualization layer)
(Cloud) Storage Manager
1: Subscribe to aggregate value "Smell" of WasteBin
2: "Temperature"=32
2.1: Notify that TemperatureDevice."Temperature"=32
2.1.1: "Temperature">30 => Recompute "Smell"
2.1.2: Get value needed for computation of "Smell""
2.1.3: "FillSensor"."FillLevel"=80
2.1.4: Compute "Smell"
2.1.5: Store resulting state of "Smell" (="BAD")
2.1.6: Publish updated value of "Smell" (="BAD")
2.1.6.1: Notify updated value of "Smell" (="BAD")

Figure 21 Calculating the aggregate value "Smell" when a new temperature value is received

The waste bin sensors report data to the SCRAL/IotDevices or data is read from the sensors by the SCRAL/IoT Devices. The data is reported into the ALMANAC Cloud via an Event Manager. The event manager forwards this data to the semantic IoT resource.

When the Semantic IoT Resource receives new temperature data, it processes the data according to its internal rules, in this case the temperature is above the threshold for which the "Smell" property should be re-calculated. A current value for the "FillLevel" is needed, so this is fetched. The "Smell" property is re-calculated, and stored in the Storage. Since the SITR is configured to notify others when its properties change, it will call the Event Manager.

The end user applications that have subscribed to changes in the SITR property "Smell" are notified by the Event Manager.

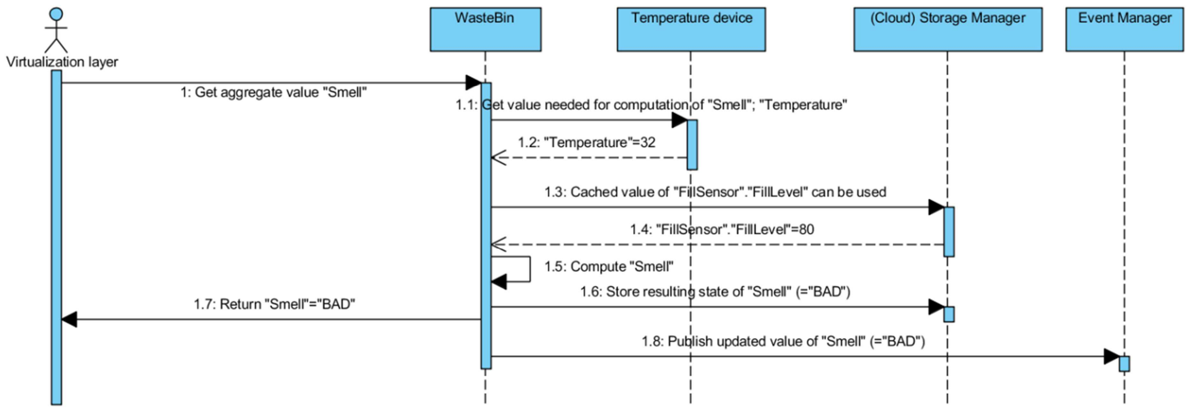

Virtualization layer
WasteBin
Temperature device
(Cloud) Storage Manager
Event Manager
1: Get aggregate value "Smell"
1.1: Get value needed for computation of "Smell"; "Temperature"
1.2: "Temperature"=32
1.3: Cached value of "FillSensor"."FillLevel" can be used
1.4: "FillSensor"."FillLevel"=80
1.5: Compute "Smell"
1.6: Store resulting state of "Smell" (="BAD")
1.7: Return "Smell"="BAD"
1.8: Publish updated value of "Smell" (="BAD")

Figure 22 Calculating "Smell" on request

An alternative scenario is that the SITR is queried for the property "Smell" by the Virtualization Layer. Since the "Cache-valid" attribute of this property indicates that I should be re-calculated, the values need to compute "Smell" are requested from other components and the value is re-calculated. The new value is stored in the Storage Manager and the new value is returned to the Virtualization layer.

9.2 Technical use case: Historic data aggregation and reasoning

When data is reported from IoT Devices or generated by SITRs, the data may be stored locally to allow rules to be able to process historical data. However, local cache is typically for short time periods. This scenario describes how data for sensors is stored and accessed in the system in a scalable way and how historical data is accessed for querying and reasoning.

The storage manager provides a limited query interface with aggregation functions for sorting, time span selection, and simple scalar functions such as min, max, average. It will also support map-reduce like functionality to write more complex queries, however, the main construct for creating aggregated data is the SITR. It can be configured to analyse historical data and publish a stream of aggregated data.

Design time scenario

The supplier of an IoT Device, the proxy for a Fill-level sensor on a waste bin in this case, "plugs" the sensor into the ALMANAC system. This results in a subscription for data from this device by the Remote Storage Manager. An instance of the Remote Storage Manager is be assigned for this device according to the internal partitioning scheme.

To get the lowest fill average level this month, the Smart city app developer defines a set of SITR that implements this functionality and reports the desired result as the property "LowestMonthlyAverageFillLevel" (or a set of properties). The query can be executed on-demand or at scheduled intervals.

Run-time scenario

The waste bin sensors report data to the Adaptation Layer. These send the data on to the event manager, which will pass the data on to the subscribers, including the Storage Manager where the data will be stored.

For aggregation of historical data, the predefined SITR is used. When the end user application queries the lowest monthly fill level, it requests the necessary historical data from the Storage Manager and sends the result back. The updated SITR property "LowestMonthlyAverageFillLevel" is also stored by the Storage Manager.

9.3 Technical use case: Smart City Resource registration

The use case shown in Figure 23 outlines how a new physical device joins the network and it is discovered and registered by the SCRAL.

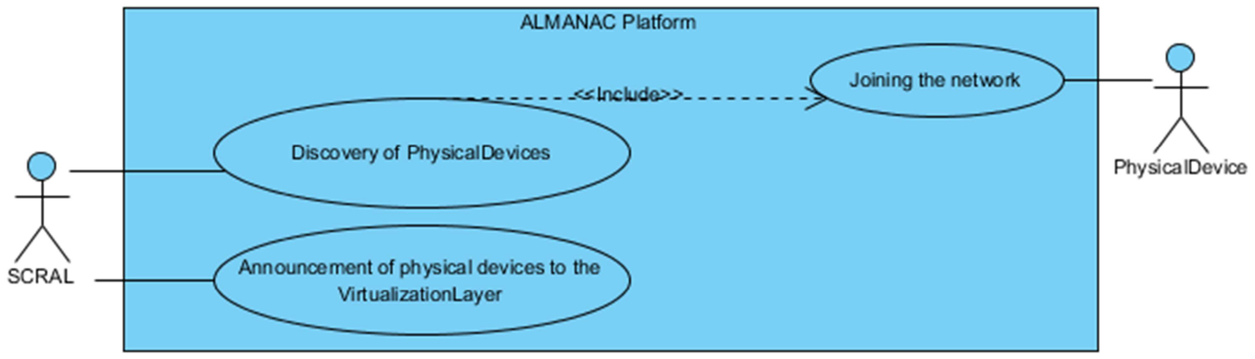

ALMANAC Platform
<<Include>>
Joining the network
Discovery of PhysicalDevices
Announcement of physical devices to the VirtualizationLayer
SCRAL
PhysicalDevice

Figure 23 Use case: device discovery and adaptation

As described in the sequence diagram in Figure 24, the SCRAL is able to discover a new physical device and to collect a list of its capabilities (if available) using one of its dedicated *SCRALDrivers*. After the discovery phase is completed, the device is exposed as a REST resource (device adaptation). Consequently, the SCRAL signals and registers the discovered device to the *VirtualizationLayer* which interacts with the *DataManagement* component to create a new or get (if already existing) a *DataManagamentFlowIdentifier.*

