Exploiting Aggregated Open Data from Smart Cities for the Future Internet Society

D5.4: Integrated SMART-FI Platform

Authors	Omer Ozdemir (Atos)
Institution lead	Atos
Version	1.0
Reviewers	Javi Cubo (UMA), Malena Donato (ATOS)
Work package	WP5
Task	5.4
Due date	30/09/2018
Submission date	30/09/2018
Distribution level (CO, PU):	Public
Abstract	This document describes the overall and specific integration activities of the FIWARE components within the scope of SMART-FI environment
Keywords	Smart-FI, FIWARE, IoT services

Document Description

Document Revision History

Version	*Date*	*Modifications Introduced*	
		Description of change	*Modified by*
V0.1	06/07/2018	Initial version	Omer Ozdemir (Atos)
V0.2	07/08/2018	Inputs from various partners	Omer Ozdemir (Atos)
V0.3	10/09/2018	Revision	Malena Donato (Atos)
V0.4	11/09/2018	Revision and inputs	Malena Donato (Atos)
V0.4	13/09/2018	Revision	Javier Cubo (UMA)

Table of Contents

Table of Figures

Table of Tables

Terms and abbreviations

FIWARE	Middleware platform, driven by the European Union, for the development and global deployment of applications for Future Internet
SMART-FI	Provides services for smart city applications, creates business opportunities and improves public's quality of life using Open Data
USDL	Master schema constructed using semantic web technologies and linked data principles (Universal Service Description Language)
HADOOP	An open-source software framework used for distributed storage and processing of very large data sets
SPARQL	Semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format
Enterprise Service Bus	Communication system between mutually interacting software applications in a service-oriented architecture (SOA
Generic Enabler(GE)	Software tools offered by FIWARE
CKAN	Open-source DMS (data management system) for powering data hubs and data portals
COSMOS	The code name for the Reference Implementation of the BigData Generic Enabler of FIWARE.
CYGNUS	Connector in charge of persisting certain sources of data in certain configured third-party storages, creating a historical view of such data.
ORION CONTEXT BROKER	The reference implementation of the Publish/Subscribe Context Broker GE.
MiDAS	(Multi-Attribute Indexing for Distributed Architecture Systems) MIDAS is an efficient method for indexing multi-attribute data
WP	Work Package

SMASSA	Municipal Society of Car Parks and Services
CartoDB	Software as a Service (SaaS) cloud computing platform that provides GIS and web mapping tools for display in a web browser.
MYSQL	Open-source relational database management system (RDBMS)
GE	Generic Enabler

Executive Summary

This purpose of this document to describe the integration activities between the SMART-FI environment and FIWARE modules by focusing on the main building blocks defined in FIWARE reference architecture[1]. SMART-FI project aims to design, implement and deploy smart urban services which are mainly for municipality officials, system integrators and 3rd party service developers. In order to achieve these goals, SMART-FI environment benefited from the open and standardized FIWARE platform components which are mainly used for legacy system management, data/context management and security mechanism implementations.

In this document, we cover candidate reference implementation components and the interoperability points within the FIWARE Reference Architecture based on the main blocks as mentioned below and explained in detail in the next sections:

- Context Management
- IoT Management
- Data Storage Management
- Security

[1] https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Architecture

1 Introduction

The aim of this document is to describe the methodologies and guidelines followed by the project partners to have a FIWARE based Smart City environment with logical components, functionalities and interfaces which are used to create, integrate and deploy smart city services by 3 pilot cities (Malaga, Malatya and Karlshamn) in the scope of the SMART-FI project. Each architectural layer, internal relations of the components, functionalities and inter-operability points will be covered with respect to the FIWARE Reference Architecture.

For this reason, the approach followed in this deliverable is twofold: on the one hand we tried to base the architecture on standard approaches and technologies derived from the most relevant international initiatives; on the other hand, it has been taken in consideration the concrete specific requirements which are addressed directly in FIWARE environment and how we used them.

The core part of this document is based on the FIWARE reference architecture description: the functionalities and interfaces following an open approach. The objective is to provide an integrated architecture that can be implemented with different technologies by pilot zones.

We follow and explain the principles which are used to implement the logical components of the SMART-FI architecture.

1.1 Document structure

Section 1 introduces the document; Section 2 discusses several candidate modules for the reference architecture implementation; Section 3 describes the NGSI standard and the FIWARE supported data models, Section 4 covers the FIWARE modules which are used in the SMART-FI environment and finally we close the deliverable with the conclusion chapter.

2 Reference implementation technologies

The ideal reference architecture should provide all the necessary functionalities that satisfy 3 different pilot requirements. There is no particular limitation for any kind of specific technology or programming language; however, we follow some criteria in order to select the best candidate which would fill the gap as a Smart City Reference Architecture:

- Compliance with SMART-FI architecture guidelines
- Transparent and open
- Standardization of the main blocks
- Interoperability
- Previous experience with the technology and modules

In this section, we analyze the possible candidates in each category that can be used to implement the technical functionalities in 3 different pilot zones. We list and explain different components/tech stacks and their functionalities in a high-level manner. There are 4 main blocks which act as the pillars of the platform:

- Context Management
- IoT Management
- Data Management
- Security

2.1 Main blocks

This section contains candidate modules which do belong to 4 different sub-categories mentioned in the previous section.

2.1.1 Context Management

2.1.1.1 FIWARE Orion Context Broker

The Orion Context Broker is an implementation of the Publish/Subscribe Context Broker GE (Generic Enabler), which follows the NGSI 9 and NGSI 10 interfaces specifications. Using these interfaces, clients can do several operations:

- Register context producer applications, e.g. a temperature sensor within a room.
- Update context information, e.g. send updates of temperature.
- Being notified when changes on context information take place (e.g. the temperature has changed) or with a given frequency (e.g. get the temperature each minute)
- Query context information. The Orion Context Broker stores context information updated from applications, so queries are resolved based on that information.

