ALMANAC

RELIABLE SMART SECURE
INTERNET OF THINGS FOR SMART CITIES

(FP7 609081)

# D4.2 Features of the ALMANAC Platform for sustainable Smart City applications

## Submission Date 29th February 2016 – Version 1.0

**Published by the ALMANAC Consortium**

**Dissemination Level: Public**

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

# Document control page

**Document file:** D4 2 Features of the ALMANAC Platform for sustainable Smart City Applications.docx
**Document version:** 1.0
**Document owner:** FIT

**Work package:** WP4 – IoT Heterogeneous Network Infrastructure
**Task**: T4.2, T4.3
**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 | Marco Jahn | 2015-12-01 | Initial structure |
| 0.2 | Marco Jahn | 2016-01-07 | Restructure, finalize ToC, Baseline scenario |
| 0.3 | Marco Jahn, Angel Carvajal Soto | 2016-01-14 | Data Management features, Scenario evolution |
| 0.4 | Raphael Ahrens | 2016-01-19 | Security features, Scenario evolution |
| 0.5 | Raphael Ahrens, Marco Jahn | 2016-02-03 | Developer features, update of security/federation scenarios |
| 0.6 | Dario Bonino, Marco Jahn | 2016-02-17 | Finalization of resource abstraction features and scenarios |
| 0.7 | Raphael Ahrens | 2016-02-22 | Finalized federation scenario |
| 0.9 | Marco Jahn | 2016-02-23 | Ready for review |
| 0.91 | Marco Jahn | 2016-02-25 | Integrate reviewers comments |
| 1.0 | Marco Jahn | 2016-02-29 | Ready for submission |

**Internal review history:**

| Reviewed by | Date | Summary of comments |
|---|---|---|
| Matts Ahlsén (CNET) | 2015-02-24 | Approved |
| Roberto Gavazzi (TIL) | 2015-02-25 | Approved with minor comments |

# Index:

**List of Figures** ..... **4**
**Terminology** ..... **5**
**1. Executive summary** ..... **6**
**2. Introduction** ..... **7**
2.1 Purpose, context and scope of this deliverable ..... 7
2.2 Background ..... 7
**3. Features of the ALMANAC Platform for sustainable Smart City Applications** **8**
3.1 Abstraction and virtualization of resources and networks ..... 8
3.2 Data Management ..... 8
3.3 Federation and Associations ..... 9
3.4 Policy and Security Management ..... 9
3.5 Application development support ..... 9
3.6 Platform Instance Monitoring ..... 10
**4. Scenarios for sustainable Smart City Applications** ..... **12**
4.1 Baseline Scenario ..... 12
4.2 Adding a new sensor networking technology ..... 13
4.3 Integrating data sources ..... 13
4.4 Securing access to IoT resources ..... 14
4.5 Creating an Association (Access control in federated deployments) ..... 19
**5. Conclusion** ..... **21**

# List of Figures

Figure 1 - Platform Instance Monitoring Page ..... 10
Figure 2 - Deployment Diagram of Baseline Scenario ..... 13
Figure 3 - Updated Deployment Diagram including Access Manager and Federated Identity Manager ..... 15
Figure 4 - Policy for granting a user access to a resource ..... 17
Figure 5 - Request to the Access Manager ..... 18
Figure 6 - TRN and AMIAT Platform Instances ..... 19

# 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 short range networks 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 802.15.4 at different frequencies, usually: 169 MHZ, 868 MHZ, 2.4 GHZ. Capillary networks are used for 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 middleware platforms and eventually with the Apps by the M2M Platform with standard APIs (http REST 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 how the ALMANAC Platform supports the development, deployment, and operation of sustainable applications for Smart Cities. The term *sustainable* in the scope of this deliverable refers to extensibility, scalability and the capability to adapt to changing conditions.

The deliverable starts with an overview of the features of the ALMANAC Platform that allow the development of such sustainable applications for Smart Cities, namely, abstraction and virtualization of data communication networks and resources, data management, federation and associations, and application development support.

Then, this deliverable describes how ALMANAC supports sustainability of Smart City Applications, using an evolving scenario. The baseline scenario describes a simple case consisting of a single Platform Instance, which will be gradually extended to more complex scenarios, explaining how the ALMANAC Platform adapts to these extensions and keeps the existing deployments sustainable.

The scenario evolution is as follows:

- The baseline scenario describes a deployment of an ALMANAC Platform Instance in the city of Turin that involves smart bins with fill-level sensors. In the first iteration, a new sensor technology should be added to the deployment. In addition to fill-level sensors based on ETSI M2M, we now want to add also new sensors based on different technologies. This iteration describes the case where the necessary SCRAL drivers are available.
- The next iteration describes the integration of an existing data source: We show how ALMANAC integrates an existing Xively deployment consisting of smart bins. Compared to the previous scenario iteration, this example describes a more complicated case where SCRAL drivers are not available for download but have to be implemented.
- Having integrated various heterogeneous data sources into the Turin ALMANAC Platform Instance, the next step is to restrict access to authorized users. The scenario is extended by the idea of giving users access to the data of their own waste bins. The necessary policies are created.
- In the last step, the scenario is extended by adding another ALMANAC Platform Instance. The city of Turin wants to share information with the local waste management company. Therefore, another Platform Instance is needed to allow federated data exchange between the two. Policies need to be defined to specify access control.