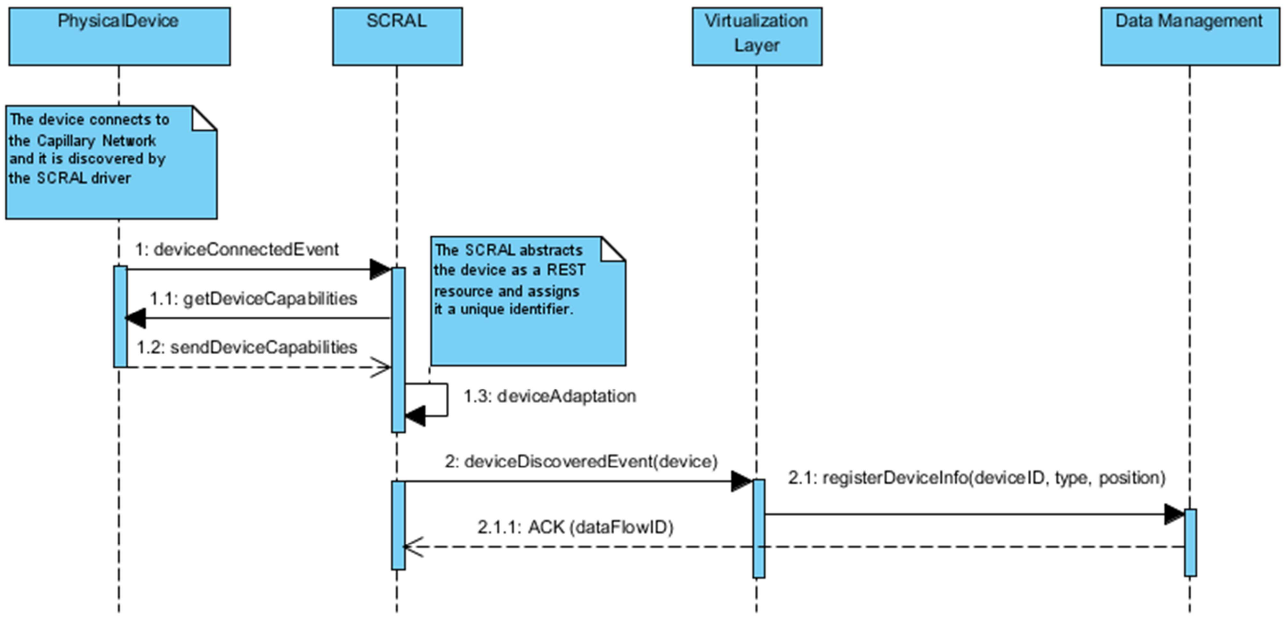

PhysicalDevice
SCRAL
Virtualization Layer
Data Management
The device connects to the Capillary Network and it is discovered by the SCRAL driver
1: deviceConnectedEvent
1.1: getDeviceCapabilities
1.2: sendDeviceCapabilities
The SCRAL abstracts the device as a REST resource and assigns it a unique identifier.
1.3: deviceAdaptation
2: deviceDiscoveredEvent(device)
2.1: registerDeviceInfo(deviceID, type, position)
2.1.1: ACK (dataFlowID)

Figure 24 Sequence diagram: device discovery and adaptation

After this sequence is completed, the SCRAL can feed autonomously raw data from device into the Data Management Layer.

9.4 Technical use case: end-user data access

In Figure 25, a citizen-centric use case is presented, describing how a generic end-user (e.g. a citizen) is authenticated by the ALMANAC platform and gets access to his personal profile and to his personal data. In this scenario, ALMANAC matches between the user identity and his personal consumption monitoring devices (e.g. the water meter provided by the Water utility). Thanks to this mapping the user can retrieve his own consumption and benchmark it against the average consumption of the city.

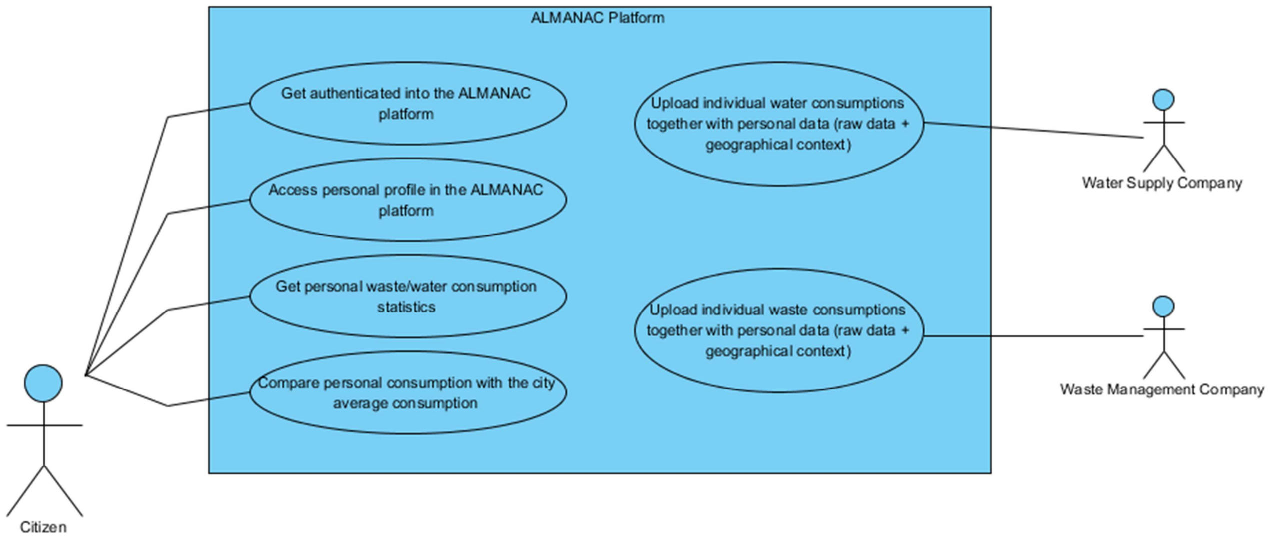

ALMANAC Platform
Get authenticated into the ALMANAC platform
Access personal profile in the ALMANAC platform
Get personal waste/water consumption statistics
Compare personal consumption with the city average consumption
Upload individual water consumptions together with personal data (raw data + geographical context)
Upload individual waste consumptions together with personal data (raw data + geographical context)
Water Supply Company
Waste Management Company
Citizen

Figure 25 Use Cases: end-user API-driven data access and comparison

Since this use-case is rather complex, it has been divided in a number of sub-cases described in the following.

9.4.1 User Authentication into the ALMANAC Platform

The process of user authentication is shown in Figure 26. The citizen sends his credentials through the Cloud-based Open APIs to the Identity Manager, which performs a credential check and replies sending back an authorization token to the user once his credentials have been verified.

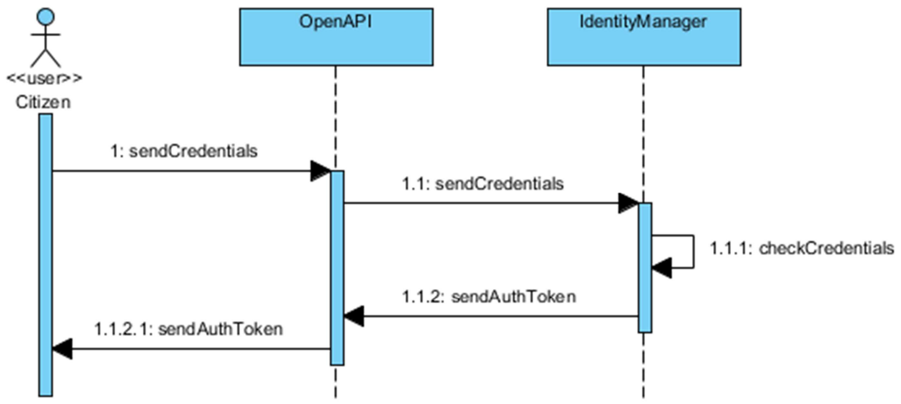


Figure 26 Sequence Diagram: user authentication into the ALMANAC platform

9.4.2 Citizen Personal Profile access

Analogously, once a user has been authenticated is able to access his personal profile, where his personal data is shown through a dashboard-like interface. This sequence is depicted in Figure 27.