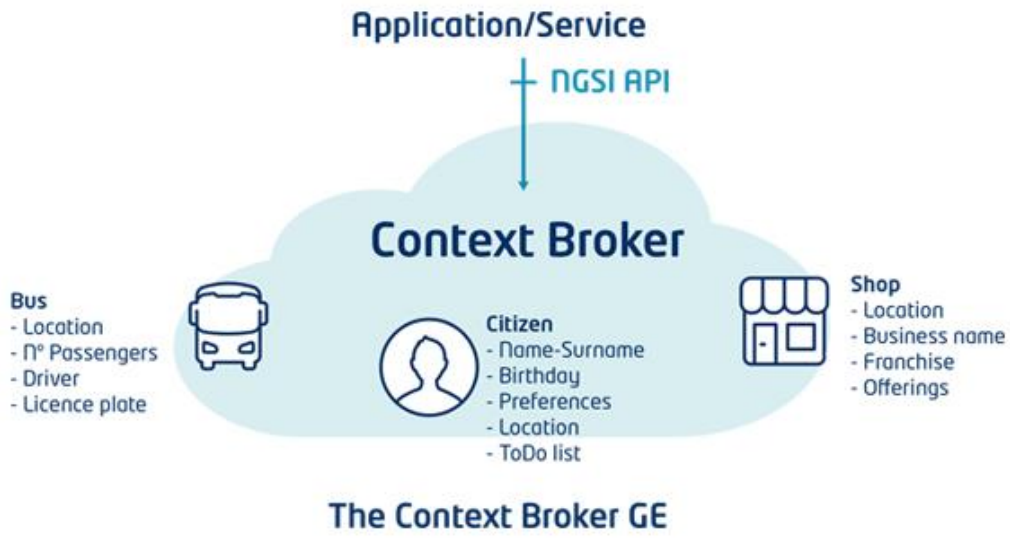

Figure 1 Context Broker Flows

Orion is a C++ implementation of the NGSI 9/10 REST API binding developed as a part of the FIWARE platform. The Orion Context Broker allows user to manage the entire lifecycle of context information including updates, queries, registrations and subscriptions. It is an NGSI 9/10 server implementation to manage context information and its availability. Using the Orion Context Broker, user can register context elements and manage them through updates and queries. In addition, user can subscribe to context information so when some condition occurs (e.g. the context elements have changed) you receive a notification. Context information is represented through values assigned to attributes that characterize those entities relevant to your application. The Context Broker can handle context information at large

2.1.1.2 Amazon Simple Notification Service

Amazon Simple Notification Service (Amazon SNS) is a web service that coordinates and manages the delivery or sending of messages to subscribe endpoints or clients.

In Amazon SNS, there are two types of clients, publishers and subscribers, also referred to as producers and consumers. Publishers communicate asynchronously with subscribers by producing and sending a message to a topic, which is a logical access point and communication channel. Subscribers (i.e., web servers, email addresses, Amazon SQS queues, AWS Lambda functions) consume or receive the message or notification over one of the supported protocols (i.e., Amazon SQS, HTTP/S, email, SMS, Lambda, MQTT) when they are subscribed to the topic.

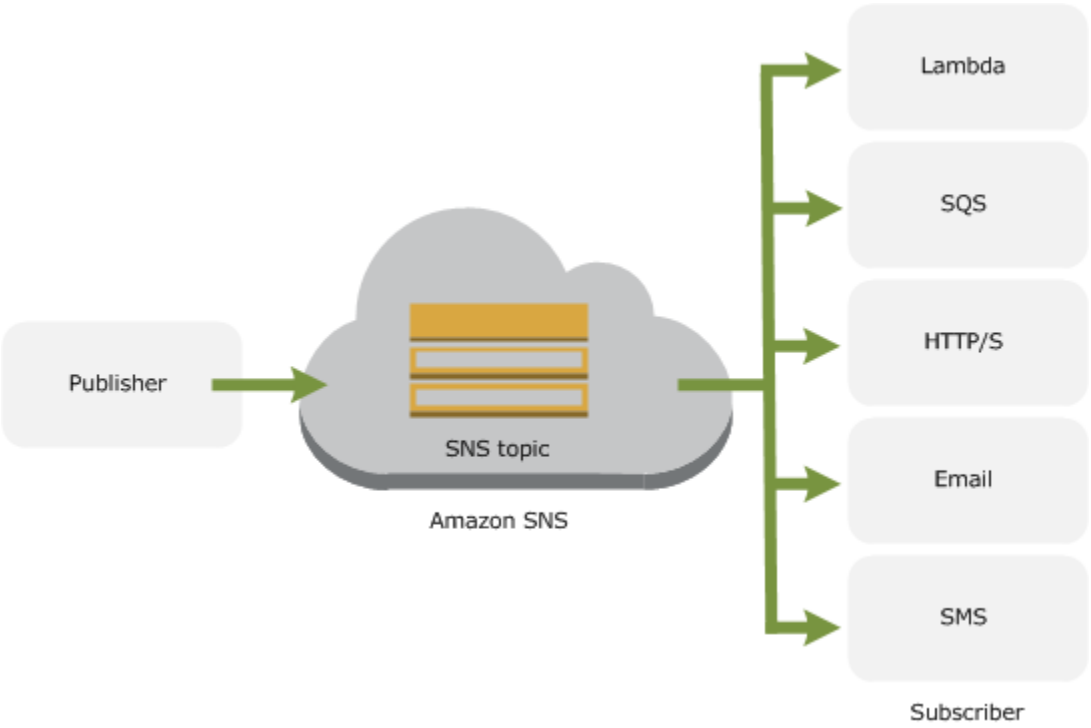

Figure 2 AMAZON SNS component

When using Amazon SNS, user creates a topic and control access to it by defining policies that determine which publishers and subscribers can communicate with the topic. A publisher sends messages to topics that they have created or to topics they have permission to publish to. Instead of including a specific destination address in each message, a publisher sends a message to the topic. Amazon SNS matches the topic to a list of subscribers who have subscribed to that topic, delivers the message to each of those subscribers. Each topic has a unique name that identifies the Amazon SNS endpoint for publishers to post messages and subscribers to register for notifications. Subscribers receive all messages published to the topics to which they subscribe, and all subscribers to a topic receive the same messages.