The evolution of the scenario exemplifies how the ALMANAC Platform supports scalable and sustainable growth and adaptability under changing conditions. For developers of Smart City Applications that only interface with the Cloud-based APIs, these changes are completely transparent, i.e. their applications stay sustainable.

# 2. Introduction

## 2.1 Purpose, context and scope of this deliverable

This deliverable describes the features of the ALMANAC Platform for sustainable application development. Using an evolving scenario, the document explains how the ALMANAC Platform can be deployed and how it stays sustainable under changing requirements. Starting from a simple, basic scenario, increasingly complex requirements are introduced. We describe how the ALMANAC Platform stays sustainable, adding heterogeneous technologies/data sources, access control and federated communication.

The focus of this deliverable is on the actual deployment. The architecture and specification of the ALMANAC Platform as a whole and also the various components are described in great detail, e.g. in Deliverable D3.1.3 – System Architecture Analysis & Design Specification 3. Here we change the viewpoint towards applicability and features regarding sustainable, federated, and secure deployments for application development.

The document is structured as follows: Chapter 3 describes the features of the ALMANAC Platform for sustainable application development. Chapter 4 describes the evolutionary scenario and for each step how ALMANAC supports the evolution and stays sustainable. The deliverable concludes with Chapter 5.

## 2.2 Background

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

# 3. Features of the ALMANAC Platform for sustainable Smart City Applications

In this chapter we provide an overview of the main features of the ALMANAC Platform for sustainable Smart City Applications. For each features or set of features it is described how they support sustainability for the platform and for application developers.

## 3.1 Abstraction and virtualization of resources and networks

In order to support Smart City application development, independence from sensing/actuating technology and network technology is crucial, as it relieves the design and development process from tackling all small nuances of communication, protocols and interaction paradigms typical of lower-level networks and devices. The Smart City Resource Adaptation Layer (SCRAL) enables applications to use, and rely on one language only, namely the ALMANAC Cloud APIs.

While the developer interfaces devices by means of standard and shared data structures, and well defined paradigms such as REST, the SCRAL takes care of harmonizing data, converting protocols and handling different low-level communication standard, from serial ports to M2M services.

The SCRAL mainly works under the hood, transparently handling / delivering data to/from the city. In this sense it might be considered less involved in providing sustainability to smart city applications. However, its abstraction role is exactly the cornerstone upon which applications are built, independently from the underlying smart city infrastructure.

Depending on the end-user nature, e.g. if he is third-party developer or city service integrator, sustainability of smart city applications might include also aspects of technological spread. The latter user typology is in fact enabled by the SCRAL to add support to a virtually unlimited number of sensors and actuators, thanks to its driver-based structure, which supports easy integration of new technologies into the smart city infrastructure.

In summary abstraction, and virtualization, of resources provided by the SCRAL can be seen as the foundation for developing sustainable, portable and scalable smart city applications that can easily cross the boundaries of single municipalities, or nations.

## 3.2 Data Management

Data management features are crucial to IoT applications, since large amounts of sensor data are becoming available. To make sense out of these data and develop useful applications, further processing is required. ALMANAC provides a rich set of data management features.

Regarding data stream handling, ALMANAC provides the following features:

- Streams Annotation: Adding context data to the streams, implemented by the Data Fusion Manager (DFM).
- Streams Aggregation: Combining streams to generate richer events, implemented by the DFM.
- Streams Routing: Transferring events from one ALMANAC Platform Instance to another, implemented by the DFM and LinkSmart GlobalConnect, and coordinated by the Virtualization Layer Core.
- Streams Storage: Selecting and storing streams, implemented by the DFM and Storage Manager.
- Real-Time Automatic Pattern Discovery: Stream mining, machine learning feature of ALMANAC platform. Implemented by Complex-Event Machine Learning (CEML) in the DFM.
- Machine Learning Chains for Pattern Discovery: Advanced feature of the DFM with CEML framework for creating automatic learning using already learnt knowledge.

Furthermore, ALMANAC provides persistence for time series data. It provides interfaces for storing and querying time-stamped events and measurements. The Storage Manager can be adapted to varying requirements. E.g. one platform may have a limited budget and limited storage needs, another might need to store huge amounts of historical data for analysis, and a third may already have and in-house storage solution set up and want to keep the data there. Furthermore, the ALMANAC Storage Manager features distributed data storage across multiple Platform Instances.

## 3.3 Federation and Associations

Federation features of ALMANAC allow different Platform Instances to transparently and securely exchange data and interface with services and resources.

Technically, federation is enabled by LinkSmart GlobalConnect, which establishes an overlay network that interconnects Platform Instances. In that way, resources and services are made available to all federation members and can be found through the Resource Managers. The Virtualization Layer Core (VLC) ensures the routing of requests to the right Platform Instance and endpoints.

