ALMANAC

RELIABLE SMART SECURE
INTERNET OF THINGS FOR SMART CITIES

(FP7 609081)

D3.1.2 System Architecture Analysis & Design Specification 2

Submission Date 2nd March 2015 – 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 Things for Smart Cities

Document control page

Document file: D3.1.2 System Architecture Analysis and Design Specification v1.0.docx
Document version: 1.0
Document owner: Matts Ahlsén (CNET)

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	Matthias Axling, Matts Ahlsén (CNET)	2015-01-14	Initial structure based on D3.1.1
0.2	Matthias Axling, Matts Ahlsén (CNET)	2015-02-10	Revised structure after architecture WS
0.3	Marco Jahn (FIT), Alexandre Alapetite (ALEX), Dario Bonino, Giampiero Bono (ISMB), Mathias Axling, Matts Ahlsén (CNET)	2015-02-13	Method update, Virtualization Layer + SotA appendix, SCRAL+ Federated Deployment + Information View, Perspectives.
0.4	Matts Ahlsen (CNET), Dario Bonino (ISMB)	2015-02-17	All changes accepted after review. Main comments remain.
0.5	Maria Teresa Delgado, Dario Bonino (ISMB), Jaroslav Pullman (FIT)	2015-02-18	Updates in sections 4.4,4.6,6.1,7.1
0.6	Marco Jahn (FIT), Mathias Axling (CNET)	2015-02-22	Architecture image update, Data Management Framework.
0.7	Matts Ahlsen (CNET), Marco Jahn (FIT)	2015-02-24	Update, Terminology
0.8	Marco Jahn, Ángel Carvajal Soto (FIT), Matts Ahlsen, Mathias Axling (CNET), Dario Bonino (ISMB)	2015-02-24	Update all sections, Technical use cases added.
0.9	Mathias Axling, Matts Ahlsen (CNET)	2015-02-27	Version for internal review
1.0	Matts Ahlsén (CNET)	2015-03-02	Final version submitted to the European Commission

Internal review history:

Reviewed by	Date	Summary of comments
Anders S Christensen (INJET)	2015-03-01	Accepted with comments
Maurizio Spirito (ISMB)	2015-02-28	Accepted with minor comments

Index:

List of Figures

Terminology

ALMANAC Platform Instance	A deployment of the ALMANAC platform. Depending on the choice of deployment this may comprise only a subset of the platform components. E.g. in some cases it may be sufficient to run an instance of SCRAL while in other cases only the Virtualization Layer and the Cloud-based APIs may be needed.
ALMANAC Platform	The ALMANAC Platform comprises a set of software components, guidelines, constraints, best practices, etc. that allow the development of Internet of Things applications for smart cities.
Capillary Network	Capillary Networks are flexible and autonomous communication networks normally used to locally collect information from sensors and actuators in the smart city. Examples of capillary networks include short-range networks based on Wireless M-Bus or DLMS-Cosem e.g. for utility metering (gas, water, electricity), collection of waste management data, pollution and traffic control sensors, smart lighting sensor, heating control sensors, etc.
Cloud-based API	Cloud-based APIs are a set of services that provide access to the ALMANAC platform for developers of smart city applications. These services can be accessed over the network through REST interfaces.
ETSI M2M	A standard defining M2M communication and platforms provided by ETSI. The basic concept of the standard is the Store and Share of data coming from smart devices. Data is stored and then shared with the Apps by the M2M Platform with standard APIs (httpREST based).
Federation	Federation describes the inter-operation between different ALMANAC platform instances (and external nodes) through a shared communication infrastructure. Each node (federate) in such a distributed system is an autonomous instance of the ALMANAC platform implementing a minimal set of components to enable communication. Further, each node manages access to its resources and services through access control policies.
IoT-ARM	The IoT Architectural Reference Model (IoT ARM) provides a collection of generic architectural concepts and constructs considered applicable to IoT system architectures. The IoT ARM does not say how to build IoT systems, it is a tool box of concepts, models and recommendations for the domain of IoT systems and their architectures.
IoTWorld Gateway	An IoTWorld Gateway is a software component that provides a logical interface towards an IoTWorld (domain). The IoTWorld gateway exposes a number of (IoT) Entities and provides a high-level API for communicating with this part of the physical world.
Machine-2-Machine (M2M)	M2M describes the ability of two devices to communicate with each other without human intervention through wired or wireless network and often through a M2M Platform. M2M communication is a prerequisite for the Internet of Things.

1. Executive summary

This deliverable describes the ALMANAC Smart City Platform architecture. The document reflects the first revision of the architecture at month 18 (mid-term) of the project.

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 component subsets are:

- Cloud-based APIs, define a set of external APIs to be used by Smart City applications and by the developers of such applications.
- The Virtualization Layer: a set of components subset that abstracts from specific Smart City resources to virtual entities and APIs, easily accessible by Smart City applications,
- Semantic Representation Framework: exploits Smart City vocabularies, ontologies and metadata, supporting the virtualization and semantic processing of resources.
- The Data Management Framework: enables storage, caching and querying of collected Smart City resource data, as well as data fusion and event management.
- The Smart City Resource Adaptation Layer: the components that provide uniform access to heterogeneous devices, over multiple protocols while also enabling standards-based interoperability with M2M networks.
- Security and Policy Framework: the components that protect 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. In particular we describe the deployment of ALMANAC Platform Instances, and the concept of ALMANAC federation.

The *information view* describes information models and data flows, in particular the semantic models and Smart City ontologies applied in ALMANAC are discussed. A set of technical use cases are provided, intended to illustrate some run-time and deployment aspects.

2. Introduction

2.1 Purpose, context and scope of this deliverable

This deliverable describes the revised architecture for the ALMANAC platform at mid-term of the project. As the requirements for the architecture will develop during also during the second half of the project, there will be 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 7 contain the different views and perspectives of the architecture; chapter 8 describes a set of technical use cases intended to illustrate run-time and deployment aspects. The document also includes annexes that report activities that supported the architecture definition process, among these an excerpt from the projects state-of-the-art library.

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 through 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 3 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.

During the first year, 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 (see section 10 Annex 1), 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 the 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.

The first revision of the architecture was then driven by the implementation of the year 1 prototypes according to the initial architecture design. Design, development, and deployment of the prototypes helped to understand better the issues with the initial architecture and led to the necessary refinements.

3.1 Year 1 architecture definition

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

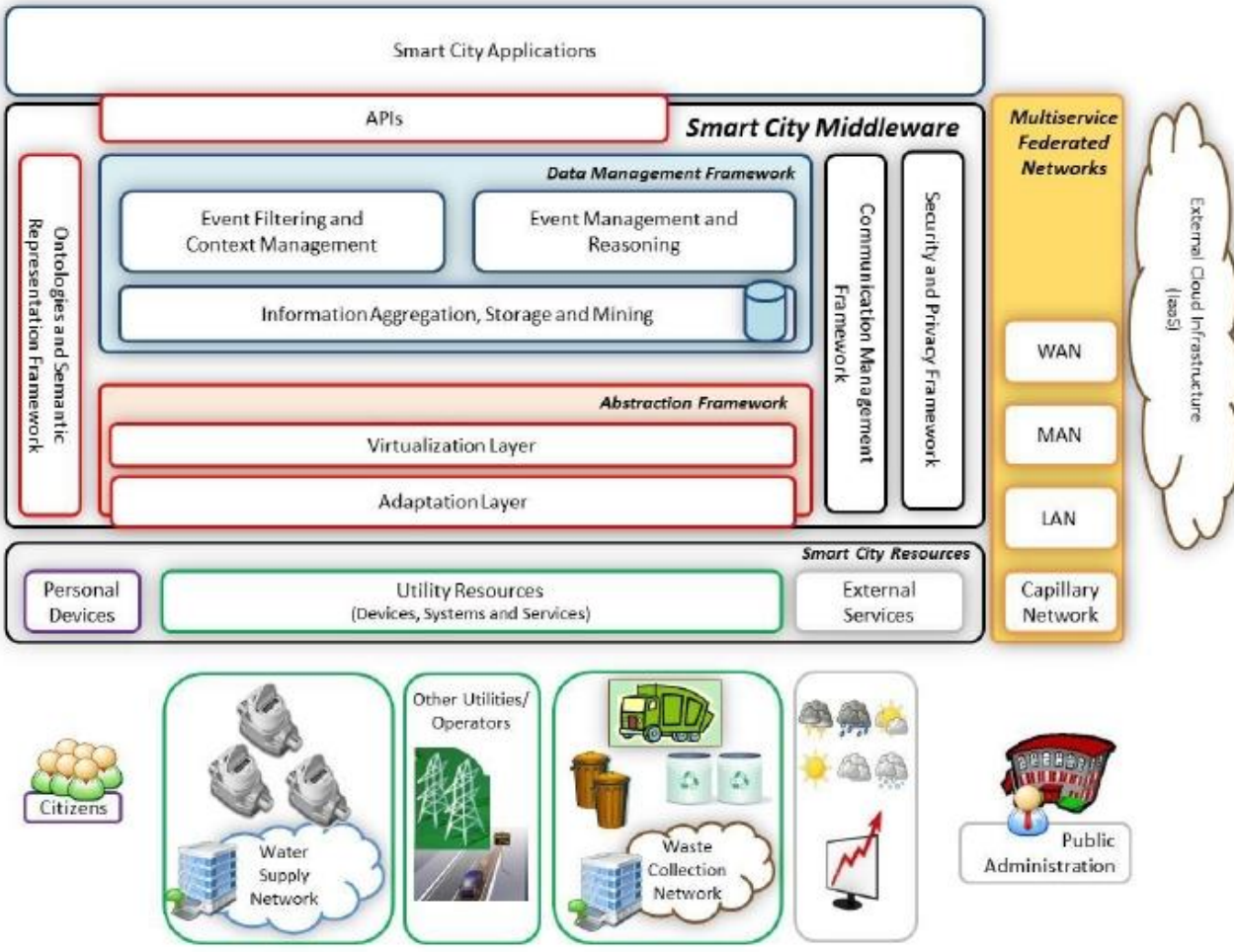


Figure 1: Initial ALMANAC architecture as presented in DoW

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.

The provided component descriptions where analyzed with respect to their position in the initial architecture. This was done in order to see how well required functionality was 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 was the initial architecture definition.

In a complementary effort, 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.

3.2 First revision of the architecture

The first revision of the architecture is based on the following aspects:

1. Lessons learned from the implementation of the prototype applications and according functionality and performance of the involved platform components
2. Security and privacy requirements of the ALMANAC platform
3. Scalability requirements of the ALMANAC platform
4. Federation between ALMANAC platform instances

Replacement of Event Manager

Addressing points (1) and (3), the event propagation was subject to major updates. While the communication paradigm didn't change, the protocols and components used did change. The Event Manager (EM) was replaced by a MQTT Broker. With this change we reduce unnecessary overhead in Event Management due to the use of HTTP/SOAP Web Services. Instead, the MQTT broker is based on plain TCP allowing for a much more lightweight and efficient event management, which is a major requirement of the ALMANAC platform. Furthermore, MQTT is a well-known standard in the IoT community.

Trust in ALMANAC

Considering points (2) and (3), we distinguish between trusted communication between components within an ALMANAC instance and untrusted communication between ALMANAC instances. In the first case, security will not go beyond basic Transport Layer Security (TLS). In the second case, ALMANAC implements security and access control at the border of such instances (see role-based access control). Furthermore, the capillary networks (represented by ETSI M2M) and external services and applications reside in the untrusted zone.

Role-based Access Control

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

LinkSmart Global Connect P2P

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

Resource Catalogue

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

The continuous refinement process in WP3 led to an update of the logical architecture as depicted in Figure 2. This updated version comprises the following subsystems of the ALMANAC platform:

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

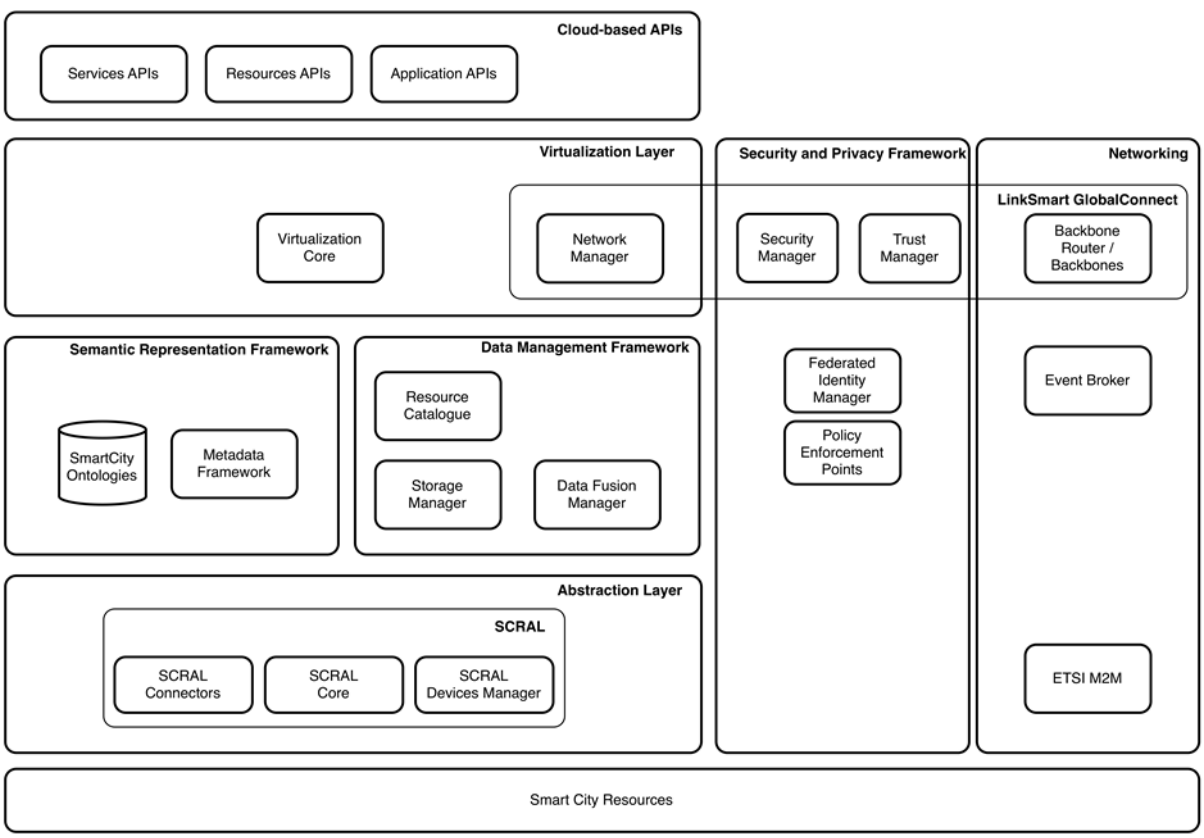


Figure 2: Updated component coverage of the architecture

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. Central to this approach 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 presents proposals in a descriptive way and UML diagrams will assist in the understanding of the proposal. It describes responsibilities, interfaces, and interactions between the functional elements.
- 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 of such a perspective is security: 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.

4. Functional view

This chapter gives an overview of the different components of the ALMANAC platform, including their functionality, interfaces, and interactions. The overall component diagram of the ALMANAC SCP is outlined in Figure 3.

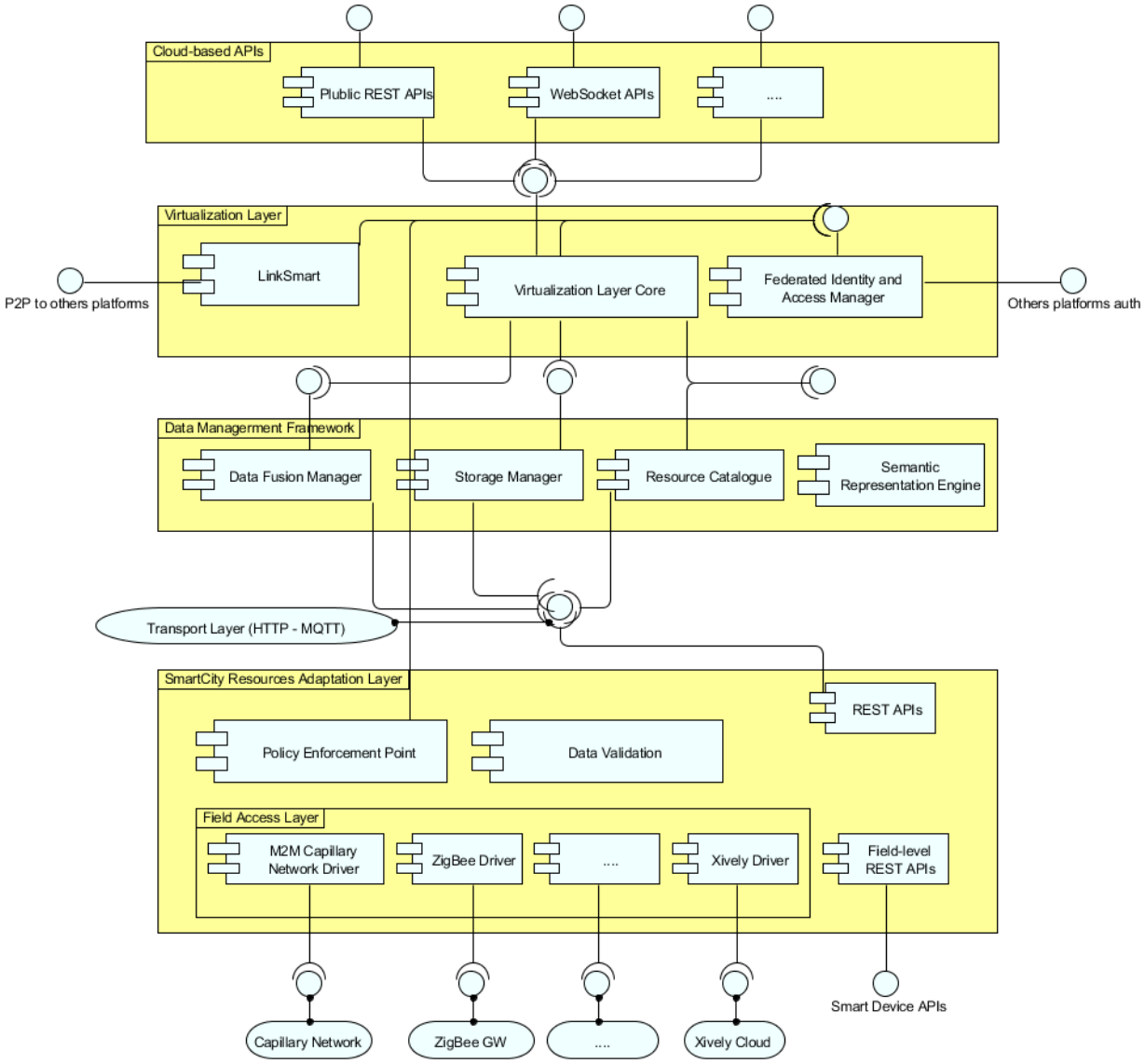


Figure 3: Component diagram of the ALMANAC platform

4.1 Overview of subsystems

The ALMANAC platform is intended to be a middleware hosting multiple forms of applications; this function is also reflected in the list of its components. The platform provides to applications 4 main services:

- Interoperability over devices: enabled by the Smart City Resources Adaptation Layer (SCRAL), applications can access any kind of devices, whichever proprietary protocol they may speak, over a uniform web-service based interface. In addition to this service, the SCRAL exposes available metadata, and semantic information for connected devices and streams, and makes

them first class citizens in the platform, e.g., by enabling the Virtualization Layer to access such an information when needed.

- Virtualization of services: enabled by the Virtualization Layer, the applications relying on the middleware do not have to know where the services or devices they consume are placed, or whether they actually exist. The Virtualization Layer provides service look-up mechanisms that bridge physical network boundaries, or can even wrap arbitrary data-sets – like historic measurements or cached values – as consumable services.
- Composition of rules and caching of data: There are multiple scenarios, where applications are not interested in current device values, but would rather like to be informed if specific thresholds are met, or see trends for particular intervals: this capability is provided by the Data Management 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 Smart City Resources Adaptation Layer

The Smart City Resource Adaptation Layer (SCRAL) provides a REST-based uniform and transparent access to physical devices, capillary networks, systems and services for monitoring and actuation in a Smart City context. Peculiar device functionalities are uniformed, abstracted and mapped to a well-known set of functions and primitives complying with (device) models handled in the semantic framework of the ALMANAC platform. Moreover, due to its nature of interface between the ALMANAC platform and the real world, the SCRAL offers primitives for applying access-control, data-validation and role-based policies on field-level data sources. While typical SCRAL instances are distributed near to physical devices, meaning that more than one SCRAL instance is usually adopted in a single ALMANAC Platform Instance (PI), at least one cloud[1] instance is typically available in a PI to support connection of smart devices, i.e., of devices able to natively exchange data conforming to the ALMANAC data model. In such a case, the main SCRAL duty is to enforce access rights, perform data validation and support needed provisioning primitives.

The SCRAL internal architecture (see Figure 4) is organized in three layers, respectively named API, Core and Field-Access. The topmost layer exploits the SCRAL Connector component which exposes REST resources to the upper layers of the ALMANAC platform and the MQTT data source, which feeds the ALMANAC PI broker with real-time data purposely uniformed, abstracted and validated by the SCRAL. The core layer hosts core SCRAL components including the SCRAL event-delivery module, the Policy Enforcement Point, the Data Validation module and the Metadata Generation component. Finally, the Field-Access layer integrates components (drivers) to wrap and isolate device-specific and technology-specific implementations used to access real physical devices and systems.