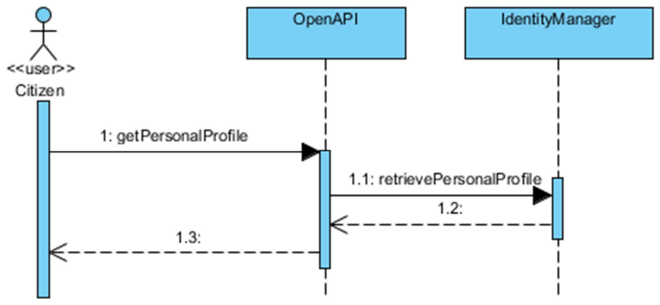
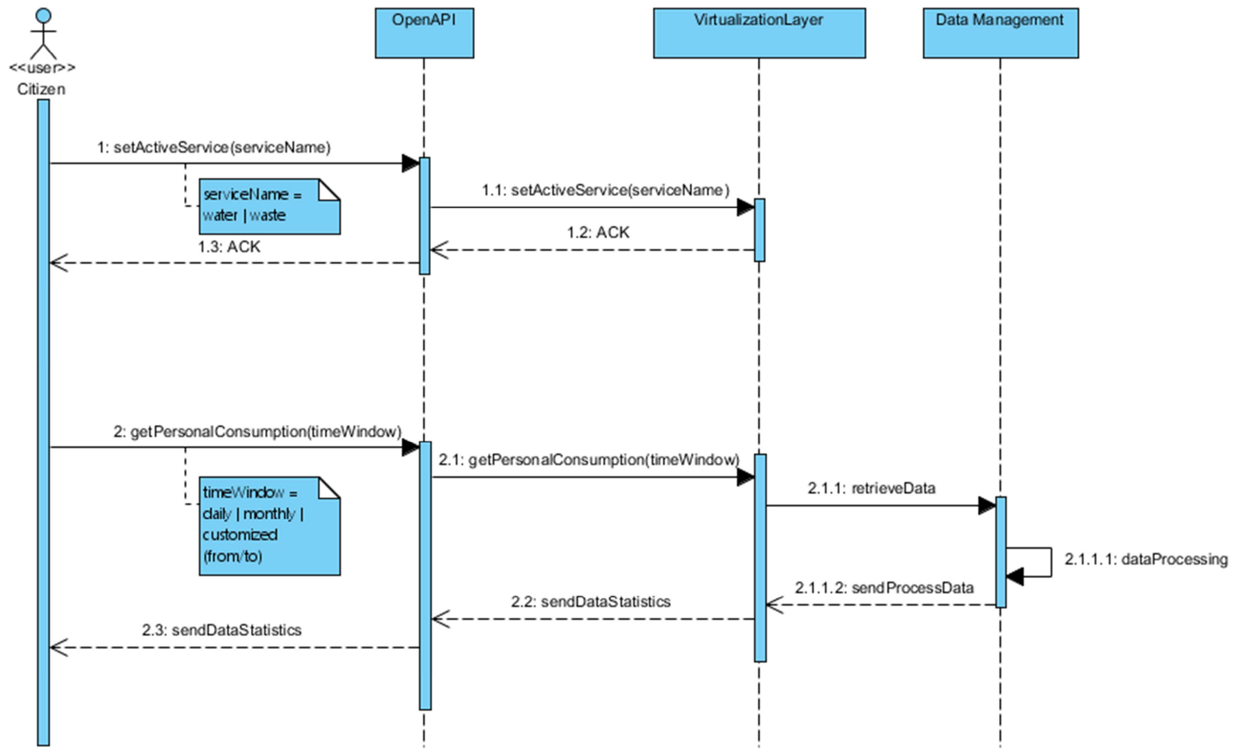


Figure 27 Sequence Diagram: citizen access his personal profile

9.4.3 API-driven Citizen Data Access

In some scenario, the citizen might need to access more specific information derived from some IoT data collection process e.g. consumption statistics, consumptions history, etc. Such data is not available through his personal profile; however he can query the *VirtualizationLayer* though the cloud-based *OpenAPIs* to retrieve the specific collection of data from the Data Management component which matches the metering devices corresponding in turn to his identity. The V*irtualizationLayer* requests the *DataManagement* module the corresponding data which is processed and sent back to the user.



<<user>>
Citizen
OpenAPI
VirtualizationLayer
Data Management
1: setActiveService(serviceName)
serviceName = water | waste
1.1: setActiveService(serviceName)
1.2: ACK
1.3: ACK
2: getPersonalConsumption(timeWindow)
timeWindow = daily | monthly | customized (from/to)
2.1: getPersonalConsumption(timeWindow)
2.1.1: retrieveData
2.1.1.1: dataProcessing
2.1.1.2: sendProcessData
2.2: sendDataStatistics
2.3: sendDataStatistics

Figure 28 Sequence Diagram: user access to personal consumptions statistics data

Figure 28 does not show how data is fed into the Data Management component. In general this is ensured by an autonomous data collection process always running across the SCRAL and the Data Management component after a successful device registration.

9.4.4 API-driven Citizen Consumption Benchmarking

Another possible action the citizen can perform, featured by the ALMANAC platform, is the possibility to benchmark his personal consumptions against the average consumption of the entire city. This process is depicted in the sequence diagram from Figure 29.

In this case, the citizen uses the cloud-based open APIs to obtain the comparison of his personal data with the city data, which is possible thanks to the *DataManager* that sends back the corresponding compared data to the user through the *VirtualizationLayer* and subsequently through the *OpenAPIs.* The citizen is able to specify various fields, to narrow down the comparison to a desired area or time interval.

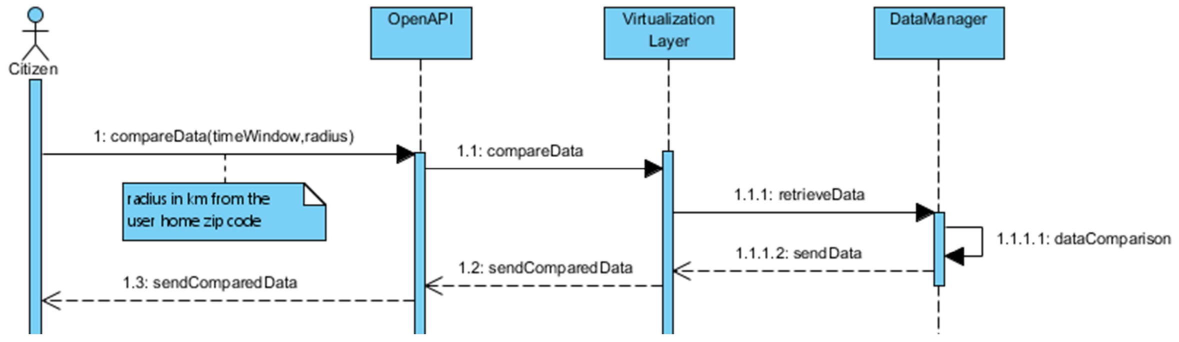


Figure 29 Sequence Diagram: citizen consumptions benchmark

9.4.5 Extended Use Case: API-driven actuation

While ALMANAC, and more in general Smart City application, do not have a strong focus on control and actuation issues, this case might happen in some specific professional scenarios e.g. in the water domain. The use case detailed in Figure 30 details how actuation could be handled in the ALMANAC architecture.

A generic user (e.g. a Citizen) with adequate access rights can query the Open Cloud-based API to retrieve the list of Actuators matching some particular properties e.g. a geographical location, or an address). The request is served by the Virtualization layer, which responds with an Actuator List eventually interrogating one or more SCRAL instances if a data synchronization is needed.