2.1.1.3 FIWARE Complex Event Processing (Proton) Generic Enabler

FIWARE Complex Event Processing (CEP) GE module analyses event data in real--time, generates immediate insight and enables instant response to changing conditions (i.e.: if the Entity attribute "temperature" is over 25 send an email to me). IBM Proactive Technology Online (Proton) [124] is an open source complex event processing engine developed at IBM Research - Haifa and it is an implementation of the FIWARE CEP GE. It provides language primitives for defining, submitting, and executing event processing networks. The goal of the system is to respond to raw events and identify meaningful events within contexts. The system comes with a set of built-in operators (such as sequence, all, etc.) for determining CEP patterns. It also has extendable APIs for adding additional custom operators. The system comes with existing source/sink adapters, allowing it to extract raw events from files or pull them from Restful services. It also provides extendable APIs for adding more adapter types.

2.1.1.4 Complex Event Processing (Apache Flink)

Apache Flink is an open-source stream processing framework for distributed, high--performing, always-available, and accurate data streaming applications. It offers different ways to process different kinds of data sets, divided into bounded (finite,

i.e., unchanging) and unbounded (infinite data sets that are continuously appended to). It allows for streaming processing, where execution is continuous for as long as data is being produced, and batch processing, that is executed in a finite amount of time. Its main features are being easily scalable horizontally, running in multiple nodes to offer better throughput. It provides event time semantics, to process results over data streams where events may arrive out of order. It also supports flexible windowing based on different parameters to extract relevant results from the processed streams.

2.1.2 IoT Management

2.1.2.1 FIWARE IoT Broker Generic Enabler

IDAS GE is an implementation of the FIWARE Backend Device Management GE and it is the component able to connect IoT devices/gateways to FIWARE-based ecosystems. To send information from the devices to the platform, user can employ specific components called IoT Agents. These components map southbound protocol requests coming from the device to NGSI requests to a Context Broker that helps user mapping device data to an NGSI Entity and attributes. On the southbound side, IDAS exposes the Device API that allows to:

- Register device to reduce the message footprint and use commands.
- Send data from the device to the FIWARE IoT Stack
- Send commands from application to the device

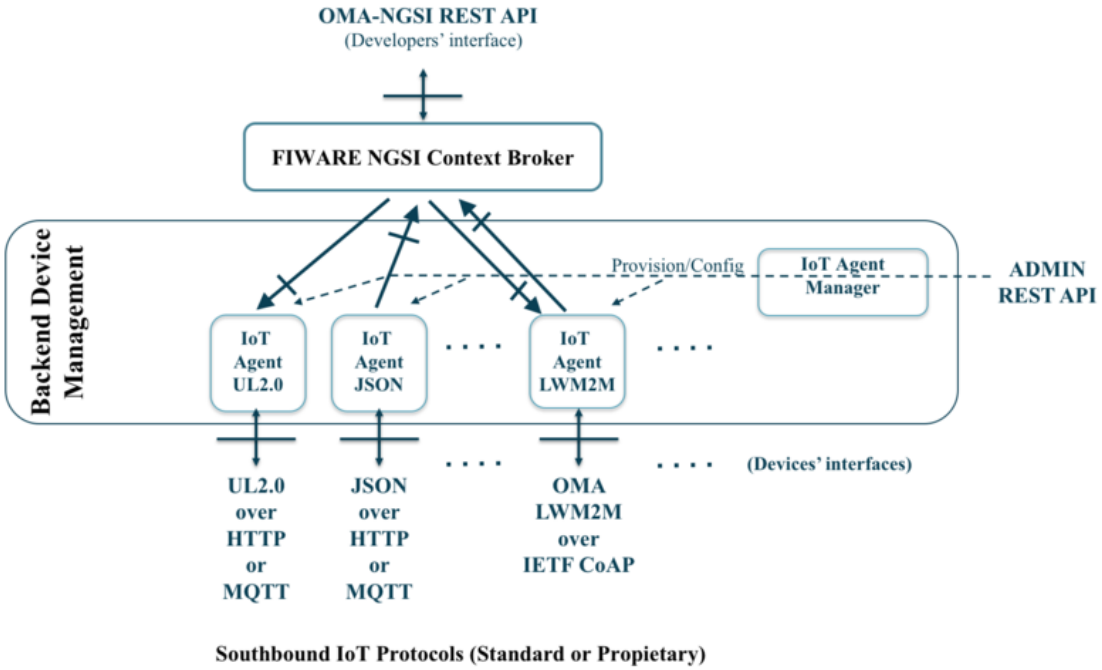

Figure 3 IoT Manager

2.1.2.2 SensiNact

SensiNact is an open source project, recently included in Eclipse Community, which provides gateway functionalities dedicated to IoT and allows interconnection of different networks to achieve access and communication with embedded devices. The main objective of SensiNact is managing heterogeneous IoT protocols providing both on demand and event-based access to data or actions of IoT devices and, in addition, it also provides a layer of APIs for access to historic data. In particular, SensiNact consists in two complementary frameworks:

- SensiNact Platform: it is the core part in charge of managing the southbound connection to IoT devices and allowing access to them with various northbound protocols. For Southbound connections SensiNact provides a list of supported protocols like ZigBee, LoRa, MQTT, XMPP and data coming from different devices are accessible by a northbound interface that supports protocols like HTTP REST, JSON RPS, and CDMI. SensiNact supports NGSI for both southbound and northbound connections so allowing a full integration with FIWARE Platforms and the other ones that use these interfaces.
- SensiNact Studio: it is an IDE Eclipse-based that allows developing IoT applications by managing IoT devices connected to SensiNact.