The SCRAL APIs[2] are organized in 4 subsets, respectively named control, streaming, metadata and enforcement endpoints, and can either be based on a standard REST transfer (for Request/Response interactions) or on an MQTT streaming protocol (for real-time data flows). Data formats are shared between the different communication channels. Following subsections summarize the four different subsets.

[1] i.e., deployed on the Internet, and not physically near to any real device/gateway.
[2] Exposed to components part of an ALMANAC platform instance.

4.2.1 Control Endpoint

A RESTful interface exposing all the sensing and actuating features made available by interfaced devices (either directly or through network-level gateways). It is reachable through REST by all components of an ALMANAC PI. While this control endpoint is typically called by the Virtualization Layer, for offering external actuation and querying APIs, it might be leveraged by components belonging to the Data Management Framework for carrying specific tasks, e.g., populating the Resource Catalogue upon request

The control endpoint does not explicitly support semantics, however, device representations and functions offered by this API are completely aligned to semantic models and representations defined at the Data Management Framework, thus enabling the platform to seamlessly integrate real-world data with the corresponding semantics-rich descriptions.

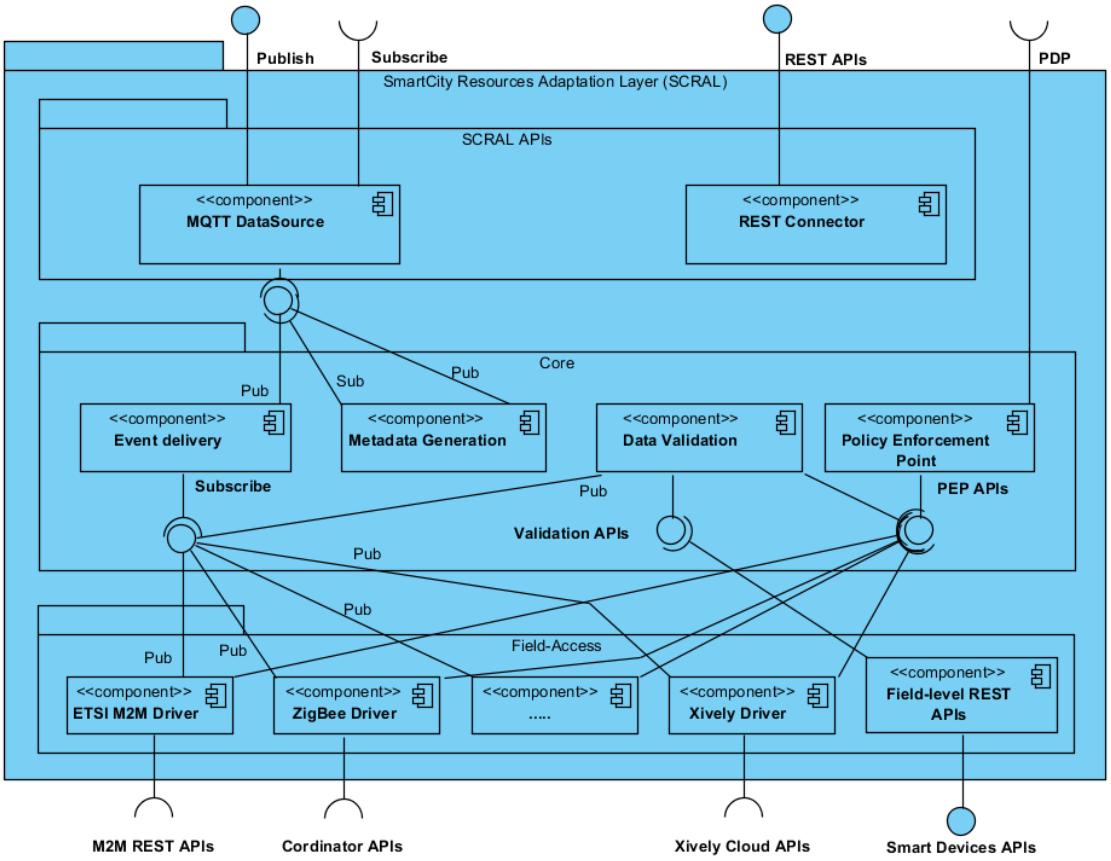


Figure 4: Smart City Resources Adaptation Layer (SCRAL) component diagram.

4.2.2 Streaming Endpoint

The SCRAL offers to the other components of an ALMANAC PI a constantly updated flow of measurement values and data, generated in real-time by connected devices and capillary networks. While each interfaced technology may have its own data generation pattern (e.g., polling vs event-based) and timing, the SCRAL converts such a data sampling into an event-based data-delivery model. In this way, measurement values, and in general observations, are conveyed asynchronously over a trusted, platform-specific MQTT data flow with ALMANAC-defined data representation standards (mainly stemming from the IoT-ARM reference data model (Bassi et al., 2013) and from well-known standards such as the OGC SensorML[3], and the IETF SenML[4]).

[3] http://www.opengeospatial.org/standards/sensorml
[4] https://tools.ietf.org/html/draft-jennings-senml-07

4.2.3 Metadata Endpoint

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

Enforcement Endpoint

Since the SCRAL lies at the boundaries between the real world and a given ALMANAC platform instance, it is endowed with the authority and capability to enforce decisions on "acceptability" of incoming data streams. According to the federated identity management model adopted in ALMANAC (see Section 5), and based on the XACML Policy Enforcement Pattern and on the SAML framework, data entering the platform must pass a set of security checks and must undergo specific policies depending on the data type, source, etc.

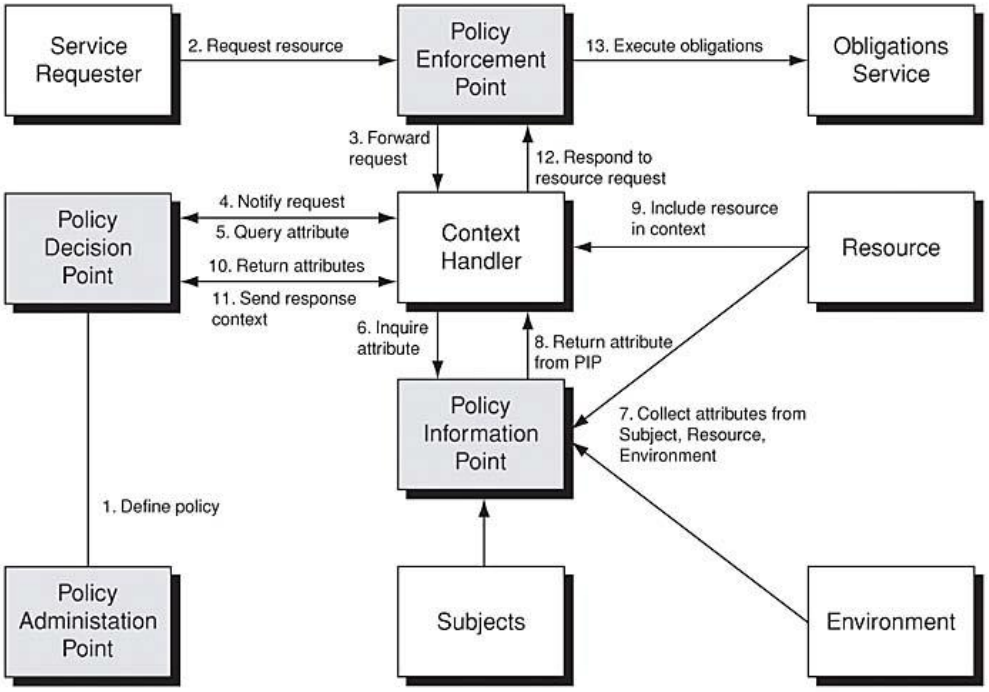


Figure 5: XACML Reference Architecture (Policy Enforcement Pattern)

The SCRAL, as boundary of the platform, implements the Policy Enforcement Point functions allowing or blocking resources from entering into the core ALMANAC platform. Such decisions are taken locally at SCRAL level by exploiting the ALMANAC federated identity manager located at the Virtualization Layer, which offers Policy Information and Policy decision services. The API used /exposed by the SCRAL to exchange information about currently connected resources and streams as well as the information needed to either authorize or refuse access to the platform services at the field-level is called Enforcement Endpoint5.

[5] This interface will be better specified during the second year of the project.

4.3 Virtualization Layer

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

The Virtualization Layer is composed of three main components, namely a Virtualization Core (with the main logic), LinkSmart Network Manager (mainly for communicating with other ALMANAC platform instances), and the Resource Catalogue (in charge of maintaining a list of IoT resources). The Virtualization Core communicates with the other ALMANAC components through HTTP REST and MQTT, and communicate with external applications though HTTP and WebSocket.

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

1. Proxying of requests/responses and events to/from the different ALMANAC internal components, making the ALMANAC instance look like a single Web service to the eye of external applications.

   Such a proxying will include a dialogue with the Security and Privacy Manager to ensure that the request is allowed.

2. Routing of requests/responses either to an ALMANAC internal component of the same ALMANAC instance, or to another ALMANAC instance when needed (through the Network Manager), leveraging the federated nature of the ALMANAC platform.

   The routing decision will be taken based on a set of internal rules, taking advantage of a naming mechanism similar to DNS (Section 5).

3. Wiring ALMANAC components in such a way that some requests requiring the collaboration of several ALMANAC internal components may be executed by external applications in a single call.

   For instance, a request may require discovering some IoT resources via the Resource Catalogue or the Semantic Layer, followed by a query for historical data to the Storage Manager, before being aggregated and returned to the external application.

4. Transforming the format of the requests/responses and corresponding payload, in selected cases not supported natively by the ALMANAC internal components. This is in particular to ease the interaction with third-party applications that are not ALMANAC-aware but may interact with ALMANAC through another standard protocol.

The Virtualization Layer may accept some queries using "real world" terms. This could be postal addresses, GPS coordinates, sensor type and exposed 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.

4.4 Data Management Framework

The Data Management Framework provides components for storing, retrieving and managing metadata and data as well as for filtering, aggregation, and fusion of data. The components of the Data Management Framework are described in more detail in deliverable *D6.1 A scalable data management architecture for Smart City environments*.

4.4.1 Data Fusion Manager

In the ALMANAC platform, events and measurement data are generated by the physical devices attached to the SCRAL. The role of the Data Fusion Manager (DFM) is to perform Data Fusion (DF) for events created across the network and derive new information from combinations of events, by

exploiting Complex Event Processing (CEP) techniques. Doing so should be possible with a simple and uniform method. The Data Fusion Language (DFL) provided by the DFM is a generalization of different Event Processing Languages (EPL) and CEP capabilities and provides an abstraction on top of different CEP Engines. The role of the Data Fusion Manager is also to manage which data is published outside the local broker and if data should be stored by the Storage Manager.

A possible future extension is a visual data fusion language to provide developers with an intuitive way of managing data fusion and event routing.

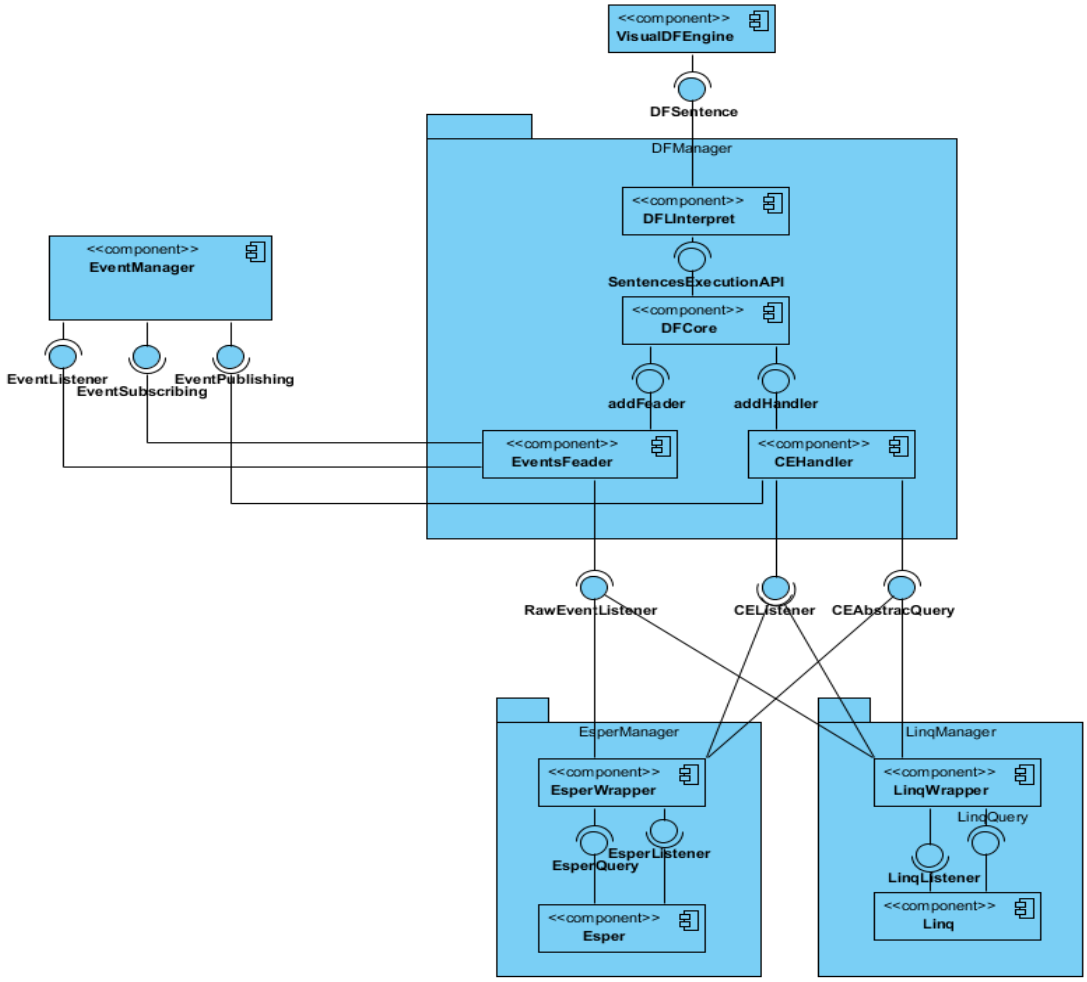


Figure 6: Data Fusion Manager Components

4.4.2 Resource Catalogue

Resources are software components that provide data from or are used in the actuation of devices. IoTResources are software objects (specifically, Network Resources in the IoT-A) that provide services for applications and end-users, e.g. retrieving and analyzing data about the physical world, invoking actions and so on. The currently available IoTResources exposed by the SCRAL are discovered and managed by the Resource Catalogue. The Resource Catalogue provides a REST-based interface to select and retrieve data about IoTResources and their services. The IoT Resource Catalogue also provides an integrated way of querying and invoking services on matching IoT Resources.

4.4.3 Storage Manager

The storage manager provides persistence for data generated by SCRAL devices and the Data Fusion Manager. It provides an interface for storage and querying of large number of time-stamped events

and measurements. The Storage Manager for a specific ALMANAC platform holds the data generated by the SCRAL and Data Fusion Manager for that platform.

The requirements and available resources of ALMANAC Platforms may vary. One platform may have a limited budget and limited storage needs, another the need to store huge amounts of historical data for analysis, and a third may already have and in-house storage solution set up and want to keep the data there.

A number of databases exist that are suitable for storing and querying large amounts of time series data [D6.1 A scalable data management architecture for Smart City environments]. The interfaces and query languages for these are not standardised and mostly product specific. The Storage Manager acts as a transparent proxy to allow for integration with different storage solutions based on the requirements and resources of the provider of a specific ALMANAC platform. The product-specific implementation is made by the Storage Adapter.

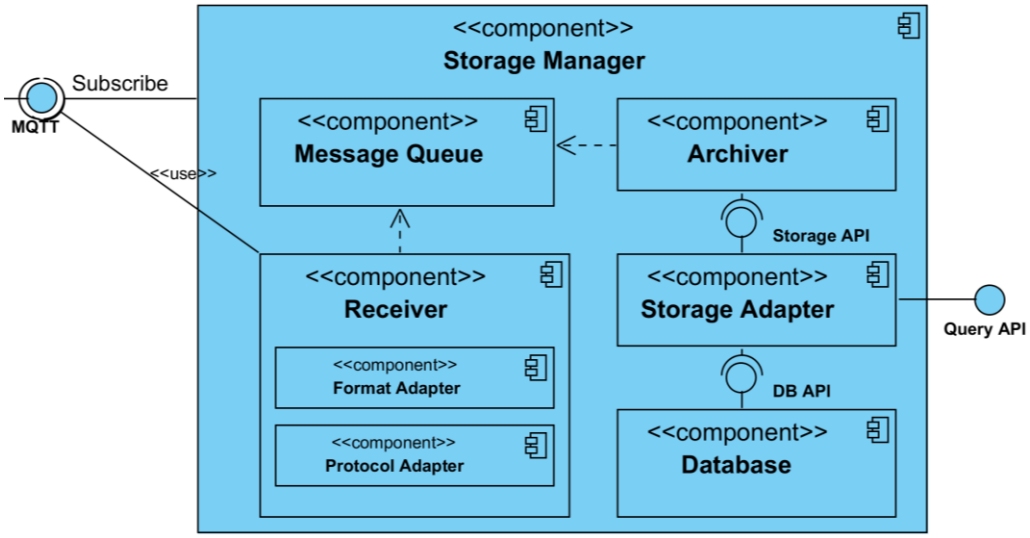


Figure 7: Storage Manager Components

The RESTful storage API of the Storage Adapter may be used directly, but in the ALMANAC platform data is received from the platform´s MQTT broker, possibly transformed to the Data Management Framework Data Model, queued and sent to the designated storage component.

The RESTful query API focuses on a limited number of filters and aggregation functionalities for the device and data fusion data that the majority of data stores are able to support. Complex queries involving type information or other metadata are resolved first using the Semantic Representation Framework (SRF), then the Storage Manager is queried for specific sets of data.

4.4.4 Semantic Representation Framework

This functional layer is responsible for the management and querying of highly structured graph-based domain models and metadata. It exposes a range of native (Java) and remote (HTTP) service interfaces allowing for various interaction modes either focusing on invocation of persistent queries (Remote Procedure Calls, RPC[6]) or the exchange of semantic data resources (REST[7]). In context of the ALMANAC architecture SRF logically complements the "Storage manager" component. While the former excels in querying of structure-rich data and its enhancement by means of semantic inference the latter is optimized for bulk storage and querying of large volumes of measurement data, events etc.

An extensive analysis and specification of the Semantic Representation Framework SRF (originally called "Semantic Representation Layer") was given in deliverable *ID 5.4 Ontologies and Semantic*

[6] http://en.wikipedia.org/wiki/Remote_procedure_call
[7] http://en.wikipedia.org/wiki/Representational_state_transfer

Representation Layer Prototype (M15). As stated there, the generic re-usable parts of SRF are being developed as part of the open-source LinkSmart[8] Metadata Framework. The Metadata framework acts as a transparent proxy to any state-of the art RDF store compatible with the SPARQL 1.1 Protocol[9] and considerably augments the standard repository functions and interface coverage, e.g. by support for persistent SPARQL queries and updates and RESTful management of graph fragments (semantic resources).

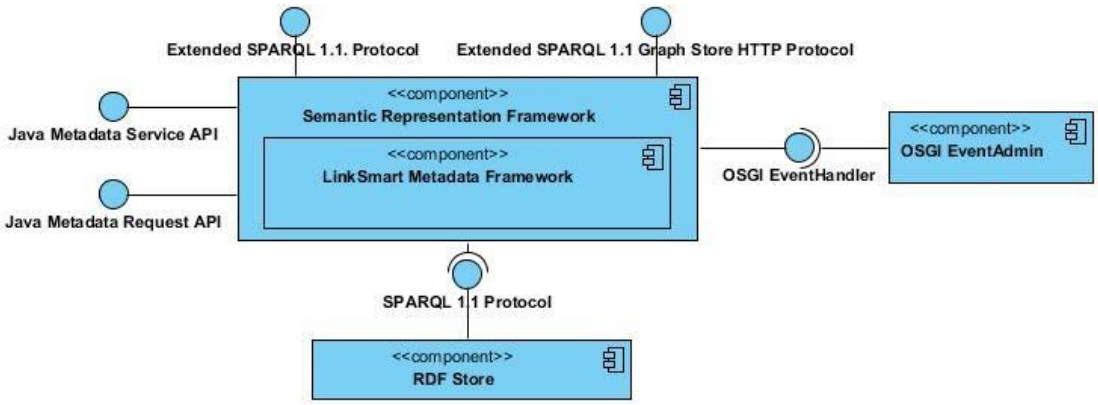


Figure 8: Semantic Representation Framework components and interfaces

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

- *Virtualization layer* mediates access to internal SRF interfaces and resources likewise other ALMANAC components allowing for location transparency and request federation.
- *SCRAL* (Metadata Endpoint) may use SRF to transparently publish, retrieve and query device metadata and service descriptions.
- *Resource Catalogue* may resolve resource identifiers based on advanced attribute queries.