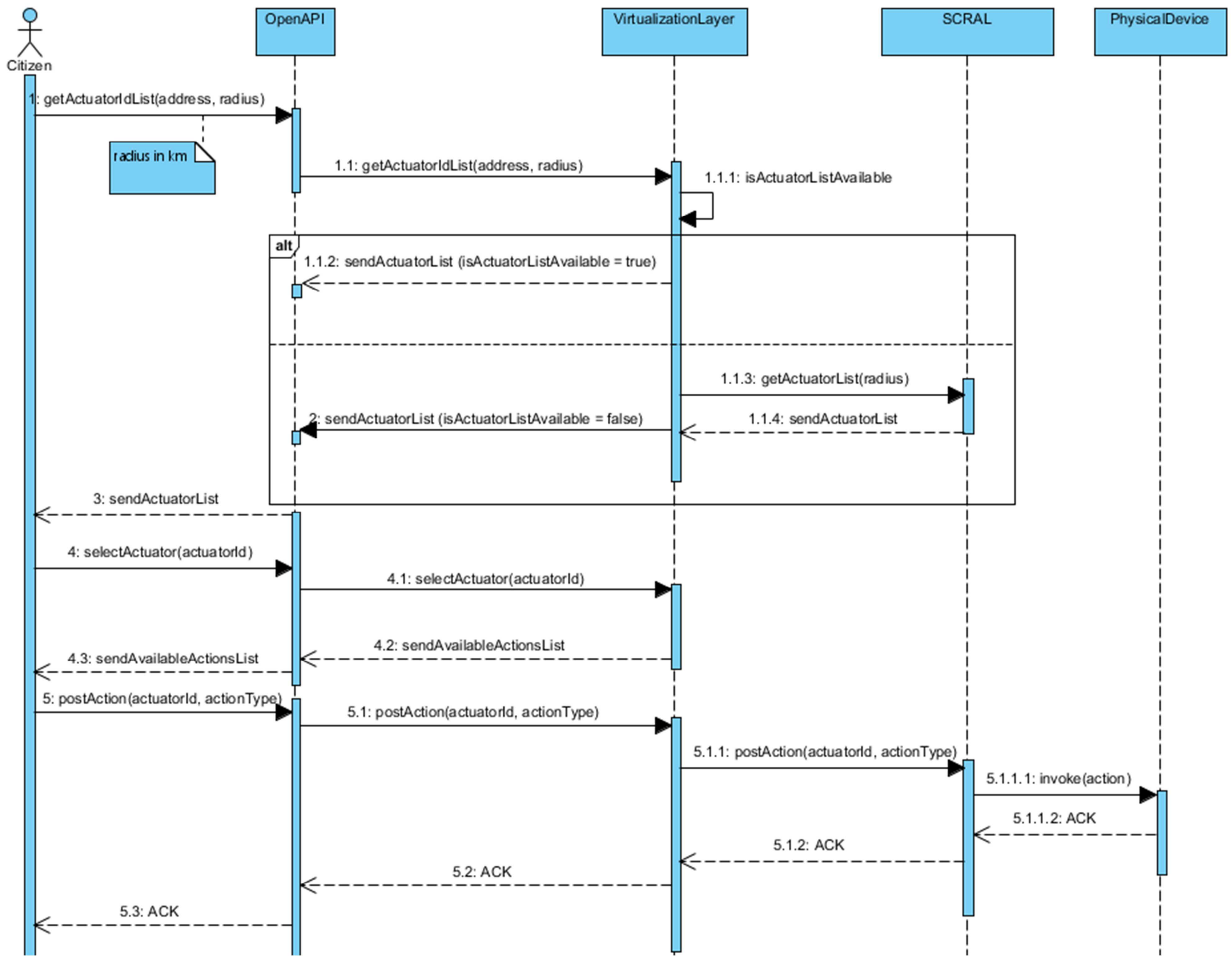

Citizen
OpenAPI
VirtualizationLayer
SCRAL
PhysicalDevice
1: getActuatorIdList(address, radius)
radius in km
1.1: getActuatorIdList(address, radius)
1.1.1: isActuatorListAvailable
alt
1.1.2: sendActuatorList (isActuatorListAvailable = true)
1.1.3: getActuatorList(radius)
2: sendActuatorList (isActuatorListAvailable = false)
1.1.4: sendActuatorList
3: sendActuatorList
4: selectActuator(actuatorId)
4.1: selectActuator(actuatorId)
4.2: sendAvailableActionsList
4.3: sendAvailableActionsList
5: postAction(actuatorId, actionType)
5.1: postAction(actuatorId, actionType)
5.1.1: postAction(actuatorId, actionType)
5.1.1.1: invoke(action)
5.1.1.2: ACK
5.1.2: ACK
5.2: ACK
5.3: ACK

Figure 30 Sequence diagram: API actuation example

The user can thus browse the features of the actuator and, having selected the action of interest, he can perform it through a postAction call delivered to the SCRAL and ultimately to the PhysicalDevice.

10. References

(IEEE1471, 2000)	IEEE Standard 1471-2000 (2000), IEEE Recommended Practice for Architectural Description of Software-Intensive Systems
(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
(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.
(ETSI, 2011)	ETSI, TS. (2011), "102 690:" Machine-to-Machine communications (M2M)." Functional architecture.
(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
(Madsen et al, 2005)	P. Madsen, E. Maler (2005), SAML V2.0 Executive Overview. Available: https://www.oasis-open.org/committees/download.php/13525/sstc-saml-exec-overview-2.0-cd-01-2col.pdf
(Reiners et al., 2009)	Reiners, R., Zimmermann, A., Jentsch, M., Zhang, Y. (2009). „Automizing home environments and supervising patients at home with the hydra middleware: application scenarios using the hydra middleware for embedded systems".CASTA '09 - Proceedings of the first international workshop on Context-aware software technology and applications, pages 9-12
(Rissanen, 2010)	Rissanen, Erik. (2010), eXtensible Access Control Markup Language (XACML) version 3.0 (committe specification 01). Technical report, OASIS, http://docs. oasisopen. org/xacml/3.0/xacml-3.0-core-spec-cd-03-en. pdf.
(Rozanski & Woods, 2005)	Rozanski, N., Woods, E. (2005), "Software Systems Architecture", ISBN: 0321112296

1. Annex: Component descriptions

Component Name

Capillary Network Gateway

Description

The capillary network gateway is an important item of the capillary network. It has two interfaces: short range radio (169 MHZ – WMBus), mobile IP (3G/4G).

Services

To connect sensors on the field with a short range radio transmitter with IP/Internet Network. The IP/Internet Network can then transport the data to the servers in the cloud.

Interfaces:

- The capillary network gateway shall interface the Middleware of the M2M Platform (LinkSmart/Adaptation Layer).
- The capillary network gateway shall interface the TX modules of the sensors (smart meters and others).

Dependencies

Might have a dependency to the sensors chosen by WP2.

Related Task

Task 4.1: Capillary Networks

Smart City Platform Layer

- Multiservice Federated Networks

API Accessibility

Will any services of this component be accessible through the open APIs?

Yes.

Component Name

NetworkManager

Description

The core component for enabling communication within LinkSmart. All devices and software that wishes to enter the network requires an associated NetworkManager. The NetworkManager directly interfaces with IdentityManager, BackboneRouter, Backbones and CommunicationSecurityManagers.

Services

Provides naming and addressing capabilities, network federation by switching between Backbones and CommunicationSecurityManagers. Routing capabilities will be added as well as more advanced discovery mechanisms.

Dependencies

None

Related Task

T4.2

Smart City Platform Layer

- Communication Management Framework

API Accessibility

- Service Discovery
- Network Management (debugging)
- Service Registration

Component Name

Policy Manager

Description

This distributed component will check at fixed points in the network and software whether required security policies are met. It will enable the overall security enforcement of information flow and will have advances features regarding anonymization of information. When external information sources are made available users will have the possibility to define rules with this component.

Services

Compose policies regarding data

Authenticates and authorizes data flows

Controls data flows whether they meet policies

Does automatic transformation of data

Dependencies

NetworkManager, TrustManager

Related Task

T3.4

Smart City Platform Layer

- Security and Privacy Framework

API Accessibility

- Compose Access Policies

Component Name

Smart City Resources Adaptation Layer (SCRAL)

Description

The SCRAL, typically running inside the gateways (i.e. co-located inside or close to the LinkSmart proxy), is the entry point for physical world data from sensors.

The high-level idea of the SCRAL is to provide a "pattern" for integrating heterogeneous devices inside proxies, supporting self-description, discovery, annotation, etc. of sensors and actuators.

Services

The SCRAL exposes an API which supports:

- Management of heterogeneous IoT device "drivers" (i.e. "integration layers") running on the gateway
- Self-description of available devices and the data they can provide
- …

Dependencies

Dependencies toward the virtualization and semantic layer. On the device side. Dependencies with all the specific integration layers of devices to be supported and also with self-description capabilities in M2M standards.

Related Task

Task T5.1

Smart City Platform Layer

- Adaption Layer

API Accessibility

Will any services of this component be accessible through the open APIs? Yes.

Component Name

TrustManager

Description

The TrustManager enables the definition of trust models and the reasoning about entities according to these models. It will enable users to define their personal trust metric regarding their data and to reason about multiple heterogeneous types of trust information. By using this component, non security experts will have the possibility to easily create secure applications.

Services

Definition of trust information to be used, way it is acquired and model it is enforced by. Provides an understandable visualization of the running trust management framework and also enables the reasoning about unexpected entities or trust evidences.

