



IRIS

Integrated and Replicable Solutions
for Co-Creation in Sustainable Cities

Project Acronym:	IRIS
Project Full Name:	Integrated and Replicable Solutions for Co-Creation in Sustainable Cities
Grant Agreement:	No 774199
Project Duration:	5 years (starting 1 October 2017)

Deliverable 4.4

Document with technical solution reference architecture for CIP components

Work Package:	WP4: City Innovation Platform (CIP)
Task:	T4.2 CIP technical solution reference architecture
Lead Beneficiary:	CIV
Due Date:	30/09/2018 (M12)
Submission Date:	10/12/2018 (M14)
Deliverable Status:	Final
Deliverable Style:	R
Dissemination Level:	PU
File Name:	D4.4 Document with technical solution reference architecture for CIP components.pdf



This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 774199

Authors

Surname	First Name	Beneficiary
Arjen	Hof	CIV
Chaudanson	Lionel	NCA
Roux	Stephane	NCA
Chaudanson	Lionel	NCA
Kruse	Thomas	UTR
Lantto	Kim	GOT
Tryferidis	Thanasis	CERTH
Tsarchopoulos	Panagiotis	CERTH
Pramangioulis	Dionysios	CERTH
Angelakoglou	Komninos	CERTH
Dartfeldt	Elin	GOT
Nordström	Camilla	GOT

In case you want any additional information or you want to consult with the authors of this document, please send your inquiries to: irissmartcities@gmail.com.

Reviewers

Surname	First Name	Beneficiary
Tsarchopoulos	Panagiotis	CERTH
Keim	Christian	EDF

Version History

Version	Date	Modifications made by
0.3	08/08/2018	First draft created by Civity and Nice
0.5	26/09/2018	Second draft by Civity
0.6	28/10/2018	Third draft by Civity
0.8	25/11/2018	Final draft by Civity
0.9	7/12/2018	New version based on the reviewers' comments
1.0	10/12/2018	Final version to be submitted to European Commission

Disclaimer

This document reflects only the author's view. Responsibility for the information and views expressed therein lies entirely with the authors. The Innovation and Networks Executive Agency (INEA) and the European Commission are not responsible for any use that may be made of the information it contains.

Executive Summary

The City Innovation Platform (CIP) reference architecture supports the technical development and implementation of projects within the IRIS-cities. This is strongly recommended to read D4.2 before going into D4.4. D4.2 describes the functional aspects of the architecture are described, whilst D4.4 focuses on the technical components.

A sustainable open urban platform architecture requires a strong connection to actual business challenges. It needs to be flexible and adaptive, because requirements are growing, technologies are changing and standards are developing. To implement a sustainable CIP, a structured approach is required with a connection to actual user stories. This structured approach is achieved by using the CurateFX toolset offered by TM Forum. All relevant aspects, from stakeholders, ecosystem, value propositions and Open API's can be discussed and documented. Chapter 2 describes a concrete use case for each Light House city, modelled in CurateFX.

Based on the use cases, the ecosystem design and the requirements from D4.2, the technical components to implement the CIP-architecture are described. The functionality within the CIP can be fulfilled with different software solutions, like database technologies, rule/event engines or API-management frameworks. The CIP should be technology agnostic. More important for a sustainable, open urban platform is the adherence with standards and Open API's. They are the glue between (interchangeable) software components. The CIP follows a standards based, open, agile and modular approach. The FIWARE-architecture and the TM Forum Open API's match these requirements. The Lighthouse (LH) cities have joined the front runner program from these organizations to collaborate on the development of an open, standardized urban platform for smart cities. A reference implementation of the CIP will use these proven software solutions and components.

The approach for the CIP is to learn from the implementation experiences in the use cases. This reference architecture is a living document, that will be improved, updated and extended with results from the actual project. The development of the Data Market, the Platform Management and the integration of BIM/CIM-models within the CIP will lead to new insights that will be incorporated in the next version of this architecture. In the case that software components are lacking or immature, collaboration will be searched with other cities, partners and the aforementioned organizations, so that generic, reusable and open components can be added to the overall architecture. The close interaction between IRIS use cases and the architecture with its components, models and standards will ensure that this document will evolve towards a practical and usable reference for implementing urban platforms.

Table of Contents

Executive Summary	3
Table of Contents	4
List of Figures.....	6
List of Tables.....	7
List of Terms and definitions	8
1. Introduction	10
1.1 Scope, objectives and expected impact.....	10
1.2 Contributions of partners.....	10
1.3 Relation to other activities.....	11
1.4 Structure of the deliverable	11
2. Methodology.....	12
3. Use Cases.....	14
3.1 CurateFX.....	15
3.2 Gothenburg BIM/CIM use case	17
3.2.1 Visualize your city.....	17
3.2.2 User story	17
3.2.3 Stakeholder Ecosystem	17
3.3 Nice e-mobility use case	21
3.3.1 User Story of “Nice e-mobility use case”	21
3.3.2 Nice e-mobility use case - Stakeholder Ecosystem	21
3.4 Utrecht civic issue tracking use case.....	25
3.4.1 Utrecht civic issue tracking use case - Stakeholder Ecosystem	25
3.5 Summary	28
4. CIP Architecture and Components.....	29
4.1 CIP Architecture	29
4.2 Data management framework.....	33
4.2.1 CKAN	33
4.2.2 FIWARE Orion Contextbroker, Cygnus and QuantumLeap.....	37
4.3 Security and Privacy	43
4.4 Data Market	47

4.4.1 IdM	48
4.4.2 Open API's and API-management	49
4.4.3 Open API's	49
4.5 Platform Management	53
4.5.1 Device connectivity and communication	54
4.5.2 Device Lifecycle management	55
4.5.3 Data transformation	55
4.5.4 Provisioning of devices	55
4.5.5 Platform management solutions	55
4.6 Proprietary systems	60
5. Conclusions	61
Annex 1: ARX	62

List of Figures

Figure 1 TM Forum and FIWARE integration within CIP	13
Figure 2 Example of ecosystem with stakeholders and roles in CurateFX.....	15
Figure 3 Categories for relationships between stakeholders.....	16
Figure 4 Types of Open API’s related to Data category relationship	16
Figure 5 Stakeholders and roles of Gothenburg BIM/CIM use case described in CurateFX	18
Figure 6 Mapping of stakeholders and roles for Gothenburg BIM/CIM use case.....	18
Figure 7 BIM/CIM Use case Gothenburg in CurateFX	19
Figure 8 Data connection between roles and stakeholders in the Gothenburg BIM/CIM use case	19
Figure 9 Mapping the Gothenburg BIM/CIM use case on the City Innovation Platform architecture	20
Figure 10 Stakeholders and roles of Nice Cote Azur e-mobility use case described in CurateFX.....	22
Figure 11 Mapping of stakeholders and roles for Nice Cote Azur e-mobility use case	22
Figure 12 Nice Cote Azur e-mobility use case in CurateFX.....	23
Figure 13 Data connection between roles and stakeholders in the Nice Cote Azur e-mobility use case...	23
Figure 14 Mapping the Nice e-mobility use case on the CIP architecture	24
Figure 15 Stakeholders and roles of Utrecht civic issue tracking use case described in CurateFX.....	25
Figure 16 Mapping of stakeholders and roles for Utrecht civic issue tracking use case.....	25
Figure 17 Civic issue tracking use case Utrecht in CurateFX	26
Figure 18 Mapping the Utrecht civic issue tracking use case on the CIP architecture	27
Figure 19 FIWARE Architecture overview	30
Figure 20 Mapping of FIWARE enablers and TM Forum Open APIs on CIP	30
Figure 21 EIP SCC Urban Platform Capability Map with capabilities per category	31
Figure 22 The Data Management Framework within the CIP.....	33
Figure 23 Table view in CKAN of dataset	34
Figure 24 Map view in CKAN of dataset	34
Figure 25 CKAN offers a generic API to access data in the datastore	35
Figure 26 Example of data and filetypes that is collected with the Harvester from ESRI ArcGIS	36
Figure 27 ETL-tools (like FME) allow to push data automatically to CKAN through its API. In this case LoD-files (Levels of Definition) for BIM/CIM-purposes.....	36
Figure 28 Data Element Structure	37
Figure 29 Context Element Structure Model	38
Figure 30 QuantumLeap Generic Enabler within the FIWARE-architecture	39
Figure 31 FIWARE Data/Context architecture.....	40
Figure 32 Security and Privacy components in CIP	43
Figure 33 Level 1 authentication with PEP Proxy	45
Figure 34 Level 2 Basic Authorization with PEP Proxy	45
Figure 35 Level 3 Advanced Authorization with PEP Proxy.....	46
Figure 36 Data Market component in CIP	47
Figure 37 TM Forum Open API's offers specifications for 50+ Open API's	50
Figure 38 Common elements within an API-management solution.	51
Figure 39 Platform Management component within CIP	53

Figure 40 FIWARE Backend Device Management architecture (IDAS)	56
Figure 41 OpenMTC & Fiware architecture	58
Figure 42 Overview ARX API.....	62

List of Tables

<i>Table 1 Definitions</i>	9
Table 2 Overview EIP SSC capabilities and CIP-components.....	32
Table 3 IDAS and OpenMTC features' comparison	59

List of Terms and definitions

Abbreviation	Definition
API	Application Programming Interface. A software intermediary that allows for distinct applications or systems to interact with one another.
BIM	Building information model The BIM-model, also called object-based model is a three-dimensional model with data information.
Capability	The abstract representation of what is needed to produce an outcome along with goals and metrics for that outcome.
CIM	City Information Model
CIP	City Innovation Platform
CIP-component	The following five components of the City Innovation Platform: <ol style="list-style-type: none"> 1. Data management framework 2. Data market 3. Security and privacy 4. Platform management 5. Proprietary systems connectivity (federated solution)
CKAN	Comprehensive Kerbal Archive Network. Open source catalogue system for open data portals.
Data Portal	A software solution (usually a website) that presents a catalogue of searchable and downloadable datasets in a user-friendly and uniform way.
Data Set	A collection of data that can be downloaded and processed further.
DCAT-AP	Data Catalogue Vocabulary (-Application Profile), is an RDF vocabulary designed to facilitate interoperability between data catalogues published on the web. The Application Profile is developed by the European Commission for interoperability optimisation between European Data Portals
DPA	Data Protection Authority
DPO	Data Protection Office
EIP-SSC	European Innovation Partnership on Smart Cities and Communities
ESPRESSO	EU-project (2016-2017) that identified a collection of open standards for smart cities that work well together (“conceptual standards framework”) and have been proven
FIWARE	“Future Internet-ware”; an open software- and standards framework
Functional requirement	Functional requirements describe the desired end function of a system to assure the design is adequate and meets user expectations. In this document also used as “Design Principles”.
GDPR	General Data Protection Regulation
IAM	Identity & Access Management
Interoperability	The ability of different information technology systems and software applications to communicate, exchange data, and use the information that has been exchanged.
IoT	Internet of Things

Linked Data	A method of publishing structured data so that it can be interlinked and become more useful through semantic queries, facilitating the sharing of machine-readable data on the web.
Metadata	Data about data
Open data	Data carrying an open licence stating it can be freely used, re-used and redistributed by anyone, for any purpose.
Open & Agile Smart Cities	Open & Agile Smart Cities (OASC) is a global initiative connecting cities, advocating de facto standards, and sharing best practices.
OUP	Open Urban Platforms (also working group of EIP-SCC) Definition: (BSI, 2016)
PbD	Privacy by Design
PEAR	Privacy Enhancing ARchitecture
PET	Privacy Enhancing Technology
PIA	Privacy Impact Assessment
PMRM	Privacy Management and Reference Model and Methodology
PRIPARE	Preparing Industry to Privacy-by-design by supporting its Application in Research
Reference architecture	A template that offers a common language and support for standards, specifications and patterns, a list of functions and interfaces (APIs) and their interactions with each other.
RDF	Resource Description Framework: a standard model for data interchange on the web.
Role	Responsibility and/or activity of a stakeholder within the CIP
SLA	Service Level Agreement
Stakeholder	A person or group with specific interest in the CIP.
Technical requirement	Technical requirements define what is required to deliver the desired function or behaviour from a system to a user’s standards.
TM-Forum	Is a neutral, non-profit member organization.
USEF	The Universal Smart Energy Framework (USEF) is an international common standard that ensures smart energy technologies and projects are connectable at lowest cost.
WP	Work Package

Table 1 Definitions

1. Introduction

The City Innovation Platform (CIP) collects, manages and exchanges data for the development of new applications and services. The CIP and its components will manage large volumes of data and information coming from vertical solutions, municipal systems, external platforms and different data sources. To implement the City Innovation Platform, we need technical guidelines and requirements, based on the CIP reference architecture (D4.2) and the use cases from the Lighthouse cities.

1.1 Scope, objectives and expected impact

The objective of this document is to translate the use cases of the Lighthouse Cities into a technical architecture for the CIP. The requirements described in D4.2 “Functional technical requirements for integrated interoperable and open solutions standards and new business models” offer guidelines for this technical architecture.

This document is a reference for each implementation of the CIP within Lighthouse and Follower cities. It describes the technical components, data processes and API’s and that help to fulfill the business needs.

The scope of this document is on the technical components that support functions like data management, security/privacy, platform management and the development of a Data Market with open API’s.

This reference architecture is the foundation for actual projects within IRIS and therefor impacts all future activities concerning data.

1.2 Contributions of partners

The technical reference architecture for the CIP is a joint effort between the three Lighthouse cities and Civity. Each of the partners has contributed with a set of use cases of urban challenges and provided insight in their current technical infrastructure. The use cases provide insight in actual needs and guide the implementation of technical components to fulfill those needs.

On the occasion of the TM Forum held in in Barcelona (November 2018), the 3 Lighthouse cities was selected as part of the [FrontRunner](#) smart cities program. This program is “a joint collaboration program to support the adoption of a reference architecture and compatible common data models that underpin a digital market of interoperable and replicable solutions for smart cities”.¹ The CIP reference architecture uses the FIWARE NGSI API, Context broker and TM Forum Open APIs for interoperability and scalability of city solutions. Collaboration with other cities, partners and organizations provide great benefits and increased opportunities for sustainable solutions.

¹ <https://www.fiware.org/news/fiware-foundation-and-tm-forum-launch-front-runner-smart-cities-program/>

1.3 Relation to other activities

This document builds upon the work that has been done in D4.1 and D4.2. D4.1 provided a baseline about the current situation in each Lighthouse city. D4.2 described the functional reference architecture, based upon an analysis of stakeholders, capabilities and components.

1.4 Structure of the deliverable

As described in the reference architecture (D4.2), CIP is conceived as vendor neutral and technology agnostic.

Chapter 2 present the methodology. When implementing CIP in each IRIS LH city, choices have to be made about which components to use. Therefore we need criteria for selecting the right technical CIP-components. These criteria are determined by the required business capabilities, technological maturity, openness, developer community, etc. Based on these criteria this document will describe a technical reference implementation for CIP, with emphasis on open source software components, standard data models and open API's.

Chapter 3 describes user stories for each LH city. With these stories a stakeholder model and ecosystem design is developed within CurateFX. By using the same toolset for all projects, a standardized description of the relationships between parties involved can be achieved. Relationships can be about contracts, data, finance, operations and the product/service. Each relationship might demand different technical solutions for performance, availability, privacy and security or billing.

Chapter 4 focuses on the CIP components needed to develop/implement the use cases. Besides the generic components, standards and models, there is specific emphasis on two important components of the architecture: the Data Market and Platform Management. The Data Market has a strong relationship with the Open API's from TM Forum (billing, performance, NGSI, etc.). For Platform Management there are several options, like existing FIWARE Generic Enablers, to support the implementation of the use cases. The Data Management Framework in CIP is the most mature part, based upon open components, like CKAN, Orion, Cygnus and Keyrock. For this component attention is needed to support BIM and CIM capabilities within CIP.

2. Methodology

The technical architecture supports the implementation of the use cases. From a technical perspective, one can, figuratively, develop and implement a luxury car, while a bicycle might be enough for the goals to be achieved. Therefore, we started the technical architecture with an analysis and description of use case from the LH cities. These cases helped to select and implement the right technical components and create an adaptable solution that can grow with future demands.

To develop the technical architecture for the CIP we started with an in-depth desk research of existing reference architectures for Urban Platforms. Information about these references is available in D4.2 “Functional technical requirements for integrated interoperable and open solutions standards and new business models”. Based on the guidelines in D4.2 we focused on technical documentation provided by FIWARE and TM Forum.

To develop a structured, repeatable and sharable solution, the TM Forum Business Framework is used. TM Forum offers an additional toolset (CurateFX) that is helpful to develop and achieve a future-proof and sustainable architecture. Each LH city has had a training and used CurateFX to describe a use case. In the context of this reference architecture a “use case” is a description of a problem/challenge for each city. These use cases are not (or loosely) connected to the cases described in T5.6, T6.6 and T7.6. The purpose of the three use cases in this document is to provide input for the priorities regarding the development of the CIP. One of the helpful activities in describing the use cases was the definition of the stakeholders and their roles and the structuring of the information in an ecosystem design. Based on the relationship between the stakeholders, the corresponding Open API’s could be identified.

The technical architecture follows in great extend the work already done by FIWARE and TM Forum. For the software components the first preference is to use existing enablers (Figure 1). All three LH cities have become front runners in the combined smart cities program from FIWARE and TM Forum and will further implement the common reference architecture, developed by these organizations. The benefit of this joint collaboration program is a bigger support for the adoption of a common reference architecture and compatible data models that underpin a digital market of interoperable and replicable solutions for smart cities.