4.5 Security and Policy Framework

The ALMANAC Security is mainly composed by 2 modules respectively handling access management and enforcing federation rules and agreements. They are referred to as the Access Manager and the Federated Identity Manager. These core elements are complemented by Policy Enforcement Endpoints distributed at the platform boundary, according to the XACML and SAML reference architectures. Following subsections better detail the role of each module.

4.5.1 Access Manager

Together with the Federated Identity Manager, the Access Manager (AM) centrally manages authentication and authorization policies for different types of ALMANAC users and resources. The AM provides the following features:

- Policy-based access control, ensuring that only authorized users can access certain data based on their access rights, possibly across multiple applications and domains
- Centralized policy management, reducing the complexity associated with managing authentication and authorization policies separately across multiple ALMANAC Platform Instances or internally across ALMANAC PI modules

[8] https://linksmart.eu/
[9] Examples of such graph stores are Apache Fuseki or the Openlink Virtuoso Server.

4.5.2 Federated Identity Manager

The Federated Identity manager (FIM) enables federation among several ALMANAC Platform Instances and with other platforms (e.g. Santander Platform) to enable access to the networks of all the platforms belonging to the federation.
The Federated Identity Manager provides the following features:

- Web single sign-on, based on the SAML standard allowing your users to sign into the federation just once
- Dynamic retrieval of user attributes among domains to facilitate access control decisions, thus enabling unified access among the federation

5. Deployment view

5.1 Federated deployment

ALMANAC defines a framework in which multiple PI's can participate into federations offering cross-city, cross-nation and cross-entity services. Such federations, typically built around some "service-exchange" agreements are deployed, on the technical standpoint, by establishing sets of "trust domains" including several (more than one) PIs. Platform instances exchange data and distribute tasks according to roles and privileges defined in the federation agreement (e.g., in a simplistic on/off case, all users/platforms belonging to the federation are allowed to perform tasks, whereas external platforms/users cannot exploit the federated services), and managed through state-of-the-art solutions for federated identity and role management (e.g., based on the SAML specification). Data and tasks are exchanged in a peer-to-peer manner, either through standard Internet protocols or through LinkSmart-controlled channels, allowing to overcome address translation issues, etc. Each PI can be part of more than one federation at time, thus providing the basis for an interconnected deployment of the ALMANAC services across cities. Figure 9 depicts a high-level view of such a federation approach.

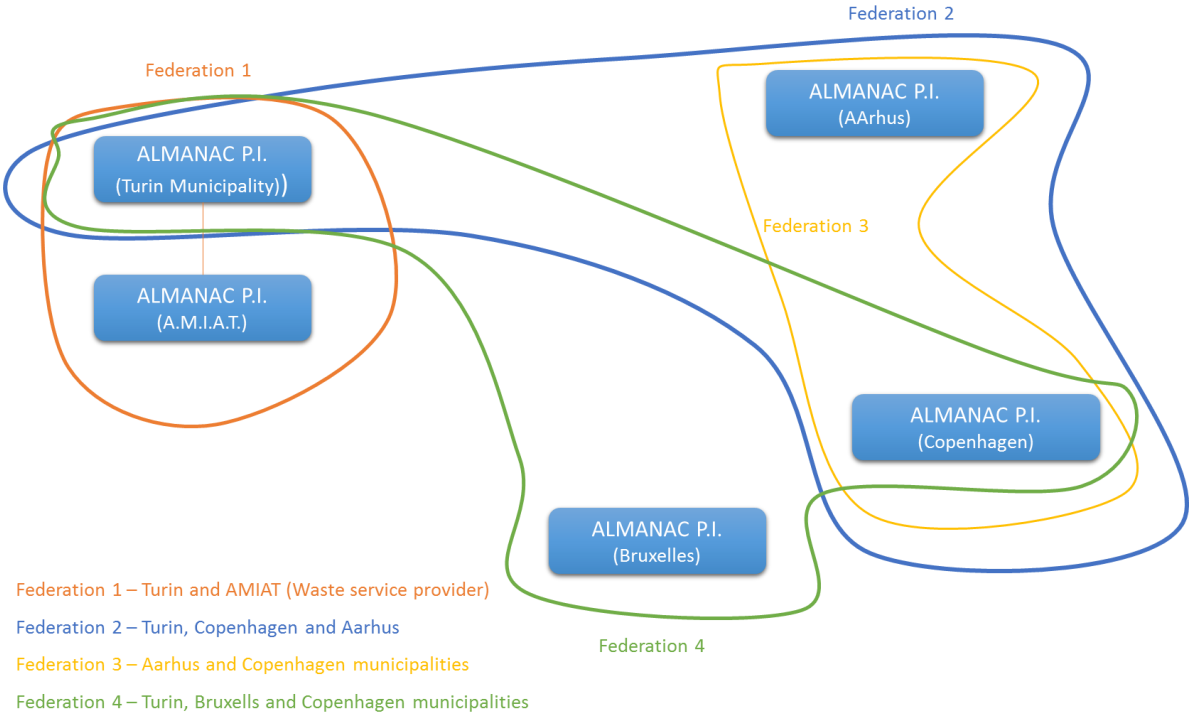


Figure 9: Multiple federations between ALMANAC Platform Instances.

Routing of requests and data through the different PIs belonging to a federation is managed at the Virtualization Layer, and exploits a sound naming / addressing scheme based on standard DNS names and CNAMEs. The ALMANAC project, in fact, defines a set of guidelines and conventions for naming platform instances and components within. Such a scheme is federation centric and adopts a twofold approach:

- Inside a platform instance, components are addressed with Internet domains structured as follows: `federation.platform.domain.component`. For example, the Storage Manager component located in the Copenhagen PI of Federation 4 (see Figure 9) will be identified by the following DNS name: `federation4.copenhagen.cnet.storagemanager.almanac.eu` where the "almanac.eu" part is the authoritative domain and might change depending on the "federation owner".

- Outside of a PI, i.e., for external applications exploiting the ALMANAC services, components will be addressed as follows: `federation.platform/component` thus hiding the platform complexity to external stakeholders.

While being mainly involved with PIs and their inter-relations, the deployment view of ALMANAC also tackles the issues and design decisions related to deployment of components inside a single PI. Two main design choices govern the consortium activity in this context:

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

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

5.2 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.

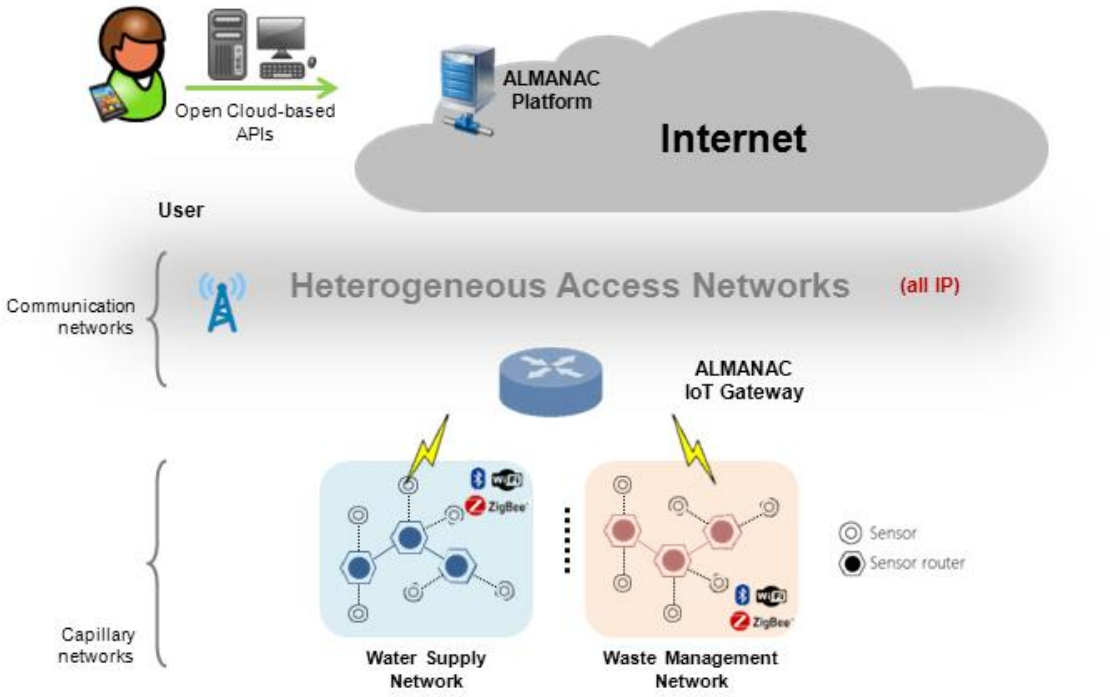


Figure 10: ALMANAC Network View

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

From the networking point of view the ALMANAC IoT Gateway has also the role of inter-connecting several types of local wireless network i.e. capillary networks. Capillary Networks are a flexible and autonomous communication networks normally use to locally collect information from sensors 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 11, 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

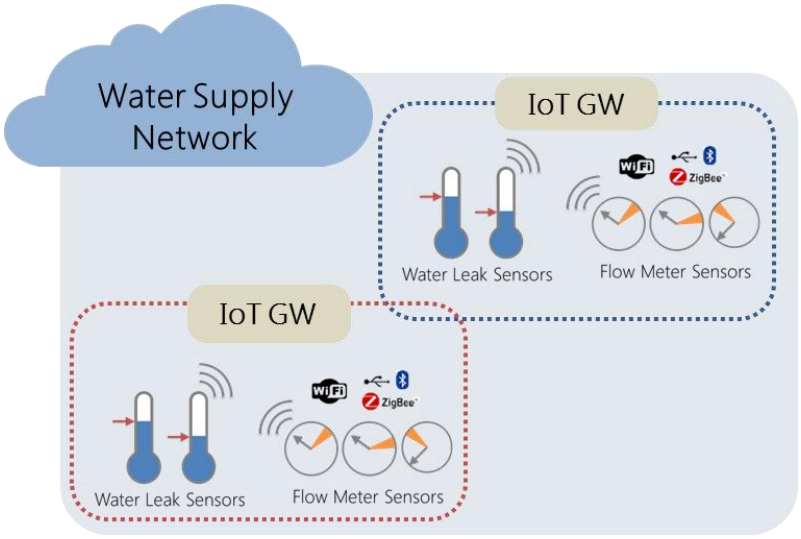


Figure 11: Water Supply Network

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.

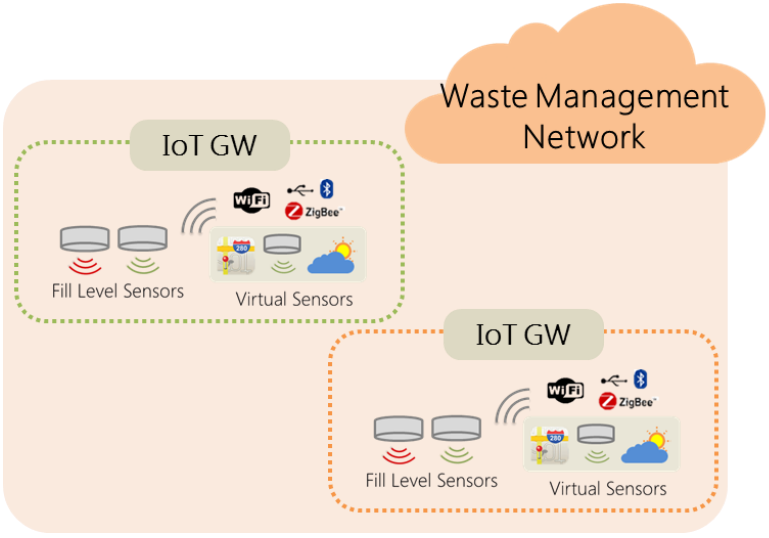


Figure 12: Waste Management Network

5.3 Cloud-Based APIs

The Cloud-based APIs are a set of external APIs to be used by Smart City applications and the developers of such applications. The Cloud-Based APIs rely on the services exposed by the ALMANAC platform components – which are more generic in nature - and expose a view of the ALMANAC system suitable for development of Smart City applications. It is accessible from any system in the LinkSmart middleware domain or from mobile or web applications, abstracting the distributed nature of the ALMANAC System and the interfaces of the specific ALMANAC components.

The functionality of the Cloud-Based APIs will have an emphasis on finding resources, requesting and subscribing to data, and aggregation of data. Most of the data related services exposed by the Cloud Based APIs are implemented by the Virtualization Layer, which is exposed directly to the external applications. It connects the components of an ALMANAC Platform as well as federated ALMANAC

platforms, making ALMANAC components and federations appear as a unified service to external applications providing support in,

- Combining services of several ALMANAC Platform components to provide a single service, e.g. for querying.
- Finding registered services and resources.
- Requesting and subscribing to ALMANAC data.
- Transformation of payload formats and adaptation to different communication protocols

The Cloud-based APIs will also provide management services, such as:

- Provisioning resources and devices to be used in the Smart City applications.
- Managing metadata for domain entities and data derived from complex event processing.
- Handling access control, such as granting and revoking application or user access to a specific resource and granting access to delegate access to a specific resource.

The subset of Cloud Services implemented will be based on requirements derived from activities in WP8 Applications Definition, Development and Evaluation.

6. Information view

6.1 Overview of semantic models used in ALMANAC

This section briefly documents semantic models used internally by the Semantic Representation Framework. They are part of the current implementation and are mainly targeted at platform developers and integrators. The following sections will present models that target end users of the ALMANAC platform covering relevant aspects of the Smart City domain.

The Semantic Representation Framework (via the LinkSmart Metadata Framework) advances the management and retrieval of graph-based semantic data by supporting operations on graph fragments. Such "semantic resources" comprise programmatically defined excerpts from the overall graph continuum. Subclasses of the `ResourceRequest` hierarchy depicted in Figure 13 express the range of supported operations – `ManagementRequest` covering the life-cycle of resources, `ModificationRequest` pertaining to partial updates on existing resources, `RetrievalRequest` dealing with retrieval of a single or a selection of resources and finally `ProcessingRequest` that cover the processing of resource's content.

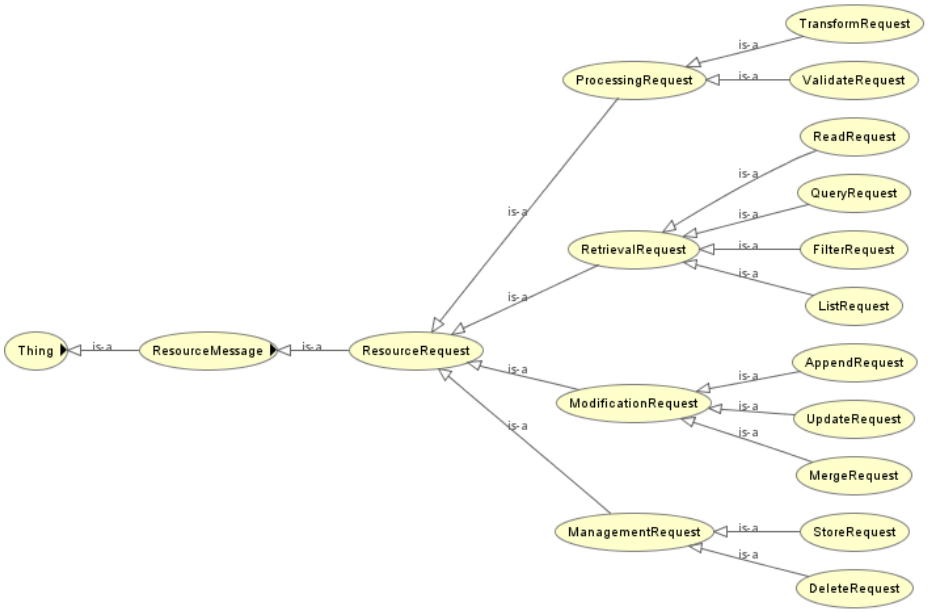


Figure 13. Resource request hierarchy.

Though not limited to, per default the handling of `ResourceRequest` messages is done by invocation of appropriate SPARQL 1.1. Query expressions. Such system- or user-provided persistent queries are modelled as instances of the `SparqlQuery` classes depicted in Figure 14. In addition to a parameter specification, description annotation etc. they contain query strings used to handle particular types of requests. `ReadRequest` messages are for example handled via pre-configured `AskQuery` instances, `ReadRequest`s result in a `ConstructQuery` call and a `FilterRequest` is handled by a `SelectQuery` instance.

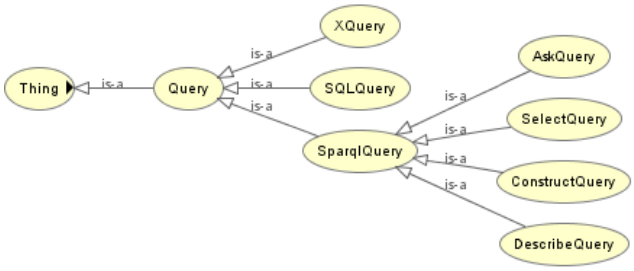


Figure 14. SPARQL 1.1. Query hierarchy

6.2 ALMANAC Smart City Ontologies

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

- The Semantic Sensor Network Ontology (SSN[10]);
- The DogOnt[11] Ontology for Intelligent Domotic Environments;
- The SCRIBE[12] IBM Smart City ontologies:
- The models listed in Smart Cities ontology catalogue[13];
- The Places[14] ontology, a light-weight ontology for describing places of geographical interest;
- The Schema.org[15] vocabulary for representing well known geographical properties, e.g., latitude and longitude;
- The GoodRelations[16] ontology for representing any kind of goods, products and services, and the relations involving their offering, exchanges, etc.;
- The MUO[17] unit of measure ontology, representing unit of measures according to the UCUM vocabulary (including International System of Measures values and other relevant units);
- The vcard[18] ontology for representing name and addresses of people and organizations;
- The GeoSPARQL[19] vocabulary and functions, to support representation and querying of geo-spatial data.
- The GeoNames[20] ontology for representing cities and other geographical entities.

A first, preliminary, modelling effort for representing smart city data with respect to waste management issues has been performed, leading to the definition of a light-weight "waste bin" ontology. The ontology focuses on waste-related issues with an open, extensible and scalable approach. According to this design goal, no assumption is performed on the remaining smart city data, and modelling is restricted to the aspects specifically related to waste collection and management,

[10] http://www.w3.org/2005/Incubator/ssn/ssnx/ssn
[11] http://elite.polito.it/ontologies/dogont
[12] http://researcher.watson.ibm.com/researcher/view_group.php?id=2505
[13] http://smartcity.linkeddata.es/
[14] http://vocab.org/places/schema.html
[15] http://schema.org/
[16] http://www.heppnetz.de/projects/goodrelations/
[17] http://idi.fundacionctic.org/muo/
[18] http://www.w3.org/TR/vcard-rdf/
[19] http://www.opengeospatial.org/standards/geosparql
[20] http://www.geonames.org/

e.g., by representing waste bins, type of disposables generated by the citizenship and so on. All represented concepts leverage existing, well known definitions of properties (and classes / super-classes) with a typical Linked Open Data approach. In such a sense, for example, the city concept is directly linked (`owl:equivalentClass`) to the corresponding schema and places concepts.

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

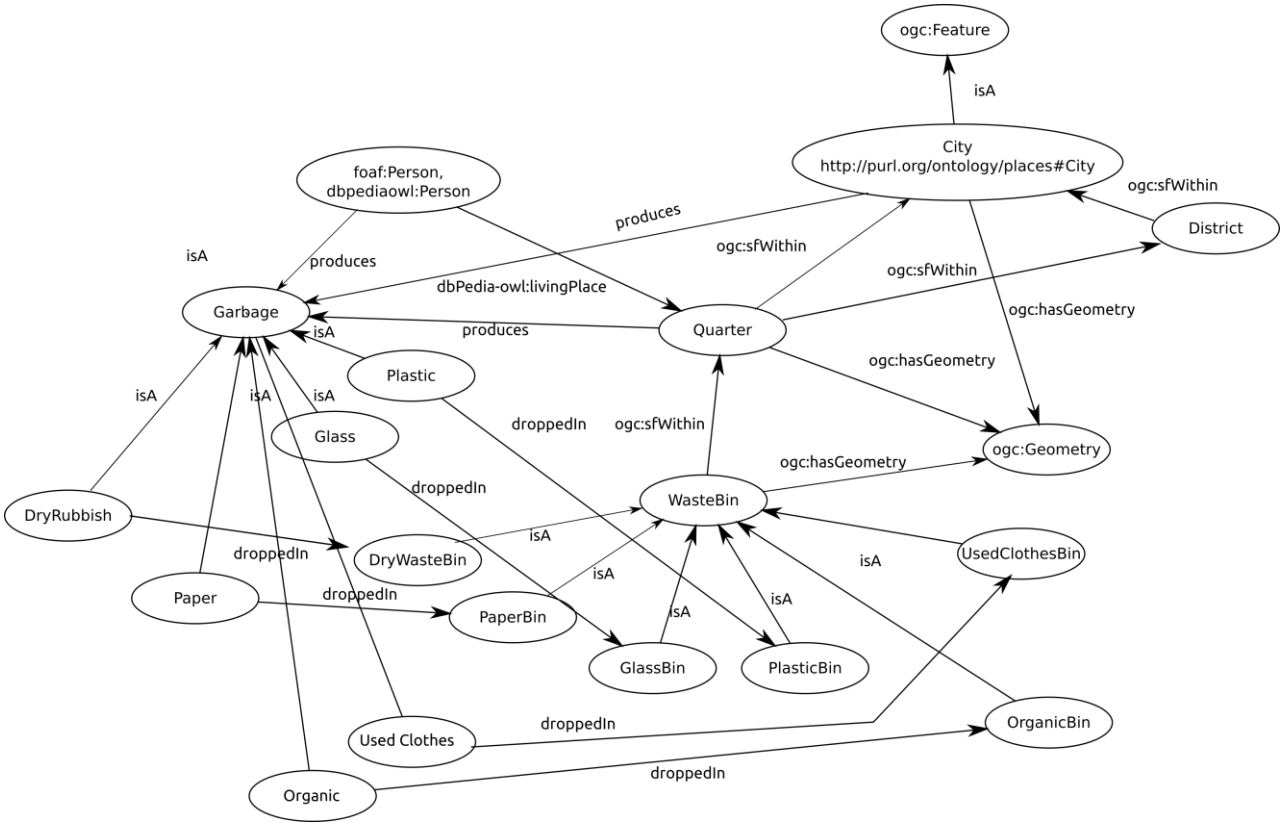


Figure 15: A part of the ALMANAC waste bin ontology.