Dependencies

PolicyManager

Related Task

T3.4

Smart City Platform Layer

- Security and Privacy Framework

API Accessibility

- Compose trust model
- Compose trust relationships
- Query trust data

Component Name

EPA – Event Processing Agent

Description

The EPA provides functionality for semantic event processing using a combination of ontology-based event descriptions and rules for processing. Multiple instances of EPAs can be used to establish an event processing network (EPN) enabling distributed and layered event management. The primary functionalities include:

- Receiving events (from different protocols).
- Event rule processing, for instance implementing the filtering, transformation and aggregation of events.
- Semantic enhancement of events, e.g., by static knowledge in the rules or based upon previous events, or, based on semantics derived from the event ontology.
- Providing event persistency and storage, for logging and future analysis.
- Routing events to the next level in the Event Processing Network.
- Triggering scheduled rules.

Services

EPA configuration, e.g.,

- peer end-point selection
- mode (pub-sub / push-pull)

Receive/dispatch event

Ontology event resolution

Event rule definition and resolution

Dependencies

- Event Manager
- Ontology Manager
- Services Orchestration Manager

Related Task

T6.3, T6.2

Smart City Platform Layer

- Data Management Framework
    - Event Filtering and Context Management
    - Event Management and Reasoning
    - Information Aggregation, Storage, and Mining
- Ontologies and Semantic Representation Framework

API Accessibility

Services to be included in the SCP event management SDK subset.

Component Name

Event Manager

Description

The Event Manager provides publish/subscribe functionality, i.e., the ability for publishers to send a notification to multiple subscribers while being decoupled from them (in terms of, e.g., not holding direct references to subscribers). The specific variant of publish/subscribe implemented is topic-based publish/subscribe where key/value pairs represent events. With this approach, any subscriber or publisher defines a topic simply by executing the "publish" or "subscribe" actions.

Source: LinkSmart

Services

The Event Manager interface provides the methods for handling the subscriptions, publications, notifications and storage of events.

- Subscription support allowing clients to subscribe to published events via a topic-based publish/subscribe scheme
- Publication support allowing client to publish event on topics
- Routing events to subscribed clients
- Event Core manages persistent subscriptions, publication to subscription matching etc.
- Event retry queuing
- Prioritization of events and subscribers
- Failed event storage

Dependencies

Interface to Link Smart Network Manager (e.g., broadcast-, multicast-, or gossiping-based dissemination).

Related Task

T6.3

Smart City Platform Layer

- Data Management Framework
  - Event Filtering and Context Management
  - Event Management and Reasoning
- Communication Management Framework

API Accessibility

Services to be included in the SCP event management SDK subset.

Component Name

OntologyManager

Description

The OntologyManager provides the backend for semantic knowledge. It should allow storing, querying and manipulation of this knowledge.

Semantic knowledge will most probably be some kind of semantic web format such as OWL or just simple RDF graphs and the storage will be a simple and easy-to-use triple-store.

Services

Services to manage semantic information based on RDF query language such as SPARQL.

The OntologyManager may provide only a SPARQL interface.

More convenient translation to e.g. JSON – if requested by other components – might be implemented in another component.

Possible interface:

- query(String sparqlQuery)
- update(String sparqlQuery)
- isTypeOf(String entityID, String className)
- …

Dependencies

Might have a dependency to the Virtualization Framework but is not clear right now how that would look like.

Related Task

T5.3 – Ontologies and Semantic Representation

Smart City Platform Layer

Ontologies and Semantic Representation Framework

API Accessibility

Will any services of this component be accessible through the open APIs?

No.

Component Name

Service Orchestration Manager

Description

The Service Orchestration Manager is a generic component that handles the sequencing and invocation of different types of HTTP-based services. The Service Orchestration Manager also implements standard HTTP security protocols and also manages cookies/headers which are needed for communication.

Services

- Initiate a request orchestration and load the supplied parameters.
- Execute a service orchestration Job.
- Run the job and makes the HTTP calls.
- Get transaction time
- …

Dependencies

- Network Manager

Related Task

T7.4 (T6.3)

Smart City Platform Layer

- Data Management Framework
    - Information Aggregation, Storage, and Mining
- Ontologies and Semantic Representation Framework
- Communication Management Framework
- Multiservice Federated Networks

API Accessibility

No external API

2. Annex: Reference Architectures

The ALMANAC Smart City Platform (SCP) is based on an Internet of Things (IoT) architecture. For this reason the architecture development in ALMANAC will build 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).

2.1 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.

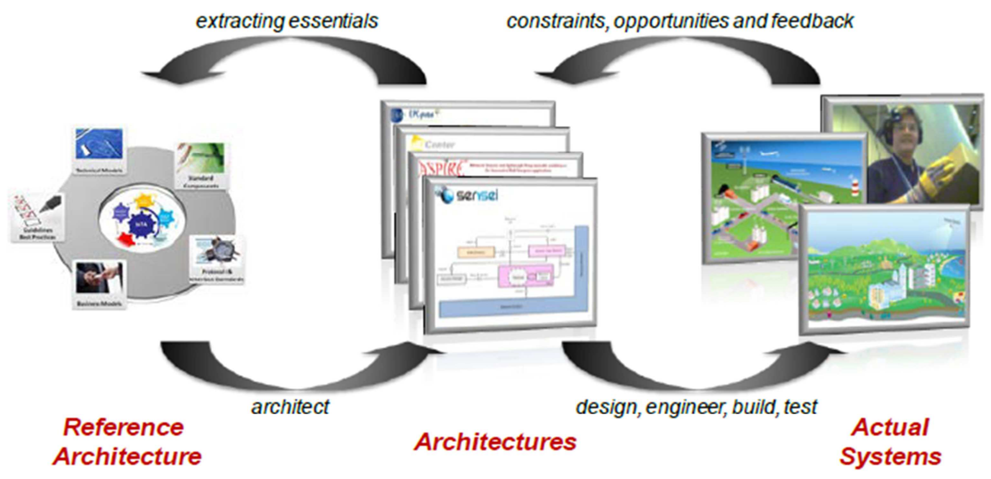


Figure 31 A reference architecture provides the instruments and guidelines for domain specific architectures from which specific system designs are derived.

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 comparisson, 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.

2.2 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 architectrures. 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[5] project and addresses IoT in terms of an overall IoT Architectural Reference Model including the subsets:

[5] 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.

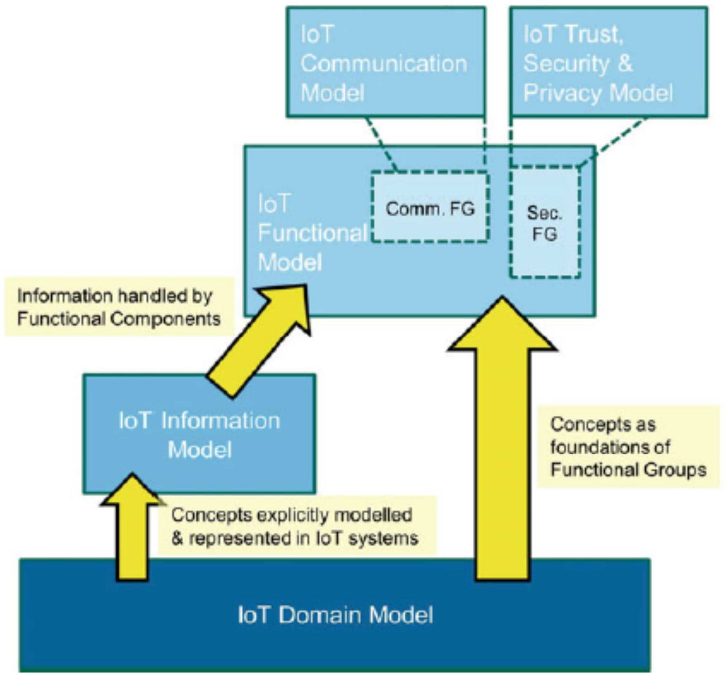

IoT Communication Model
IoT Trust, Security & Privacy Model
IoT Functional Model
Comm. FG
Sec. FG
Information handled by Functional Components
IoT Information Model
Concepts as foundations of Functional Groups
Concepts explicitly modelled & represented in IoT systems
IoT Domain Model

Figure 32 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.

2.2.1 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.