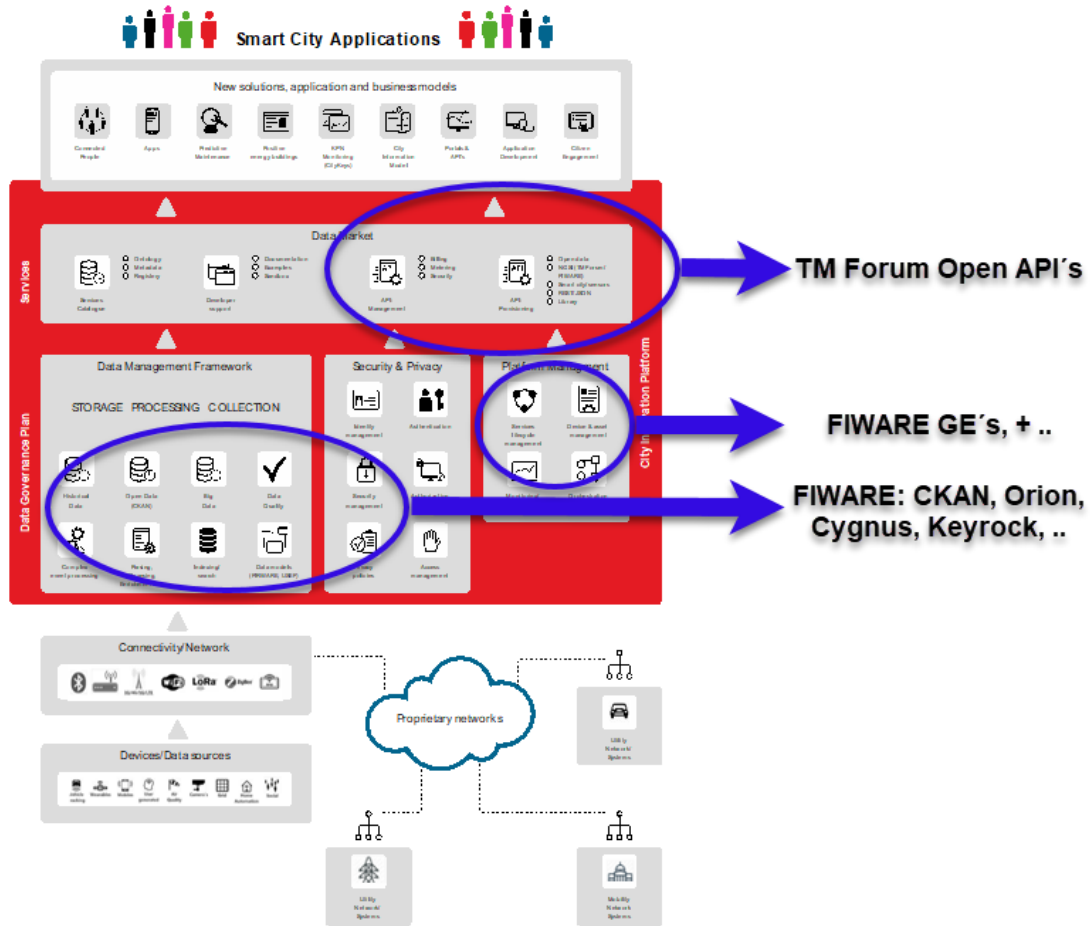


Figure 1 TM Forum and FIWARE integration within CIP

3. Use Cases

The CIP technical architecture needs to support the successful implementation of use cases of the LH cities (WP: 5,6,7).

With use cases we refer to actual urban challenges in cities. The ambition of this approach is to anticipate what the integrated solutions will be in order to set up the CIP adequately. Each use case is composed of both a “story user” and a “role model scheme” (using CurateFx).

The use cases are on purpose defined with a high level of detail in order to facilitate the replication in other cities. Also, the use cases are described according the maximal theoretical state of the art. This way, each use case can be specified with more specific details in a later stage if needed.

These use cases are not described in technical terms, but in functional requirements. The expectations and needs from these use cases help to make the translation to technical choices for the standards, components, data models, etc. and ultimately offer the tools to fulfill the business needs. This chapter contains a use case from each LH city. To structure the description and approach for each use case, the toolset offered by TM Forum (CurateFX) is used. There is a relationship with D4.2, regarding the stakeholders and capabilities.

The selection of use cases is not meant to be complete or comprehensive. In T5.6, T6.6 and T7.6 more exhaustive uses cases are described. The use cases in this chapter are meant as input for learning to work with CurateFX and to provide generic insight in technical foundation needed to support implementation of use cases in general. This is then used to identify the needed programmatic, component interfaces that have to be ensured to make them interoperable.

3.1 CurateFX

CurateFx is a combination of strategic planning, visualization tools, business and technical perspectives, integrated framework models, and collaborative capabilities to enable organizations to make faster, more confident business decisions about complex business scenarios and ecosystems. It is used to bring digital ecosystems to life and design business model with multiple stakeholders. It helps to define, agree and document relationships, roles and processes between stakeholders. It also helps to identify quickly standards and APIs needed to operationalize digital products and services.²

To further detail the roles and the associated stakeholders, the corresponding platform design within the IRIS project, we will use CurateFX to structure our work. This is work in progress and will continue during the project. An example helps to provide a better understanding. In the scheme below (Figure 2), part of the use case in Nice Cote Azur is shown. The circles and squares **(1)** refer to roles involved in this use case. The lines **(2)** indicate the relationship between such roles. In this example only “data” relations are shown. There are 5 different categories for the relationships between the different identified roles (Figure 3). For each relationship, the type can be defined **(3)**. This type refers directly to the TM Forum Open API’s (Figure 4). In this example the type refers to the product inventory API **(2)**. By structuring the use case by its roles, it provides insight in capabilities and API’s (and technical components) needed to fulfill the user story. This is just one of the tools available within CurateFX. When further developing the uses cases, the other CurateFX tools will also be used.

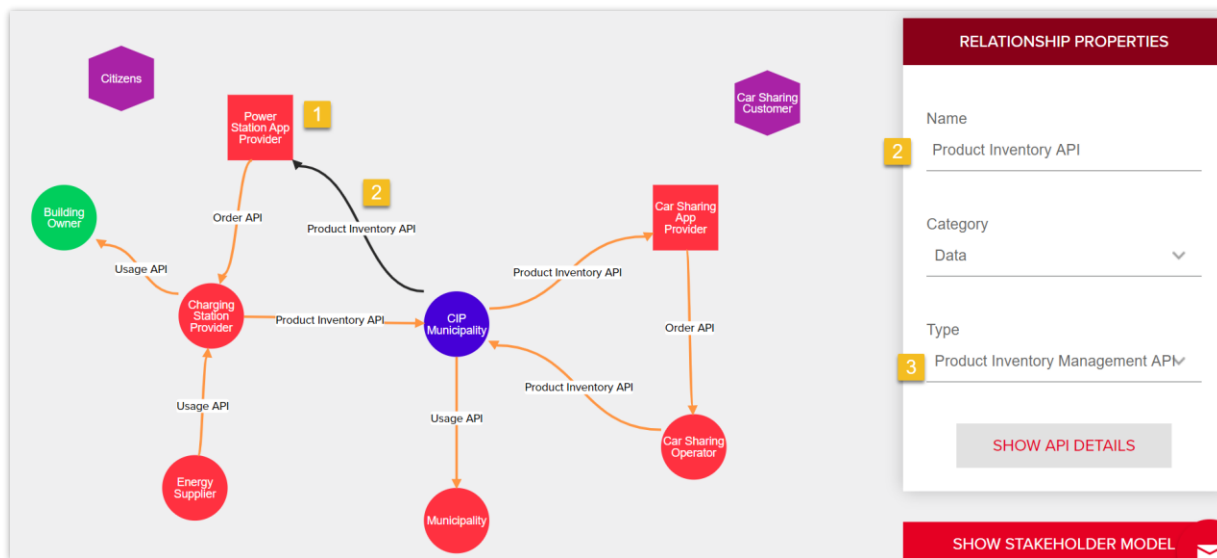


Figure 2 Example of ecosystem with stakeholders and roles in CurateFX

² <https://www.tmforum.org/curatefx/>

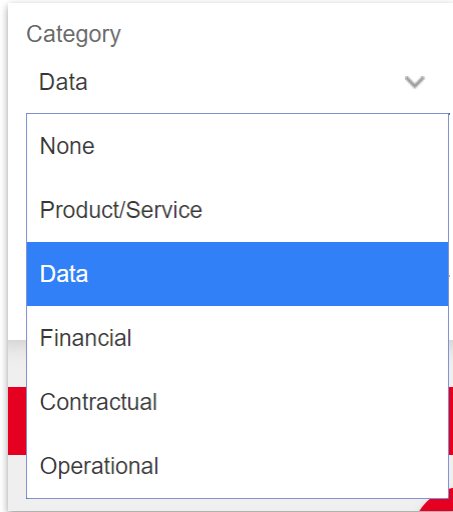


Figure 3 Categories for relationships between stakeholders

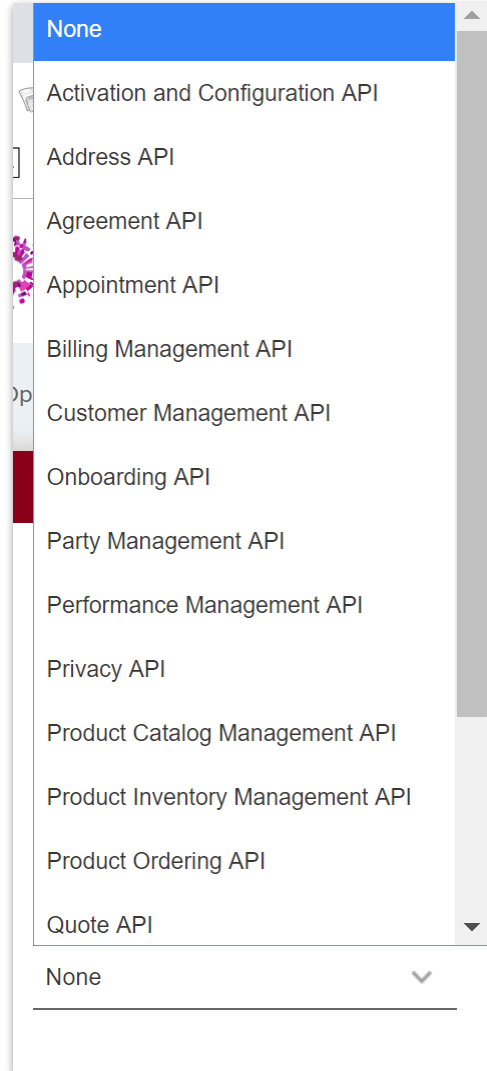


Figure 4 Types of Open API's related to Data category relationship

3.2 Gothenburg BIM/CIM use case

3.2.1 Visualize your city

The main objective is to give citizens and users an easier way to access/acknowledge projects and means to influence the planning process in order to achieve better and smarter planning through participatory design. Furthermore, the City of Gothenburg and the Urban Transport Administration need to ensure the improvement of city operation performance while improving the citizens' life.

To achieve such objectives, the Urban Transport Administration has to share information and give the citizens a chance to co-create and be a part of the decision-making process. The citizens have to be able to report issues and give feedback that improves the city. By sharing data and information it is expected that the efficiency for the city planners increase. During planning, contracting and building the quality increases by using co-reviewing and collision controls. It may also increase engagement for all parts as it will be easier to be a part of the decisions making process.

3.2.2 User story

Before building the planned bridge to Hisingen Island the City officials would like to have feedback from their citizens. They make sure the plans are shown in the City's relevant application and ask the citizens to give feedback.

The citizen wants good mobility in their neighborhood. They check out the plans in the City's tool for ongoing and future infrastructure constructions to see what the city's plans are.

The citizen discovers that the planned bridge to Hisingen island does not have satisfactory bike lanes and comments that. The project administration receives the comment and determines that the comment is relevant and decides to make an amendment in the construction. The project managers and city officials redraw the bike lanes accordingly in their design tool and resubmits the BIM (Building Information Model) file to the CIP. The Citizen can look at the changed plan for the bridge via the City's application that visualizes data coming from the CIP. The project brings the comment into the lessons learned during its retrospective for future constructions to consider.

To support the use case, a CIM (City Information Model) data API needs to be developed, where current or new visualization tools easily can access the outcomes of ongoing infrastructure projects in combination with existing geodata and other data. There is also a need for a tool where the BIM providers can submit their BIM-data in a standardized way.

3.2.3 Stakeholder Ecosystem

Based on the user story above, the stakeholders and their roles are defined in CurateFX (see figures below). Gothenburg defined 14 different stakeholders, with 21 different roles (Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε., Σφάλμα! Το αρχείο προέλευσης της αναφοράς δεν βρέθηκε.).



Figure 5 Stakeholders and roles of Gothenburg BIM/CIM use case described in CurateFX

STAKEHOLDER	ROLES
The Urban Transport Administration	Owner of traffic infrastructure and data x Data supplier x Owner x
DESCRIPTION	Creator of BIM Files x Finance x KPI x Monitors usage x
	Manager Accessibility and availability x Builder x
STAKEHOLDER	ROLES
City Planning	Owner of city planning infrastructure x Data supplier x Owner x
DESCRIPTION	Creator of BIM Files x Finance x KPI x Monitors usage x
	Manager Accessibility and availability x Builder x
STAKEHOLDER	ROLES
Intraservice	Owner of computer and information infrastructure x Data holder x
DESCRIPTION	Owner x Finance x KPI x Manager of Platform for information x

Figure 6 Mapping of stakeholders and roles for Gothenburg BIM/CIM use case

In the CurateFX ecosystem design module each role is connected. This model shows all different relationships between such roles (eventually having the same or less number of stakeholders). There are 5 different relationships. The schema in Figure 8 shows only the “data” connections between roles, which is relevant for developing and configuring the CIP’s APIs and components.

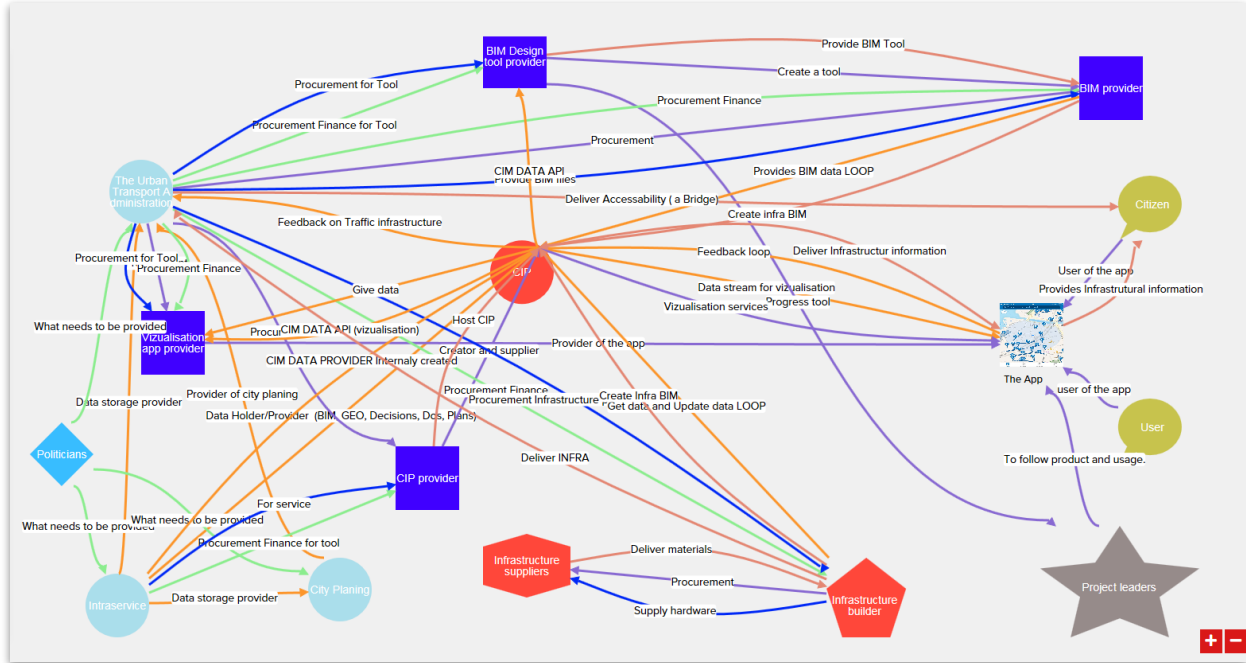


Figure 7 BIM/CIM Use case Gothenburg in CurateFX

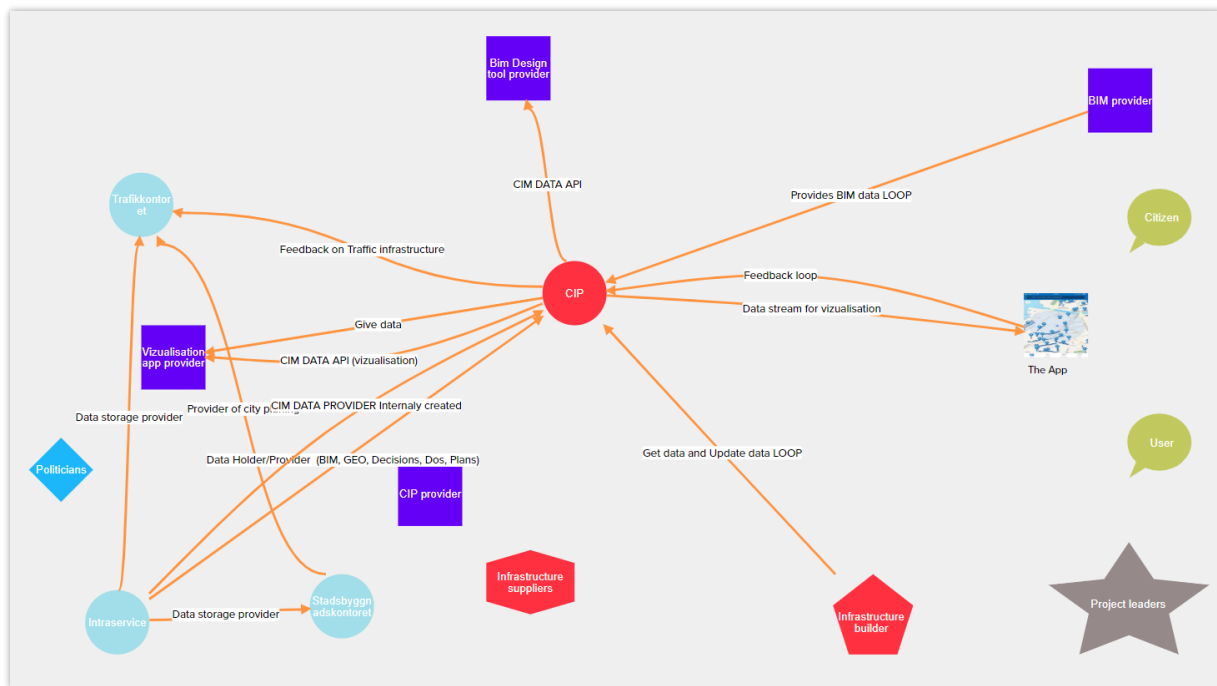


Figure 8 Data connection between roles and stakeholders in the Gothenburg BIM/CIM use case

From the functional requirements in the Gothenburg use case we can derive the following requirements and priorities for the development of the CIP. The basic components of the CIP are shown in the schema of the CIP-architecture below (Figure 9).