ALMANAC allows federation members to create Associations, i.e. specifying and controlling access to their resources and services based on mutual agreements. This is done by policy and security management, which is described in the next section.

## 3.4 Policy and Security Management

With ALMANAC every PI can configure which Federated Identity Managers (FIM) it trusts to authenticate users. This allows the entity which operates the PI to define its own users with its own FIM. But it also allows the PI to accept requests from users which are registered at another FIM from a federated entity.

Through the use of the Access Manager (AM) it is possible to have fine-grained access control for all interactions a PI has. In combination with the user attributes (user role, FIM ID, log in time, ...) the policies of the AM become quite a powerful feature, because the PI gets full control on whom it trusts how. For example, with the policies the PI operators could restrict the access for a user group which has been registered at a specific FIM to only access stored data that was collected by a specific sensor.

The policies, the users and their attributes, and the trusted FIMs can be freely configured and the PI is fully autonomous in these decisions. This allows the carrier of the PI to react to changes by opening interfaces to new federation partners, without giving away the keys to the castle. The PI carrier can even allow the new partner to manage there own user base without risking to disclose information.

By default all information inside a PI is treated as confidential and cannot be accessed, except when the operator of the PI has created a policy and thereby give access to authenticated users

## 3.5 Application development support

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.

The following collection of Cloud-based APIs is available:

| Name | Description |
|---|---|
| Smart City Resource Library API | The Smart City Resource Library Services API (SCRLS API) s used to query for IoT Resources and Things based on metadata. |
| Historical Data API | The Historical Data API provides access to stored observations in data streams (time series data) from sensors or data fusion queries. |

| | |
|---|---|
| Live Data API | The Live Data API is used to query resources for data directly and subscribe to events from sensors or data fusion queries using web sockets. |
| Data Fusion Services API | The Data Fusion Service API allows users to define data fusion tasks by using a Data Fusion Language. |
| Provisioning API | The Provisioning API allows to connect IoT Resources to semantic metadata in the Smart City ontologies. |
| Management API | The Management API provides the means to secure ALMANAC Platform Instances and IoT Resources through user management and policies. |

The Cloud-based APIs are a key feature for supporting sustainability of ALMANAC deployments. Regardless of underlying heterogeneity of the integrated technologies and systems, developers only work with the Cloud-based APIs in a unified way.

Detailed descriptions of the Cloud-based APIs is provided in D7.3 - Cloud-based APIs for Smart City Applications – Developer's Guide.

## 3.6 Platform Instance Monitoring

To foster sustainability of ALMANAC deployments, monitoring and management of a Platform Instance is necessary.

The PI Monitoring Page allows the PI operator to monitor a deployed ALMANAC PI in terms of the services and their availability.

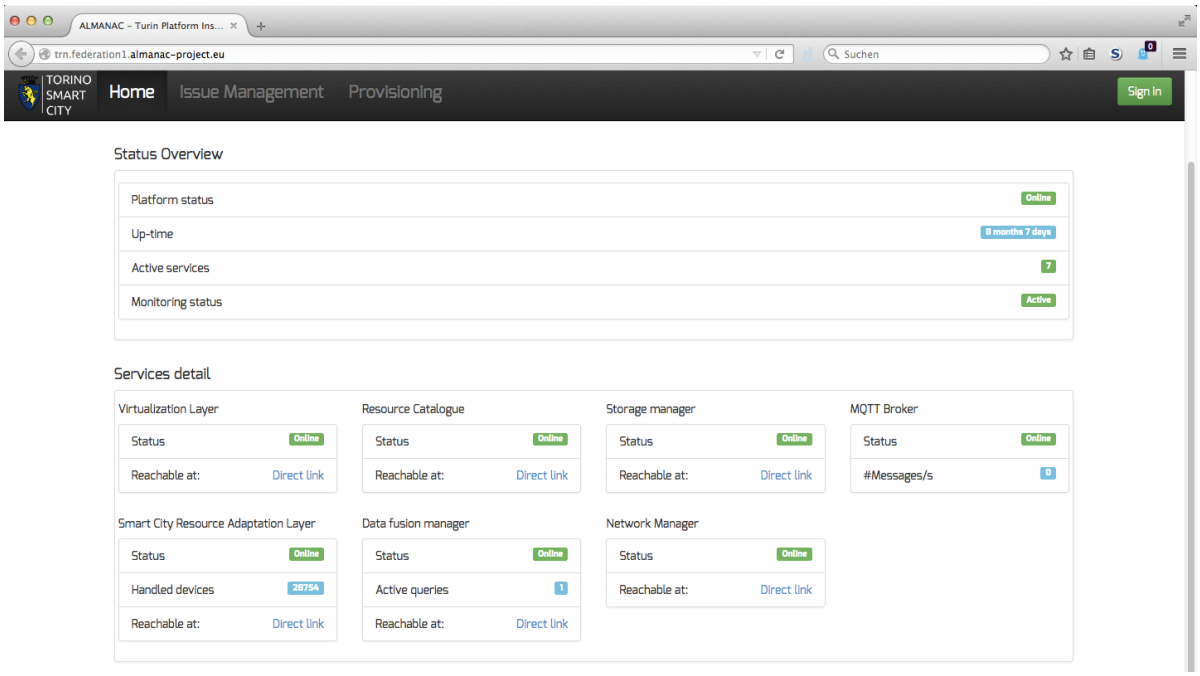


Figure 1 - Platform Instance Monitoring Page

Figure 1 shows the PI Monitoring Page of the TRN Platform Instance, i.e. the ALMANAC Platform Instance owned and managed by the city of Turin.