From the architecture point of view, SensiNact can be represented using the following five functional groups:

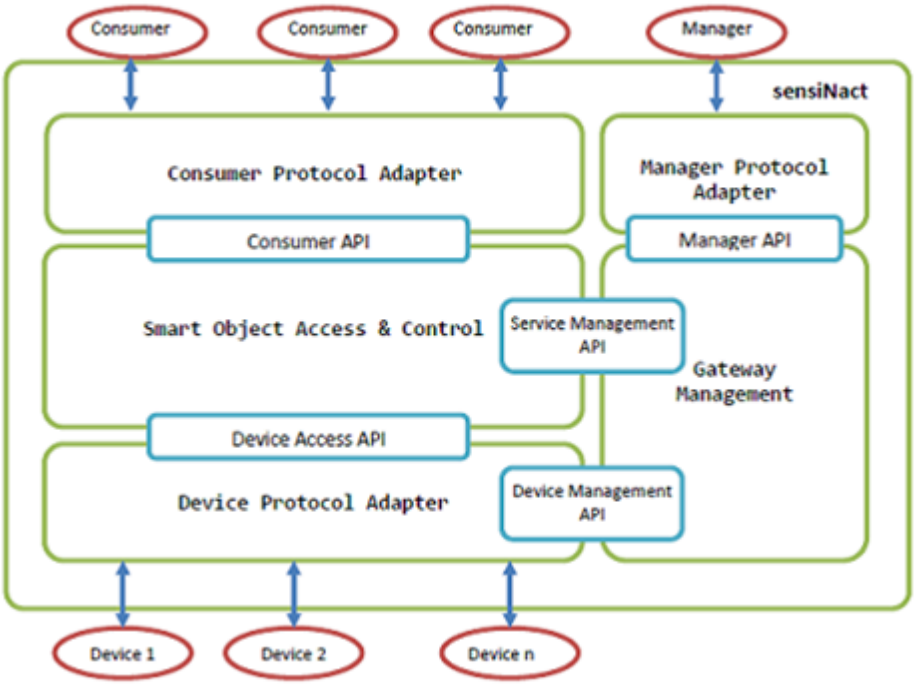

Figure 4 SensiNact platform

The main functionalities of each block are:

- Device Protocol Adapter: Represents all the bridges that allow southbound access to heterogeneous IoT protocols. It exposes Device Access API to interact with northbound interfaces
- Smart Object Access and Control: Includes all the core functionalities such as devices and resources discovery, secure communication, etc. It exposes its services through the Consumer API to consumers.
- Consumer Protocol Adapter: Represents all the bridges that translate information coming through Consumer API interface and provide it to consumer using specific application protocols (REST, NGSI, etc.).
- Gateway Management: Manages devices connected to SensiNact, by using the Device Management API, and the other functionalities such as cache, resource directory and security services by using the Gateway Management API.
- Manager Protocol Adapter: Adapts Gateway Management API to protocols used by external management entities.

2.1.3 **Data Storage Management**

2.1.3.1 FIWARE Cosmos Generic Enabler

Cosmos is the code name for the Reference Implementation of the Big Data Generic Enabler of FIWARE, a set of tools and developments helping in the task of enabling a Hadoop as a Service (HaaS) deployment, Cosmos can serve:

- A set of administration tools such as HDFS data copiers
- An OAuth2 tokens generator
- A web portal for users and accounts management, running MapReduce jobs and doing I/O of big data
- A custom authentication provider for Hive
- A REST API for running MapReduce jobs in a shared Hadoop cluster
- A specific OAuth2-base proxy for HTTP/REST operations

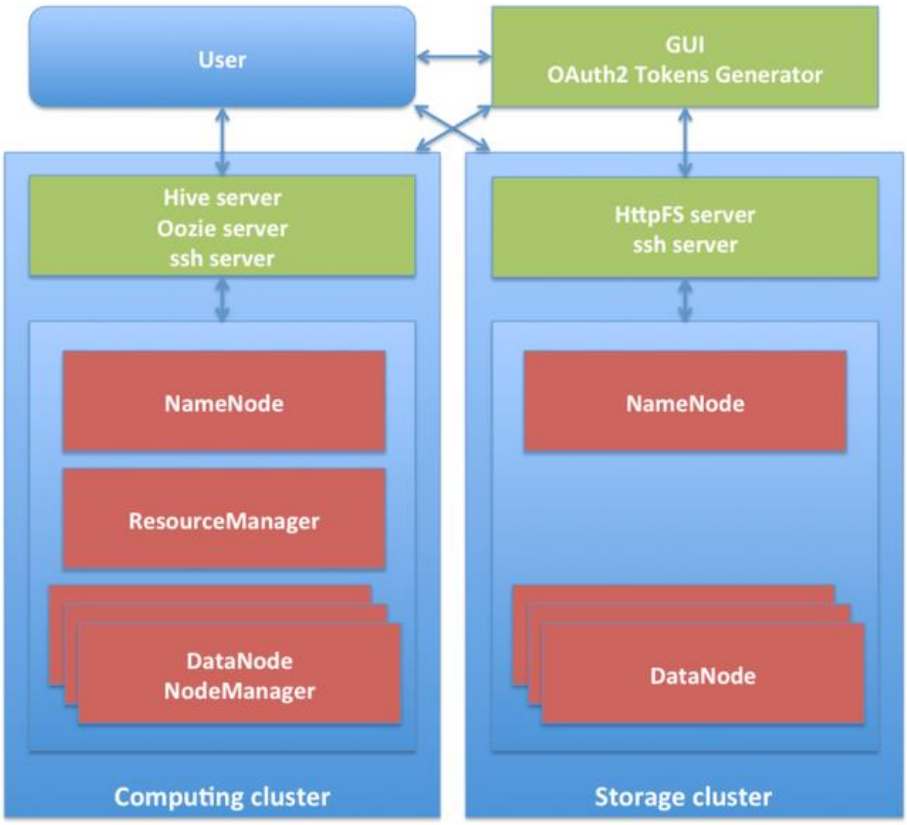

Figure 5 Cosmos Generic Enabler

The basic design principles of the Big Data Analysis GE are:

- To hide the complexity behind the process of creating Big Data environments, where some software packages must be appropriately configured in order they work in a coordinated fashion.
- To offer a wide set of processing and querying technologies for being installed in the environment. The GE exposes a catalogue, which can grow up in an easy and opened way.
- To focus the efforts on extracting insights and value-added information. To achieve that, there are components allowing for complex SQL-like queries design, reusing binaries and to compose processing sequences thanks to an easy and intuitive web-based interface.