1. The need for an open API to provide access to BIM and CIM data. This API is not available within the list of Open API's of TM Forum and could be derived from existing standards from the Open Geo Consortium. The TM Forum Open APIs can add functionalities aimed for example at performance improvements or increased availability.
2. There is a need to store or connect to data from BIM data providers or municipal systems. For open (geo)data, and sensor data the combination of CKAN and Cygnus can serve this purpose. Available data sources can be found in the CKAN register catalog.
3. To manage access to data, an IdM-solution as Keyrock can be added. Extra options for managing access to API's are handled through the API-management layer.
4. Feedback from citizens can be handled in different ways, for example with the Open311 API (part of FIWARE Data models). Reports can be stored as open data (use of CKAN API) and sent to the municipality.

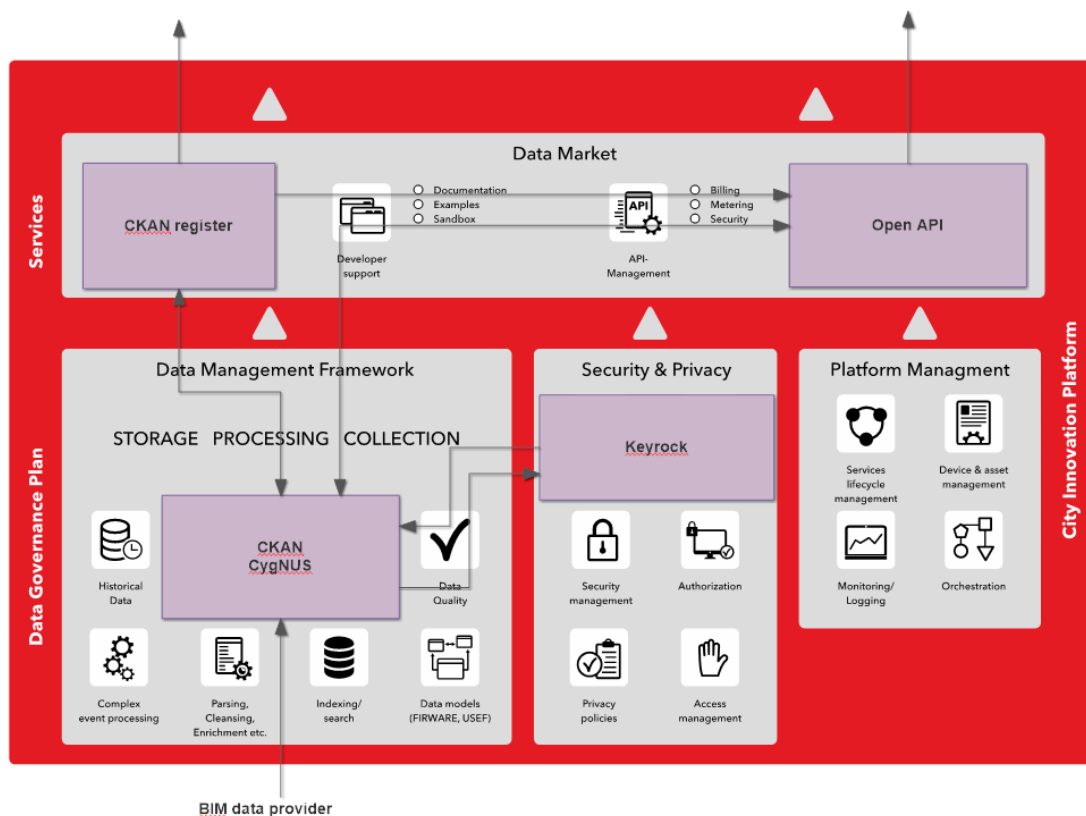


Figure 9 Mapping the Gothenburg BIM/CIM use case on the City Innovation Platform architecture

3.3 Nice e-mobility use case

As part of Task T6.5 a scenario has been developed to support smart mobility in Nice. The name of this use-case is “Nice e-mobility use case”. The overall objective of that scenario is to optimize the management of an Electric Vehicle (EV) free-floating pool, which will be facilitated by an increase in the number of charging points.

T6.5 aims at developing an Integrated Solution (IS) linked to mobility. At the time of defining the technical requirements and specifications of the CIP to permit the collection of data produced by this Mobility IS, the “use case” approach is adopted with twofold objectives:

- identifying potential dysfunctions in T6.5 scenario through the draft of user-experience, that can be corrected with anticipated changes.
- listing a large-scope of potential requirements and specifications (still to be prioritized) that might come with the development of further ISs.

3.3.1 User Story of “Nice e-mobility use case”

A citizen that comes to Nice Meridia with a car from a car sharing pool wants to park the car and looks for a charging point to plug the car. This charging point can be located either in the street (as currently) or in a private property. The EV user must be able to leave and access the private car park. The user must be able to restart the EV-car so it is mandatory to check the connection to the network and the user must be able to drive out of the private car park. In this scenario the private car park has to anticipate security aspects (access to the car-park, but no access to buildings) and have policies in place, for example to manage insurance-issues.

For using the charging point, a billing mechanism needs to be in place. The charging costs might depend on the availability of the energy in the building (T6.3), the availability of the energy in the district (T6.4), the operator’s and the user’s needs. These factors could lead to different charging fees as for example:

- Free for slow recharge or for recharge monitored by the building or the public charging point operator
- Pay for quick charge or for a charge at peak hours
- User has the choice of different pricing schemes for the recharge depending on power and duration of the foreseen charging.

To support the requirements mentioned above, data needs to be available with appropriate license and service models. Data that is needed for this use case concerns for example charging points availability, car park availability, number of EV available on each location for a specific period, at best with the battery level. Privacy and security concerns have to be addressed (as for e.g. access to buildings or license plate recognition) and also the management of (new) sensors.

3.3.2 Nice e-mobility use case - Stakeholder Ecosystem

The Nice e-mobility use case is also modeled in CurateFX (Figure 10, Figure 11, Figure 12, Figure 13).

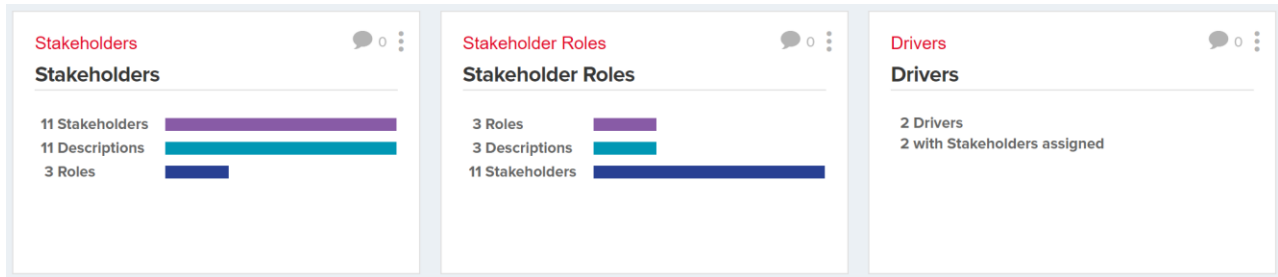


Figure 10 Stakeholders and roles of Nice Cote Azur e-mobility use case described in CurateFX

STAKEHOLDER	ROLES	
CIP Municipality	Data Producer x	+ 🔗 🗑️
DESCRIPTION		
The administration of the city		
STAKEHOLDER	ROLES	
Citizens	Data Producer x Customer x	+ 🔗 🗑️
DESCRIPTION		
Inhabitants of a city using an E.V		
STAKEHOLDER	ROLES	
Charging Station Provider	Data Producer x Data Consumer x	+ 🔗 🗑️
DESCRIPTION		
Supplier of Charging Stations and other related technologies		

Figure 11 Mapping of stakeholders and roles for Nice Cote Azur e-mobility use case

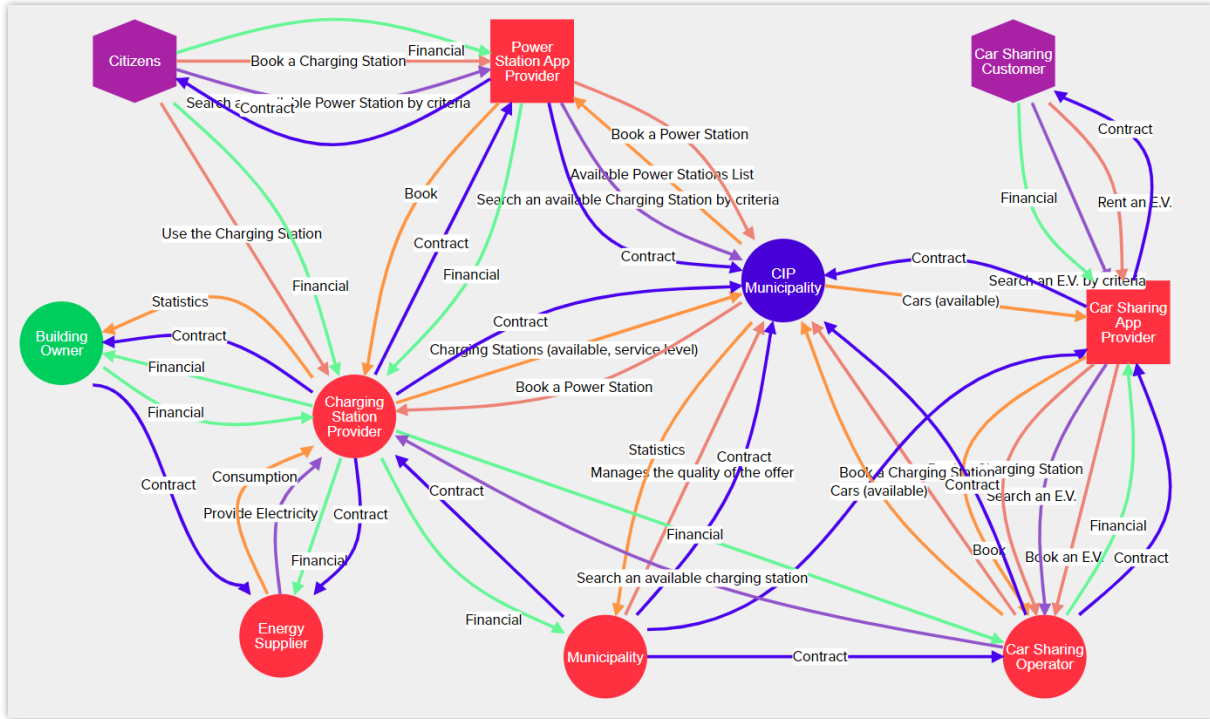


Figure 12 Nice Cote Azur e-mobility use case in CurateFX

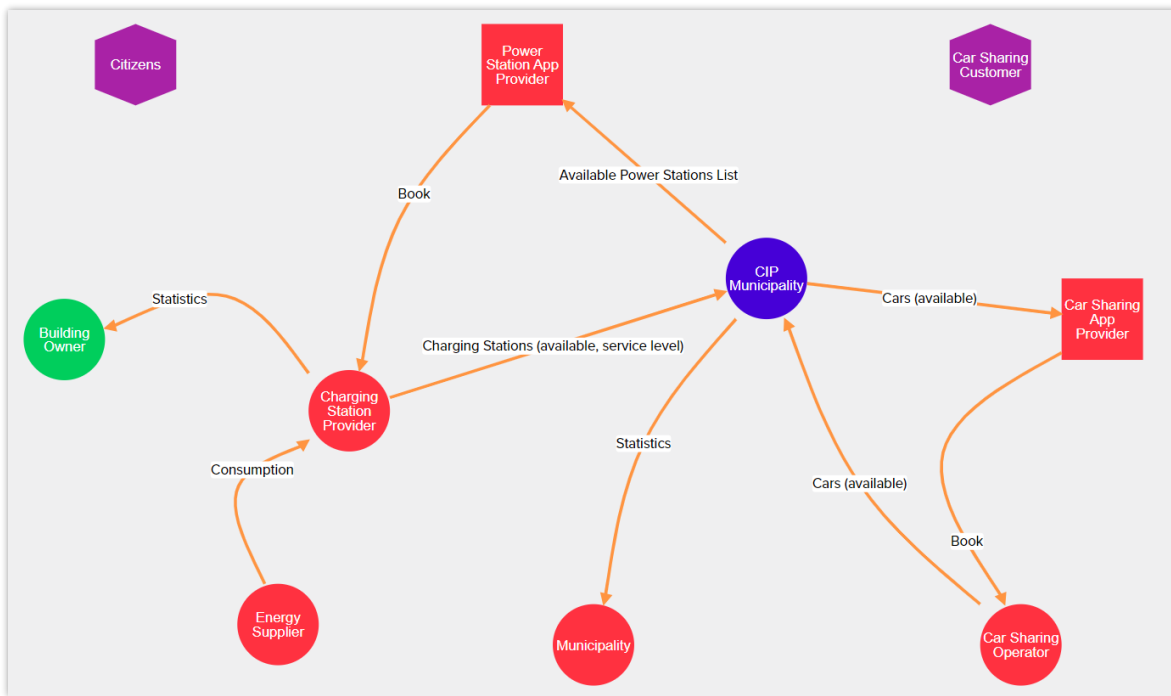


Figure 13 Data connection between roles and stakeholders in the Nice Cote Azur e-mobility use case

From the functional requirements in the Nice use case we can derive the following requirements and priorities for the development of the CIP. The basic components of the CIP are shown in the schema of the CIP-architecture below (Figure 14).

1. Real time information from different sources needs to be connected to the CIP (charging pole availability, energy usage, etc.) and from there to an app to the end user.
2. Many data processes are connected to events and triggers and lead to transactions (billing)
3. The services, transactions and data are sensitive. Appropriate measures to guarantee privacy and security are needed. Components like IdM and API-management are prerequisites.
4. Service levels, performance, scalability and platform management are crucial. Many of the TM Forum Open API's are aimed at these topics.

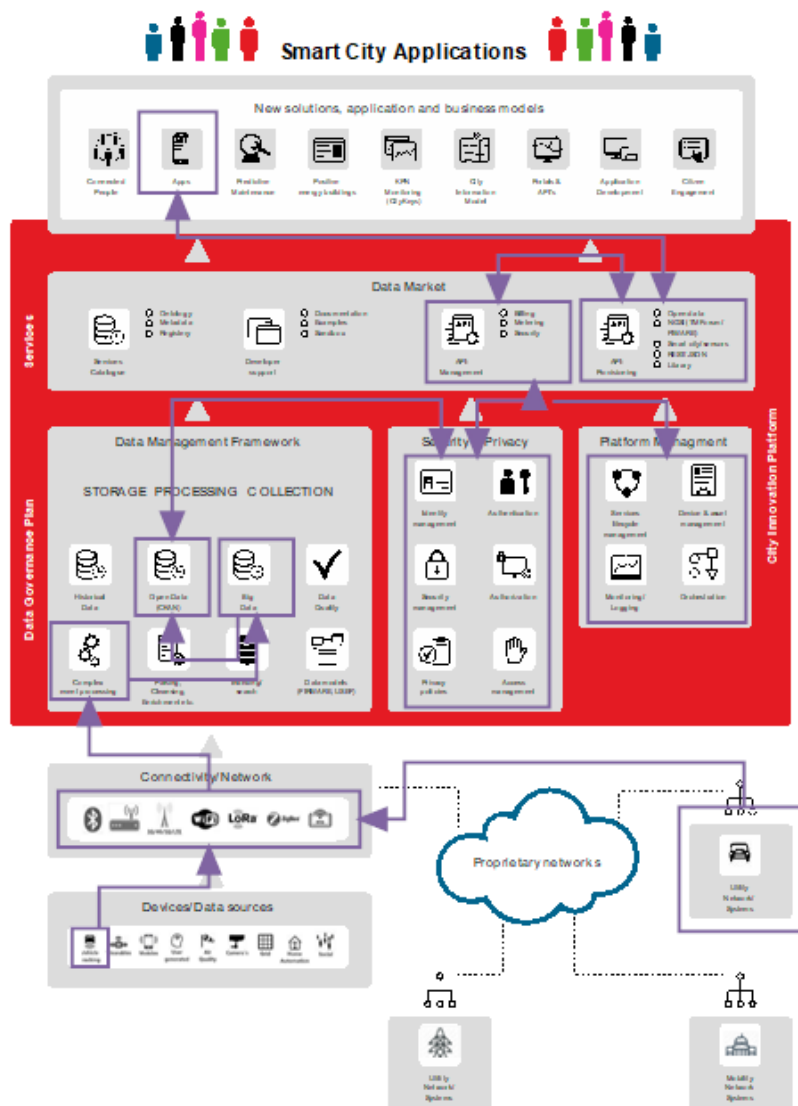


Figure 14 Mapping the Nice e-mobility use case on the CIP architecture

3.4 Utrecht civic issue tracking use case

Reports made by citizens about public space are very diverse: a broken bus stop or streetlight, graffiti on a building, a garbage bin that is overloaded or noise pollution. Engaged citizens use several channels to report these issues: apps, e-forms, phone and e-mail.

Utrecht wants to improve quality of public space, but also planning, staff scheduling, budgeting and policies for maintaining public spaces.

Citizens should be enabled to make reports without adding information already available in (open) datasets. The process should be easy and integrated through the CIP with back-office processes in municipal systems. Where appropriate, reports should directly be sent to subcontractors, ensuring to maintain contractual and service agreements (KPI's).

Utrecht wants insight in reports (dashboard) to improve data driven policies, aiming at predictive analytics.