In the Status Overview it indicates if the PI is online and the related uptime. Furthermore, it shows the number of active services deployed in this PI.

The Services Detail section provides further details on each of the deployed services, showing if it is online and a link to the endpoint. For SCRAL the number of handled devices is shown and for the Data Fusion Manager the PI operator can see the number of active queries.

# 4. Scenarios for sustainable Smart City Applications

This chapter describes different scenarios for sustainable Smart City Applications. Starting from a baseline scenario, various extensions and changing requirements are introduced, explaining how the ALMANAC Platform deals with these changes and how the sustainable evolution of the Smart City Applications is supported by ALMANAC.

The baseline scenario describes a simple case consisting of a single Platform Instance, which will be gradually extended to more complex scenarios, explaining how the ALMANAC Platform adapts to these extensions and keeps the existing deployments sustainable.

## 4.1 Baseline Scenario

In this scenario, the city of Turin owns public waste bins, which are outfitted with fill-level sensors. Concentrators for data collection are deployed throughout the city, acting as gateways to the ALMANAC Platform.

The city of Turin deploys an ALMANAC Platform Instance that provides access to the data collected from the fill-level sensors. These data should be open to developers to implement innovative smart city applications.

The prerequisite is that

1. Fill-level sensors and concentrators have been deployed, e.g. by an external company working for Torino City council;

Steps to deploy the TRN Platform Instance:

2. Integrate the fill-level sensors:
    a. Deploy SCRAL including M2M drivers
3. Deploy Data Management Framework
    a. Deploy any MQTT Broker
    b. Deploy Data Fusion Manager
    c. Deploy Storage Manager
    d. Deploy Resource Catalogue
4. Download and deploy Virtualization Layer Core

The Platform Instance installation and maintenance could be managed by the IT department of the city of Turin or by an external service provider.

With this setup, developers can use the following Cloud-based APIs:

- Smart City Resource Library API: to query for available IoT Resources, i.e. the fill-level sensors.
- Historical Data API: to access stored observations of the fill-level sensors measurements.
- Live Data API: to subscribe to events from the fill-level sensors or data fusion queries.
- Data Fusion Services API: to define data more complex streams out of the fill-level events.

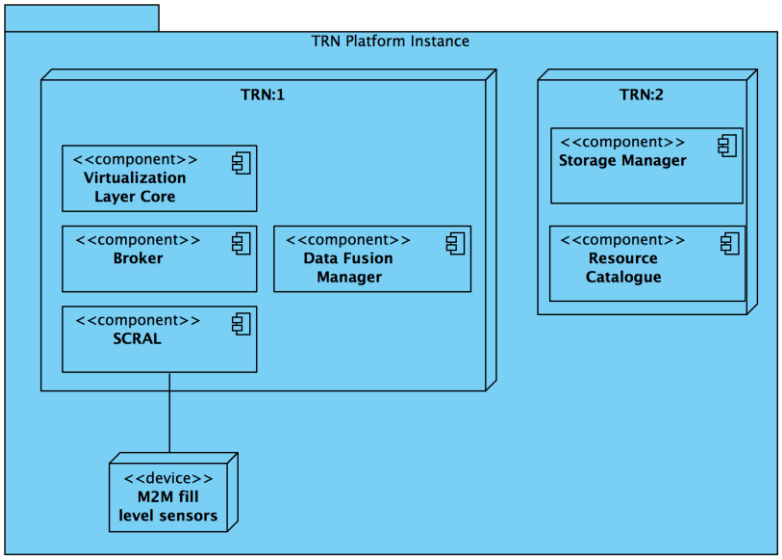


Figure 2 - Deployment Diagram of Baseline Scenario

Figure 2 shows the deployment diagram of the TRN Platform Instance. This simple deployment of a single ALMANAC Platform Instance allows the city of Turin to provide access to the public waste bin data to its citizens but also to the waste utility of Torino city.

In the next scenario we describe how this scenario is extended with another sensor networking technology.

## 4.2 Adding a new sensor networking technology

Waste bins in another quarter are outfitted with a different sensor technology. The city of Turin wants to include these waste bins too. The new sensors run different capillary network technology.

To integrate the sensors, the developer or device integrator simply has to install a new driver for the SCRAL. In this case, a new technology driver for the SCRAL could be already implemented and could be installed in the existing deployment. On integration, the SCRAL updates the Resource Catalogue with the available metadata provided by the new driver.