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

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

The above described waste ontology has recently been linked to the latest version of the well-known DogOnt[21] ontology which was recently updated to represent generic IoT devices and entities. Such a link, currently instantiated for the waste bin concept only, establishes a direct relationship between declarative knowledge on waste-bin usage and location modelled in the ALMANAC-developed ontology and the functional descriptions of "smart waste bins" provided by DogOnt. This can, for example, be exploited through code-generation for defining the interfaces exposed by the SCRAL.

6.3 Information flow

6.3.1 Data

Measurement data, state changes in devices or resources and system events are delivered asynchronously through MQTT events. Data is generated by the devices attached to the SCRAL and the DFM queries. The routing of data is controlled by the DFM, which propagates events originating in one ALAMANC platform instance to another.

It is also possible to request data from the SCRAL REST APIs via the Resource Manager to get a snapshot of the current state of a device.

6.3.2 Metadata

Metadata for devices is gathered by the SCRAL and primarily delivered to other platform components asynchronously through events. The same data may also be requested on-demand through REST APIs in the SCRAL to get the current information about connected devices and resources.

The SRF also provides HTTP APIs to the Smart City ontologies that may be used by other components to publish, retrieve and query domain information, device metadata and service descriptions, e.g., the Virtualization Layer or the Resource Catalogue may resolve the resources needed to gather data for specific domain entities.

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

7. Perspectives

7.1 Security perspective

According to the platform architecture updates and lessons learned, as reported in Section 3, the ALMANAC platform instance architecture can be partitioned into a trusted area and an untrusted one. The former involves core platform components and does not imply major security risks for the platform, provided that inter-component communications are properly secured (e.g., through encryption). Likewise, it is possible to identify a set of entry points where potential security risks exist. Components in the trusted zone are exempt of any security or policy overhead. Boundary components must provide access to the trusted zone only to legitimate users/entities, and therefore deal with security mechanisms to guarantee access to shared resources within the trusted zone. The components in the ALMANAC platform instance architecture that need to deal with access control mechanisms are:

i. The Abstraction Layer, more specifically the SCRAL
ii. The Virtualization Layer, more specifically the Virtualization Layer Core (VLC)
iii. The LinkSmart Network Manager.

The untrusted area, instead, encompasses the capillary network - represented by the ETSI M2M platform - as well other instances of the ALMANAC platform, and any kind of external services or applications. Any other inner interaction among ALMANAC platform instance components is considered trusted, and applied security does not go beyond simple TLS[22] or any standard, secure networking mechanism.

The last iteration of the ALMANAC PI architecture design includes a new component, namely the "Federated Identity Manager". The FIM is in charge of enabling reliable communications among Smart City users and resources belonging to different Federate/PI's or domains. It does so in cooperation with the Access Manager. Authentication, authorization and access control mechanisms (based on the RBAC[23] approach) are implemented by these two components (working in strict collaboration), and relying on the Policy Enforcement Points (PEPs) distributed across ALMANAC platform instances to guarantee a secure and reliable communication, both among different PIs or between a PI and external services/applications from other domains. It is important to notice that both the Federation issue and the Scalability perspective are tightly related to Security, and will be closely considered for the development of activities in this matter.

7.2 Scalability perspective

Scalability in Smart City platforms is a major design concern, which is better tackled by defining solutions able to natively scale, either vertically or horizontally. Being natively scalable means being able to easily and gracefully scale to deployments where exchanged data and integrated components may vary by several orders of magnitude, from a few sensors to billion data sources. To address such an ambitious goal, the ALMANAC platform adopts a design-driven approach to scalability mainly based on the well-known and widely accepted Micro-services Architecture[24], having the following advantages:

- Each micro-service is relatively small, which implies that it is easier for developers to understand and work on;
- Each service can be deployed independently of other services, i.e., it is easier to deploy new versions of services frequently;
- It is easier to scale development. Micro-services permit to deploy the development efforts around multiple teams. Each team is responsible of a single service, and can develop, deploy and scale its own service independently of all of the other teams;

[22] Transport Layer Security
[23] Role-Based Access Control
[24] http://microservices.io/patterns/microservices.html

- Improved fault isolation. For example, if there is a memory leak in one service then only that service will be affected. The other services will continue to handle requests. In comparison, one misbehaving component of a monolithic architecture can bring down the entire system;
- Each service can be developed and deployed independently;
- Eliminates any long-term commitment to a technology stack;

The following subsections better detail the single design solutions, and patterns implemented to support both horizontal and vertical scalability.

7.2.1 ALAMANC Platform Storage Scalability

Scalability issues with regards to storage occur in three areas in an ALMANAC Platform: Data insertion, data querying and storage volume.

Data insertion throughput

The storage system must be able to store data at the rate generated by the platform. Data is generated by multiple nodes.

Some design changes have been introduced to cope with a high number of insertions. The number of storage requests using HTTP POST was too high for a single machine web server so the data delivery mechanism was changed to MQTT. During bursts with high frequency of received data, the time required to store data was longer than the time between messages received from MQTT. Therefore, a queue was introduced in between MQTT client and storage.

There is of course a limit to the possibilities to handle a large amount of data to store by software design. The data insertion mechanism may be scaled out by adding (real or virtual) hardware using load balancing clusters. These may be provisioned in the cloud.

Query throughput

A very large amount of data affects indexes and memory use and thus affects query response times, especially for the most frequently used function: getting the cache of the last value seen from a sensor (most often called "current value").

We have designed the Storage Manager so that "current value" cache is separate from the long-term storage. The Y1 demo setup indicates that on a single ALMANAC platform with 10^6 data streams, we may keep this information in memory. However, we will have to deploy the "current value" cache and the historical values on different nodes.

Storage volume

The federation of ALAMANC Platform Instances provides storage scalability as there is no single data store and each PI will provision the necessary storage for its needs.

In the Y1 demo, after a week of use with ~28.000 sensors reporting at 3 minute intervals, the amount of data became too large for the hardware the single instance MongoDB was running on. For any hardware specification, this will eventually happen.

In the general case, without any specific functional requirements on data access, we scale out by adding an additional cluster node (buy more hardware or cloud resources).

By analyzing storage requirements and access patterns however, we can devise additional partitioning mechanisms, e.g., we could partition by time, so that old values are located at another storage that requires longer access time but is cheaper. Very old values may be archived, meaning that they cannot be instantly queried, but may be unpacked and analyzed.

An ALMANAC PI cannot be assumed to have infinitely scalable storage. For the scalability efforts of Y3, the project needs to define non-functional requirements for scalability taking hardware into account. The project needs to achieve these by software design and describe the mechanism for managing what happens once the limit is reached. Such a mechanism could be based on scalable storage units, i.e., how much hardware needs for V events per X time in Y amount of time with Z size, e.g.,

- Single machine Data Receiver (REST, MQTT Client, Message Queue) (With a default specification of RAM, processor and storage)
  - X messages to store/second
  - Y current value requests/second
  - Z HTTP POST requests/second
- Single machine database server (With a default specification of RAM, processor and storage)
  - X messages to store/second
  - Y current value requests/second
  - Maximum A GB of total storage
  - Maximum B GB of storage per data stream/sensor

For the ALMANAC PI, we need a provisioning mechanism where the set-up of a platform defines the amount of storage needed, as well as how to add additional storage to a platform when the limit of the storage is reached.

- A mechanism to add these units when the storage capabilities needs to be upgraded
- Also maybe migration mechanisms to migrate from one hardware to a new one.

We need to detect when the platform limit is reached and specify how to add additional storage once the foreseen limit is reached, manually or automatically.

Overloading mechanisms or strategies may act in several steps. The first response is to scale out the system with additional resources – as defined for the platform. If the specified limit for hardware (or cost) is reached, the system may start to erase old data, or migrate it to an "archive". The platform manager should be able to define the actions to take. The last step when the limit is reached and no more resources can be added may be graceful degradation of a platform's storage. Levels of storage may be: "current", "historical", "archived" (which means that archived data may be unzipped and batch analyzed but no real-time queries are possible).

8. 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).

8.1 Data Management

In first System Architecture deliverable (D3.1.1) the initial approaches of data reasoning and data management, both through rules, were presented. After the first revision of the ALMANAC platform architecture, a more exhaustive and precise view of the data management will be presented, introducing new use cases handling the challenges to be addressed in the second year.

Regarding streams and events, six basic local operations have been identified: *Store*, *Query*, *Fuse*, *Aggregate*, *Annotate* and *Route*. The architecture allows to define more complex operations, based on the above primitives and, dealing with transformation or manipulation of streams (e.g. deploying queries as rules). As – beside Store – most of these basic operations are very similar, only Store, Route, and Annotation will be presented as use cases.

Regarding federation two uses cases and three sub use cases have been identified. The use cases are Publish Stream and Subscribe Stream (through federation) and the sub use cases are Event Propagation, Request Access and Authentication. These use cases tackle the transportation of data from one platform instance to another.

Use cases grouped into the Data Querying UC deal with the issues of querying distributed data, in a scalable manner. They encompass: the Data Gathering and the Query Partition UCs. The Data Querying UC is also leveraged in 2 new UCs, namely the Request Data and Broadcast Request Data.

Figure 16 provides a general overview of the currently defined technical use cases.

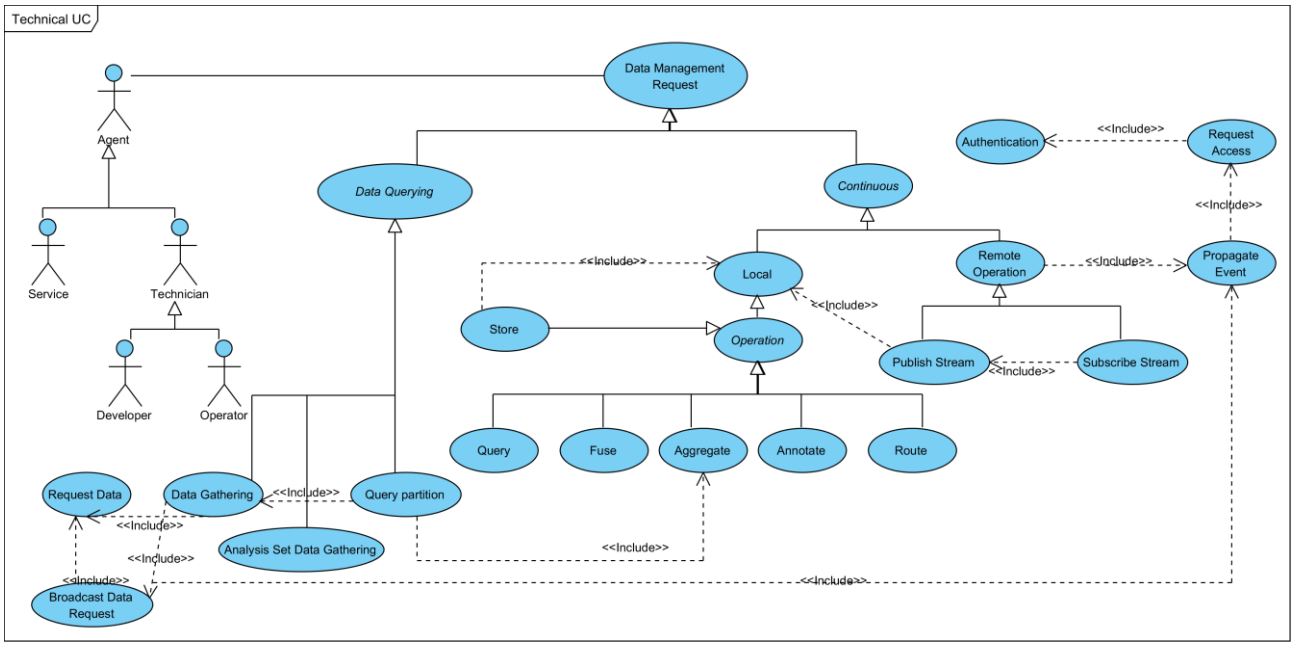
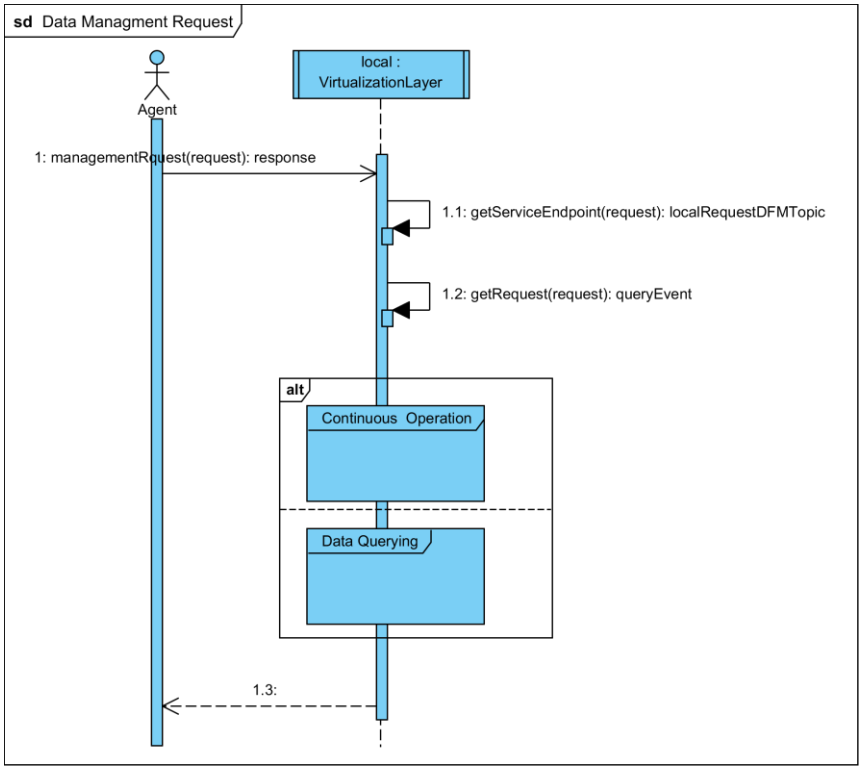


Figure 16: Technical Use Case overview

8.1.1 Data Management Request

This use case represents the starting point where a use case agent has to make a request. The agent will send a request to the Virtualization layer, which will trigger any other use case.



Either a Continuous Operation UC or the Data Querying UC is started. Both UCs are abstract UCs which represent logical separation of the UCs.

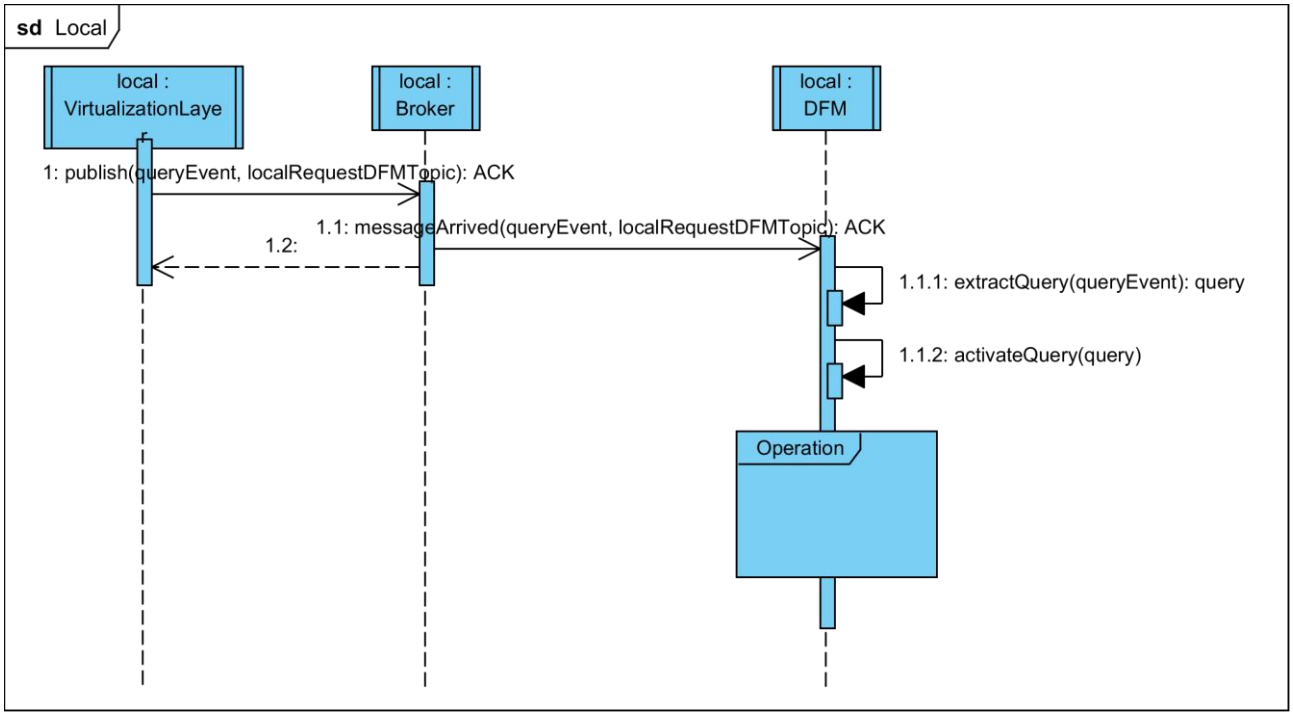
8.1.2 Continuous operations

This is a subset of use cases started by the Management Data Request, representing the cases which are performed in streams. Therefore these operations are done "continuously" for each event in the stream until the request is paused or removed.

Local

These are the operations done by the Data Management Framework. The Storage operation includes an interaction between the Data Fusion Manager and the Storage Manager. As examples, three local operations will be presented.