3.4.1 Utrecht civic issue tracking use case - Stakeholder Ecosystem

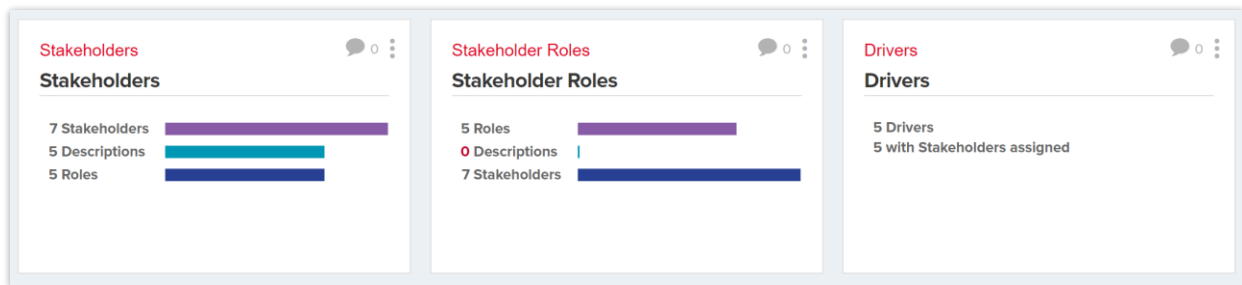


Figure 15 Stakeholders and roles of Utrecht civic issue tracking use case described in CurateFX

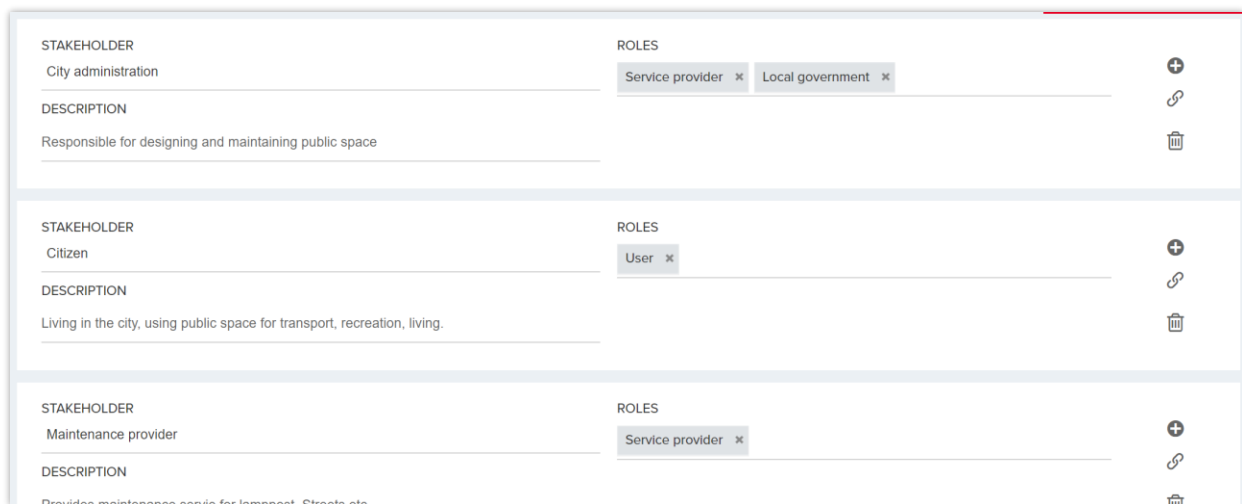


Figure 16 Mapping of stakeholders and roles for Utrecht civic issue tracking use case

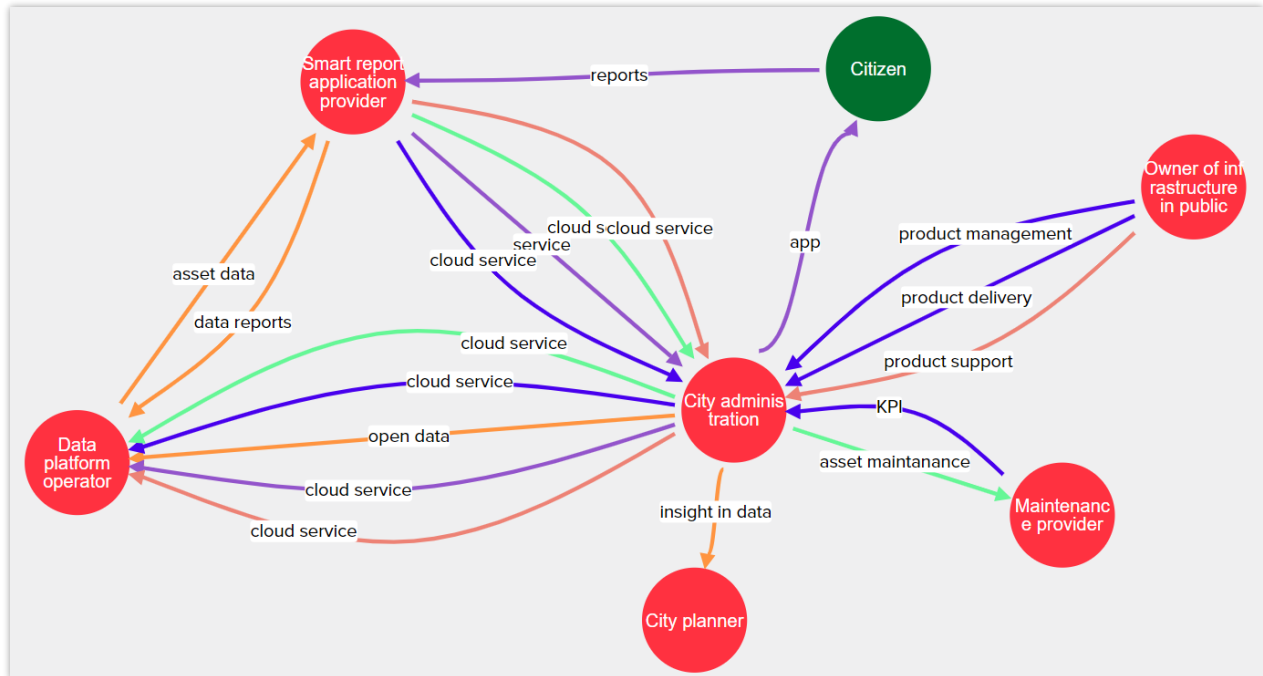


Figure 17 Civic issue tracking use case Utrecht in CurateFX

From the functional requirements in the Utrecht use case we can derive the following requirements and priorities for the development of the CIP. The basic components of the CIP are shown in the schema of the CIP-architecture below (Figure 18).

1. Datasets on public space assets are stored in an open data platform (CKAN)
2. Data is provided through an Open API, based upon a FIWARE Data Model (Civic Issue Tracking³ = open311)
3. Data is sent and retrieved from municipal (proprietary) systems for handling reports
4. Data about report are published as open data (CKAN) and made available for reuse (dashboard, apps)

³ <https://fiware-datamodels.readthedocs.io/en/latest/IssueTracking/doc/introduction/index.html>

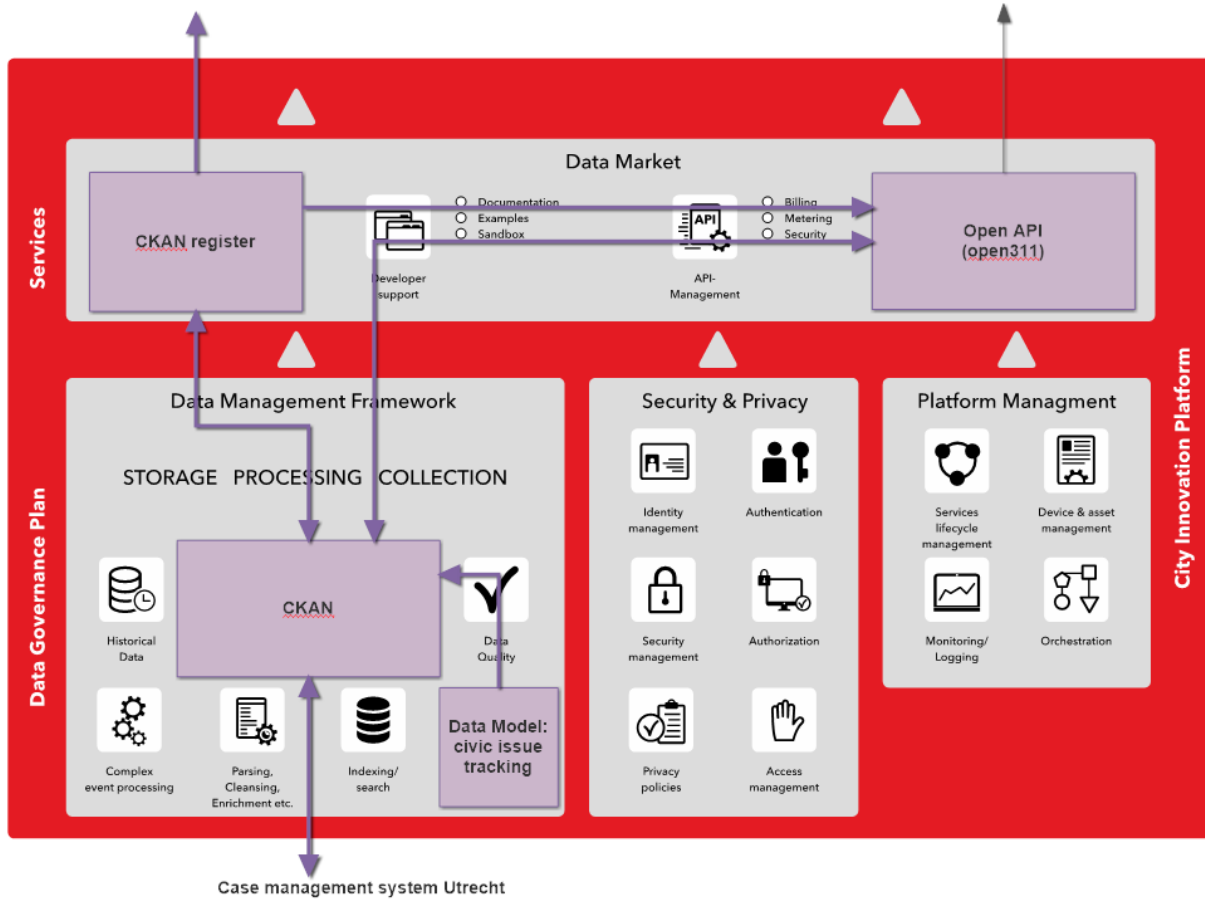


Figure 18 Mapping the Utrecht civic issue tracking use case on the CIP architecture

3.5 Summary

In each description of the use cases above, a rough overview is given about the CIP architecture to visualize data flows and the CIP-components which are touched. The next chapter elaborates more on these flows, components and API's.

The descriptions of the use cases, its roles and the associated interactions, also offer generic requirements that apply to the technical aspects of the CIP. From the use cases the following guidelines are derived:

- 1 Provide (open, geo) data in a standardized way, including the metadata, so that it can be easily used in applications. Standardization applies to topics like syntax, format, measurements, words, identifiers and schemas.
- 2 There is a huge need for visualizing data, based upon BIM and CIM data. Common agreements between cities need to facilitate the exchange of BIM and CIM solutions and apply standardized Data Models and Open API's.
- 3 The use Open API's and supporting API-management to manage access, privacy, performance for different stakeholders and different solutions are needed for event and transaction based solutions.
- 4 The use cases need support for different license and service arrangements.
- 5 To support the development of vibrant business ecosystems a comprehensive business framework is needed with policies, processes and technology. A structured approach like CurateFX helps to define and describe all related topics.

4. CIP Architecture and Components

This chapter describes the technical components, as mentioned in the reference architecture (D4.2), that are needed to support the execution of the presented use cases in the previous chapter and the foreseen pilot in tasks T5.6, T6.6 and T7.6. The CIP is thought/designed as technology agnostic and will not prescribe the usage of certain named software components.

This technical reference architecture for CIP (D4.4) is also a living document, that will be updated based on the results from demonstrations, Proof of Concepts (PoCs) and knowledge from other (Lighthouse) projects.

4.1 CIP Architecture

To create one technical reference implementation of CIP we comply with (as much as possible) the FIWARE and TM Forum architecture. The reason for this is that FIWARE and TM Forum aim for open innovation by providing generic enablers (open source, open standards), data models and open APIs. There are several schemas online that explain the FIWARE-architecture. The most used schema is shown below (Figure 19)⁴. It shows the different blocks, like Security, Data, Internet of Things (IoT) and the Apps/Business Framework. For further explanation of the different blocks we refer to the FIWARE documentation.

⁴ https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE_Architecture_R3

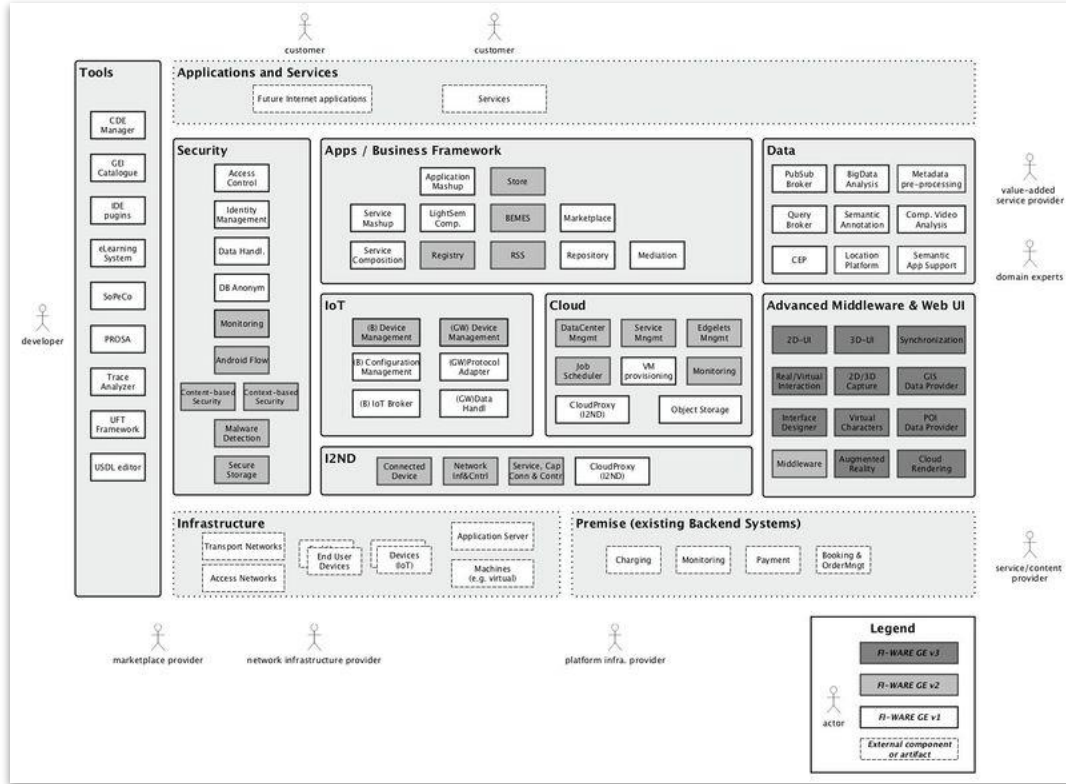


Figure 19 FIWARE Architecture overview

The CIP architecture was already developed to match the FIWARE-architecture and the most important FIWARE Generic Enablers fit within the CIP architecture, as shown in Figure 20.

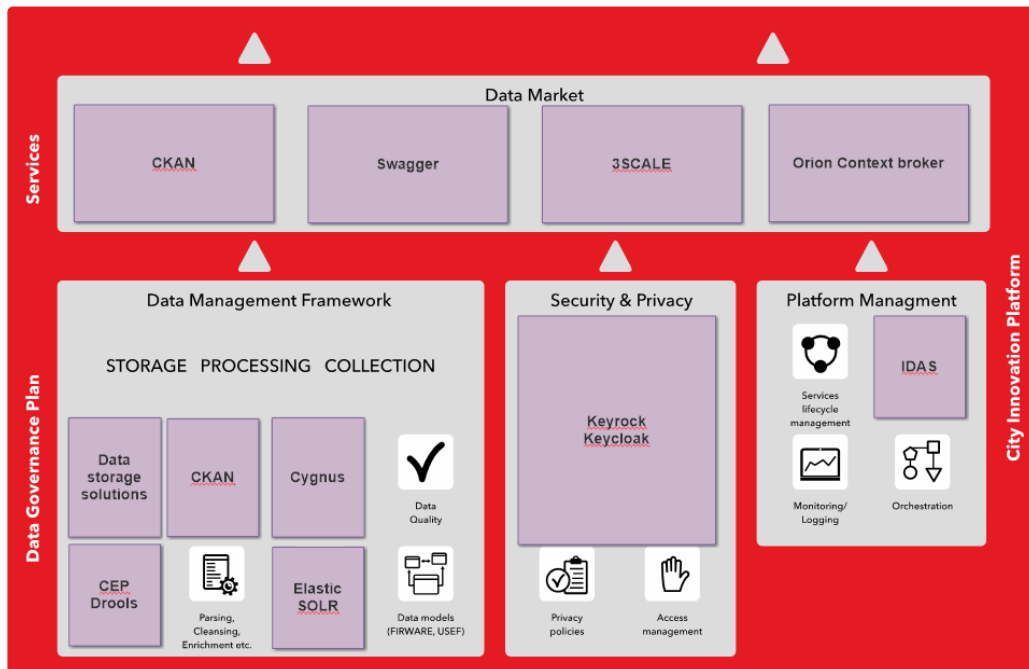


Figure 20 Mapping of FIWARE enablers and TM Forum Open APIs on CIP

The CIP technical reference architecture also uses the capability map of EIP-SSC (see D4.2) as reference for implementation of the different components. Figure 21 shows this in an overview. Each of the capabilities is connected to a technical component, as described in the Table 2.