For application developers this process is completely transparent, they will simply see new sensors in the Resource Catalogue which they may or may not use for application development. Developers don't have to know anything about the new sensor and network technologies. They can use the new sensors using the same cloud based APIs they are using for other sensors.

It is not always the case that access to raw sensor data is provided and that a SCRAL driver already exists. Such a case is described in the next scenario.

## 4.3 Integrating data sources

In the world of IoT, a wide range of IoT integration platforms exist, e.g. Xively[1]. Such platforms allow to upload sensor data to the cloud.

In our scenario, we now want to make use of an existing Xively deployment in Turin. It is an experimental deployment of parking sensors and air quality sensors in a quarter of Turin, which the city is allowed to access and publish.

[1] https://xively.com/

Sensor data on Xively is represented by so-called devices and channels. Devices are software representatives of real sensors/actuators and offer one or more data channels. Channels, in turn, represent homogeneous streams of measures / data about a single observed property, e.g., temperature. For the sake of clarity, if we consider the air quality sensors deployed in our example, each sensor will correspond to a Xively device, whereas for each of the physical quantities measured by the sensor, e.g., temperature, humidity CO2 level, etc., a channel will be available.

To integrate generic Xively device into the set of data sources handled by ALMANAC, a suitable driver shall be configured and deployed at the SCRAL level. The SCRAL already includes a generic Xively driver as part of the base ALMANAC distribution. Such a driver allows attaching devices and channels tagged with a given identifier (by default "project:name=almanac") and polling their value at a given frequency. Attached devices, i.e., devices tagged as above, are then associated to device specific drivers able to interpret channel data and represent it meaningfully inside the platform. Currently supported drivers include fill-level, temperature, light, humidity and pressure sensors.

As in our example scenario drivers are unfortunately unavailable, the integration process follows two main steps (the second would be skipped for devices already supported by the SCRAL). The first step requires either tagging the target Xively devices (and channels), if they are under the developer control, or to configure the Xively driver to attach different tags. The second step, instead, is more complex and involves development of new SCRAL drivers. In the example, this task is relatively easy as the base network driver already exists. In such a case the developer must only:

1. Map the device with an already supported device type (interface).
    a. If a device type does not exist, which matches the target device, a new type shall be defined by following the SCRAL development guidelines provided as part of the SCRAL documentation. This task shall possibly be carried by a domain expert to avoid proliferation of similar but slightly different device interfaces.
2. Implement the device interface exploiting the services offered by the Xively base driver, i.e., by matching the device type and by configuring the needed polling time.
3. Integrate the just developed driver into the SCRAL distribution.

Once developed the driver, e.g., the parking sensor driver, sensor integration is just a matter of tag matching. All sensors tagged as specified, e.g., "project:name=almanac" and matching the set of quantities handled by the driver will automatically be exposed as sensors in the ALMANAC platform, and will then be available for further integration / elaboration. It must be noted that drivers act on "prototype devices", i.e., they can match a virtually unlimited number of similar devices, thus distributing the development costs among possibly high numbers of integrated devices (scale economy).

Thanks to the modular nature of the SCRAL integrating new sensors in the platform is quite easy, difficulty might vary from simple configuration (and/or tagging in the Xively case) to driver development, but in all cases points of intervention are well defined, and the development scope has clear boundaries. This sustains scalable deployment of the platform as well as adaptability to changing requirements / environments.

## 4.4 Securing access to IoT resources

As the next step in the scenario, the city of Turin wants to raise awareness for recycling and waste minimization by creating an app for interested citizens. With this app, citizens can monitor the amount of waste they personally produce and can compare themselves with the average Turin citizens.

To realize this scenario, the TRN deployment needs to support a way for citizens to be registered for the app. The registration is necessary because

- it is an opt-in approach and the citizens need a way to join, and
- users might not be willing to have this information available to someone else which requires TRN to provide a form of authentication which requires a registration.

The registration for citizens can be done online. After the registration is confirmed, the user (citizen) should have his login credentials for an account at the Federated Identity Manager (FIM) of the TRN PI and this account should have been registered for the user with the new app.

To adapt the TRN PI to these new requirements, TRN only has to create the new accounts in the Federated Identity Manager for every registered citizen and TRN has to create the policies in the Access Manager (AM), which controls the user access. Therefore, both the Federated Identity Manager and Access Manager have to be deployed in the TRN PI. The updated deployment is shown in Figure 3 (this figure also shows the new sensors and Xively data integrated through SCRAL).