2.1.3.2 Cygnus

Cygnus (more specifically, cygnus-ngsi agent) plays the role of a connector between Orion Context Broker (which is a NGSI source of data) and many FIWARE storages such as CKAN, Cosmos Big Data (Hadoop) and STH Comet. Of course, as previously said, you may add MySQL, Kafka, Carto, etc. as other non FIWARE storages to the FIWARE architecture.

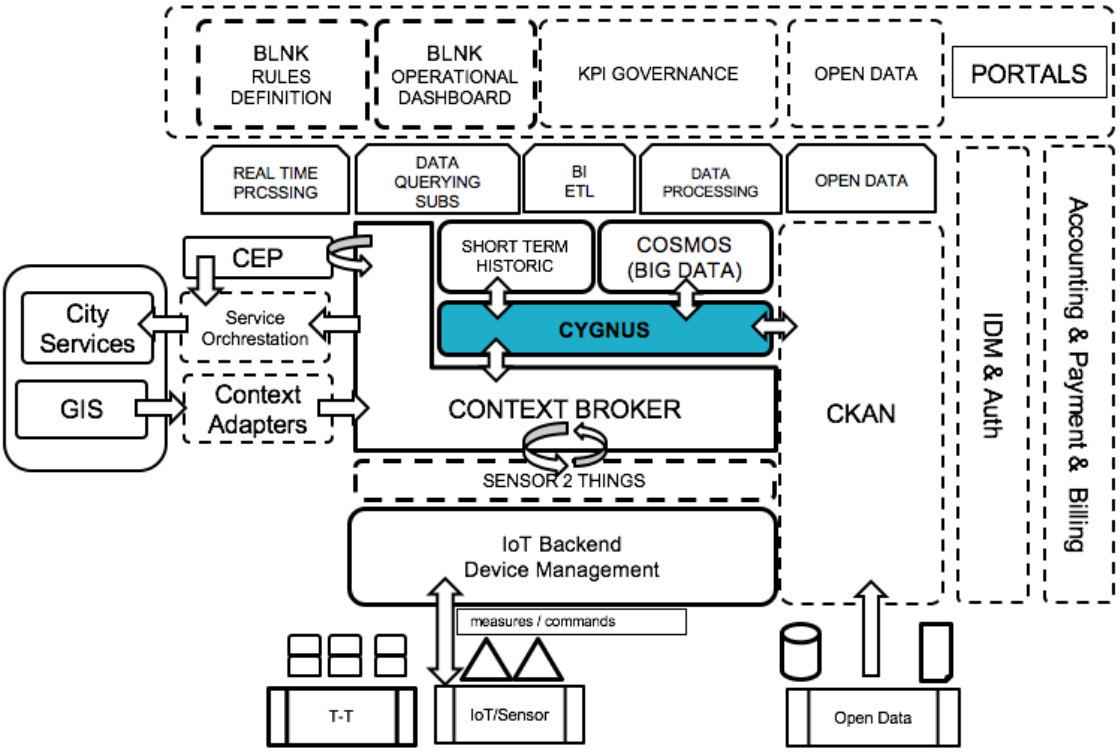

Figure 6 Cygnus connector

2.1.3.3 CKAN

CKAN is a powerful data management system that makes data accessible – by providing tools to streamline publishing, sharing, finding and using data. CKAN is aimed at data publishers (national and regional governments, companies and organizations) wanting to make their data open and available. It is also aimed at the data users who go to CKAN instances to find open data and start using it. CKAN is built with Python on the back-end and JavaScript on the front-end and uses The Pylons web framework and SQLAlchemy as its ORM. Its database engine is PostgreSQL and its search are powered by SOLR. It has a modular architecture that allows extensions to be developed to provide additional features such as harvesting or data upload. CKAN uses its internal model to store metadata about the different records and presents it on a web interface that allows users to browse and search this metadata. It also offers a powerful API that allows third-party applications and services to be built around it.