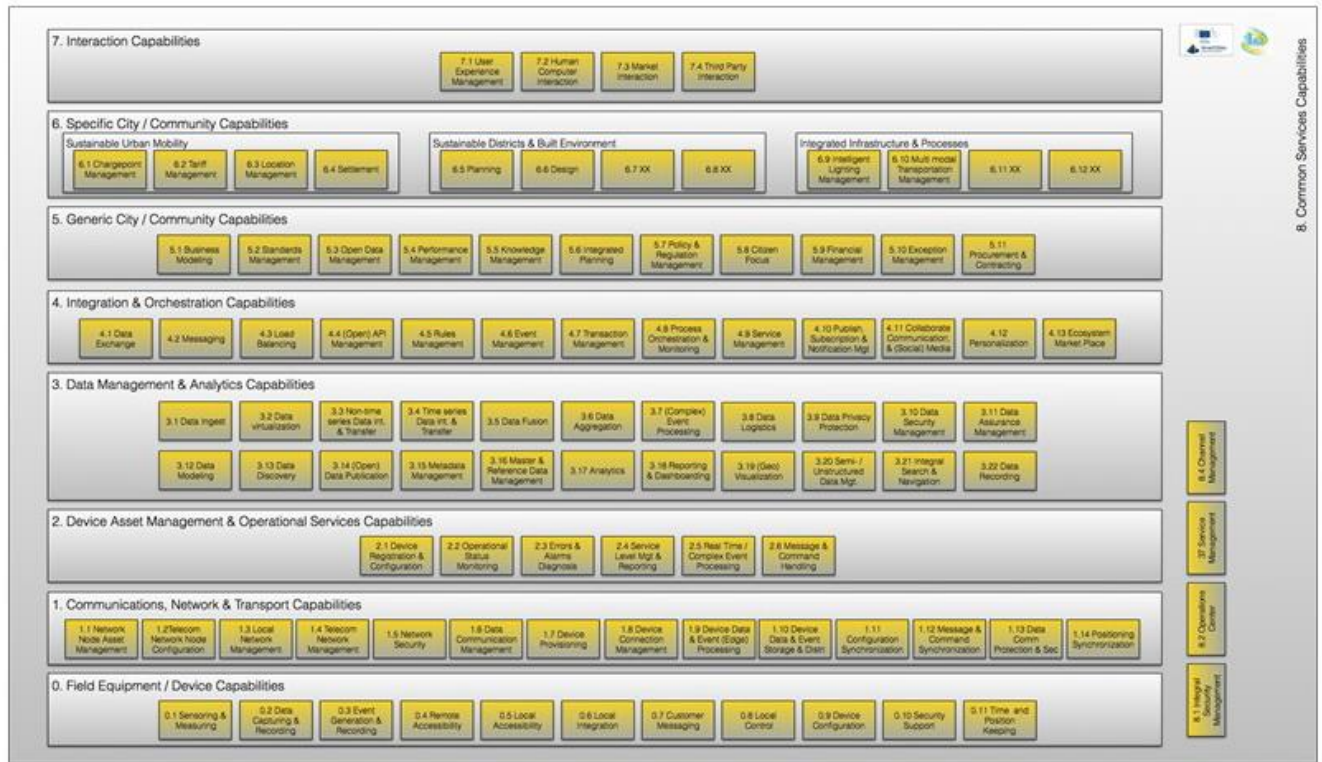


Figure 21 EIP SCC Urban Platform Capability Map with capabilities per category

No.	Capability category designation	Component
0	Field Equipment/Device capabilities	Out of scope
1	Communications, Network and Transport capabilities	Decisions on tooling will be made later
2	Device Asset Management and Operational Services capabilities	FIWARE Backend Device Management (IDAS) FIWARE Open MTC
3	Data Management and Analytics capabilities	CKAN (open data portal) Swagger (API documentation) QuantumLeap (big data) Orion Context Broker SOLR and Elastic (Search, Indexing) Database-technology (PostgreSQL, CrateDB, Hadoop)
4	Integration, Choreography and Orchestration capabilities	Drools, CEP, .. Decisions on additional tooling will be made later
5	Generic City and Community capabilities	Decisions on tooling will be made later
6	Specific City and Community capabilities	Decisions on tooling will be made later
7	Stakeholder Engagement and Collaboration capabilities	Decisions on tooling will be made later
8	Privacy and Security capabilities	Key Rock

		API Management (different solutions like 3Scale, Apinf/API-umbrella)
9	Common Services capabilities	Decisions on tooling will be made later

Table 2 Overview EIP SSC capabilities and CIP-components

All of the use cases mentioned in the previous chapter need access to different data sources. Within the CIP this is handled by the Data Management Framework (or 3: Data Management capabilities in the EIP SSC Map).

4.2 Data management framework

For the data management framework, the CIP is using existing components, if possible, from FIWARE. The Data Management Frame in CIP, in the Figure 22, is represented by the block in the bottom left corner. It consists of tools for collecting, storing and processing different kinds of data. The components that are part of the CIP Data Management Framework are described in the following paragraphs.



Figure 22 The Data Management Framework within the CIP

4.2.1 CKAN

[CKAN](#) is an open source Data Management System (DMS), with over 200 extensions available. CKAN is one of the important building blocks in the FIWARE-architecture⁵.

CKAN allows for storing public and private datasets. Public datasets are open to everyone. Private datasets are only accessible for people with the appropriate rights. With regard to the use cases in the previous chapters, CKAN is the component to store any data from political documents to events and (static) geodata like the location of charging points.

⁵ <https://fiwaretourguide.readthedocs.io/en/latest/publishing-open-data-in-fiware/fiware-extended-ckan/>

The example below (Figure 23, Figure 24) shows how CKAN stores data about charging poles (in this case from a city in the Netherlands) and how it can be applied to the Nice e-mobility use case to provide information about the locations of charging poles.

_id	GEMEE...	LOCATIE	POSTC...	PLAATS	TYPE	STATUS	X	Y	LAT	LONG
1	Diemen	Meidoor...	1112EK	Diemen	SNELLA...	OPENB...	125434....	483381....	52.337564	4.953402
2	Diemen	Martin L...	1111LS	Diemen	SNELLA...	OPENB...	126096....	484182....	52.3448	4.963046
3	Diemen	Henry D...	1111ZH	Diemen	SNELLA...	OPENB...	126528....	483814....	52.341513	4.969415
4	Diemen	Kruidenh...	1112PS	Diemen	SNELLA...	OPENB...	125720....	482927....	52.333494	4.95763
5	Diemen	Van Med...	1111CX	Diemen	GEWON...	OPENB...	126028....	483576.81	52.339349	4.962106
6	Diemen	Rietgors 11	1111VP	Diemen	SNELLA...	OPENB...	126830....	484784....	52.350244	4.973764
7	Diemen	Johan va...	1111BD	Diemen	GEWON...	OPENB...	125511....	483693....	52.340371	4.954502
8	Diemen	Keulse V...	1111RN	Diemen	GEWON...	OPENB...	126799....	483134....	52.335418	4.973454
9	Diemen	Distelvin...	1113KC	Diemen	GEWON...	OPENB...	127345....	484444....	52.347219	4.98135

Figure 23 Table view in CKAN of dataset

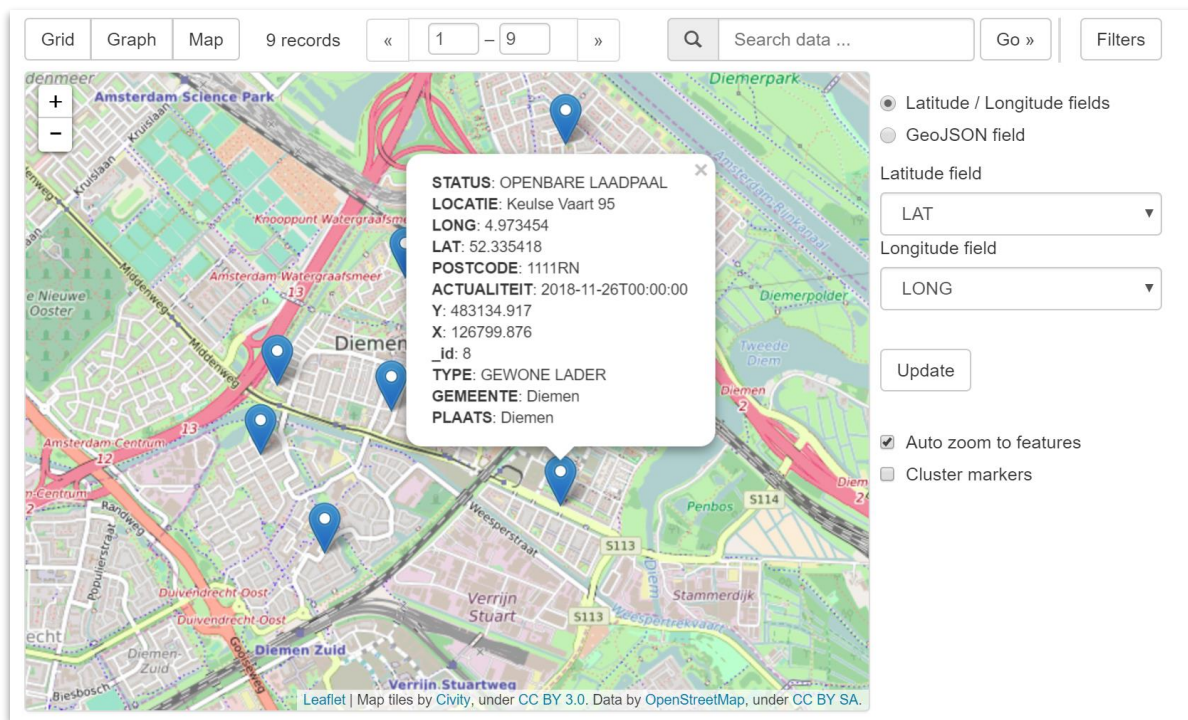


Figure 24 Map view in CKAN of dataset

CKAN⁶ offers a CKAN DataStore extension that enables storing structured data from CKAN resources. Data can be pulled out of resource files (like CSV-files) and stored in the DataStore. Once stored in the DataStore, the data is accessible through a generic (CKAN) API (see example below).

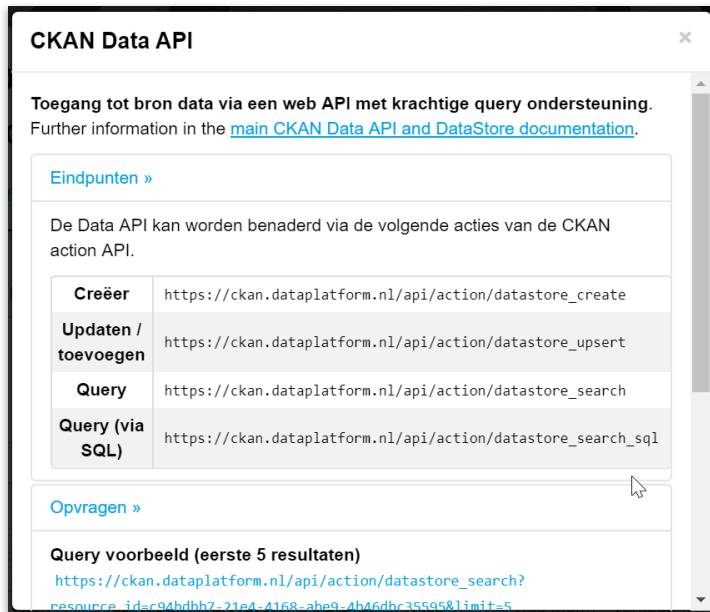


Figure 25 CKAN offers a generic API to access data in the datastore

Another important extension⁷ for CKAN is for supporting the DCAT-AP metadata standard. This extension allows CKAN to expose and consume metadata from other catalogs using RDF documents serialized using DCAT. The Data Catalog Vocabulary (DCAT) is "an RDF vocabulary designed to facilitate interoperability between data catalogs published on the Web". Data catalogs are for example the National and European open data catalogs. The metadata-information is specifically important in the catalog function of the CIP (Data Market).

CKAN allows for manual and automated uploads of data sources. The use cases demand near to real time and high quality data. Preferably automated and scheduled processes are used to upload data from their source to CKAN. This can be done with the CKAN Harvester or through ETL-scripts. The harvester⁸ allows to synchronize data with CSW-servers, Web Accessible Folder (WAF) and single metadata documents. Data from GIS databases like ESRI or Geoserver can be connected with the harvester and schedules updates to guarantee that data is in synch with the source.

⁶ <https://docs.ckan.org/en/latest/maintaining/datastore.html>

⁷ <https://github.com/ckan/ckanext-dcat>

⁸ <https://github.com/ckan/ckanext-harvest#the-harvesting-interface>



Figure 26 Example of data and filetypes that is collected with the Harvester from ESRI ArcGIS

Where the harvester is part of CKAN and synchronizes with the source data, ETL-tools allow to push data through CKAN via its API. Common ETL-tools are Safe FME, Kettle and Talend. Scripts can be written to schedule automatic upload of data to CKAN.

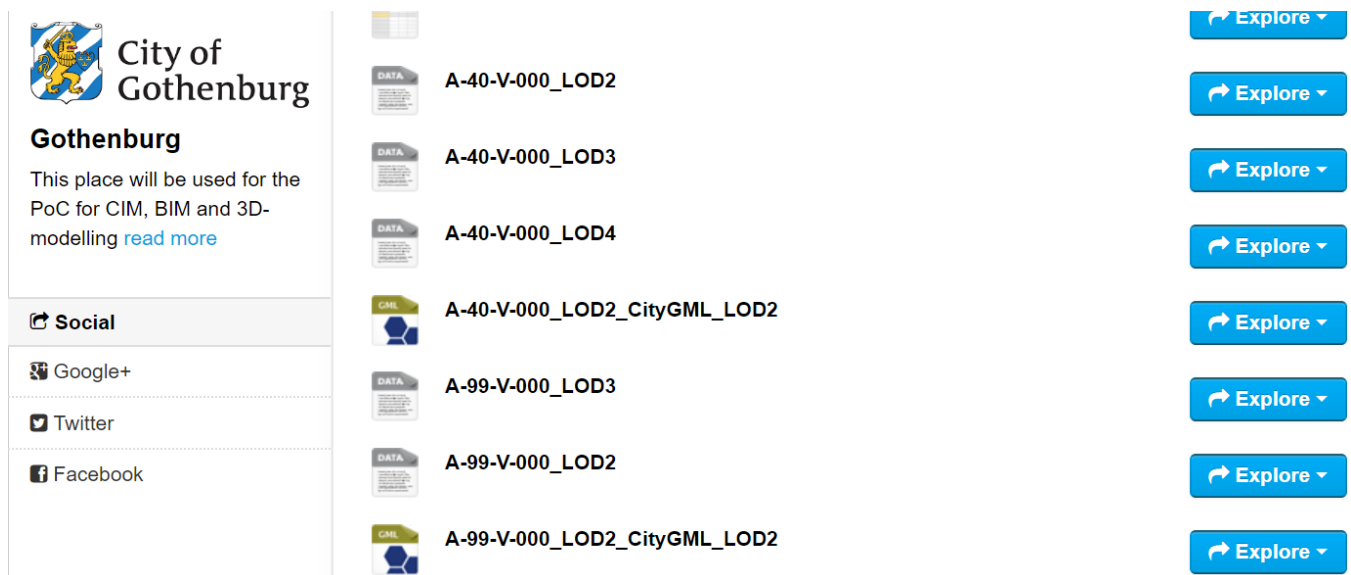


Figure 27 ETL-tools (like FME) allow to push data automatically to CKAN through its API. In this case LoD-files (Levels of Definition) for BIM/CIM-purposes.

Within the Data Management Framework, CKAN is a core component for collecting, storing and provisioning of data.

To further extend the options of the Data Management Framework other (FIWARE) components are part of the reference architecture.

4.2.2 FIWARE Orion Contextbroker, Cygnus and QuantumLeap

To support the collection, storage and provisioning of real time data, the Data Management Framework of CIP contains two FIWARE components. To understand the functions of these components, it's important to explain the underlying concepts in the FIWARE-architecture⁹. The description below is based on information from FIWARE.

Data in FIWARE refers to information that is produced, generated, collected or observed that may be relevant for processing, carrying out further analysis and knowledge extraction. A cornerstone concept in FIWARE is that data elements are not bound to a specific format representation. The structure associated to a **data element** is represented in the figure below.

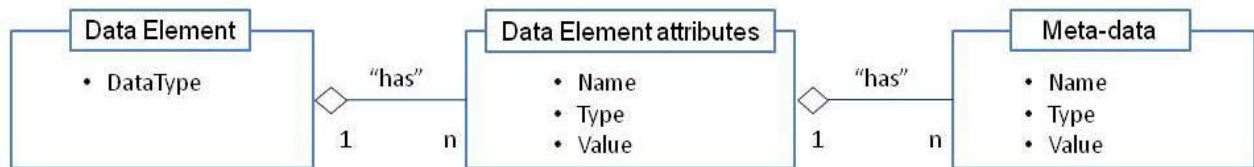


Figure 28 Data Element Structure

Data has associated a **data type** and a **value**. A **data element** refers to data whose value is defined as consisting of a sequence of one or more <name, type, value> triplets referred as **data element attributes**, where the type and value of each attribute is either mapped to a basic data type and a basic data value or mapped to the data type and value of another data element. Each data element has an associated data type in this formalism. This data type determines what concrete sequence of attributes characterizes the data element. There may be **meta-data** (also referred as semantic data) linked to attributes in a data element.

Applications may assign an identifier to data elements in order to store them in a given **Data Storage**, e.g. a Database. Such identifier will not be considered part of the structure of the data element. A given application may decide to use the value of some attribute linked to a data element as its identifier in a given Data Storage but there is no identifier associated to the representation of a data element.

Context in FIWARE is represented through **context elements**. A context element extends the concept of **data element** by associating an EntityId and EntityType to it, uniquely identifying the entity (which in turn may map to a group of entities) in the FIWARE system to which the context element information refers to. In addition, there may be some attributes as well as meta-data associated to attributes that we may define as mandatory for context elements as compared to data elements.

The structure of a context element is represented in the Figure 29 below.

⁹ https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/Data/Context_Management_Architecture_R3

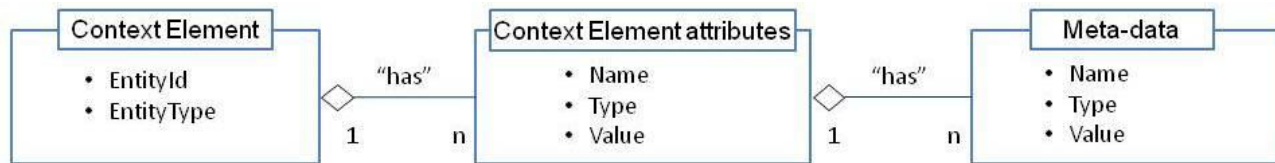


Figure 29 Context Element Structure Model

An **event** is an occurrence within a particular system or domain; it is something that has happened, or is contemplated as having happened in that domain. Events typically lead to creation of some data or context element, which enables applications or event-aware GEs (complex event processing: CEP) to handle the information described or associated to it. As an example, a sensor device may be measuring the temperature and pressure of a given boiler, sending a context element every five minutes associated to that entity (the boiler) that includes the value of these to attributes (temperature and pressure) or just the one that has changed. The creation and sending of the context element is an event, i.e., something that has occurred (the sensor device has sent new measures). As another example, a mobile handset may export attributes like "Operating System" or "Screen size". A given application may query for the value of these two attributes in order to adapt the content to be delivered to the device. As a result, the mobile handset creates and replies a context element back to the application. This response may be considered as well an event, i.e., something that has occurred (the mobile handset has replied to a request issued by an application).