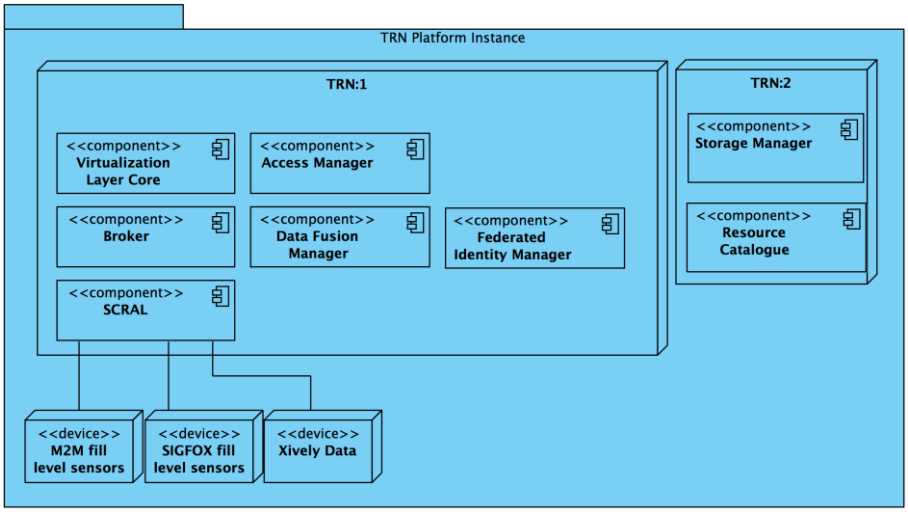


Figure 3 - Updated Deployment Diagram including Access Manager and Federated Identity Manager

In the app, the authentication process works as follows: If a citizen uses the app, he would be asked to authenticate himself via OpenID Connect at the FIM. After a successful authentication request the app will receive in the response a signed token with which the app can access certain ALMANAC services.

The token has the following form

```
{
  "sub": "d8190172-ce8b-4a19-8ae6-123746c21f96",
  "exp": 1454543174,
  "iat": 1454507174,
  "iss": "https://almanac.eu:8543/auth/realms/SHOWCASE",
  "aud": ["81shr123ad"],
  "rls": ["WasteAwareAppUser"]
  ...
}
```

where

- `sub` is the subject identifier of this citizen.
- `exp` and `iat` are the expiration date and the date when the token was issued in seconds from 01.01.1970.
- `iss` is the issuer of the token e.g. the Federated Identity Manager.
- `aud` are the targeted audiences for which the token was created e.g. the Virtualization Layer Core of the PI the user wants to contact.
- `rls` is an non standard claim which hold the roles of the citizen, which will be used in this scenario to identify user of the new app. Other claims could be included into the token and then be used to more complex access rules

Alongside the registration of the user, a new policy was added to the Access Manager, granting to the user access to fused data streams of his waste bins. The policy implementing this rule is shown in Figure 4. It states that it grants the user with the subject identifier `d8190172-ce8b-4a19-8ae6-123746c21f96` access to the resource `https://almanac.eu/dfs/v0_6/data-fusion/chains/100`.

```
<?xml version="1.0"?>
<Policy xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17
 http://docs.oasis-open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd"
 PolicyId="policy01" RuleCombiningAlgId="urn:oasis:names:tc:xacml:3.0:rule-combining-
 algorithm:deny-unless-permit" Version="1.0">
    <Description>This Policy allows the user with the id 'd8190172-ce8b-4a19-8ae6-
     123746c21f96' to read from the resource 'https://almanac.eu/dfs/v0_6/data-
     fusion/chains/100'.
    </Description>
    <Target>
     <AnyOf>
       <AllOf>
         <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:string-equal">
          <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
            d8190172-ce8b-4a19-8ae6-123746c21f96
          </AttributeValue>
          <AttributeDesignator
           AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id"
           DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"
           Category="urn:oasis:names:tc:xacml:1.0:subject-category:access-subject"/>
         </Match>
         <Match MatchId="urn:oasis:names:tc:xacml:1.0:function:anyURI-equal">
           <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#anyURI">
             https://almanac.eu/dfs/v0_6/data-fusion/chains/100
           </AttributeValue>
           <AttributeDesignator
            AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id"
            DataType="http://www.w3.org/2001/XMLSchema#anyURI" MustBePresent="true"
            Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource"/>
         </Match>
       </AllOf>
     </AnyOf>
    </Target>

    <Rule RuleId="" Effect="Permit">
     <Condition>
       <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string equal">
         <Apply FunctionId="urn:oasis:names:tc:xacml:1.0:function:string-one-and-only">
          <AttributeDesignator
           AttributeId="urn:oasis:names:tc:xacml:1.0:action:action id"
           DataType="http://www.w3.org/2001/XMLSchema#string" MustBePresent="true"
           Category="urn:oasis:names:tc:xacml:3.0:attribute category:action"/>
         </Apply>
         <AttributeValue
          DataType="http://www.w3.org/2001/XMLSchema#string">Read</AttributeValue>
       </Apply>
     </Condition>
    </Rule>
</Policy>
```