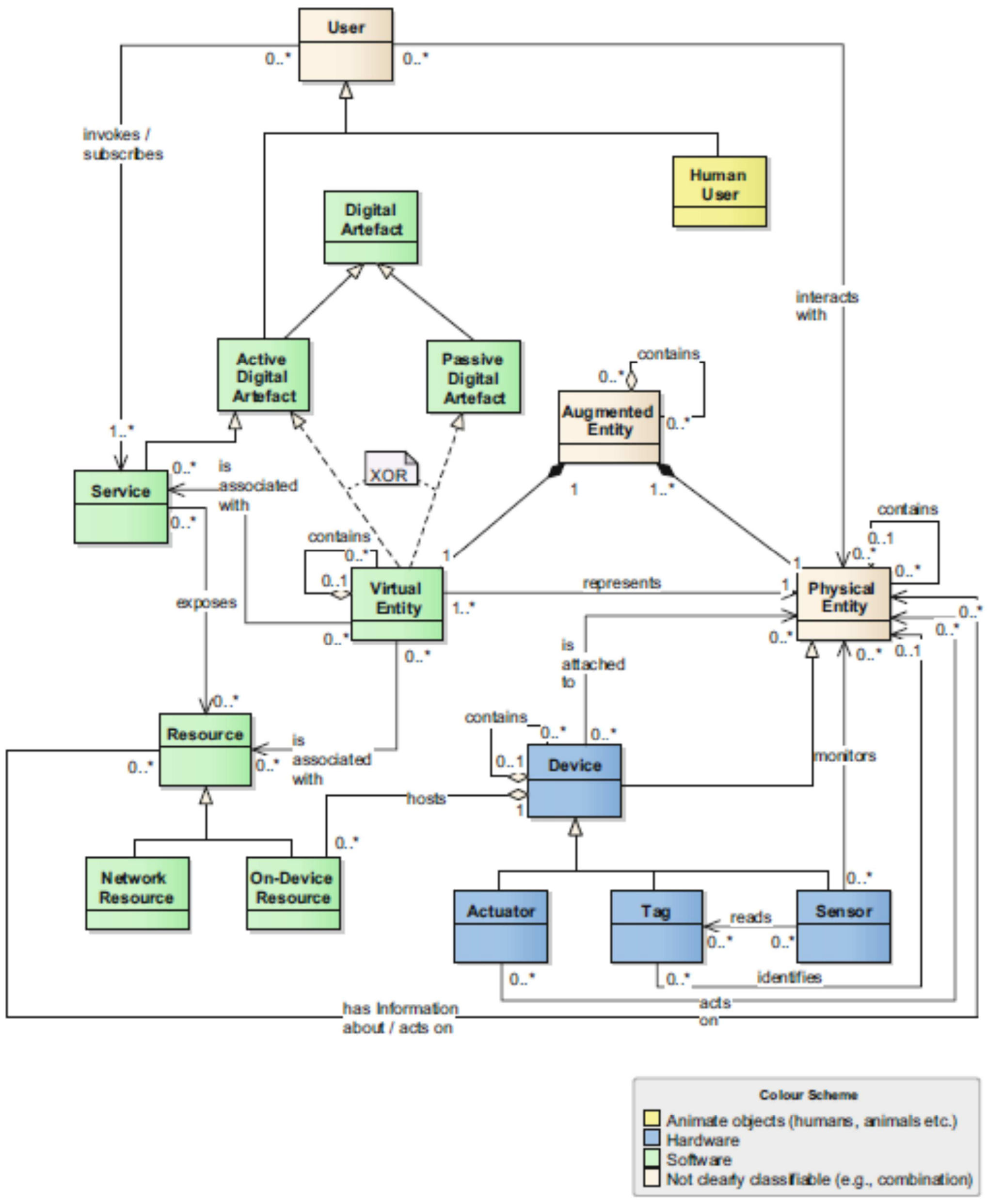


Figure 33 UML version of the IoT-A Domain Model

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.

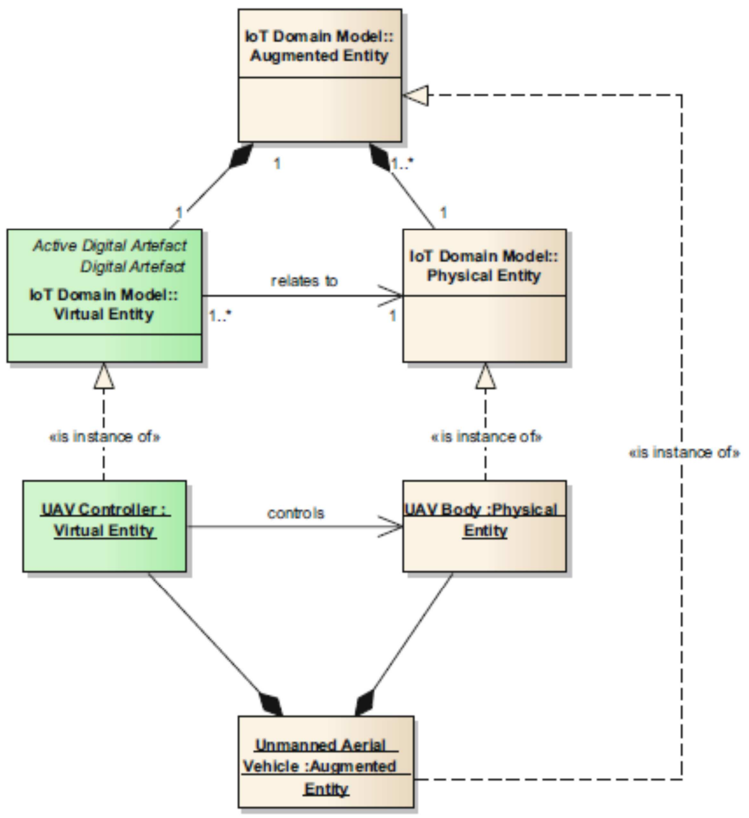
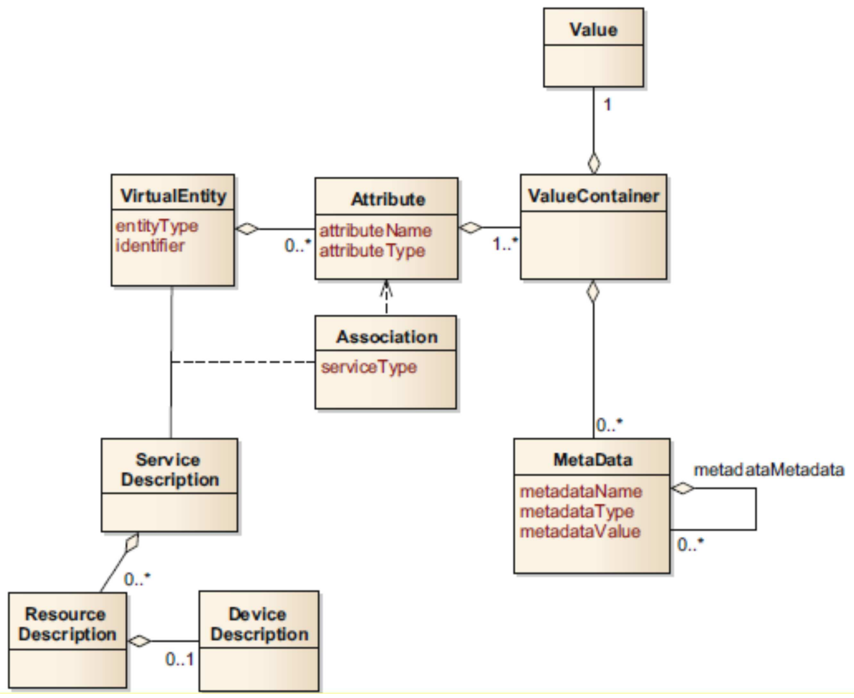

IoT Domain Model:: Augmented Entity
1
1..*
1
1
Active Digital Artefact
Digital Artefact
IoT Domain Model:: Virtual Entity
relates to
1..*
1
IoT Domain Model:: Physical Entity
«is instance of»
«is instance of»
«is instance of»
UAV Controller : Virtual Entity
controls
UAV Body :Physical Entity
Unmanned Aerial Vehicle :Augmented Entity

Figure 34 Example modelling using the Domain Model.

2.2.2 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.