Events get visible in the system by updates in data/context elements and therefore, it is common to refer to data/context elements related to events simply as "events". For convenience, we also may use the terms "data event" and "context event". A "data event" refers to an event leading to creation of a data element, while a "context event" refers to an event leading to creation of a context element.

The word **event object** is used to mean a programming entity that represents such an occurrence (event) in a computing system [EPIA]. Events are represented as event objects within computing systems to distinguish them from other types of objects and to perform operations on them, also known as **event processing**.

In FIWARE, event objects are created internally to some GEs like the Complex Event Processing GE or the Context Broker GE. These event objects are defined as a data element (or a context element) representing an event to which a number of standard event object properties (similar to a header) are associated internally. These standard event object properties support certain event processing functions. The concrete set of standard event object properties in FIWARE is still to be defined but we may anticipate that one of these properties would be the time at which the event object is detected by the GE (arrives to the GE). This will, for example, allow supporting functions that can operate on events that exceed certain age in the system. Tools will be provided enabling applications or admin users to assign values to those event object properties based on values of data element attributes (e.g., source of the event or actual capture time). An event object may wrap different characteristics of the data element (i.e., DataType) or the context element (i.e., EntityId and EntityType).

Several tools to support the mechanisms described above are available. Most significant are the Orion Context Broker and the QuantumLeap Big Data. The Orion Context Broker allows 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. Using the Orion Context Broker, you are able to create context elements and manage them through updates and queries. In addition, you can subscribe to context information so when some condition occurs (e.g. the context elements have changed) you receive a notification.¹⁰

FIWARE started with Cygnus¹¹, 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.

For time-series data FIWARE is adding another Generic Enabler: Quantum Leap. This enabler supports the storage of FIWARE NGSIv2 data into a time series database (CrateDB). The FIWARE-documentation¹² states “The typical usage scenario for QuantumLeap would be the following (notice the numbering of the events presented in the arrows):

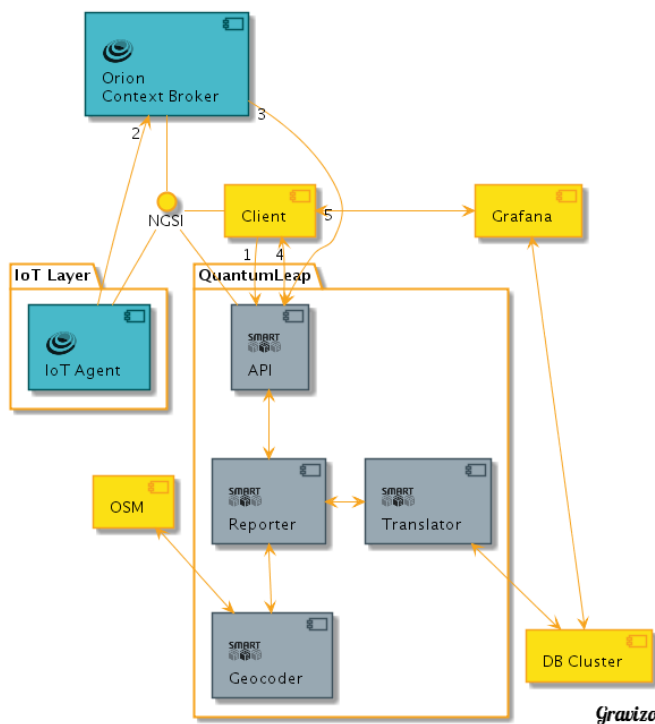


Figure 30 QuantumLeap Generic Enabler within the FIWARE-architecture

¹⁰ <https://fiware-orion.readthedocs.io/en/master/>

¹¹ <https://readthedocs.org/projects/fiware-cygnus/>

¹² <https://quantumleap.readthedocs.io/en/latest/>

The idea of QuantumLeap is pretty straightforward. By leveraging on the NGSIV2 notifications mechanism, clients first create an Orion subscription (1) to notify QuantumLeap of the changes in the entities they care about. This can be done either through QuantumLeap's API or directly talking to Orion. Then, new values arrive in Orion Context Broker (2) for the entities of interest, for example from a whole IoT layer governed by 1 or more IoT Agents pushing data in NGSI format. Consequently, notifications will arrive to QuantumLeap's API /v2/notify endpoint (3).”

The data and context management components of the FIWARE architecture enable the CIP to:

- Subscribe for being notified about and query for context information coming from different sources.
- Model changes in context as events that can be processed to detect complex situations that will lead to generation of actions or the generation of new context information.
- Processing large amounts of context information in an aggregated way.
- Process data streams coming from different sources in order to generate new data streams as well as context information that can be further exploited.
- Process metadata that may be linked to context information, using standard semantic support technologies.
- Manage some context information, such as location information, presence, user or terminal profile, etc., in a standard way.

Figure 31 shows the main components (Generic Enablers) that comprise the third release of FIWARE Data/Context chapter architecture.

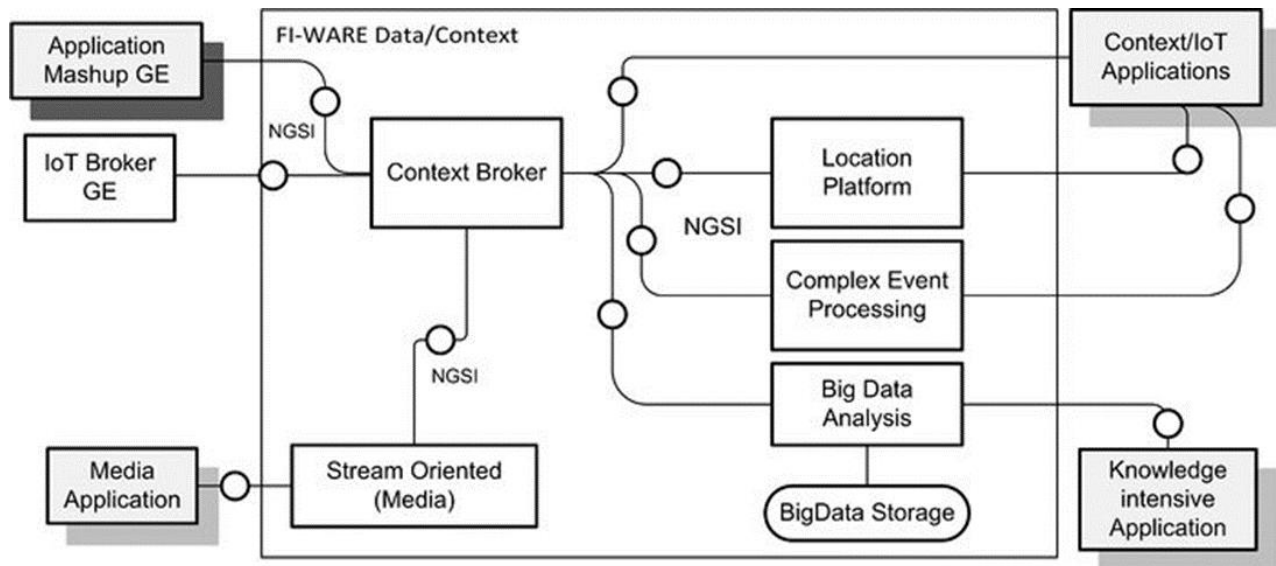


Figure 31 FIWARE Data/Context architecture

Also tooling for complex event processing is needed. At this moment it is uncertain whether the FIWARE CEP-component is mature enough or most suitable for the CIP. Open alternatives, like Drools¹³ (Red Hat Drools Fusion¹⁴) might be an alternative. During demonstrations or Proof of Concepts the tooling for the CIP will be evaluated and enhanced where needed.

The reference implementation of the Data Management Framework of the CIP (at least) consists of CKAN, Orion and QuantumLeap. All components use the NGSI APIs of FIWARE and the Orion Context Broker is the core of the integration.

The NGSI API is the glue between all the technical components. The FIWARE-NGSI v2¹⁵ is intended to manage the entire lifecycle of context information, including updates, queries, registrations, and subscriptions.

Datamodels

In the D4.2 CIP reference architecture the FIWARE data models are described. To facilitate the use cases in the previous chapter the CIP needs to support 3D models (BIM and CIM). To create a digital representation of the city at least the OGC CityGML standard needs to be supported.

CityGML

CityGML is an open, standardized data model and exchange format of the Open Geospatial Consortium (OGC) to store digital 3D models of cities and landscapes. It has been realized as an open data model and is implemented as a GML application schema. Because CityGML is based on GML, it can be used with the whole family of GML-compatible web services for data access, processing, and cataloging, such as Web Feature Services, Web Processing Services, and Catalog Services.

CityGML defines ways to describe most of the common 3D features and objects found in cities (such as buildings, roads, rivers, bridges, vegetation and city furniture) and the relationships between them. It also defines different standard Levels Of Detail (LODs) for the 3D objects, which allows representation of objects for different applications and purposes.

CityGML mainly describes the geometry, attributes and semantics of different kinds of 3D city objects. For visualization purposes, these can be supplemented with textures and/or colors in order to give a better impression of their appearance. Specific relationships between different objects can also be stored using CityGML, e.g. that a building is decomposed into three parts, or that a building has both a carport and a balcony.

The types of objects stored in CityGML are grouped into different modules. These are:

- Appearance: textures and materials for other types
- Bridge: bridge-related structures, possibly split into parts

¹³ <https://www.drools.org/>

¹⁴ <https://docs.jboss.org/drools/release/6.2.0.CR3/drools-docs/html/DroolsComplexEventProcessingChapter.html>

¹⁵ <https://orioncontextbroker.docs.apiary.io/#introduction/specification/terminology>

- Building: the exterior and possibly the interior of buildings with individual surfaces that represent doors, windows, etc.
- CityFurniture: benches, traffic lights, signs, etc.
- CityObjectGroup: groups of objects of other types
- Generics: other types that are not explicitly covered
- LandUse: areas that reflect different land uses, such as urban, agricultural, etc.
- Relief: the shape of the terrain
- Transportation: roads, railways and squares
- Tunnel: tunnels, possibly split into parts
- Vegetation: areas with vegetation or individual trees
- WaterBody: lakes, rivers, canals, etc.

It is possible to extend the list above with new classes and attributes by defining so-called Application Domain Extensions¹⁶ (ADEs). There are already several ADEs, for example for energy estimation in an urban context, and for estimating solar potential.

In its most common implementation, which is the one generally used to disseminate and exchange data, CityGML datasets consist of a set of plain text files (XML files) and possibly some accompanying image files that are used as textures. Each text file can represent a part of the dataset, such as a specific region, a specific type of object (such as a set of roads), or a predefined LOD. The structure of a CityGML file is a hierarchy that ultimately reaches down to individual objects and their attributes. These objects have a geometry that is described using GML.

An example of software that has implemented CityGML is 3D City DB¹⁷, which stores CityGML in a database.

¹⁶ <https://www.citygml.org/ade/>

¹⁷ <http://www.3dcitydb.org/>

4.3 Security and Privacy

Security and privacy are generic functions that apply to all capabilities and all layers within the CIP (Figure 32).

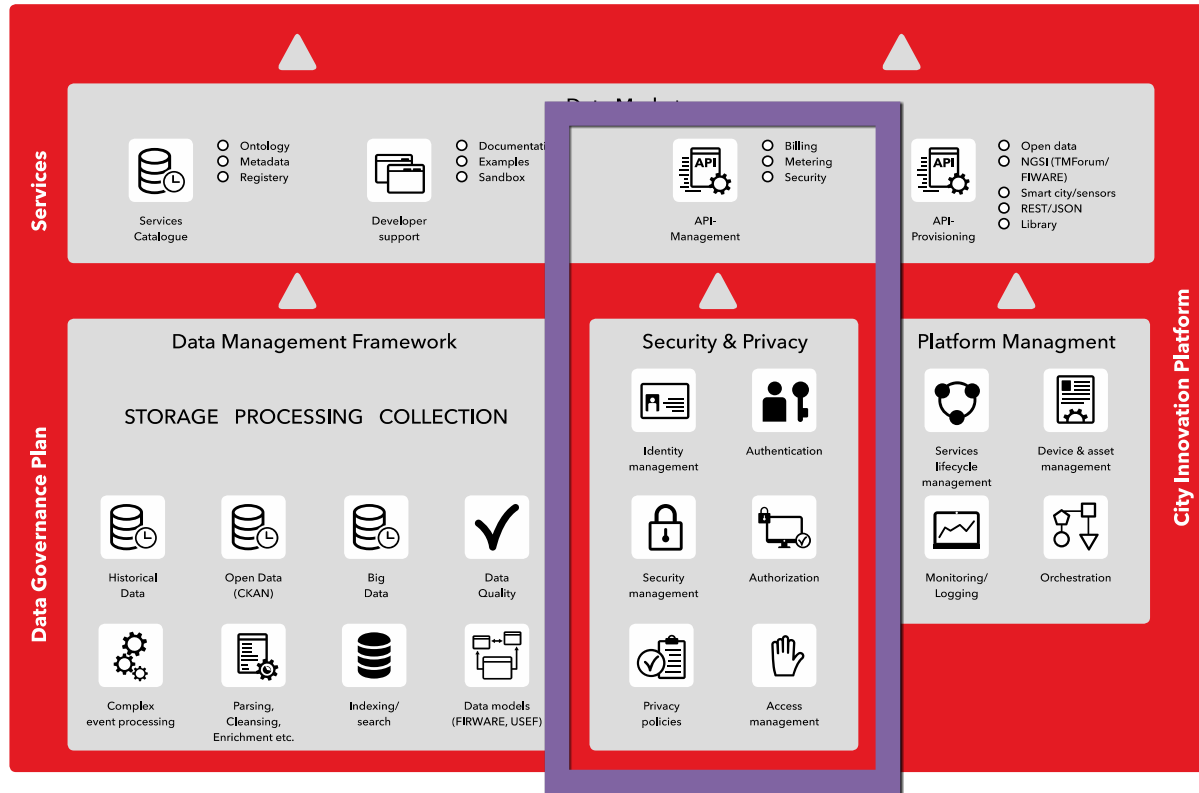


Figure 32 Security and Privacy components in CIP

For the technical architecture, the following general requirements apply:

- CIP should provide tool for anonymization and pseudomization of data (see annex ARX)
- CIP should offer tools to comply with GDPR, like opt-in and opt-out, insight in personal data for citizens and the right to be forgotten
- CIP should offer tools for communication and storage encryption

Two security components within the FIWARE architecture are the **PEP Proxy Wilma GE**¹⁸ and the **Authorization PDP GE**¹⁹.

The PEP Proxy GE (in combination with Identity Management and Authorization PDP GE's), adds authentication and authorization security to backend applications, so that only users with specific permissions and policies can access components, resources and services. PEP Proxy provides a security layer for adding authentication and authorization filters to FIWARE GE's and any backend service. It is the

¹⁸ <https://catalogue-server.fiware.org/enablers/pep-proxy-wilma>

¹⁹ <https://catalogue-server.fiware.org/enablers/authorization-pdp-authzforce>

PEP (Police Enforcement Point) of the FIWARE Security Chapter. So together with the Keyrock IdM GE and the Authorization PDP GE it provides security to FIWARE backends.

Wilma is the reference implementation of the PEP Proxy Generic Enabler and it completely integrates with the FIWARE ecosystem. It works with OAuth2 and XACML protocols, the standards for authentication and authorization chosen in FIWARE. Furthermore, Wilma is the component that all GE's are including on top of their REST APIs, making it applicable for many different scenarios. The PEP Proxy GE is a backend component, without frontend interface.

The Authorization PDP GE provides an API to get authorization decisions based on authorization policies, and authorization requests from PEPs. The API follows the REST architecture style and complies with XACML v3.0. (eXtensible Access Control Markup Language), an OASIS standard for authorization policy format and evaluation logic, as well as for the authorization decision request/response format. The PDP (Policy Decision Point) and the PEP (Policy Enforcement Point) terms are defined in the XACML standard. The Authorization PDP GE plays the role of a PDP.

The Authorization PDP helps to externalize the authorization logic and take advantage of flexible and standard-compliant Attribute-Based Access Control features. Combined with the Identity Management GE (Keyrock) and the PEP proxy GE, this gives a comprehensive access control solution for applications.

The basic use cases for the components mentioned above are scenarios in which users of a front-end application will access resources in a backend application. And where only authorized users can access those resources. This scenario applies also to most of the use cases in the previous chapter and is in accordance with the CIP-architecture.

The three FIWARE security GE's are designed to perform three levels of security for backend REST APIs²⁰.

1. **Level 1: Authentication**

When the frontend application sends a REST request to the backend application it has to include the OAuth2 token (access token) of the user. So, the first step is to create a user and an application with the appropriate account. When PEP Proxy receives the request, it extracts the access_token from the HTTP header (X-Auth-Token) and sends a request to the FIWARE Identity Management GE (Keyrock)

²⁰ https://fiware-pep-proxy.readthedocs.io/en/latest/user_guide/

in order to validate it (Figure 33).

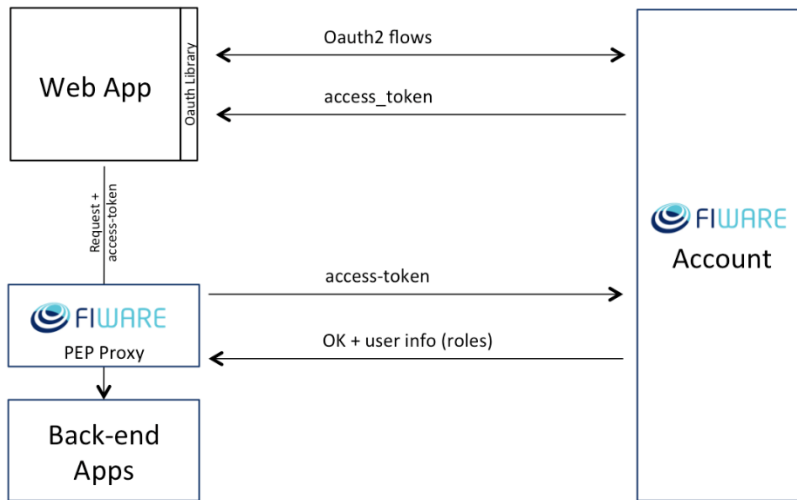


Figure 33 Level 1 authentication with PEP Proxy

2. Level 2 Basic Authorization

The first step is (again) to create a user and an application with the appropriate account. To allow basic authorization the roles and permissions for that user in that application also need to be configured. PEP Proxy checks if the `access_token` included in the request corresponds to an authenticated user. If the validation is successful, the response includes the user information for that application. In this information is included the list of roles that the user has in the application. PEP Proxy checks with Authorization PDP if the user has the permissions to access the resource of the request (Figure 34).