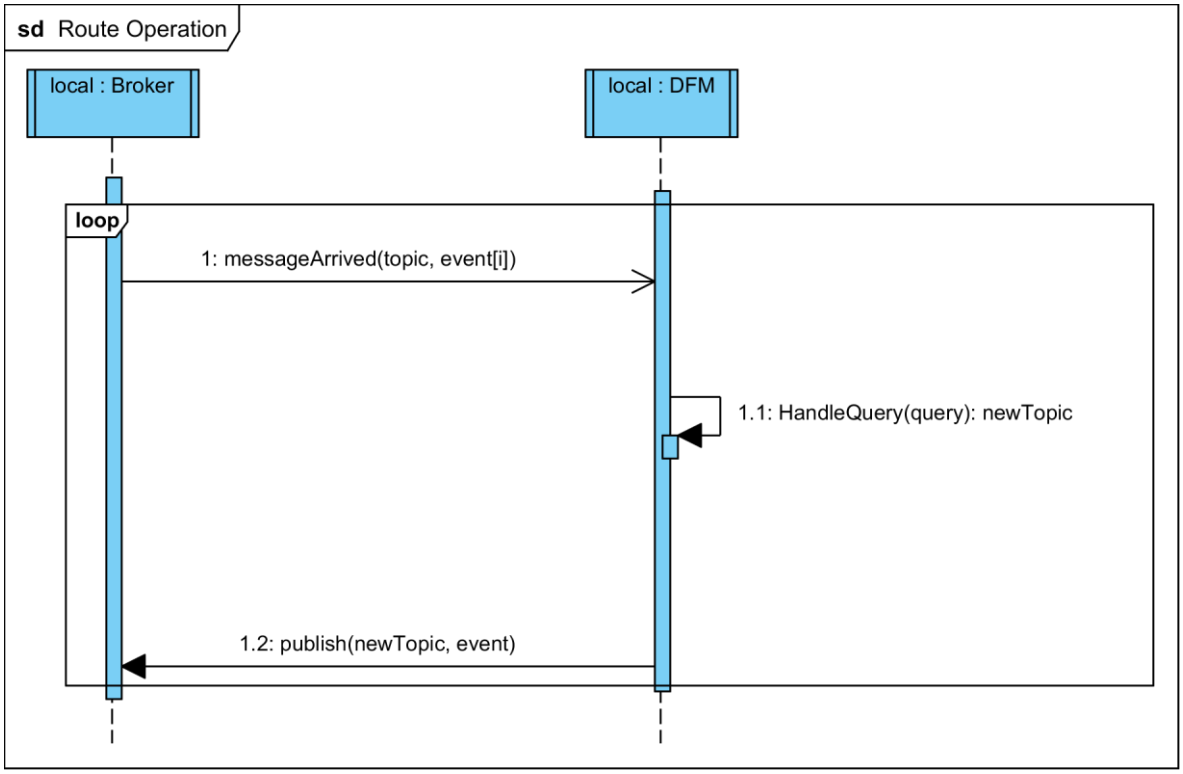
In this use case the request (which was obtained in the Data Management Request use case will be analysed, extracted and finally sent to the DFM through the Broker. Then any of the *Operation* use cases can start



Route Operation

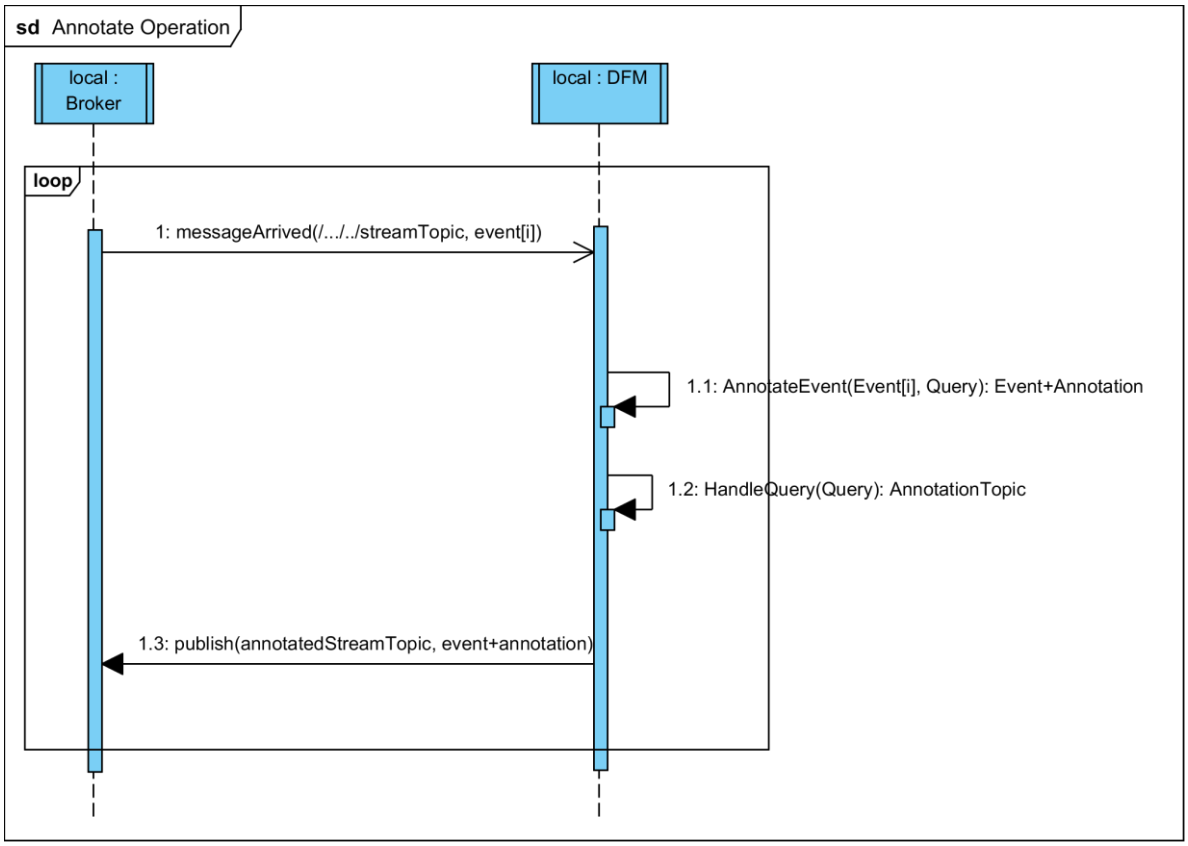
The Route Operation use case is the most basic operation regarding events or streams. It is a building block for other use cases intended for the management of single events or streams. In this manner, an event, or stream, can be routed so that a component triggers a particular behaviour. For instance, in the Storage Use Case this operation to notify the Storage Manager that one event or stream needs to be stored.

In the use case a route request arrived to the Data-Fusion Manager. Then the Data-Fusion Manager takes the event or events (stream) and republishes them in a new topic.



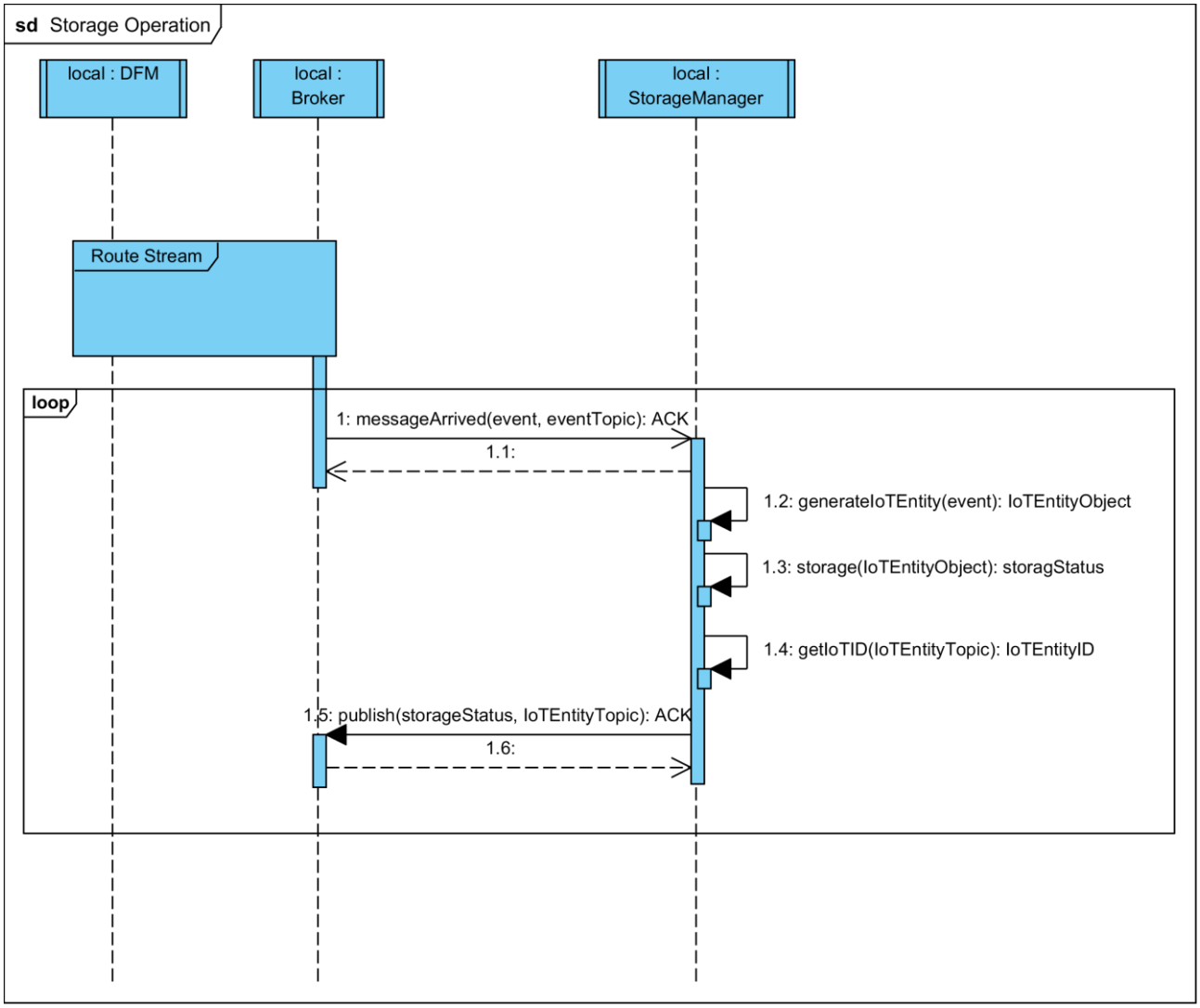
Annotate Operation

Similar to the Route Operation, the Annotate Operation is a building block use case. In the use case an event or an event stream is republished under another topic. Before being republished additional information might added to original event.



Storage Operation

This use case enables the platform to deploy storage rules, i.e., which events or streams should be stored. In this use case the request is received by the Data Fusion Manager which then re-routes the events under a topic the Storage Manager is listening to.



8.1.3 Federated Operations

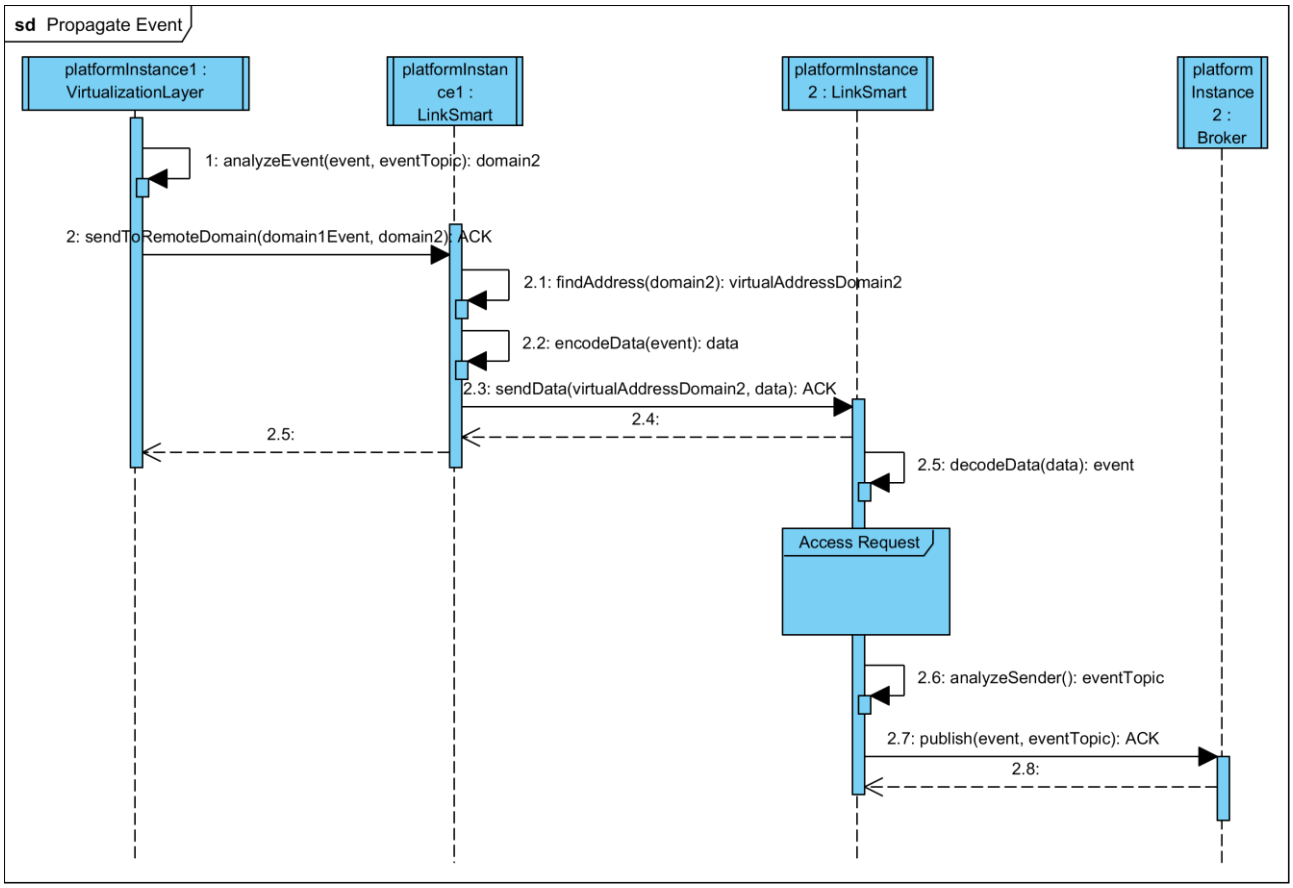
Federated operations are the use cases that concern communication between two or more ALMANAC platform instances. Interoperability between platform instances is exemplified by the use cases named Event Propagation and Broadcast Data Request.

The federal operations Publish Stream and Subscribe Stream leverage from the Data Propagation use case. Therefore the Event Propagation is presented in this chapter.

Event Propagation

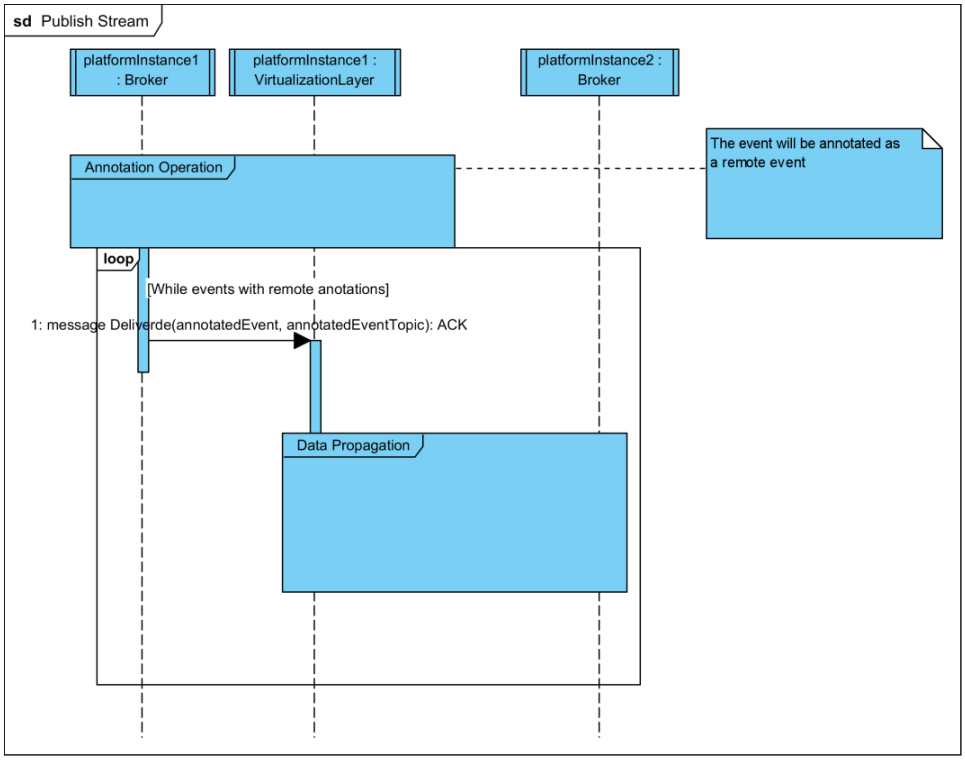
This sub use case describes the propagation of an event through the federation of more than one ALMANAC platform instances, i.e. the propagation of an event from one ALMANAC platform instance to another.

Event transportation is carried by LinkSmart given a Virtualization Layer request. LinkSmart transports the event from one platform instance to the other. On arrival of the event, the remote LinkSmart proves the access rights of the event. In case the event is allowed in the platform instance, it is published into the local Broker.



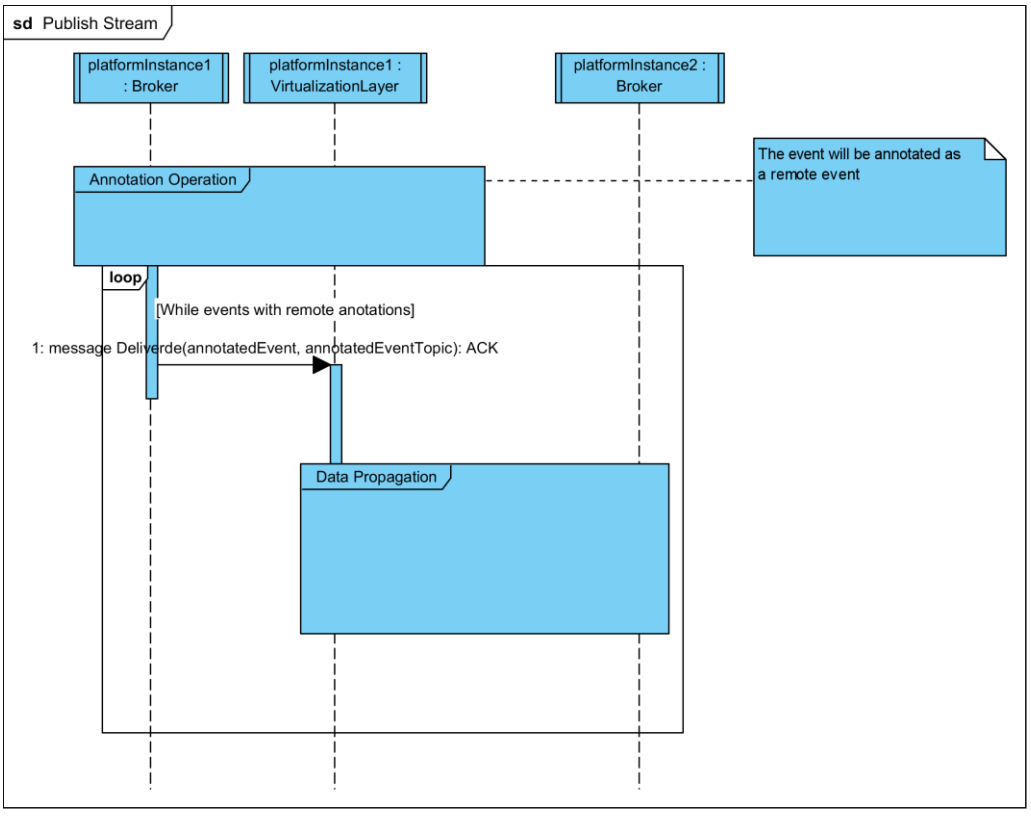
Publish Stream

This use case allows sending events from one ALMANAC platform instance to another. It is similar to the Route Operation use case but in this case the event will be routed through federation. The use case consists of two parts. First, the event to be published in a remote platform instance is annotated (as described in the Annotation Operation use case). Second, the Virtualization Layer starts to propagate the event (as described in the Propagate Event use case) relying on the annotation information.



Subscribe Stream

This use case allows receiving events generated in a remote platform instance. This use case is similar to the Publish Stream use case with the difference that the publication of the event is requested by the receiver and not by the producer. The use case starts when the receiver sends a publish request to a remote instance by propagating the request. Then if the request is accepted in the remote instance, the events will be published as in the Publish Stream use case.



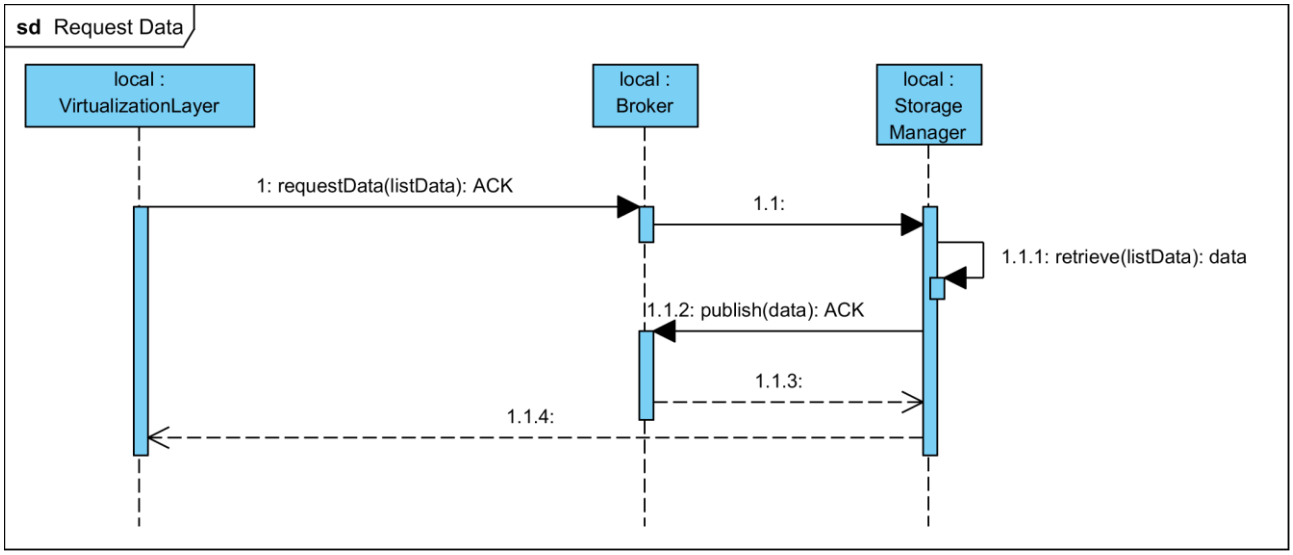
8.1.4 Data Querying

This use case is a special case which involves all aspects of the platform. In this use case the agent wants to query stored data which is distributed through federation. Due to the complexity of the use case it is split into two. Both use cases provide the same functionality, but their strategies are different. The result is that the Data Gathering use case is simple but less scalable whereas the Query Partitioning is more complex but also more scalable.

For the proper realization of the Data Querying use cases two sub use cases are introduced in this chapter: Data Request and Broadcast Data Request.