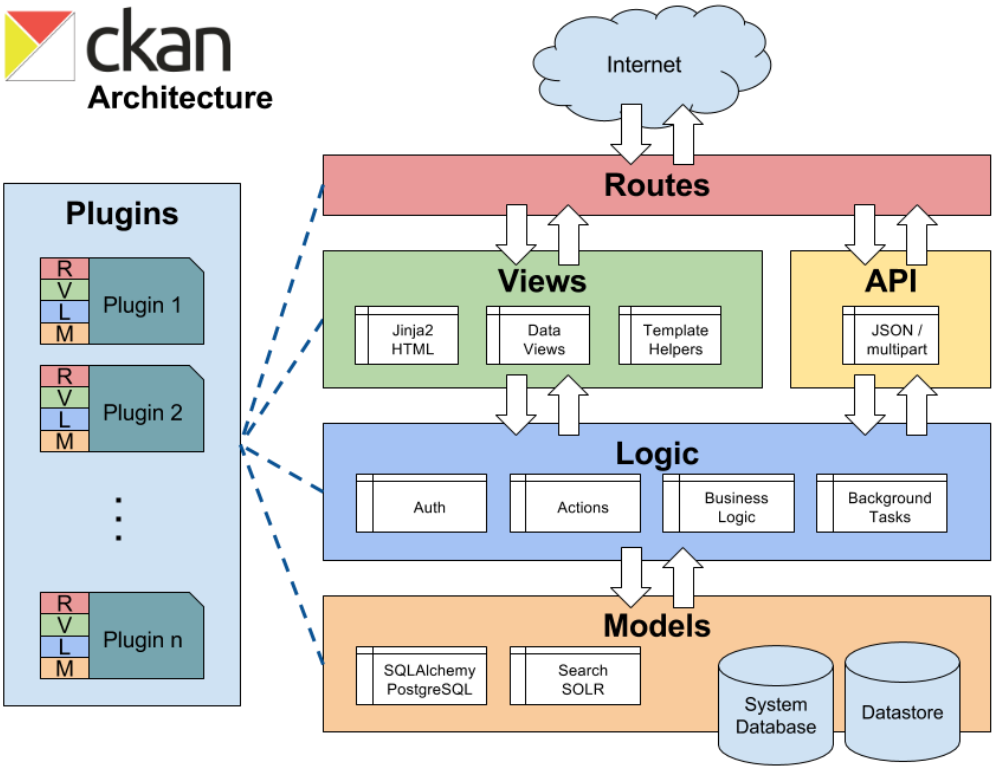

Figure 7 CKAN architecture

2.1.4 **Security**

2.1.4.1 FIWARE IDM Generic Enabler

Identity Management covers several aspects involving users' access to networks, services and applications, including secure and private authentication from users to devices, networks and services, authorization & trust management, user profile management, privacy-preserving disposition of personal data, Single Sign-On (SSO) to service domains and Identity Federation towards applications. The Identity Manager is the central component that provides a bridge between IdM systems at connectivity-level and application-level. Furthermore, Identity Management is used for authorizing foreign services to access personal data stored in a secure environment. Hereby usually the owner of the data must give consent to access the data; the consent-giving procedure also implies certain user authentication.

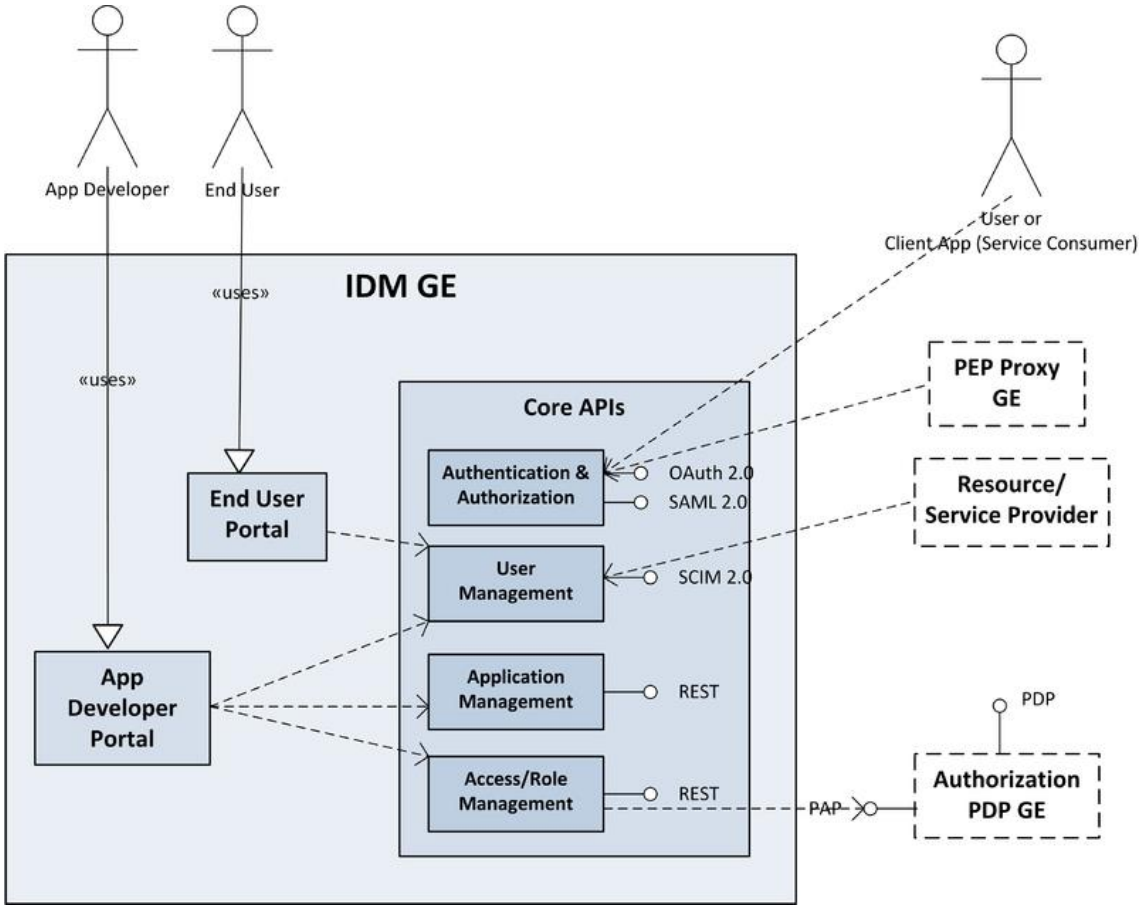

Figure 8 IDM GE

3 Interoperability points and standards for APIs and data models

This section analyses the NGSI data model and the list of operations that allow to interact with context entities, these operations and standards are the interoperability points in overall SMART-FI platform.

This section specifies the NGSI-9 and NGSI-10 Interfaces with the following functions:

- Register and retrieve the availability of Context Entities and/or Context Information
- Update Context Information in accordance to a specified Context Information Model.
- Query for and subscribe to Context Information about Context Entities. Another relevant aspect is the usage of common data models that can add benefits in SynchroniCity in terms of interoperability and reusability of application. The following proposal is based on FIWARE Data Models project that provides a wide set of data models based on NGSI specification.

3.1.1 NGSI data model and interfaces

NGSI is a protocol developed by OMA to manage Context Information, which provides following functionalities:

- Manage the Context Information about Context Entities.
- Access (query, subscribe/notify) to the available Context Information about Context Entities.

Context Entities are entities that are described by Context Information and they are described by the Context Information Model. The Context Information Model details how Context Information is structured and associated to Context Entities to describe their situation. In this model, Context Information is organized as Context Elements, which contain set of Context Attributes and associated metadata.