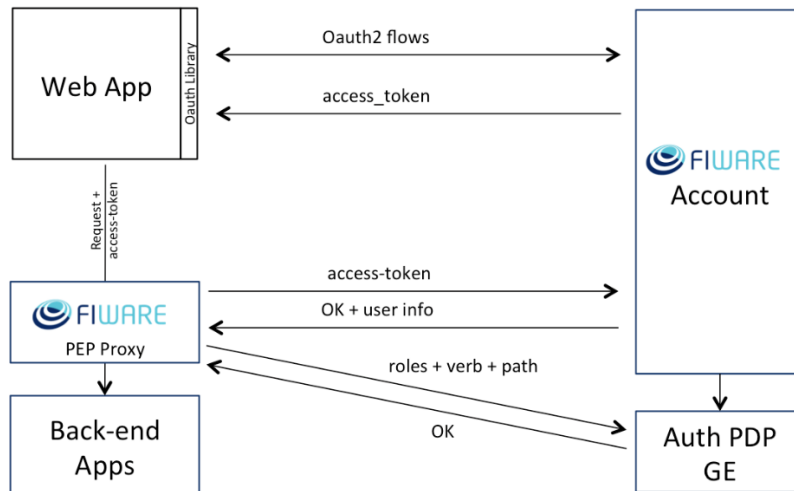


Figure 34 Level 2 Basic Authorization with PEP Proxy

3. Level 3: Advanced Authorization

The first step is (again) to create a user and an application. After that the roles and XACML policies

for that user in that application are configured. As this case is thought to check advanced parameters of the request such as the body or custom headers, it depends on the specific use case (Figure 35).

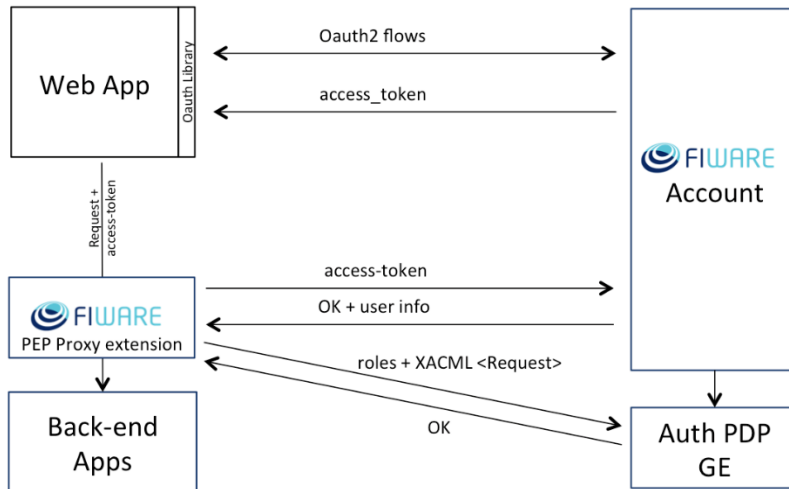


Figure 35 Level 3 Advanced Authorization with PEP Proxy

4.4 Data Market

The Data Market is an online store for different data products and types (open, commercial datasets and information products) from different sources/organizations and aimed at different users. Data types can be mixed and structured in a variety of ways and can be made available as download, API or other arrangements.

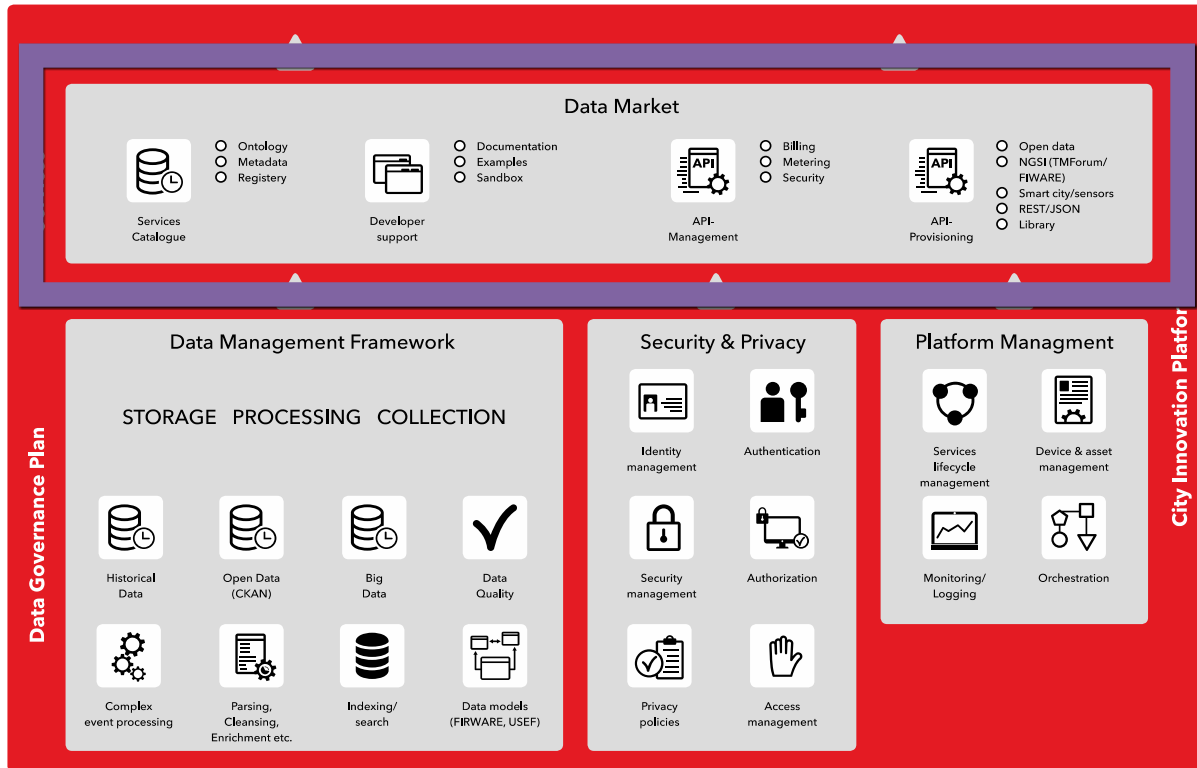


Figure 36 Data Market component in CIP

The Data Market connects to components already described in the Data Management Framework. Additionally, the CIP uses the FIWARE Business-API-Ecosystem²¹ that was developed in collaboration with TM Forum.^{22 23} The FIWARE Business API Ecosystem is not a single software repository, but it is composed of different projects which work coordinately to provide the complete functionality. The Business API Ecosystem Generic Enabler (GE) is a joint component made up by integrating the FIWARE Business Framework with a set of standard open APIs (and its reference implementations) provided by TM Forum.

²¹

<https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Apps.BusinessAPIEcosystem>

²² <https://github.com/FIWARE-TMForum/Business-API-Ecosystem>

²³ <https://catalogue-server.fiware.org/enablers/business-api-ecosystem-biz-ecosystem-ri>

Concretely, the Business API Ecosystem (CIP Data Market) consists of reference implementations of the TM Forum APIs for Catalog Management, Product Ordering Management, Product Inventory Management, Party Management, Customer Management, Billing Management and Usage Management. Furthermore, it consists of a Rating, Charging, and Billing backend²⁴, Revenue Settlement and Sharing System²⁵, Authentication, API Orchestrator, and Web portal²⁶.

The Business API Ecosystem GE allows the monetization of different kind of assets (both digital and physical) during the whole service life cycle, from offering creation to its charging, accounting and revenue settlement and sharing. In this way, the Business API Ecosystem provides sellers the means for managing, publishing, and generating revenue of their products, apps, data, and services. Using the Business API Ecosystem within CIP offers the following key features:

- Support for the management of catalogs, products, and offering
- Support for rich pricing models, including recurring payments, pay-per-use, etc.
- Support for accounting callbacks
- Support for billing and charging
- Integrated support for PayPal, including customer charges and seller payments
- Support for revenue sharing, including models with multiple stakeholders involved

In the next paragraphs components of the CIP Data Market are described in more detail.

4.4.1 IdM

Identity Management (IdM) is a key component of the CIP. The FIWARE-GE for IdM is Keyrock²⁷. This GE covers a number of aspects involving users' access to networks, services and applications, including secure and private authentication from users to devices, networks and services, authorization and trust management, user profile management, privacy-preserving disposition of personal data, Single Sign-On (SSO) to service domains and Identity Federation towards applications.

IdM offers tools for administrators to support the handling of user life-cycle functions, like enforcement of policies and procedures for user registration, user profile management and the modification of user accounts. For end users, the IdM provides a convenient solution for registering with applications since it gives them the means to re-use attributes like address, email or others, thus allowing an easy and convenient management of profile information (including Single Sign On).

The KeyRock Identity Management GE complies with existing standards for user authentication and it provides access information to services acting as a Single Sign-On platform.

²⁴ <https://github.com/FIWARE-TMForum/business-ecosystem-charging-backend>

²⁵ <https://github.com/FIWARE-TMForum/business-ecosystem-rss>

²⁶ <https://github.com/FIWARE-TMForum/business-ecosystem-logic-proxy>

²⁷ https://fiware-idm.readthedocs.io/en/master/user_guide.html

4.4.2 Open API's and API-management

As described in the reference architecture (D4.2) two important component of the Data Market are Open API's and API-management.

API management is the practice an organization implements to manage that the APIs they expose, either internally or externally are consumable, secure, and available to consumers in conditions agreed upon in the APIs terms of use. API-management requires tools and solutions to provide and manage those API's. The functionality considered to be a part of API management is described in D4.2. Main functionalities are an API registry and gateway, publishing tools, a developer portal and reporting and analytics tools. To support the CIP Data Market all these functionalities should be available.

Within the FIWARE-architecture API-management is still under development²⁸. It will be built upon API-umbrella²⁹ and APInf³⁰ and made available in an upcoming release of FIWARE. There are several other mature open source API-management solutions that can be used within the CIP. During the demonstrations and PoC several solutions will be evaluated. Utrecht for example now uses APIman³¹ as solution for its Dataplatform and will migrate to 3Scale³², a fully open source solution by Red Hat. Nice will evaluate the FIWARE components API-Umbrella and APInf.

4.4.3 Open API's

The API Ecosystem within the CIP will be based upon the Open APIs from TM Forum³³. Concretely, it includes the catalog management, ordering management, inventory management, usage management, billing, customer, and party APIs³⁴ (Figure 37). FIWARE and TM Forum jointly developed the [Business API Ecosystem Open Specification](#). The IRIS LH Cities have joined the Front Runner program from TM Forum and FIWARE which will further strengthen the development of common Smart City Data

²⁸ <https://www.fiware.org/2017/07/28/apinf-contributes-top-class-api-management-technologies-to-fiware-platform/>

²⁹ <https://apiumbrella.io/>

³⁰ <https://www.apinf.com/>

³¹ <http://www.apiman.io/latest/>

³² <https://www.redhat.com/en/technologies/jboss-middleware/3scale/features>

³³ https://projects.tmforum.org/wiki/display/API/Open+API+Table?_ga=2.91744794.1082033315.1539175670-1726701653.1525631387

³⁴ <http://business-api-ecosystem.readthedocs.io/en/latest/>

Models and Open API's.³⁵




TM Forum Open APIs	Document Number	Swagger (Apache 2.0 or RAND)	API Specification (RAND)	Conformance Profile (RAND)	CTK	Reference Implementation Code	Postman Collection	Release	Lifecycle Status
events. Notice that for the management of customers there is a specific Customer Management API. Party Role management API manages the following data resources: PartyRole									
Payment Management API The Payments API provides the standardized client interface to Payment Systems for notifying about performed payments or refunds. Examples of Payment API originators (clients) include Web servers, mobile app servers, Contact center dashboards or retail store systems.	TMF676				R18.0 Refresh in progress	R18.0 Refresh in progress	R18.0 Refresh in progress	18.0.1	Updated
Payment Methods API This API supports the frequently-used payment methods for the customer to choose and pay the usage, including voucher card, coupon, and money transfer.	TMF670				Not yet available	Not yet available		17.5.0	

Figure 37 TM Forum Open API's offers specifications for 50+ Open API's

The growing exposure of data through APIs for use by apps and the developers that create them, requires policies to ensure security, support for developer (documentation, sandbox, etc.), analytics and monetization. Security is essential when organizations expose their backend systems via APIs. It is about authenticating and authorizing access to APIs, but also about policies to manage quotas, block attacks and ensure that sensitive data is not accidentally or intentionally leaked. Another reason API management is important, is that it provides logs and audit trails to support both offline analysis and real-time troubleshooting. Developer experience is key to adoption and success of APIs. They are useless if nobody uses them. A (self-service) developer portal (as part of the Data Market) with all APIs in one place, allows for easy discovery and testing.

API-management has usually a hierarchical data model that consists of a number of primary elements (Figure 38). The description below is based on documentation from APIman³⁶.

³⁵ <https://www.fiware.org/news/fiware-foundation-and-tm-forum-launch-front-runner-smart-cities-program/>

³⁶ <http://www.apiman.io/latest/crash-course.html>

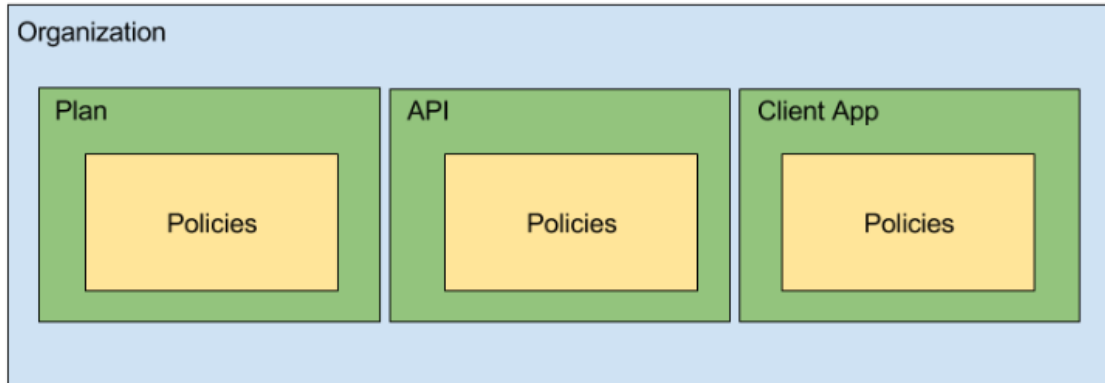


Figure 38 Common elements within an API-management solution.

- Policies** - Policies are at the lowest level of the data model, but the most important concept because they represent the unit of work done at runtime (when the API Gateway applies the policies to all API requests). Everything defined in an API Management solution is there to enable that solution to apply policies to requests made to APIs. When a request to an API is made, a chain of policies to be applied to that request is created. Policy chains define a specific sequence order in which the policies are defined are applied to API requests. You can think of a policy as a rule or set of rules that are enforced by the API Gateway. There are multiple types of policies. Some policies allow or block access to APIs based on the IP address of the client application, while others allow or restrict access to specific resources provided by an API, while still others enable you to control or “throttle” the rate at which requests are made to an API.
- Plans** - a Plan is a set of policies that together define the level of service that the API management solution provides for an API. Plans enable users to define multiple different levels of service for their APIs. It is common to define different plans for the same API, where the differences depend on configuration options. For example, an organization may offer both a “gold” and “silver” plan for the same API. The gold plan may be more expensive than the silver plan, but it may offer a higher level of API requests in a given (and configurable) time period.
- APIs** - These represent real back-end APIs that are being “managed” by the API Management solution. An API can either be Public (available to any invoker at a static endpoint) or Private (only invocable by Client Apps that are modeled within the API Management solution). This is the primary entity that an API Provider is responsible for editing within the API Manager.
- Client Apps** - A Client App represents a piece of software that needs to consume managed APIs that are offered through the API management solution. Each Client App can consume multiple APIs by creating a contract to the API through one of its plans. This is the primary entity that an API consumer is responsible for editing within the API Manager.

- **Organizations** - The Organization is at the top level of the data model. An organization contains and manages all elements used by a company, university, group inside a company, etc. for API management. All Plans, APIs and Client Apps are defined in an Organization. In this way, an Organization acts as a container of other elements.

When further developing the CIP Data Market, a strategy and implementation roadmap for API-management needs to be developed.

4.5 Platform Management

One of the components in the CIP is “Platform Management” and it consists of different functions. The idea behind “Platform Management” is that an Urban Platform need facilities/tools to manage connections between suppliers of data and consumers of data. For example, in the Nice use case there is a dependency between the app builder (free spot for EV-charging) and the supplier of data. When new charging poles are added, sensors are replaced or of the availability changes, the app builder needs to get this information to keep the app working. The way to solve these kinds of questions need to be worked out in the use cases, because there are different solutions possible. There is also a relationship with the TM Forum Open API’s on service level management, usage, billing and so on.



Figure 39 Platform Management component within CIP

Regarding the Platform Management component within CIP, the important question is: “who does what”. In a landscape with multiple stakeholders, multiple systems, platforms and sensors, we have to define what kind of logging, monitoring, orchestration or device management functions are needed within the CIP. The answer to this question can differ per use case.

With reference to the use cases (as for e.g. the Nice use case) the currently required functionality within the Platform Management component should support device management for connected things, sensors/actuators and other devices like sensor gateways. Platform Management consists of a toolset for IoT-network operators to maintain, manage and guarantee service levels for data and sensors.

From a capability perspective Platform Management combines functions from the EIP SSC Capability Map described in layer 0 (Field Equipment/Device capabilities), layer 1 (Communications, Network and Transport capabilities) and layer 2 (Device Asset Management and Operational Services capabilities).