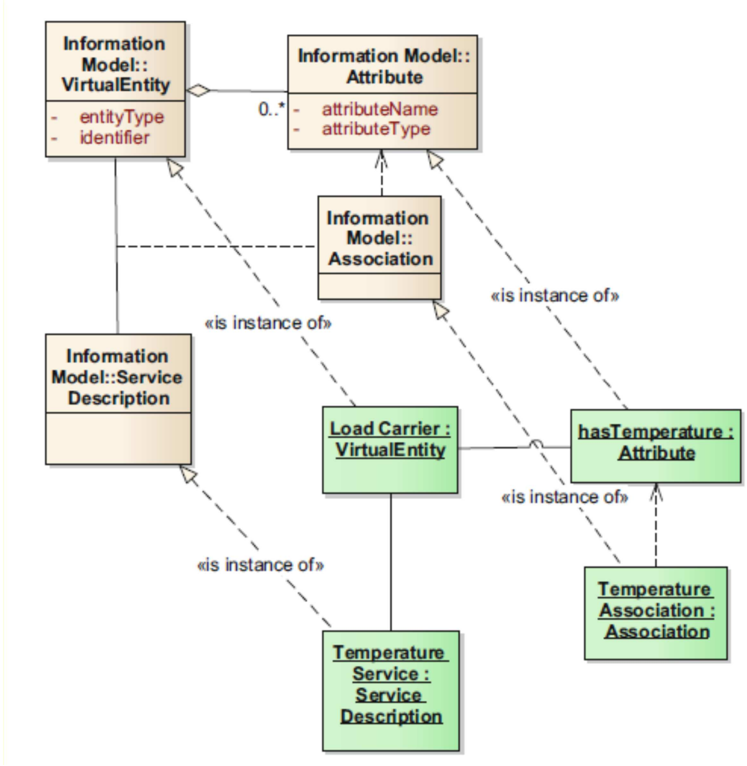

Information Model:: VirtualEntity
- entityType
- identifier
Information Model:: Attribute
- attributeName
- attributeType
0..*
Information Model:: Association
Information Model::Service Description
«is instance of»
«is instance of»
«is instance of»
«is instance of»
Load Carrier : VirtualEntity
hasTemperature : Attribute
Temperature Association : Association
Temperature Service : Service Description

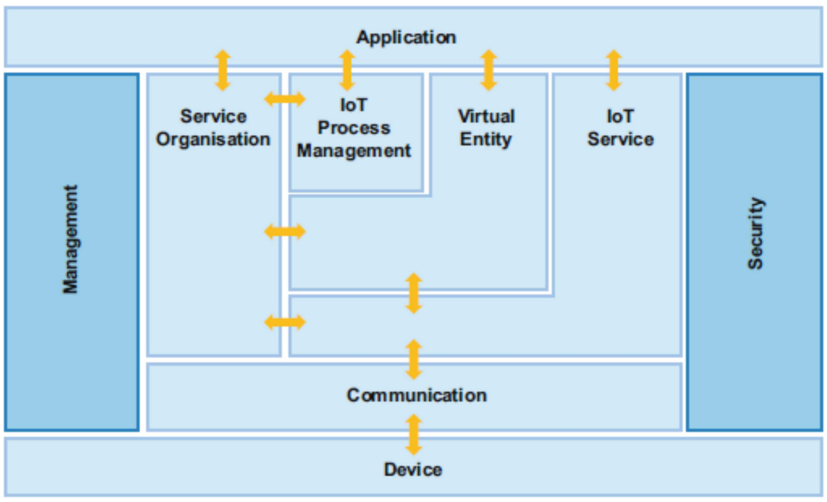
Figure 35 Example instantiation of the Information Model

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.

2.2.3 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.



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

2.2.4 Views in the Reference Architecture

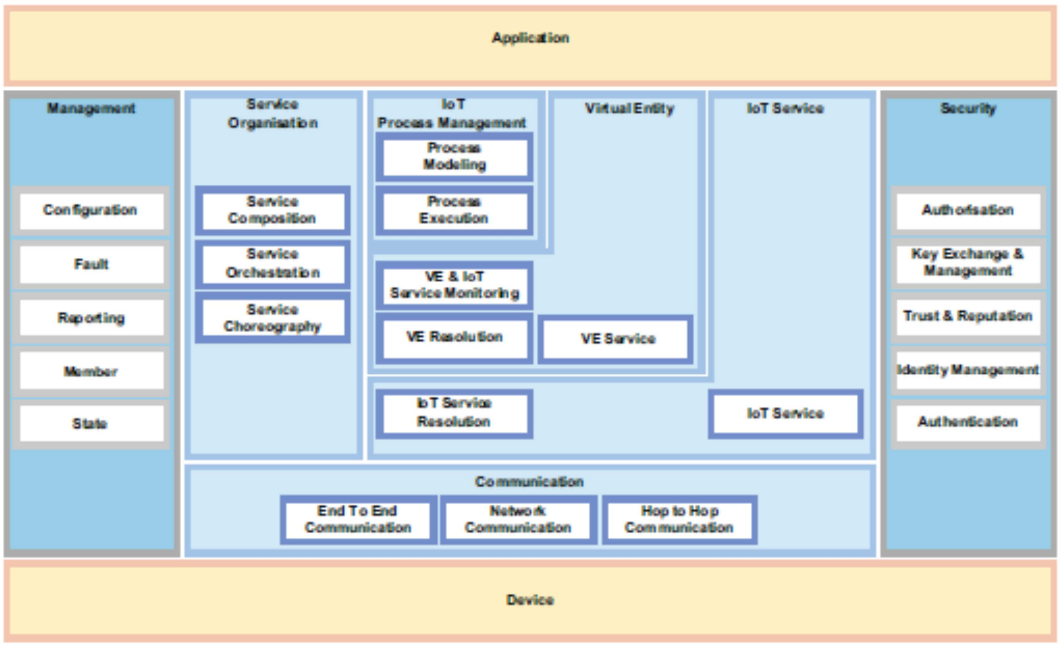
The *reference architecture* defines the Views, View Points relevant for IoT systems architecture design. Following the conventional approach the ARM describes three views, each one with a number of View Points focusing specific aspects of a view,

- 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.



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).

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.

2.2.5 Perspectives (architectural qualities) in the IoT ARM

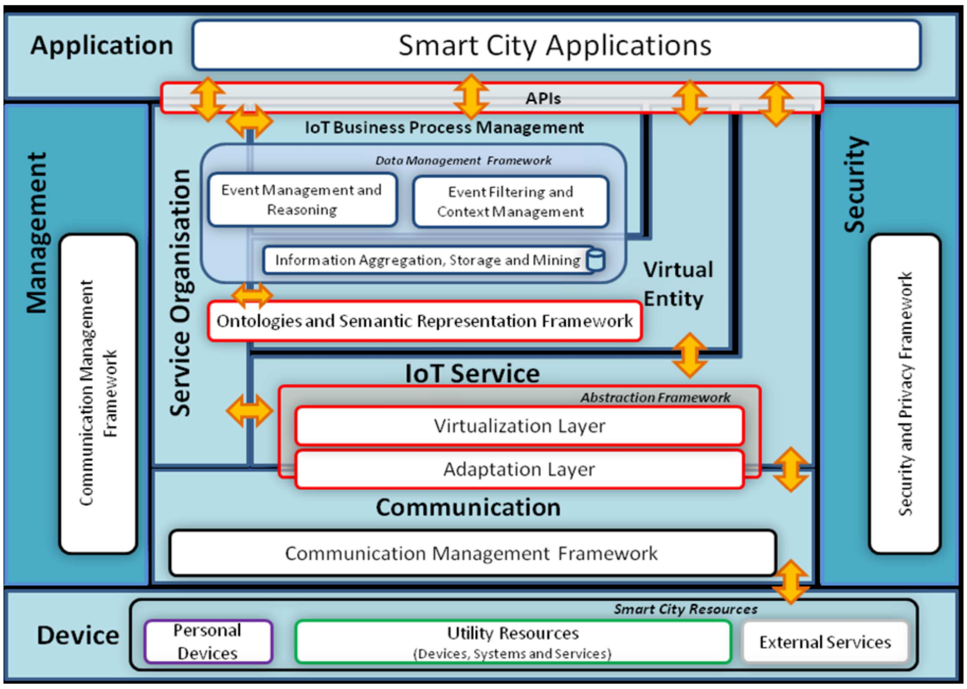
Perspectives represent non-functional requirements on a systems design , which are orthogonal to the Views of the reference architecture.

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

2.3 ALMANAC in relation to IoT ARM

Initial mapping of the ALMANAC IoT architecture to the IoT-A reference model.