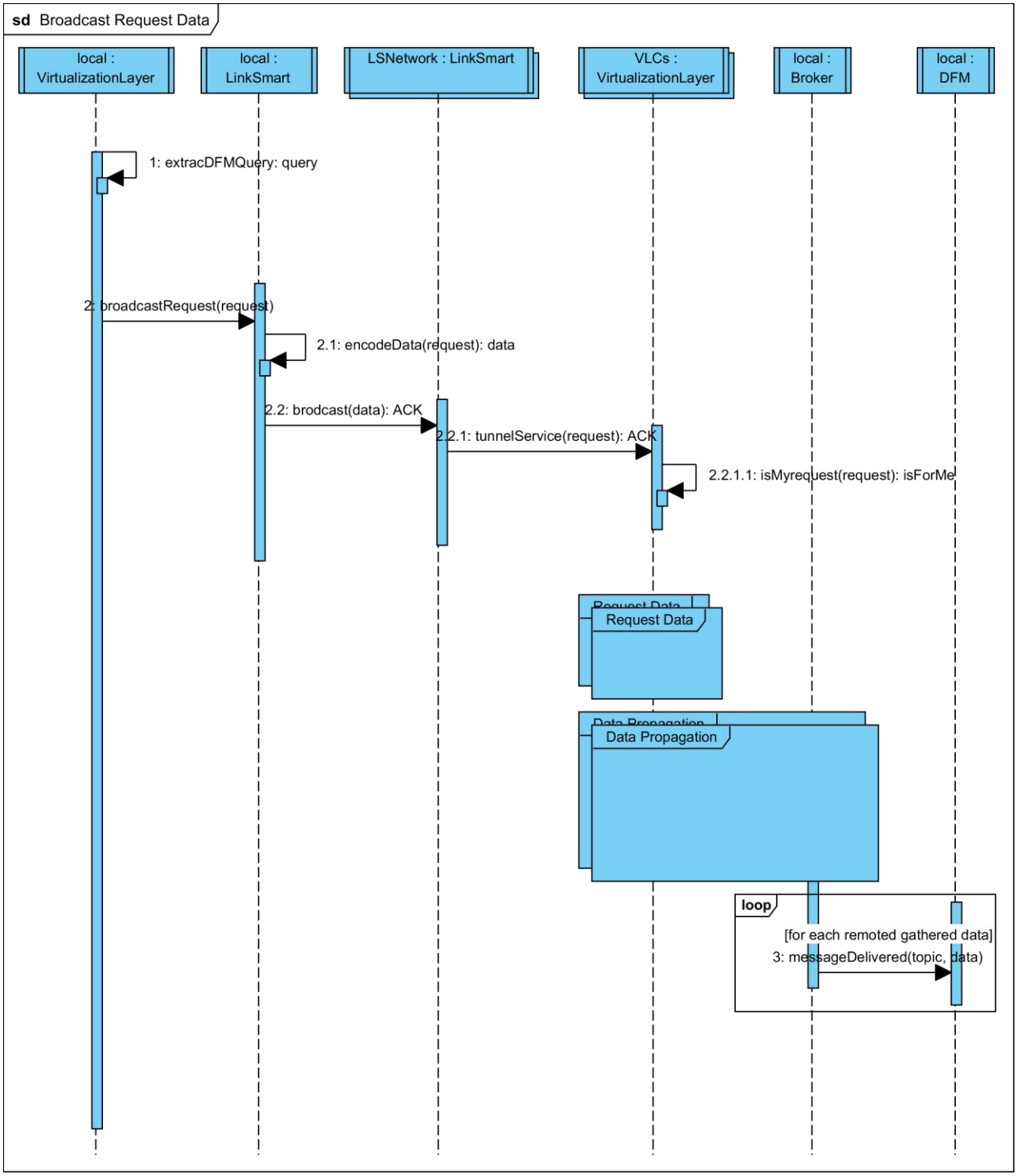
Request Data

This sub use case is started by the Virtualization Layer. It requests to the Storage Manager to publish data so that the data is available to other platform components.



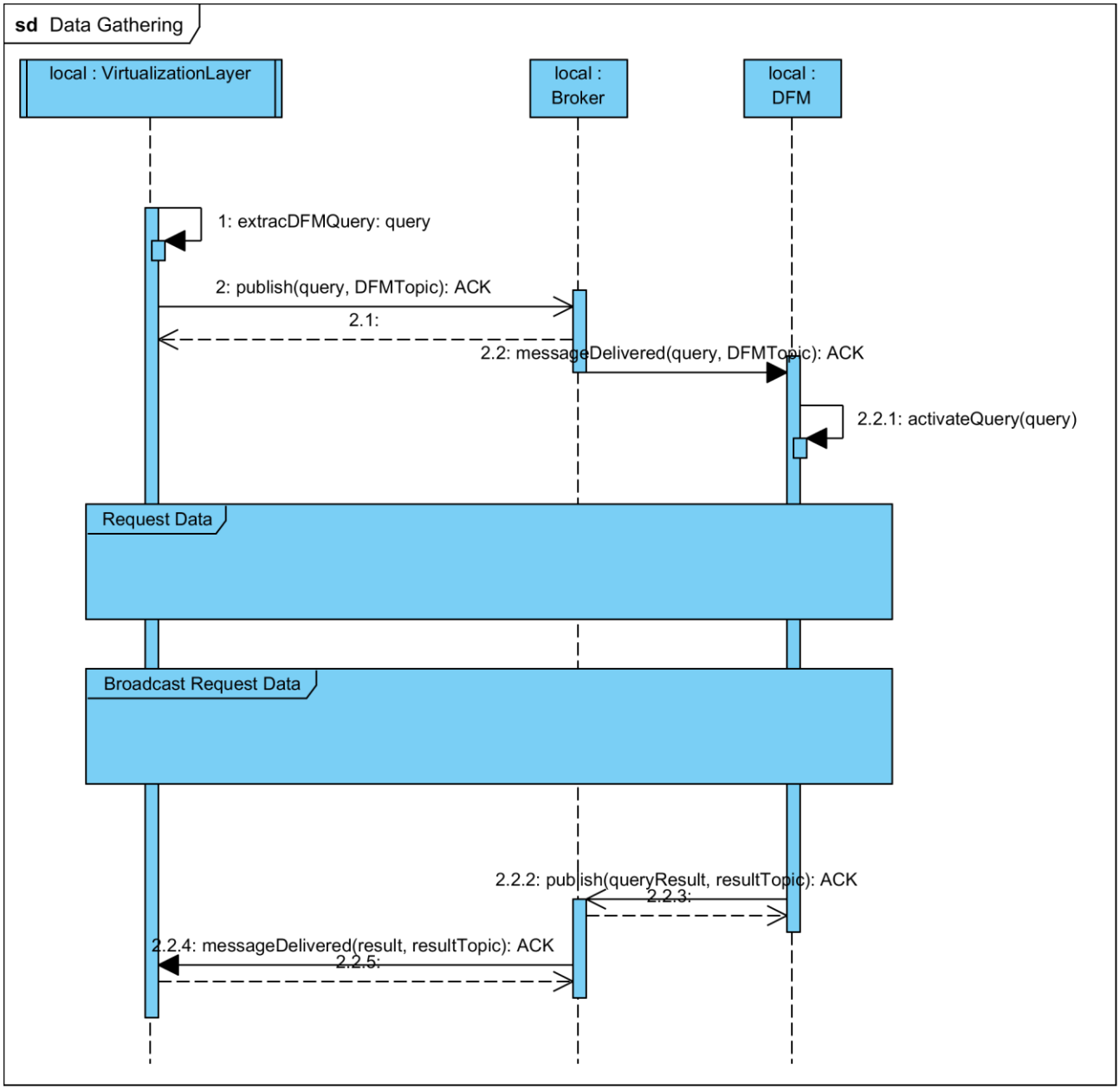
Broadcast Data Request

This sub use case is similar to the Request Data use case except that the data is requested by all federated ALMANAC platform instances. The Virtualization Layer broadcasts the data request through LinkSmart to each Virtualization Layer instance. Then each instance requests data from the Storage Manager to finally propagate the requested data back to the instance that initiated the original request.



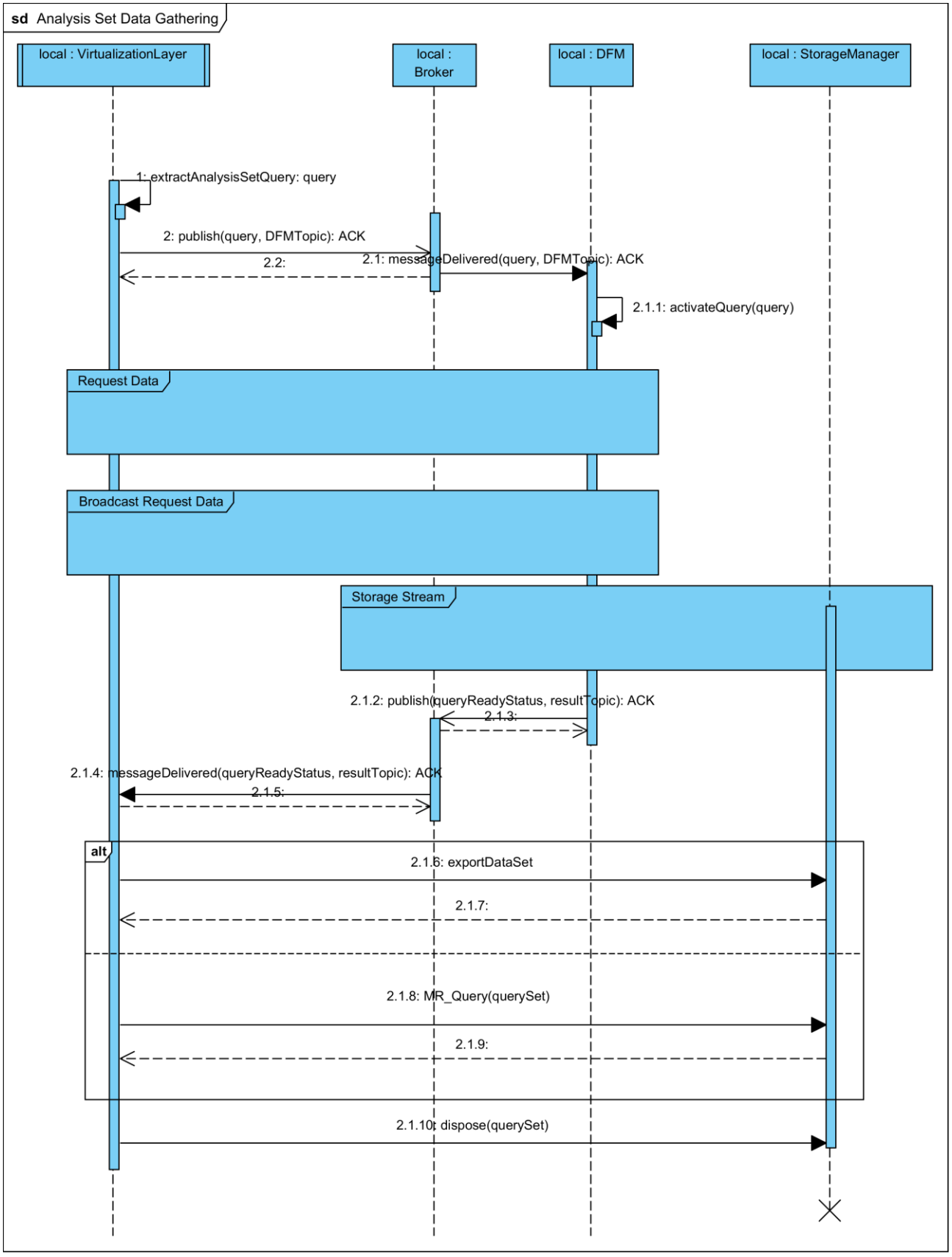
Data Gathering

This use case handles the Data Querying use case by gathering all data needed distributed in a federated network. While the data is being pulled, at the same time it is being queried by a continuous query. Therefore when all data is gathered in the platform the query produces the result expected by the "Agent".



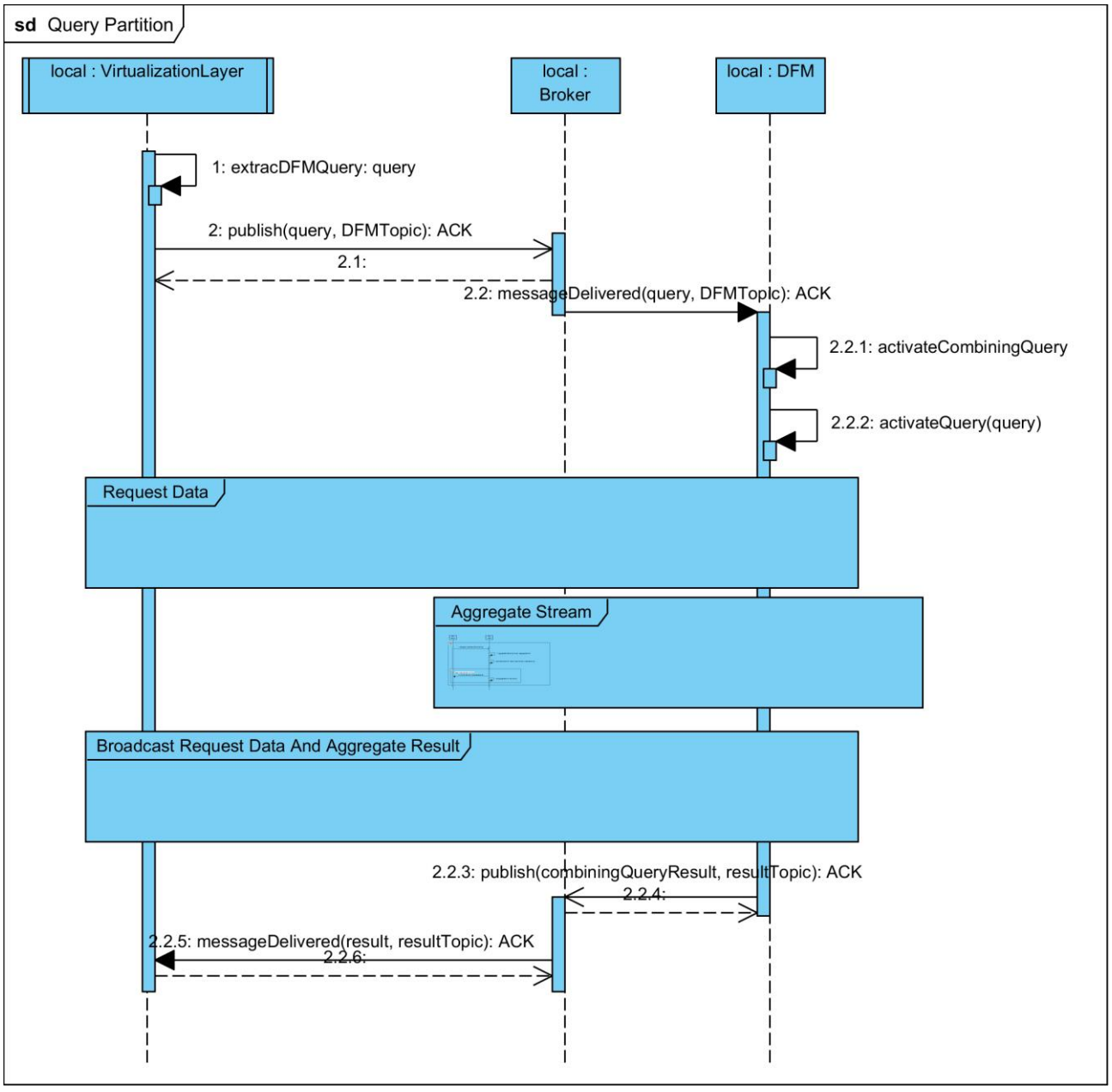
This use case makes it transparent to the agent where the requested data is located. Unfortunately, this solution may result in bursts of data, which may lead to performance issues. Therefore, this use case is currently designed for querying smaller amounts of data.

Analysis Set Data Gathering



There are a number of available tools to analyze big data sets, including statistical tools like R[25], Google BigQuery[26], Azure Machine Learning[27] as well as running MapReduce[28] jobs against the data set in a data store that supports this (e.g. MongoDB, Cassandra and Hadoop based data stores). To allow subsets of the ALMANAC data to be used for statistical analysis, to construct predictive models and for data mining, a subset of ALAMAC data may be stored as an analysis set. This use case is largely similar to the "Data Gathering" use case. The query is propagated and query results are sent back to originating platform instance. However, they are stored at a designated analysis storage at the originating platform instance. The analysis set at the designated storage may be used to run map-reduce queries and the data may also be exported to services like Azure ML, Google BigQuery, or analyzed using R scripts and similar tools.

Query partition



[25] http://www.r-project.org/
[26] https://cloud.google.com/bigquery/
[27] http://azure.microsoft.com/en-us/services/machine-learning/
[28] http://research.google.com/archive/mapreduce.html

This use case includes the Data Gathering use case with additional steps to distribute the workload. The query consists of three parts: the data request, the result set grouping query and the final combining query to be executed on the result sets. The final combining query resides on the originating node and reduces the intermediate results to the final result. The data request and grouping query are broadcasted to the other platform instances where the data is located. In parallel, on each platform instance, the data is requested from the SM and DFM maps it to a partial result set. This result is sent to the originating node, where the combining query is run over the result sets.

Example: We want to know the mean "FillLevel" for each subtype of the type "WasteBin" on all platform instances in a federation for the last month. The data request and grouping query are run on the originating node and broadcast to all other nodes. On each platform instance, for each the subtype, the number of observations, number of instances and the sum of values for the "FillLevel" is calculated (E.g., `{{"PaperBin", 35478, 1, 24835}, {"GlassBin", 45372, 3, 25460}}`) and sent back to the originating platform instance. At the originating platform instance, these partial results are combined to calculate the mean "FillLevel" for every subtype of "WasteBin" in the federation. (In MapReduce terms, there is only a single "reducer" here, the platform instance where the query originated. We could group the results by "WasteBin" subtype and assign one reducer per group, if this use of another ALMANAC platform instance's processing capacity is allowed by the federation agreement.)

9. References

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

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

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

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

(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.

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

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

(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.

(Rozanski & Woods, 2005) Rozanski, N., Woods, E. (2005), "Software Systems Architecture", ISBN: 0321112296.

10. Annex 1: Supporting infrastructure - LinkSmart

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

10.1 Introduction of LinkSmart

The LinkSmart middleware has been developed in the European project Hydra[29]. Hydra was a 4-year integrated project co-funded by the European Commission within the Sixth Framework Programme.

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

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

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

10.2 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 17 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.

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

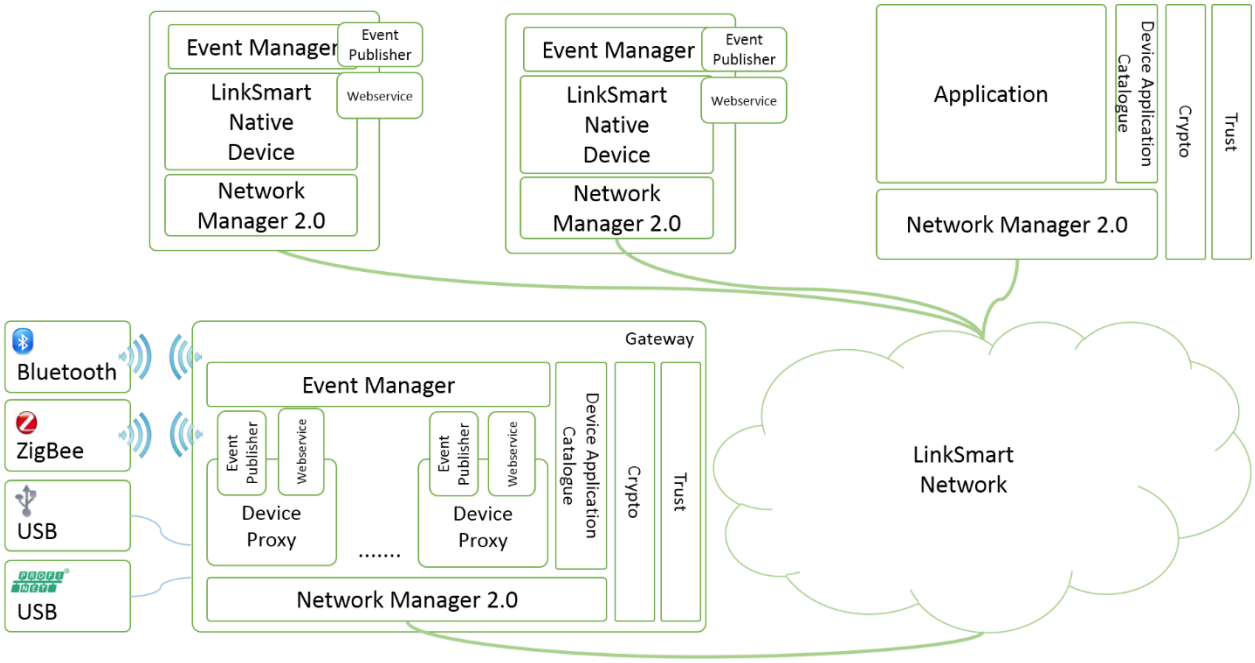


Figure 17: LinkSmart Example Concrete Deployment

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 18, 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.