Figure 4 - Policy for granting a user access to a resource

Now, when the VLC receives a request it will send a policy request to the Access Manager to check if the user if authorized to access the requested resource. For example, when the user with the sub-id `00000000-0000-0000-0000-a7745f0ab71b` wants to read from the resource

https://almanac.eu/dfs/v0_6/data-fusion/chains/100 the VLC sends the request to the Access Manager as shown in Figure 5.

```
<?xml version="1.0" encoding="utf-8"?>
<Request
 xsi:schemaLocation="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17 http://docs.oasis-
 open.org/xacml/3.0/xacml-core-v3-schema-wd-17.xsd" ReturnPolicyIdList="false"
 CombinedDecision="false" xmlns="urn:oasis:names:tc:xacml:3.0:core:schema:wd-17"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Attributes Category="urn:oasis:names:tc:xacml:1.0:subject category:access-subject">
    <Attribute IncludeInResult="false"
     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
       00000000-0000-0000-0000-a7745f0ab71b
     </AttributeValue>
    </Attribute>
    <Attribute IncludeInResult="false"
     AttributeId="urn:oasis:names:tc:xacml:1.0:subject:subject-id">
      <AttributeValue DataType="http://www.w3.org/2001/XMLSchema#string">
        00000000-0000-0000-0000-a7745f0ab71b
      </AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:resource">
    <Attribute IncludeInResult="false"
     AttributeId="urn:oasis:names:tc:xacml:1.0:resource:resource-id">
      <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#anyURI">
        https://almanac.eu/dfs/v0_6/data-fusion/chains/100
      </AttributeValue>
    </Attribute>
  </Attributes>
  <Attributes Category="urn:oasis:names:tc:xacml:3.0:attribute-category:action">
    <Attribute IncludeInResult="false"
     AttributeId="urn:oasis:names:tc:xacml:1.0:action:action-id">
      <AttributeValue
       DataType="http://www.w3.org/2001/XMLSchema#string">
        Read
      </AttributeValue>
    </Attribute>
  </Attributes>
</Request>
```

Figure 5 - Request to the Access Manager

Since this user has no policy to for accessing https://almanac.eu/dfs/v0_6/data-fusion/chains/100 the Access Manager will return a deny message to the VLC and the VLC will then act accordingly, i.e. denying access to the user to the resource.

For the application developer the policy management is transparent, once he is logged-in. The process of authentication and authorization is hidden from the developer (except for the registration and log-in of course) and he can use the Cloud-based APIs as usual. In this case the developer would employ the Data Fusion Services API to create the sum and average of waste produced based on fill-level sensors. How such application can be implemented with the Cloud-based APIs is out of scope of this deliverable but is described in detail in D7.3 - Cloud-based APIs for Smart City Applications – Developer's Guide.

## 4.5 Creating an Association (Access control in federated deployments)

In this iteration of the scenario, a new player is introduced: AMIAT the local waste management company who wants to become a part of the Turin Smart City. AMIAT owns several underground ecological islands (UEIs) for collecting waste in quarters that don't have door-to-door collection. Each UEI comprises four types of bins (generic, glass/can, plastic, organic) which are equipped with fill-level sensors based on M2M capillary networks.

AMIAT and the city of Turin (TRN) agreed on a contract that allows Turin to access the data collected by the AMIAT fill-level sensors. However, the data that belongs to AMIAT should not be accessible by the public but only by a limited group of users from the city of Turin (TRN).