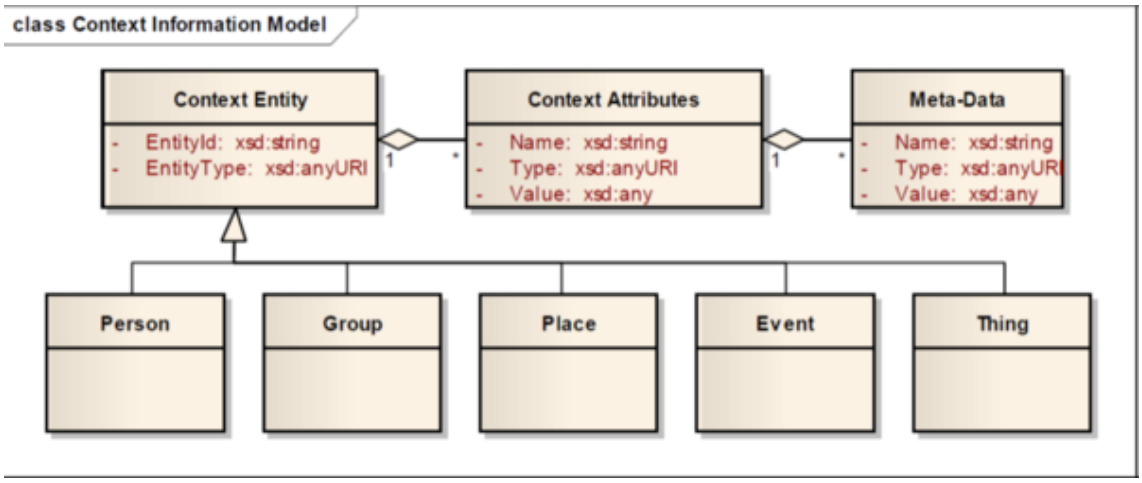

Figure 9 Context Information Model

NGSI defines the following two interfaces for managing information based on this Context Information model:

- NGSI-9: provides operations to obtain the availability information about context entities and their attributes; this interface contains operations to register context entities and to discover context information providers.
- NGSI-10: provides operations for exchanging information about entities and their attributes; this interface contains operations to perform queries, update or activate subscription on context entities.

NGSI 9	Description	NGSI 10	Description
registerContext	Allows registering and updating of registered Context Entities, their attribute names and availability	queryContext	This operation allows for the synchronous retrieval of Context Information.
discoverContextAvailability	Allows the synchronous discovery of the potential set Context Entities, types of Context Entities and related Context	subscribeContext	This operation allows the asynchronous retrieval of Context Information. It is used for subscription to Context Information. The subscription triggers

	Information that can be provided.		the notifications about the matching Context Entities based on the defined Notify Condition information passed in the subscribeContextRequest operation.
subscribeContextAvailability	Allows the asynchronous discovery of the potential set Context Entities, types of Context Entities and related Context Information that can be provided	updateContextSubscription	This operation allows updating a previous subscription to Context Information.
updateContextAvailabilitySubscription	Updates a previous subscription to discover Context Information.	unsubscribeContext	This operation allows unsubscribing a previous subscription to Context Information.
UnsubscribeContextAvailability	Deletes a previous subscription to discover Context Information.	notifyContext	This operation allows receiving the notification about the Context Information subscribed to by the subscriber that implements the notification interface.
notifyContextAvailability	Allows receiving the notification about the potential set of Context Registrations subscribed to by the subscriber that implements the notification interface.	updateContext	This operation allows updating a set of Context Information, related attributes and metadata.

Table 1 NGSI 9&10 operation list

3.1.2 **Data Models**

The FIWARE Community is promoting the development of re-usable and harmonized Data Models under the umbrella of the FIWARE Data Models initiative. These data models are reusing and extending the work performed under the GSMA IoT Big Data initiative. The work conducted under FIWARE is evolving daily by taking into consideration requirements from Data Models' adopters. All FIWARE Data Models, coherently with the centric role of the NGSI API, are expressed to be used with such API.

In line with this principle FIWARE has harmonized so far, the following set of data models:

- Environment. A model to enable the monitoring of air quality and other environmental conditions for a healthier living. Covered entity types include:
  - AirQualityObserved: an observation of air quality conditions at a certain place and time.
  - WaterQualityObserved: capture all the parameters involved in Water Quality scenarios.
  - NoiseLevelObserved: represents an observation of those parameters that estimate noise pressure levels at a certain place and time.
- Civic Issue tracking. This set provides entity types for civic issue tracking interoperable with the de-facto standard Open311. Covered entity types include:
  - ServiceType. A type of service a citizen can request.
  - ServiceRequest. A specific service request (of a service type) made by a citizen.
- Street Lighting. It models street lights and all their controlling equipment towards energy-efficient and effective urban illuminance. The covered entity types include:
  - Streetlight: a instance of a streetlight. A streetlight is composed by a lantern and a lamp. Such elements are mounted on a column (pole), wall or other structure.
  - StreetlightGroup: a group of streetlights being part of the same circuit and controlled together by an automated system.
  - StreetlightModel: a model of streetlight composed by a specific supporting structure model, a lantern model and a lamp model. A streetlight instance will be based on a certain streetlight model.
  - StreetlightControlCabinet: an automated equipment, usually on street, typically used to control a group(s) of streetlights, i.e. one or more circuits.
- Device. This set of entity types describes IoT devices (sensors, actuators, wearables, etc.) with their characteristics and dynamic status. The covered entity types include:
  - Device: an electronic apparatus designed to accomplish a task.
  - DeviceModel: the static properties common to multiple instances of a Device.
- Transportation. Transportation data models for smart mobility and efficient management of municipal services. The covered entity types include:
  - TrafficFlowObserved: a recorded observation of traffic flow.
  - Road: a geographic and contextual description of a Road.
  - RoadSegment: a geographic and contextual description of a road segment.
  - Vehicle: a specific vehicle instance.
  - VehicleModel: a model of vehicle, capturing its static properties such as dimensions, materials or features.
- Indicators. It models key performance indicators intended to measure the success of an organization or of a activity in which the organization is