Application
Smart City Applications
APIs
IoT Business Process Management
Data Management Framework
Event Management and Reasoning
Event Filtering and Context Management
Information Aggregation, Storage and Mining
Virtual Entity
Ontologies and Semantic Representation Framework
Management
Communication Management Framework
Service Organisation
IoT Service
Abstraction Framework
Virtualization Layer
Adaptation Layer
Communication
Communication Management Framework
Security
Security and Privacy Framework
Device
Smart City Resources
Personal Devices
Utility Resources (Devices, Systems and Services)
External Services

3. Annex: 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.

3.1 Devices

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

Name	Reference	Description	Relevance
Smart Bin	Producer websitehttp://smartbin.com/how-it-works/smartbin-sensors.html[6]	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[7]http://postscapes.com/internet-of-things-hardware	List of prominent IoT hardware	Helps with technology scouting about IoT hardware blocks
Smart irrigation controllers	Producer website[8]	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[9]	Waste container monitoring service	ALMANAC could target this example for the waste management use case
Xtreme RFID	Producer website[10]	Asset tracking for Municipal Solid Waste[11] [12]	Consider compatibility for waste management use case
Water and leak detection sensors	Website[13]	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[14]	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)

6 http://smartbin.com/how-it-works/smartbin-sensors.html
7 http://postscapes.com/internet-of-things-hardware
8 http://postscapes.com/smart-irrigation-controllers
9 http://www.enevo.com/
10 http://www.xtremerfid.com/applications/municipal-solid-waste
11 http://www.xtremerfid.com/news/xtreme-rfid-helps-cincinnati-grand-rapids-enhance-recycling-and-waste-collection-operations
12 http://www.rfidarena.com/2012/9/27/a-push-towards-recycling-with-rfid.aspx
13 http://www.smarthome.com/_/Sensors/Water_Leak_Detection/_/L/1SD/nav.aspx
14 http://www.thecitiesoftomorrow.com/solutions/waste/solutions/vacuum-waste-collection-systems

Urbiotica sensors	Producer website[15]	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[16].
Insteon sensors	Producer website[17]	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.

3.2 Systems

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

Name	Reference	Description	Relevance
IBM Intelligent Water	Website Leafet (PDF)[18]	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 articlehttp://txchnologist.com/post/5033695	Sewer system monitoring, analysis and control optimization. "EmNet's Real Time Intelligence and Optimization technology helps utilities	Example/Competitor to water management application. Could also provide some ideas for the water management application.

[15] http://www.urbiotica.com/products-and-solutions/
[16] http://www.metropolis.org/sites/default/files/news/urbiotica-2.pdf
[17] http://www.insteon.com/index.html
[18] http://public.dhe.ibm.com/common/ssi/ecm/en/gws03010usen/GWS03010USEN.PDF

	1628/going-against-the-flow-green-tech-sensors-and[19]	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."	
Sewage Grid: Drifting Sensors that Monitor the Wastewater Collection System (Scientifc paper)	Sewage grid document[20]	They use wireless floating sensors to detect leakage in waste water systems.	Innovative solution to water system monitoring.
SeWatch - wastewater and sewage wireless monitoring system Company: Telematics Wireless[21]	Website[22]	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[23]http://postscapes.com/connected-	List of projects using IoT for city applications	Helps with technology scouting about IoT for cities

[19] http://www.emnet.net/index.php/whatwedo

[20] https://confluence.fit.fraunhofer.de/confluence/login.action%3bjsessionid=CD08ED1A4382C4F9E19C41EF401F233A?os_destination=%2Fnotpermitted.action%3Fversion%3D1%26modificationDate%3D1391184940244%26api%3Dv2

[21] http://www.telematics-wireless.com/index.php?page_id=1

[22] http://www.telematics-wireless.com/index.php?page_id=138

	city		
Enevo ONe Collect	Producer website[24]	Waste monitoring solution	A competing solution to the waste management use case
Hitachi's water infrastructure solution	Producer website[25]	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[26]	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)

3.3 Services

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

Name	Reference	Description	Relevance
Xively (formerly Pachube.com)	Producer website[27]	Cloud service to upload and process IoT data	Example of data aggregation service for inspiration. Consider partial compatibility
IFTTT	Producer website[28]	If This Then That: Channels, triggers, actions... E.g. with SmartThings[29]	End-user empowerment
Node-RED	Producer website[30]	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	End-user empowerment

[23] http://postscapes.com/connected-city
[24] http://www.enevo.com/one-collect/
[25] http://www.hitachi.com/products/smartcity/smart-infrastructure/water/solution.html#plink05
[26] https://m2m.telefonica.com/discover-m2m/smart-cities
[27] https://xively.com/showcase/
[28] https://ifttt.com/wtf
[29] http://smartthings.com/news/ifttt/
[30] http://nodered.org

		browser-based editor and deploy on small devices such as Raspberry Pi. ´	
Google App Engine	Producer website[31]	Google's Platform-as-a-Service. Provides storage, load balancing and other services.	Cloud provisioning
Windows Azure	Producer website[32]	Microsofts Platform-as-a-service. Provides storage, load balancing and other services.	Cloud provisioning

3.4 Research Applications

Examples: experiences from on-going projects, etc.

Name	Reference	Description	Relevance
EU projects on IOT	EU Project Website[33]	CORDIS search	Live list of projects regarding IoT funded by the European Commission
SmartSantander	EU Project Website[34]	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[35]	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
IoT.est	EU Project Website[36]	EU project: Internet of Things Environment for Service Creation and Testing	Abstraction, testing, semantic annotations
Mobosens	Website[37]	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

[31] https://cloud.google.com/products/app-engine/
[32] http://www.windowsazure.com/en-us/
[33] http://cordis.europa.eu/projects/index.cfm?fuseaction=app.search&TXT=iot
[34] http://www.smartsantander.eu/
[35] http://urb-grade.eu/
[36] http://ict-iotest.eu/iotest/
[37] http://nanobionics.mntl.illinois.edu/mobosens/

3.5 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[38]	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	http://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	http://sensus.com/web/usca/products/water	Metering, Water system-level network solutions,	inspirational
Watersave SmartMeter	https://www.watersave.com.au/smartmeter/commercial	Australian Smart Water Solution	Inspirational – system descriptions. Management, system structure, Metering citizen interface
IntelliH2O smart water meter	http://www.intelli-h2o.com/products/technology		American – M2M wireless connectionand Management solutions to arrive

3.6 Standards

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

[38] http://www.techweekeurope.co.uk/news/m2m-bins-tell-council-when-they-are-full-92401

Name	Reference	Description	Relevance	Proposed by
OGC Sensor Web Enablement (framework)	OGC Sensor web enablement Group[39]	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	**ALEX**
W3C Semantic Sensor Network	W3C Incubator Group Report[40]	Ontology to describe sensors and sensor networks for use in sensor network and sensor web applications	Candidate standard for the ontology needs	**ALEX**
ETSI M2M	ETSI M2M website[41]	ETSI M2M is the European standard for M2M (Machine to Machine) applications	Used in the ALMANAC architecture and developments	**TIL**
OneM2M	One M2M standards website[42]	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	**TIL**
CDMI	ISO standards catalogue[43]	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.	**CNET**

3.7 Fi-WARE components

Examples: sections of generic enablers from FIWARE which are useful for us. (Added as a decision during the workshop on Dec. 13 in Stockholm)

Name	Reference	Description	Relevance	Proposed by
Object Storage	FI-WARE Catalogue[44]	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	**CNET**
Cosmos - Big Data analysis	FI-WARE Catalogue[45]	Implementation of the Big Data GE, based on Hadoop ecosystem, including support for MapReduce	Test of usability of Hadoop & Map Reduce in ALMANAC data	**CNET**

[39] http://www.opengeospatial.org/projects/groups/sensorwebdwg
[40] http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/
[41] http://www.etsi.org/technologies-clusters/technologies/m2m
[42] http://www.onem2m.org/
[43] http://www.iso.org/iso/catalogue_detail.htm?csnumber=60617
[44] http://catalogue.fi-ware.eu/enablers/object-storage-ge-fi-ware-implementation
[45] http://catalogue.fi-ware.eu/enablers/bigdata-analysis-cosmos

			management	
Wire Cloud	FI-WARE Catalogue[46] FI-WARE Mashup YouTube link[47]	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.	**CNET**
Device Management	FI-WARE Catalogue[48]	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	**CNET**

[46] http://catalogue.fi-ware.eu/enablers/application-mashup-wirecloud
[47] http://www.youtube.com/watch?v=yzQqstBAUeo#t=1
[48] http://catalogue.fi-ware.eu/enablers/backend-device-management