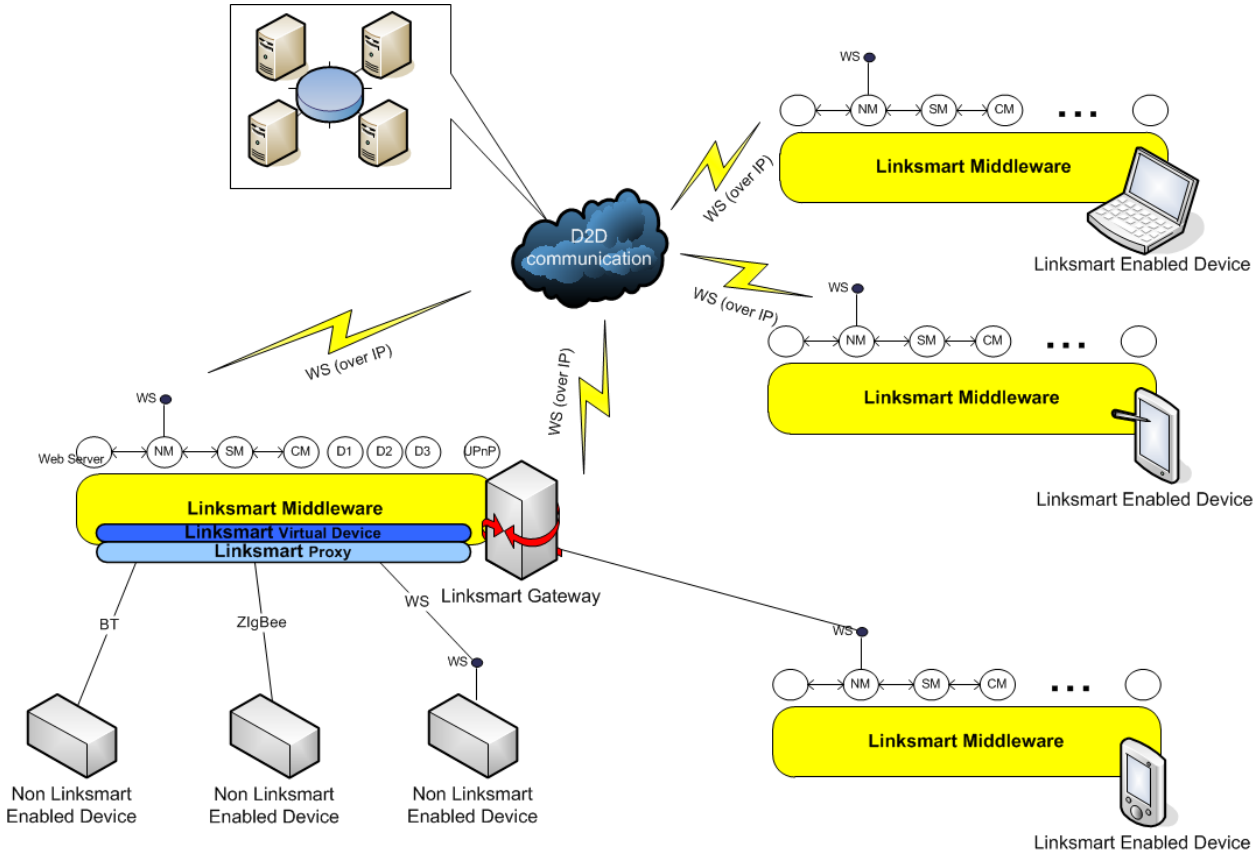


Figure 18: LinkSmart Example Device Network

11. Annex 2: IOT-ARM and ALMANAC

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

11.1 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.

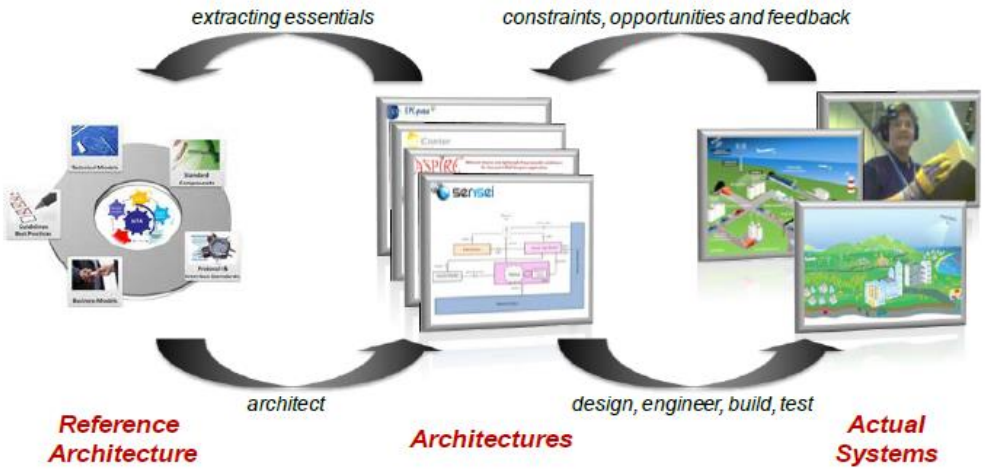


Figure 19: A reference architecture provides the instruments and guidelines for domain specific architectures from which specific system designs are derived.

A Reference architecture serves the following roles and usages,

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

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

11.2 Elements of the IoT ARM

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

The reference model framework was developed by the IoT-A[30] project and addresses IoT in terms of an overall IoT Architectural Reference Model including the subsets:

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

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

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

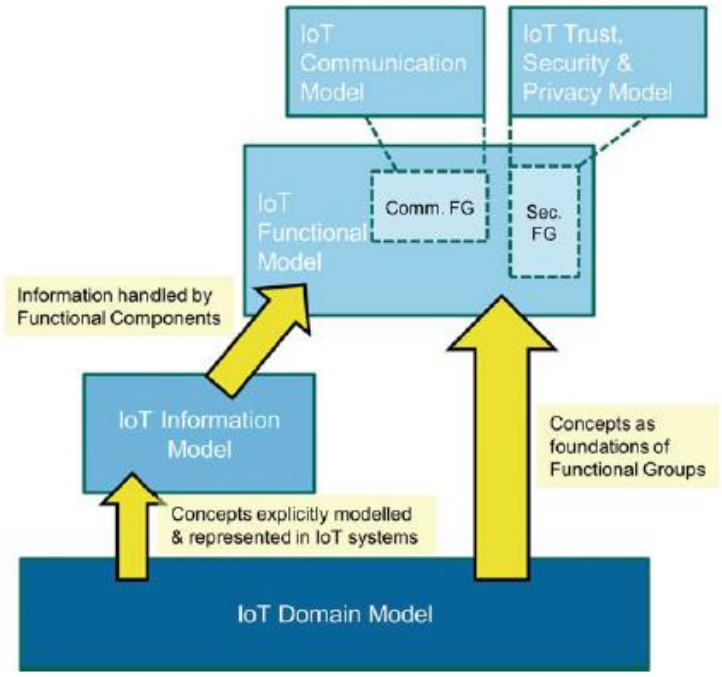


Figure 20: Sub-models of the IoT Reference Model (From (Bassi et al., 2013))

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.

11.2.1 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.

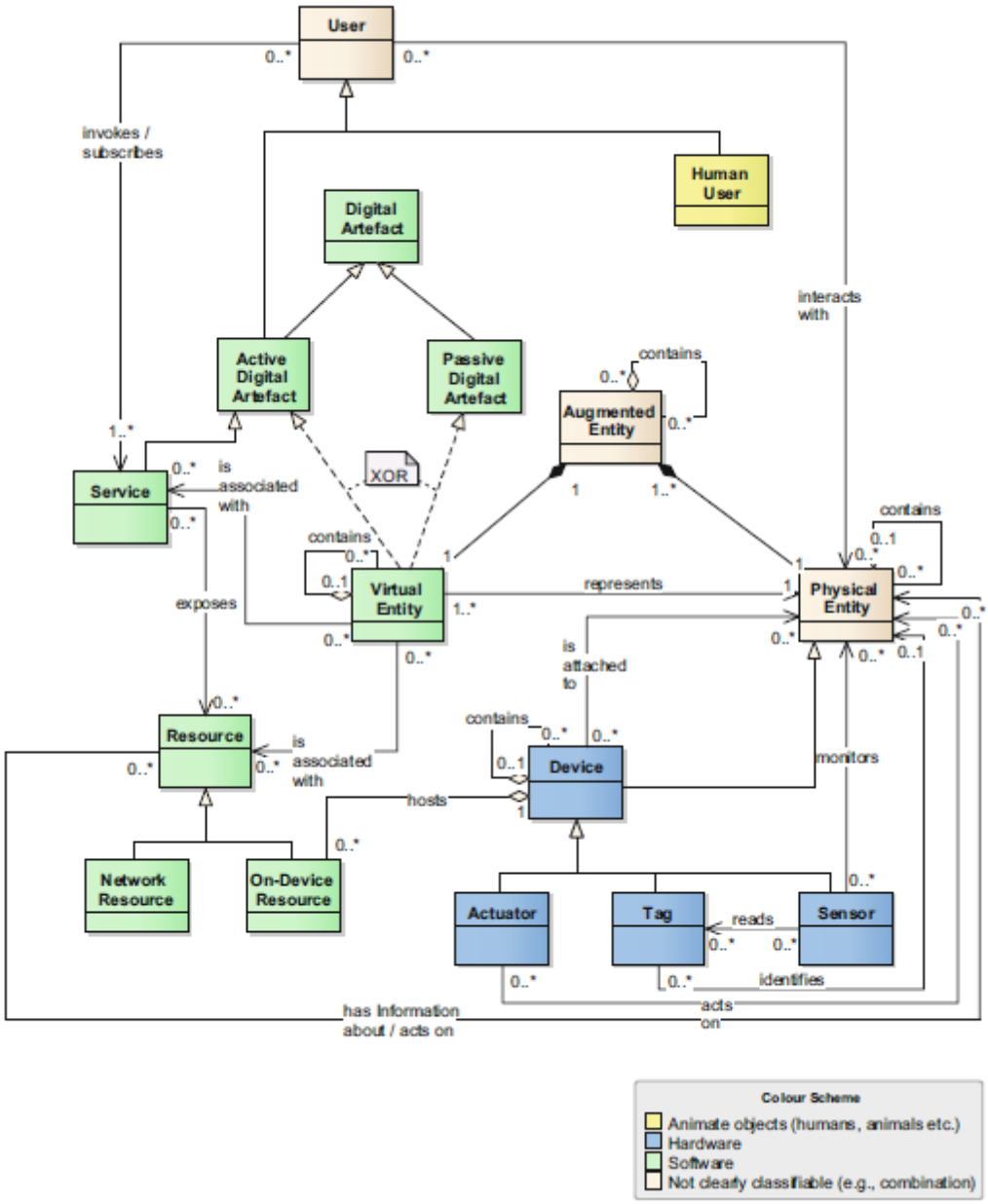


Figure 21: UML version of the IoT-A Domain Model

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

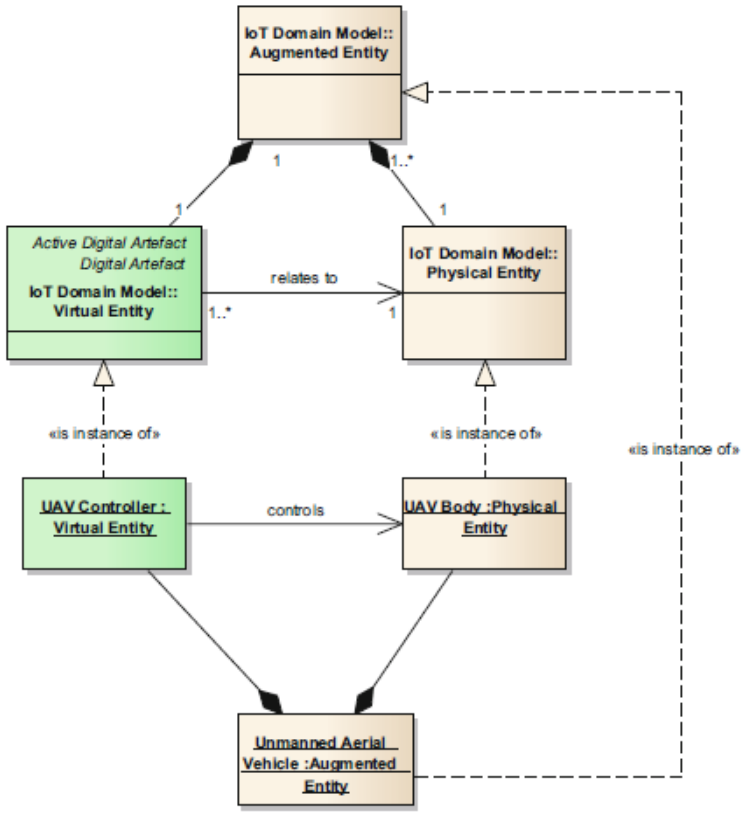
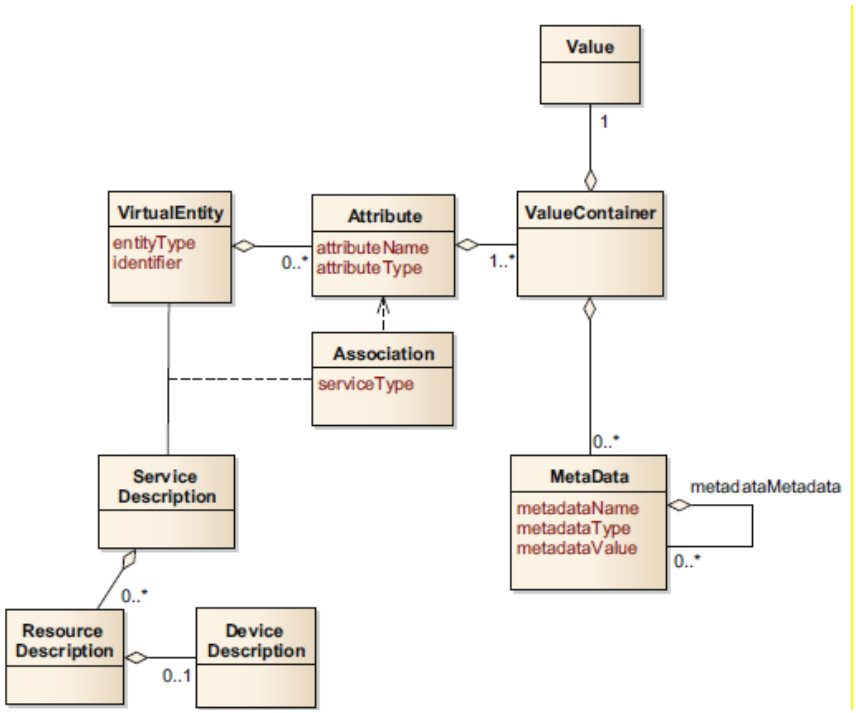


Figure 22: Example modelling using the Domain Model.

11.2.2 Information Model

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



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

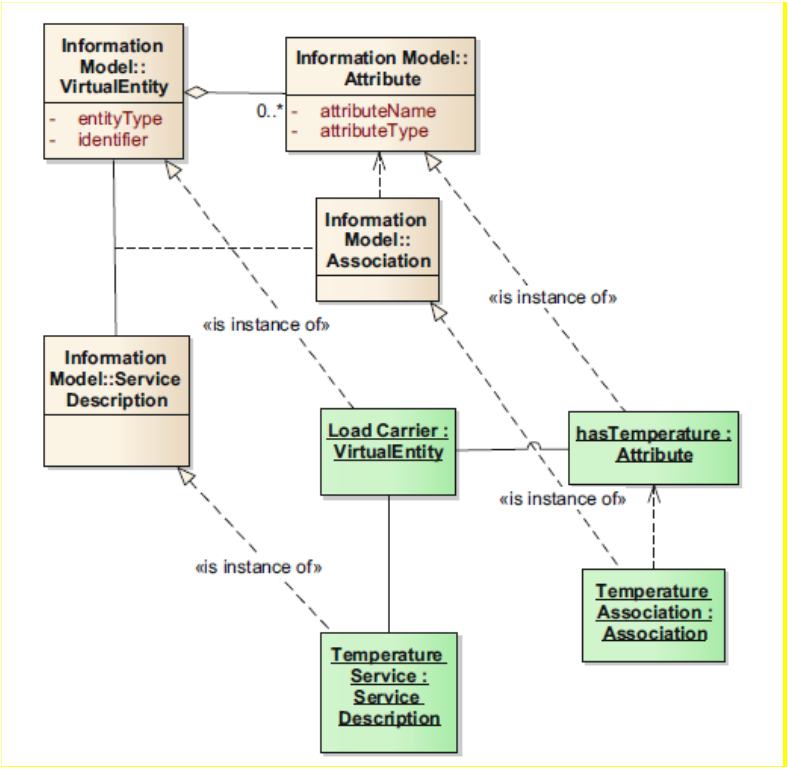


Figure 23: Example instantiation of the Information Model

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.

11.2.3 Functional model

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

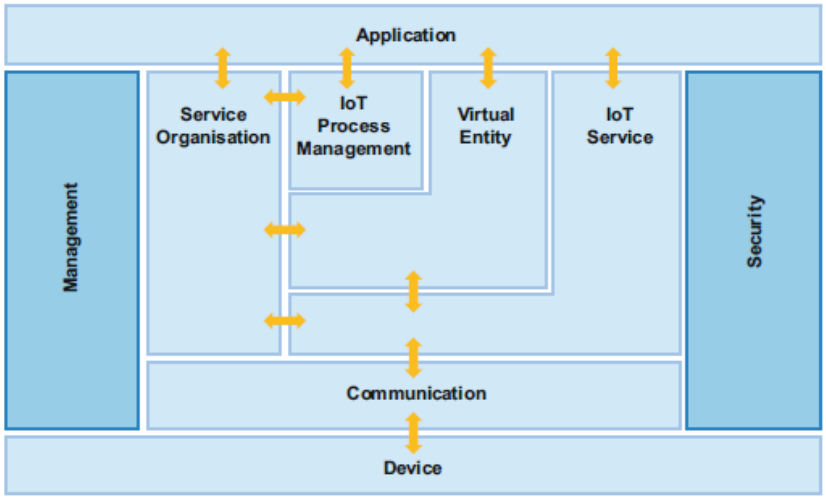


Figure 24: Functional Model

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

11.2.4 Views in the Reference Architecture

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

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

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

(Layered) Functional view

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

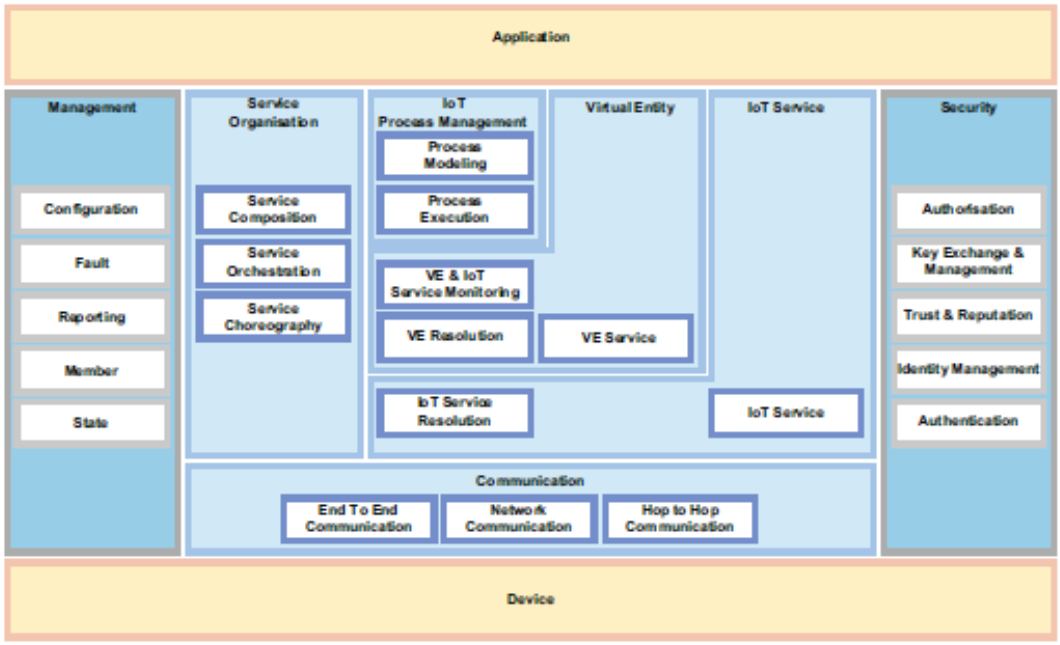


Figure 25: Function Groups

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

- IoT Service
- IoT Service Resolution

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

……..