engaged.
- Waste Management. This model enables efficient, recycling friendly, municipal or industrial waste management using containers, litters, etc. The covered entity types include:
  - WasteContainerIsle: the isle that holds one or more containers.
  - WasteContainerModel: a model of waste container, capturing its static properties such as dimensions, materials or features.
  - WasteContainer: an instance of waste container placed at a isle or place.
- Parking. This model provides real time and static parking data (on street and off street) interoperable with the EU standard DATEX II. This model includes the following entity types:
  - OffStreetParking: an off-street parking site with explicit entries and exits.
  - ParkingAccess: an access point to an off-street parking site.
  - OnStreetParking: an on street, free entry (but might be metered) parking zone which contains at least one or more adjacent parking spots.
  - ParkingGroup: a group of parking spots.
  - ParkingSpot: an individual, usually monitored, parking spot.
- Weather. Weather observed, weather forecasted or warnings about potential extreme weather conditions.

4 FIWARE integration in SMART-FI environment

In this section we make a brief analysis on how the FIWARE components are used in SMART-FI environment.

Each pilot city has several open and non-standardized data (in different domains) coming from their own environments. These data must be standardized, collected and kept securely in FIWARE based solutions under the SMART-FI umbrella. Here we define our approach based on this familiar smart city issue below:

- Define data based on the pilot use cases
- Collect data
- Standardize data based on the NGSI standards with the help of FIWARE data models
- Export the standardized and normalized data periodically to FIWARE Orion Context Broker
- Configure FIWARE Cygnus component so that data would be sent to 3rd party applications like CKAN or databases.
- Implement the security mechanisms by the help of FIWARE IDM Generic Enabler

To support the workflow above, we do not need all the components defined in FIWARE reference architecture but these below:

FIWARE Orion Context Broker

Orion is a C++ implementation of the NGSIv2 REST API binding developed as a part of the FIWARE platform.

Orion Context Broker allows developers to manage the entire lifecycle of context information including updates, queries, registrations and subscriptions. It is an NGSIv2 server implementation to manage context information and its availability.

Orion Context Broker allows to create context elements and manage them through updates and queries. In addition, it allows to subscribe to context information so when some condition occurs (e.g. the context elements have changed) a notification is being sent.

FIWARE Cygnus

Cygnus is a connector in charge of persisting certain sources of data in certain configured third-party storages, creating a historical view of such data.

Internally, Cygnus is based on Apache Flume, a technology addressing the design and execution of data collection and persistence agents. An agent is basically composed of a listener or source in charge of receiving the data, a channel where the source puts the data once it has been transformed into a Flume event, and a sink, which takes Flume events from the channel in order to persist the data within its body into a third-party storage.

Cygnus is designed to run a specific Flume agent per source of data.

Current stable release can persist the following sources of data in the following third-party storages:

NGSI-like context data in: HDFS, the Hadoop distributed file system. MySQL, the well-known relational database manager. CKAN, an Open Data platform. MongoDB, the NoSQL document-oriented database. STH Comet, a Short-Term Historic database built on top of MongoDB. Kafka, the publish-subscribe messaging broker. DynamoDB, a cloud-based NoSQL database by Amazon Web Services. PostgreSQL, the well-known relational database manager. Carto, the database specialized in geolocated data. Twitter data in: HDFS, the Hadoop distributed file system.

FIWARE IoT Manager

The IoT Agent Manager works as a proxy for scenarios where multiple IoT Agents offer different southbound protocols. The IoTA Manager appears as a single administration endpoint for provisioning tasks, redirecting provisioning requests to the appropriate IoTAgent based on the declared protocol.

The IoTAgent Manager also offers a cache of all the provided device Configurations, to fasten the retrieval of certain information from the Agents.

FIWARE IDM

Keyrock is the FIWARE component responsible for Identity Management. Using Keyrock (in conjunction with other security components such as PEP Proxy and Authzforce) enables developers to add OAuth2-based authentication and authorization security to the services and applications.

The main identity management concepts within Keyrock are:

Users

- Have a registered account in Keyrock.
- Can manage organizations and register applications.

Organizations

- Are group of users that share resources of an application (roles and permissions).
- Users can be members or owners (manage the organization).

Applications

- has the client role in the OAuth 2.0 architecture and will request protected user data.
- Can authenticate users using their Oauth credentials (ID and secret) which unequivocally identify the application
- Define roles and permissions to manage authorization of users and organizations
- Can register Pep Proxy to protect backends.
- Can register IoT Agents.

Keyrock provides both a GUI and an API interface.

5 Conclusion

This document defines several candidate reference technologies both from FIWARE environment and from other different sources, by giving brief information on the functionalities and features of the modules which would act as baseline technology stack in SMART-FI environment. We have divided the overall approach into 4 main building blocks as mentioned in the beginning of the document which we have talked about the possible solutions:

- Context management
- IoT management
- Data storage management
- Security

Then we switch to the core part of the document which is related with the data normalization activities that is based on NGSI specification and later mentioned the FIWARE components used in the hands-on implementation activities. As FIWARE and NGSI specification is selected to build the baseline of the platform, we have covered and analyzed the must-have FIWARE modules in the final chapter in order to give a brief idea of the applications that we use in SMART-FI platform.

6 References

1. EU Internet Handbook – Keep it simple http://ec.europa.eu/ipg/content/tips/volume_en.htm
2. http://www.city-go.eu/
3. www.fiware.org
4. https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Main_Page . Accessed: 16.08.2018.
5. Nastic, Stefan, Javier Cubo, Malena Donato, Schahram Dustdar, Örjan Guthu Mats Jonsson, Ömer Özdemir, Ernesto Pimentel, and M. Serdar Yümlü. "SMART-FI: Exploiting Open IoT Data from Smart Cities in the Future Internet Society." In Internet of Everything, pp. 153-173. Springer, Singapore, 2018.
6. SMART-FI Proposal 2016. http://www.smart-fi.eu/