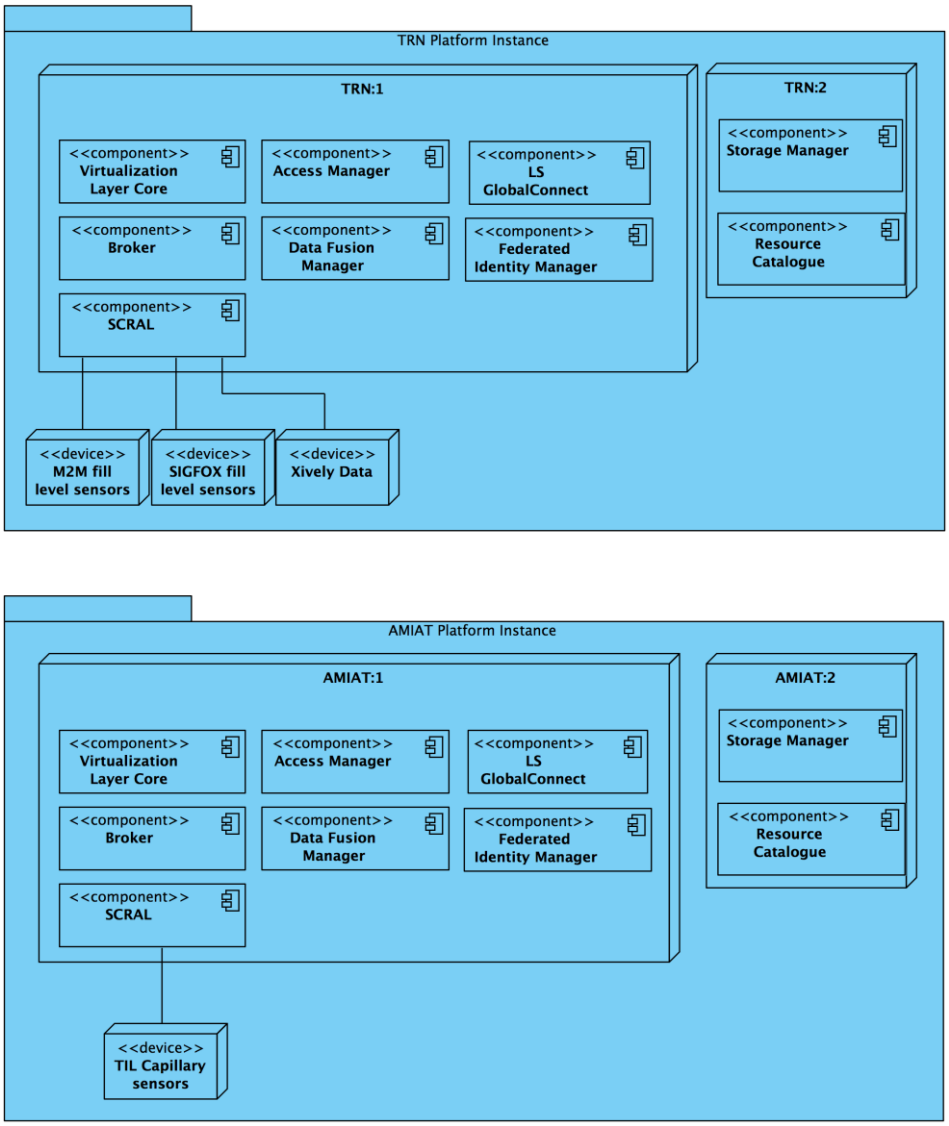


Figure 6 - TRN and AMIAT Platform Instances

As a first step, an AMIAT Platform Instance will be deployed. As shown in Figure 6, the AMIAT PI is similar to the TRN PI, except that AMIAT owns capillary sensors, which are integrated in their PI. The basic (technical) requirement for creating an Association between both PIs is that they both have to belong to the same Federation. Therefore, both PIs have to deploy LinkSmart GlobalConnect which allows to transparently share information between both PIs. Technically, the instances of LinkSmart GlobalConnect establish an overlay network by connecting all PIs within a Federation. Therefore, each

instance of LinkSmart GlobalConnect connects to the LinkSmart SuperNode, which is done by a simple configuration parameter.

After establishing the Federation, AMIAT and TRN have to come to an agreement on which TRN users will be allowed to access the AMIAT sensor data. For this scenario, we say that a group of developers of TRN will have access. To establish this Association we need to do the following:

After establishing the Federation, AMIAT and TRN have to come to an agreement on which TRN users will be allowed to access the AMIAT sensor data. For this scenario, we say that a group of developers of TRN will have access. To establish this Association we need to do the following:

1. AMIAT needs to add the TRN-FIM to the trusted FIMs. This done by adding the authentication information of the TRN-FIM to the AMIAT PI.
2. Further, AMIAT must create policies for the developers to give them access to the capillary sensors. In the new policies the following rules should be checked
    a. that the subject id of the requesting user is equal to the subject id of one of the allowed developers from TRN,
    b. which services this specific user is allowed to use,
    c. which actions the user is allowed to perform with the service,
    d. from where the requests need to come from (by comparing the Id of the sending GC instance with the one of TRN), and
    e. that the issuer of the token is the TRN-FIM.
3. AMIAT needs to register the basic services at their LinkSmart GlobalConnect instance so they can be found by the TRN instance within the federation.

The first step is necessary since the AMIAT PI needs to be able to verify the users of TRN. The tokens from the FIM are either signed via a public key signing mechanism or via a message authentication code (MAC). To verify its authenticity, the security enforcement points at the AMIAT PI need to have the certificate for the signatures or the secret key for the MAC.

The policies in the second step are an extension of the policies defined in the previous section. The rules a, b, and c are the same for both scenarios. The rules d and e define, which PIs are allowed to access the resources.

The rule d allows AMIAT to allow only requests issued from a specific PI, in this case the TRN PI. This allows the TRN PI to reuse its FIM for multiple PIs, which might not all need to have access to AMIATs capillary network.

The rule e might seem redundant since in step one it was ensured that the token is a valid token from a trusted FIM. But it may be the case that the subject id of a user on one trusted FIM might be equal to the subject id of another user from a different trusted FIM. This means that in an Association with multiple FIMs a user is identified over the combination of subject id and issuer id.

This also shows that the policies from the previous section would need to be extended as well, since they only use the subject id, which was possible while only one trusted FIM was configured.

The third and final step activates the remote access to the basic ALMANAC services, which can then be used by the developers of TRN.

# 5. Conclusion

The deliverable described the features of the ALMANAC Platform for sustainable Smart City Applications and how these features contribute to sustainability of ALMANAC Platform Instances.

Smart City Application developers can rely on sustainable ALMANAC Platform Instances to support their development processes. Various layers of abstraction and the Cloud-based APIs as entry point for developers make sure that existing applications will not be affected by further evolution of a scenario. New sensors or technologies can be integrated seamlessly. Federation features of ALMANAC allow for transparent resource exchange between different ALMANAC Platform Instances. The Security Framework provides the features to secure this communication by providing user and policy management.

All of these features allow ALMANAC Platform Instances to adapt to changing conditions while at the same time keeping application sustainable.