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

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

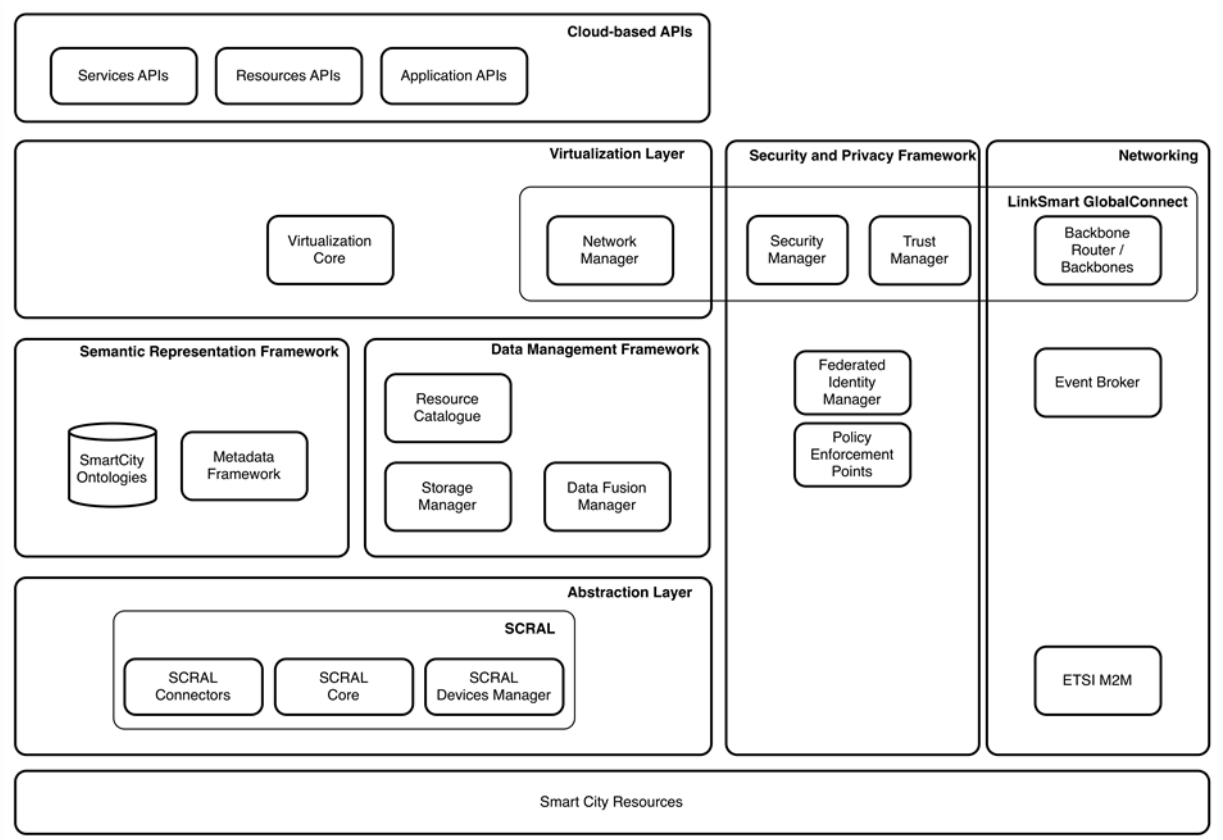


Figure 26: Function Groups in the ALMANAC architecture

Information view

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

Deployment and operation view

The following viewpoints are elaborated:

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

11.2.5 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

12. Annex 3: State of the art library

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

12.1 Devices

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

Name	Reference	Description	Relevance
Smart Bin	Producer website[31]	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[32]	List of prominent IoT hardware	Helps with technology scouting about IoT hardware blocks
Smart irrigation controllers	Producer website[33]	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[34]	Waste container monitoring service	ALMANAC could target this example for the waste management use case
Xtreme RFID	Producer website[35]	Asset tracking for Municipal Solid Waste[36] [37]	Consider compatibility for waste management use case
Water and leak detection sensors	Website[38]	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[39]	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)

[31] http://smartbin.com/how-it-works/smartbin-sensors.html
[32] http://postscapes.com/internet-of-things-hardware
[33] http://postscapes.com/smart-irrigation-controllers
[34] http://www.enevo.com/
[35] http://www.xtremerfid.com/applications/municipal-solid-waste
[36] http://www.xtremerfid.com/news/xtreme-rfid-helps-cincinnati-grand-rapids-enhance-recycling-and-waste-collection-operations
[37] http://www.rfidarena.com/2012/9/27/a-push-towards-recycling-with-rfid.aspx
[38] http://www.smarthome.com/_/Sensors/Water_Leak_Detection/_/L/1SD/nav.aspx
[39] http://www.thecitiesoftomorrow.com/solutions/waste/solutions/vacuum-waste-collection-systems

Urbiotica sensors	Producer website[40]	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[41].

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

Insteon sensors	Producer website[42]	A wide variety of sensors for the following applications: Remote control lighting, Control and status on smartphones and tablets, Remote control heating and air conditioning (HVAC), Scene lighting, Timers, Occupancy sensing, Leak sensing, Humidity sensing and control, Garage door sensing and control, Email and text (SMS) alerts, Access control (e.g. door locks), Audio-video control, Appliance management, Irrigation control, Energy measurement, Energy savings.	Useful for the waste and water use cases, specially: Occupancy sensing, Leak sensing, Humidity sensing and control.
Texas Instruments Sensors	Website[43] Water meter[44]	A variety of sensors, used in Texas smartgrid projects, f.ex. CenterPointEnergy Specific design sheet for water metering	Interesting web site with connections to especially SmartEnergy, but meters seem to have cross-domain architectural features (MeterUIDs etc.). A number of meters documented in different domains
Contiki OS	Website[45]	"The Open Source OS for the Internet of Things", with low footprint, and including advanced routing	OS to consider for sensors, and for interaction with LinkSmart
RIOT OS	Web site[46]	"The friendly Operating System for the Internet of Things", can run on ARM microcontroler boards such as Arduino Due	OS to consider for sensors, and for interaction with LinkSmart
Tiny OS	Website[47]	OS for low-power wireless devices, such as those used in sensor networks	OS to consider for sensors, and for interaction with LinkSmart

[42] http://www.insteon.com/index.html
[43] http://www.ti.com/lsds/ti/apps/smartgrid/metering/overview.page
[44] http://www.ti.com/solution/water_meter
[45] http://www.contiki-os.org
[46] http://www.riot-os.org
[47] http://www.tinyos.net

12.2 Systems

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

Name	Reference	Description	Relevance
IBM Intelligent Water	Website Leafet (PDF)[48]	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[49] Article[50]	Sewer system monitoring, analysis and control optimization. "EmNet's Real Time Intelligence and Optimization technology helps utilities maximize existing and planned resources to minimize overflows and save money. EmNet combines the use of traditional hydraulic modelling and novel real time information technology to deliver actionable insight."	Example/Competitor to water management application. Could also provide some ideas for the water management application.
Sewage Grid: Drifting Sensors that Monitor the Wastewater Collection System (Scientifc paper)	Sewage grid document[51]	They use wireless floating sensors to detect leakage in waste water systems.	Innovative solution to water system monitoring.

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

SeWatch - wastewater and sewage wireless monitoring system Company: Telematics Wireless[52]	Website[53]	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[54]	List of projects using IoT for city applications	Helps with technology scouting about IoT for cities
Enevo ONe Collect	Producer website[55]	Waste monitoring solution	A competing solution to the waste management use case
Hitachi's water infrastructure solution	Producer website[56]	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[57]	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)

[52] http://www.telematics-wireless.com/index.php?page_id=1
[53] http://www.telematics-wireless.com/index.php?page_id=138
[54] http://postscapes.com/connected-city
[55] http://www.enevo.com/one-collect/
[56] http://www.hitachi.com/products/smartcity/smart-infrastructure/water/solution.html#plink05
[57] https://m2m.telefonica.com/discover-m2m/smart-cities

CenterPointEnergy	Website[58] Security design[59]	The publicly owned non-profit electric transmission infrastructure owner in Texas defining a market for smart metering for competitive business	inspiration; mostly business

12.3 Services

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

Name	Reference	Description	Relevance
Xively (formerly Pachube.com)	Producer website[60]	Cloud service to upload and process IoT data	Example of data aggregation service for inspiration. Consider partial compatibility
IFTTT	Producer website[61]	If This Then That: Channels, triggers, actions... E.g. with SmartThings[62]	End-user empowerment
Node-RED	Producer website[63]	A visual tool for wiring the Internet of Things. Built on top of Node.js. Users can create data flows from IoT Devices to e.g. Twitter in a browser-based editor and deploy on small devices such as Raspberry Pi. ´	End-user empowerment
Google App Engine	Producer website[64]	Google's Platform-as-a-Service. Provides storage, load balancing and other services.	Cloud provisioning
Windows Azure	Producer website[65]	Microsofts Platform-as-a-service. Provides storage, load balancing and other services.	Cloud provisioning
Datahub	datahub.io	Publish and consume open data (based on the open source ckan software platform)	Publish data
OpenDataSoft datastore	public.opendatasoft.com	Cloud platform to publish and consume open data (using OpenDataSoft platform)	Publish data

[58] http://www.centerpointenergy.com/cehe/about/
[59] http://www.centerpointenergy.com/staticfiles/CNP/Common/SiteAssets/doc/123062_SmartMeterDataSecurity.pdf
[60] https://xively.com/showcase/
[61] https://ifttt.com/wtf
[62] http://smartthings.com/news/ifttt/
[63] http://nodered.org
[64] https://cloud.google.com/products/app-engine/
[65] http://www.windowsazure.com/en-us/

freeboard.io	freeboard.io	Visualisation of real-time (~1 second delay) IoT data with widgets, open source, works well with dweet.io as data input (Alex's example https://freeboard.io/board/536e2a2df1776c1c2e0001a0) and with PubNub.com's real-time network	Visualisation, user interfaces
dweet.io	dweet.io	Publish IoT data with a Twitter-like approach. Publish/subscribe	Publish/subscribe
OpenRemote	www.openremote.com	Open source middle-ware	Inspiration for middle-ware
PubNub	www.pubnub.com	Commercial real-time publish/subscribe service	Publish/subscribe
Busit	www.busit.com	Small French competitor of IFTTT. Interesting user interface	End-user empowerment

12.4 Research Applications

Examples: experiences from on-going projects, etc.

Name	Reference	Description	Relevance
EU projects on IOT	EU Project Website[66]	CORDIS search	Live list of projects regarding IoT funded by the European Commission
SmartSantander	EU Project Website[67]	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[68]	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

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

IoT.est	EU Project Website[69]	EU project: Internet of Things Environment for Service Creation and Testing	Abstraction, testing, semantic annotations
Mobosens	Website[70]	US research project, which provides citizens with a platform for collecting and sharing environmental data, from stream quality to drinking water safety	End-user involvement. Inspiration for the use-case about water
SmartOpenData	smartopendata.eu	EU project: Linked Open Data infrastructure (including software tools and data) fed by public and freely available data resources	Geospatial Data http://www.w3.org/2014/03/lgd/report
OpenIoT	openiot.eu	EU project: Open Source cloud solution for the Internet of Things	
UrbanWater	urbanwater-ict.eu	EU FP7 project: Almería, Spain, is installing smart meters in order to achieve greater efficiencies in water management http://cordis.europa.eu/news/rcn/122370_en.html	Water scenarios

12.5 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[71]	A large-scale pilot (6000 Smart Bins) in Groningen (NL). Citizen can unlock bins using an RFID badge + sensors are used to monitor the fill level.	It could be used as inspiration for the Smart Waste use case. Smart Bin could be considered for testing (if available on the market)
HydroPoint WeatherTrak	www.hydropoint.com/products/outdoor-solutions/	Measurement, irrigation control based on weather data, equipment for sensors, wired link & wireless comm., central adm server	Competing system
Sensus Intelligent Water management soluions	sensus.com/web/usca/products/water	Metering, Water system-level network solutions,	inspirational

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

Watersave SmartMeter	www.watersave.com.au/smartmeter/commercial	Australian Smart Water Solution	Inspirational – system descriptions. Management, system structure, Metering citizen interface
IntelliH2O smart water meter	www.intelli-h2o.com/products/technology		American – M2M wireless connectionand Management solutions to arrive
PCScale Tower Software	Website[72]	Software to map, route & dispatch trucks, track inventory, scale management & invoice for residential and commercial waste companies	inspirational
AccuTrax Mobile	http://www.accu-trax.com/images/accu-trax_pdf_low_res.pdf	Hardware and software solution for Outgoing business, with software to map, route, document, take pictures, maintain customer / citizen adm, etc.	inspirational
RapportFraStedet	rapportfrastedet.dk	A simple cross-domain, open-source issue reporting app developed in Denmark and shared by a larger number of Danish Municipalities. Available on web and in app stores	inspirational
WaspMote SmartWater	http://www.libelium.com/smart-water-sensors-monitor-water-quality-leakages-wastes-in-rivers-lakes-sea/	a Smart Water wireless sensor platform to simplify remote water quality monitoring	inspiration
CenterPointEnergy.com	www.centerpointenergy.com/cehe/about/, security design at http://www.centerpointenergy.com/staticfiles/CNP/Common/SiteAssets/doc/123062_%20SmartMeterDataSecurity.pdf	The publicly owned non-profit electric transmission infrastructure owner in Texas defining a market for smart metering for competitive business	inspiration

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

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

12.6 Standards

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

Name	Reference	Description	Relevance
OGC Sensor Web Enablement (framework)	OGC Sensor web enablement Group[73]	interoperability interfaces and metadata encodings that enable real time integration of heterogeneous sensor webs into the information infrastructure	Candidate framework for the various system interfaces in the ALMANAC architecture
W3C Semantic Sensor Network	W3C Incubator Group Report[74]	Ontology to describe sensors and sensor networks for use in sensor network and sensor web applications	Candidate standard for the ontology needs
ETSI M2M	ETSI M2M website[75]	ETSI M2M is the European standard for M2M (Machine to Machine) applications	Used in the ALMANAC architecture and developments
OneM2M	One M2M standards website[76]	OneM2M will be the worldwide standard for M2M (Machine to Machine) applications. It is going to include also ETSI. M2M	When available considered in the ALMANAC architecture and developments
CDMI	ISO standards catalogue[77]	Cloud Data Management Interface (CDMI), ISO/IEC 17826:2012, specifies the interface to access cloud storage and to manage the data stored therein.	WP6 cloud based storage management.
OMA M2M enablers	openmobilealliance.org/about-oma/work-program/m2m-enablers/	Overview of OMAs foreseen standardization efforts for LWM2M ("Light-Weight M2M	Device mgmt, SCRAL
EEBus	White paper[78]	Technology for comprehensive networking of devices and load management between power suppliers, grid operators and end users. Abstracts many existing protocols in a consistent IPv6/XML format. EEBus approach are also targeted at eHealth, Ambient Assisted Living and Security.	Platform partially competing with ALMANAC. Could be considered during interoperability testing, to replace some ALMANAC / LinkSmart parts.

[73] http://www.opengeospatial.org/projects/groups/sensorwebdwg
[74] http://www.w3.org/2005/Incubator/ssn/XGR-ssn-20110628/
[75] http://www.etsi.org/technologies-clusters/technologies/m2m
[76] http://www.onem2m.org/
[77] http://www.iso.org/iso/catalogue_detail.htm?csnumber=60617
[78] http://www.eebus.org/fileadmin/Mediapool/2011_06_EEBus_Whitepaper_en.pdf

HYPER/CAT	1248.io/casestudies/hypercat.pdf wiki.1248.io/doku.php?id=hypercat	HyperCat is designed for exposing information about IoT assets over the web. It allows a server to provide a set of resources to a client, each with a set of semantic annotations. Open, lightweight JSON-based hypermedia catalogue format for exposing collections of URIs, each with any number of RDF-like triple statements about it.	To be considered for exposing IoT semantic properties on the Web.
OASIS OData 4.0	OData Version 4.0 Part 1: Protocol[79] OData JSON Format Version 4.0[80]	Standards for an Open, Programmable Web https://www.oasis-open.org/news/pr/oasis-approves-odata-4-0-standards-for-an-open-programmable-web The Open Data Protocol (OData) enables the creation of REST-based data services, which allow resources, identified using Uniform Resource Locators (URLs) and defined in an Entity Data Model (EDM), to be published and edited by Web clients using simple HTTP messages. OData JSON Format extends the core specification by defining representations for OData requests and responses using a JSON format.	To be considered for data interoperability over an HTTP / REST interface
MQTT	mqtt.org	Machine-to-machine (M2M)/"Internet of Things" connectivity protocol, lightweight publish/subscribe messaging transport. Used e.g. by Xively. Clients available in several languages http://eclipse.org/paho/ . Under OASIS standardisation.	Example of popular M2M protocol to be compatible with
XMPP (Extensible Messaging and Presence Protocol) and its extensions	Overview[81] Publish/subscribe extension[82]	Decentralised, real-time communication, routing of XML messages	Consider adding an XMPP gateway to be compatible with third-party services based on it
PubSubHubbub protocol	code.google.com/p/pubsubhubbub/	Server-to-server webhook-based publish/subscribe protocol for any Web accessible resources. Short discussion on XMPP vs. PubSubHubbub http://documentation.superfeedr.com/subscribers.html#whatapitochoose	To be considered for instance to have a distributed network of Storage Managers / Virtualization Layers
JSON-LD (JavaScript Object Notation for Linked Data)	www.w3.org/TR/json-ld/ www.w3.org/TR/json-ld-api/	Method of transporting Linked Data using JSON APIs to expend/compact/flatten the data-structure to get different guarantees, transform from/to RDF,	To consider as ALMANAC's main data-interchange format, at least for semantic data

[79] http://docs.oasis-open.org/odata/odata/v4.0/os/part1-protocol/odata-v4.0-os-part1-protocol.html
[80] http://docs.oasis-open.org/odata/odata-json-format/v4.0/os/odata-json-format-v4.0-os.html
[81] http://xmpp.org/about-xmpp/technology-overview/
[82] http://xmpp.org/extensions/xep-0060.html#subscriber-subscribe

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

12.7 Reference Architectures

Name	Reference	Description	Relevance
IoT-A reference architecture	Website[83]	Describes the definition process and the initial functional blocks.	WP3
IoT-A terminology	Website[84]	List of concepts and terms used in the IoT reference model	WP3
IoT-A converged architectural reference model	Website[85]	The complete (converged) architectural reference model for the IoT (November 2012)	WP3
IoT-A Solutions for Privacy and Security	Website[86]	Details the security building blocks and their functionality in the IoT-A architecture.	WP3
IoT-A main book reference	DOI:10.1007/978-3-642-40403-0	Describes the IoT projects objectives & results, IoT ARM reference, usage guides and examples.	WP3
ISO/TC/268/SC1 Smart urban infrastructure metrics	Website[87]	ISO work on metrics for SC (also check CEN groups)	
UK Authority/ BSI	Website[88]	UK Smart City Vocabulary	
IoT Design Patterns	Website[89]	Interesting view discussing "Design Patterns" in IoT architecture	WP3
Eclipse SmartHome	eclipse.org/smarthome/ www.openhab.org	Open Source Services, Interfaces and Tools for IoT. Based on Java/OSGi. Targeting smart homes, but seems to overlap with some of the ALMANAC components	WP3
Intel IoT Platform	Website[90]	End-to-end reference architecture, including data collection, gateways, security, cloud storage, data retrieval, data visualisation	WP3

[83] http://www.iot-a.eu/public/public-documents/d1.2/view
[84] http://www.iot-a.eu/public/terminology
[85] http://www.iot-a.eu/public/public-documents/documents-1/1/1/D1.4/at_download/file
[86] http://www.iot-a.eu/public/public-documents/d4.2/view
[87] http://www.iso.org/iso/standards_development/technical_committees/other_bodies/iso_technical_committee.htm?commid=656967
[88] http://www.ukauthority.com/tabid/64/Default.aspx?id=4603#
[89] http://iot-datamodels.blogspot.it/2014/05/design-patterns-for-internet-of-things.html?m=1
[90] http://www.intel.eu/content/www/eu/en/internet-of-things/overview.html

12.8 Fi-WARE components

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

Name	Reference	Description	Relevance
Object Storage	FI-WARE Catalogue[91]	Provides robust, scalable object storage functionality through an open, standardised interface: it exposes a CDMIinterface on top of OpenStack Swift.	WP6 (cloud based) Storage Management
Cosmos - Big Data analysis	FI-WARE Catalogue[92]	Implementation of the Big Data GE, based on Hadoop ecosystem, including support for MapReduce	Test of usability of Hadoop & Map Reduce in ALMANAC data management
Wire Cloud	FI-WARE Catalogue[93] FI-WARE Mashup YouTube link[94]	Wirecloud is an open source, reference implementation of the FIWARE Application Mashup. Creating mash-ups by wiring existing components/widgets. REST	Consider for end-user creation of mash-up UIs to ALMANAC platform services.
Device Management	FI-WARE Catalogue[95]	A REST API for M2M application developers and a device communication API, receiving ETSI M2M events (and other protocols) together with historic info.	WP4 to decide on relevance
REST for OMA NGSI 9/10	Website[96]	FI- ware version of the OMA NGSI 10 interface which is a RESTful API via HTTP. Its purpose is to exchange context information. The three main interaction types are • one-time queries for context information • subscriptions for context information updates (and the corresponding notifications) unsolicited updates (invoked by context providers)	ALMANAC architecture, WP3,4,5,6

[91] http://catalogue.fi-ware.eu/enablers/object-storage-ge-fi-ware-implementation
[92] http://catalogue.fi-ware.eu/enablers/bigdata-analysis-cosmos
[93] http://catalogue.fi-ware.eu/enablers/application-mashup-wirecloud
[94] http://www.youtube.com/watch?v=yzQqstBAUeo#t=1
[95] http://catalogue.fi-ware.eu/enablers/backend-device-management
[96] http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/OMA_NGSI_10