Subsequently, Platform Management should provide the following functionalities: remote accessibility, device configuration and security support, device provisioning and connection management, configuration synchronization, device registration and configuration, operational status monitoring, error and alarm diagnostics, device service level management and reporting, and message and command handling.

Platform management is hence a set of tools for an IoT network operator to manage devices connected to the CIP during their complete life cycle (from installation to decommissioning). This set of tools is also in charge to exchange data between devices and the FIWARE Orion context broker.

The CIP aims to be compatible with the OneM2M³⁷ specifications and architecture³⁸ from ETSI, OMA and 3GPP, which can be considered as the standard to follow for platform management features. The FIWARE Context Information model provides an interoperability framework that allows applications and services to consume information and/or trigger actions from IoT systems such as OneM2M.

For Platform Management the following features are relevant:

- Device connectivity and communication
- Device lifecycle management
- Data transformation
- Provisioning of devices

4.5.1 Device connectivity and communication

The device connectivity functionality feeds the FIWARE context broker with data coming from sensors/devices through the NGSI v2 protocol. It might be able to support standardized protocols and standard transports but should also be able to integrate easily any protocol.

The CIP must be able to communicate with devices/sensors through main standard transport and device management protocols and gateway software:

- HTTP(s), MQTT, COAP
- LPWAN, such as LoRa, Sigfox, LTE-M, NB-IOT
- Low range networks such as Bluetooth, BLE, ZigBEE, Z-Wave and BacNet
- Industrial networks such as OPC-UA or Modbus
- [LWM2M](#) (lightweight M2M), for constrained devices, [OMA-DM](#) or/and [BBF TR-069](#) for less constrained devices, such as gateway, or edge server, as soon as device supports it natively or through an additional client library.

³⁷ <http://www.onem2m.org/> and <https://www.etsi.org/about/what-we-do/global-collaboration/onem2m>

³⁸ <http://www.onem2m.org/application-developer-guide/architecture>

4.5.2 Device Lifecycle management

Functionalities that belong to this topic are:

- Provisioning of devices: register device or device group into the platform, accept incoming data from it or send commands to it.
- Device configuration: ability to write configuration parameters/attributes on a connected device, update the firmware of a device and set the geographic location.
- Storage of device dependent informative attributes, such as firmware version, manufacturer, or some pictures allowing easy recognition of device.
- Device security (authentication, encryption) to manage device credentials and parameters, such as API key, authorization tokens and encryption keys.
- Device monitoring: working status, communication, battery status.
- Device depreciation (the opposite of provisioning) to unregister or disable a device to prevent data acquisition

4.5.3 Data transformation

Before sending data from devices to the FIWARE context broker, as well before sending commands to devices, it might be necessary to convert, calibrate or validate those data.

4.5.4 Provisioning of devices

To be able to register a device or a group of devices (service) in the platform, to enable the acquisition of data coming from those devices and give the possibility to send commands to those devices.

4.5.5 Platform management solutions

There are no clear open solutions to implement the Platform Management functionality in the CIP. The FIWARE catalog proposes two different Generic Enablers to manage devices and communicate with them: IDAS and OpenMTC. Both enablers expose a RESTful API to interact with but don't provide any web user interface.

Fiware Backend Device Management (IDAS)

The FIWARE Backend Device Management³⁹ GE aims to be the central enabler at the IoT backend for most common scenarios. The architecture of this component is shown in Figure 40.

39

<https://forge.fiware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.ArchitectureDescription.IoT.Backend.DeviceManagement>

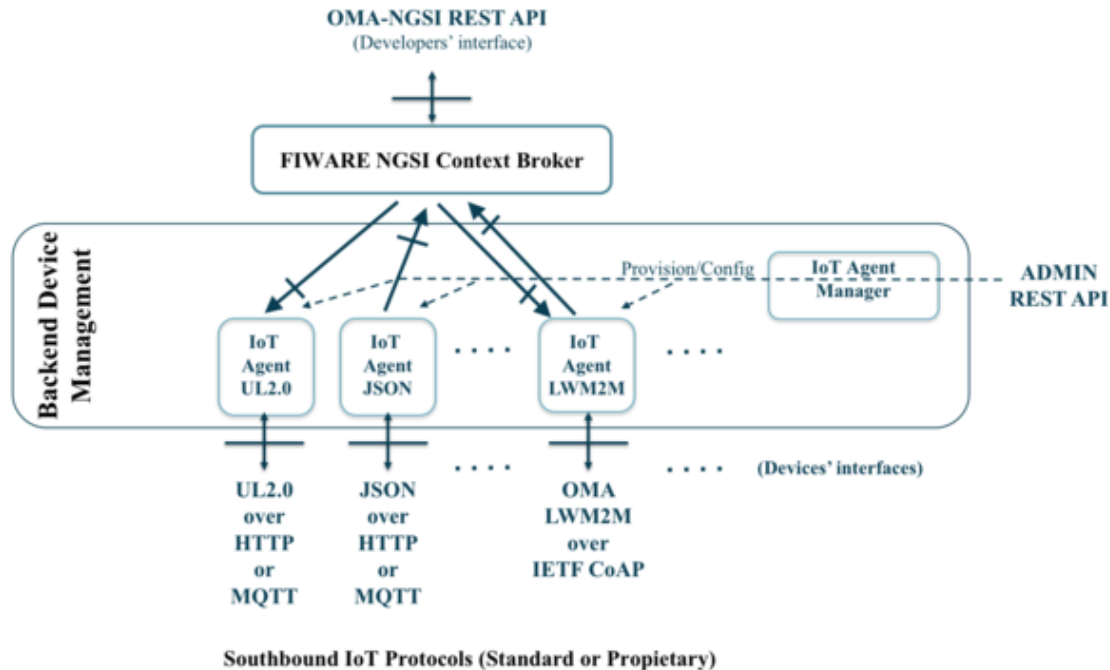


Figure 40 FIWARE Backend Device Management architecture (IDAS)

Basically, the FIWARE Backend Device Management provides the following features:

- **Connects physical devices to a FIWARE platform.**
Based on different standard or proprietary communication protocols and APIs.
- **Manages IoT-related NGSI Context Entities.**
It handles the connection (north-bound) to a FIWARE NGSI Broker to create one Context Entity per physical connected device. For most cases, FIWARE Application developers will only interact with those NGSI Entities. On the other hand, IoT integrators will also interact with the devices' ADMIN API and the devices' protocols APIs.
- **IoT Edge Management.**
This functionality is provided within the "IoT Manager" module and provides IoT Integrators with the ability of transforming device specific Data Models into the Data Models defined at the NGSI level.

The IDAS⁴⁰ component is an implementation of the Backend Device Management GE. It allows to connect objects to gather data or interact with them, in typical IoT use case scenarios. Depending on the southbound protocol it is recommended to use one of the existing IoT Agents that are part of IDAS. The current list of supported IoT Agents (all implemented with node.js) is:

⁴⁰ <https://catalogue-server.fiware.org/enablers/backend-device-management-idas>

- **IoTAgent-LoRaWAN⁴¹**
This IoT Agent allows to connect different LoRaWAN network servers, to forward data produced by LoRaWAN nodes into FIWARE (through an Orion Context Broker). Currently supported by The Things Network and LoRaServer.io network servers.
- **IoTAgent-JSON⁴²**
This IoT Agent is designed to be a bridge between HTTP/MQTT+JSON based protocol speaking devices and the FIWARE NGSI standard used in the Orion Context Broker.
- **IoTAgent-LWM2M⁴³**
This IoT Agent is designed to be a bridge between Lightweight M2M protocol speaking devices and the FIWARE NGSI standard used in the Orion Context Broker
- **IoTAgent-UL⁴⁴**
This IoT Agent is designed to be a bridge between an UltraLight2.0 protocol speaking devices and the FIWARE NGSI standard used in the Orion Context Broker
- **IoTAgent-node-lib⁴⁵**
This is a library to create new agents. This core library is used in any other implemented IOT agent and allows to develop new agents for specific southbound protocols/standards/messages.

Additionally, there exist others IoT agents, based on the above library, for other known protocols/transport:

- **IoTAgent-OPC-UA⁴⁶**
An IOT agent that can exchange data with an [OPC-UA](#) server (in an industrial context, to communicate with PLCs)
- **IoTAgent-sigfox⁴⁷**
This IoT Agent is designed to be a bridge between the Sigfox callbacks protocol and the OMA NGSI protocol used by the Orion Context Broker.

The RESTful API in the FIWARE Backend Device Management is implemented at the library level and therefore common and consistent across all the IoT agents. It is possible to manage devices and groups of devices through the API (services). This API posts JSON-content which defines the mapping of device attributes to context entity attributes in the Orion Context Broker. It is possible to transform (convert, calibrate and validate) measurement data using “[expression language](#)” in the attribute definition, before sending it to the Orion Context Broker. There’s no way to validate the value of the measurement within a range of minimum and maximum value, to avoid taking in count bad measurement values.

⁴¹ <https://github.com/Fiware/iot.IoTagent-LoraWAN>

⁴² <https://github.com/Fiware/iot.IoTagent-JSON>

⁴³ <https://github.com/Fiware/iot.IoTagent-LWM2M>

⁴⁴ <https://github.com/Fiware/iot.IoTagent-UL>

⁴⁵ <https://github.com/Fiware/iot.IoTagent-node-lib>

⁴⁶ <https://github.com/BEinCPPS/idas-opcua-agent>

⁴⁷ <https://github.com/telefonicaid/sigfox-iotagent>

Fiware Open MTC

The other FIWARE GE that can be useful to fulfill the requirements for the CIP Platform Management component is OpenMTC⁴⁸ (Python based) OpenMTC is an IoT middleware consisting of gateway and backend functionalities. The gateway hosts protocol adapters that interconnects various devices and decodes device specific information. Additionally, local applications can preprocess data before providing them to higher levels. The backend aggregates information of multiple gateways and provides connection information to the devices. OpenMTC unifies information from heterogeneous data sources by adding semantics and storing the data in a tree-based structure. Through the usage of meta information data can be easily discovered.

OpenMTC complies with [ONEM2M](#)⁴⁹ ETSI specifications but is still in incubation in the FIWARE catalog. The OpenMTC software is developed by Fraunhofer institute⁵⁰, a member of the ONEM2M consortium During the implementation of demonstration projects in the LH Cities, further investigation of the maturity and opportunities of OpenMTC within CIP needs to be evaluated. An advantage might be that via oneM2M-based REST APIs and event-based notifications M2M/IoT application developers can easily and dynamically access data from various sensors, without struggling with underlying sensor / actor technology specifics.

According to the documentation, the OpenMTC can be seen as alternative implementation of the existing IoT Backend and IDAS (IoT Agent) Generic Enablers. The architecture and interconnection with the FIWARE Context Broker are shown in Figure 41.

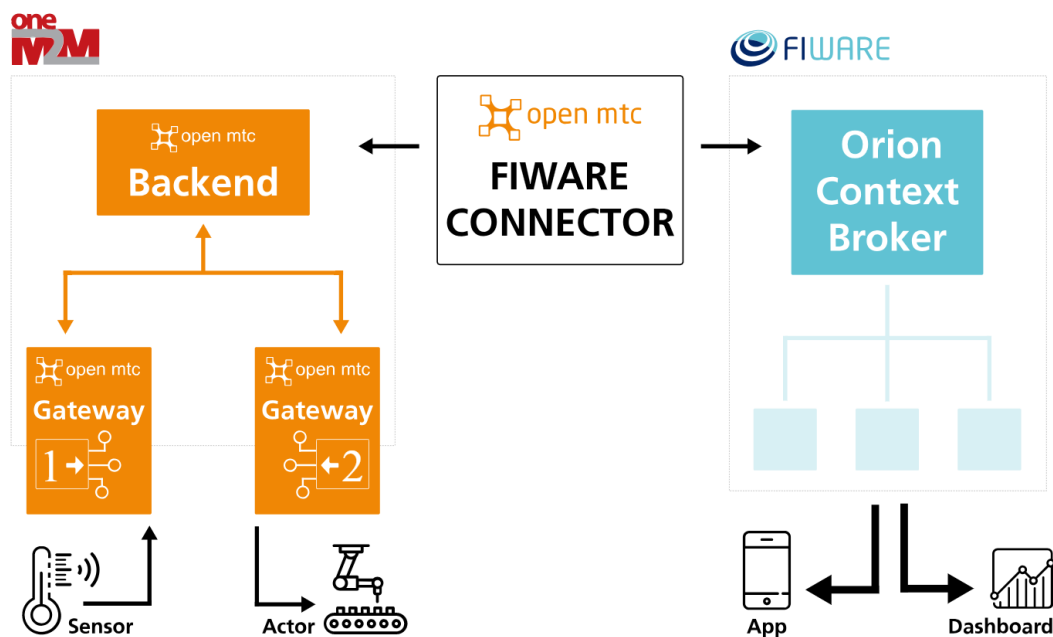


Figure 41 OpenMTC & Fiware architecture

⁴⁸ <https://catalogue-server.fiware.org/enablers/openmtc>

⁴⁹ <http://www.onem2m.org/>

⁵⁰ <https://www.fokus.fraunhofer.de/en/ngni/workingareas/m2m>

IDAS and OpenMTC feature comparison

Table 3 compares the features available in the two FIWARE products.

Feature	IDAS	open MTC
Device connectivity		
• HTTP transport	yes	yes
• MQTT transport	yes	yes
• LWM2M protocol support	yes	yes through IPE
• LoRaWAN protocol support	yes	
• Sigfox support	yes	
• Ultralight 2.0 protocol support	yes	
• JSON support	yes	yes (only ONEM2M API JSON or XML)
• OPC-UA support	yes	planned IPE
• Modbus		planned IPE
• Profinet		planned IPE
• ONE M2M protocol		yes
• Zigbee		planned IPE
Device Management		
• Provisioning of devices	yes through API	yes through API
• Device configuration	yes through API	yes through API
• security authentication	yes API key	yes SSL + key pair
• security encryption	http only, https can be provided through external proxy	yes: end to end https support and SSL/TLS for MQTT
• Monitoring of devices	yes through API	yes through API
• Devices deprovisioning	yes remove device, through API, but no disable feature	yes remove device, through API, but no disable feature
• Set device geographic location	yes	
Data transformation	yes (simple expression)	yes
Data validation	no	

Table 3 IDAS and OpenMTC features' comparison

4.6 Proprietary systems

In the CIP architecture there is a distinction between sensor data and data from proprietary systems. Both connections are about the supply of data (from sensor and systems) and making it available in a secure way. The reason the distinction is made, is that existing domains (like energy or mobility) in many cases already have their own platforms for connecting partners and exchanging data. There are often other requirements to connect to these systems. Take for example the energy ecosystem. It can be considered as a domain platform, connecting a lot of different stakeholders, using their own standards and exchanging specific data. In relationship to the CIP there might only be a limited set of data that can be shared or is relevant for the challenges IRIS LH cities face. In that sense, connecting to each system requires a different approach, with different agreements about connecting to the data sources.

Another example is the so called vertical solutions for domains like waste management, parking or air quality monitoring. Companies that offer solutions for these specific topics usually have their own systems and services as well. Providing access to their data (to develop new applications) requires agreements on API-usage, security and billing. From the CIP perspective these vertical solutions are proprietary systems.

For each of those systems there are different ways to connect, like a direct API, upload of data with ETL-tooling, harvesting or just a reference.

The generic tools within the CIP, like API-management, Data Management Framework, Security and Privacy and Platform Management can also be used in different configurations to connect to these proprietary systems.

5. Conclusions

A sustainable open urban platform architecture requires a strong connection to actual business challenges. It needs to be flexible and adaptive, because requirements are growing, technologies are changing and standards are developing. Therefore, the technical architecture for CIP is more than a description of software components and will be work in progress to be able to respond to new use cases, processes and other business requirements. This technical architecture is a living document. By implementing projects new insights will be gathered and the CIP-architecture can be extended and improved.

The functionality within the CIP can be fulfilled with different software solutions, like database technologies, rule/event engines or API-management frameworks. The CIP should be technology agnostic. More important for a sustainable, open urban platform is the adherence with standards and Open API's. They are the glue between (interchangeable) software components.

The architecture for the CIP seeks connection with broader standardization initiatives and proven solutions, like FIWARE and TM Forum. The LH cities have joined the front runner program from these organizations to collaborate on the development of an open, standardized urban platform for smart cities. A reference implementation of the CIP will use these proven software solutions and components.

The CIP resembles in many ways the open FIWARE architecture, with its generic enablers, data models and TM Forum Open APIs. These requirements of openness, modularity and broad community support provide trust for a future proof, sustainable architecture. Nevertheless, there will be blank spots, incomplete data models and standards to be developed. In the case that software components are lacking or immature, collaboration will be searched with other cities, partners and the aforementioned organizations, so that generic, reusable and open components can be added to the overall architecture. The close interaction between IRIS use cases and the architecture with its components, models and standards will ensure that this document will evolve towards a practical and usable reference for implementing urban platforms.

In order to implement a sustainable CIP, a structured approach must be used, with a connection to actual user stories to refine the requirements. This structured approach can be achieved by using the toolset offered by CurateFX. All relevant aspects, from roles, ecosystem, value propositions and Open API's can be discussed and documented. All LH cities have used the same approach for different use cases. It provided input to structure the relationships between stakeholders and helped to create an ecosystem design. From this design, with its different kind of relationships, the TM Forum Open API's are easily connected.

Annex 1: ARX

ARX is an open source tool for transforming structured (i.e. tabular) personal data using selected methods from the broad areas of data anonymization and statistical disclosure control. It supports transforming datasets in ways that make sure that they adhere to user-specified privacy models and risk thresholds that mitigate attacks that may lead to privacy breaches. ARX can be used to remove direct identifiers (e.g. names) from datasets and to enforce further constraints on indirect identifiers. Indirect identifiers (or quasi-identifiers, or keys) are attributes that do not directly identify an individual but may together with other indirect identifiers form an identifier that can be used for linkage attacks. It is typically assumed that information about indirect identifiers is available to the attacker (in some form of background knowledge) and that they cannot simply be removed from the dataset (e.g. because they are required later for analyses). ARX also supports methods for protecting sensitive attributes from disclosure and semantic privacy models, which require fewer assumptions to be made about the goals and the background knowledge of attackers.

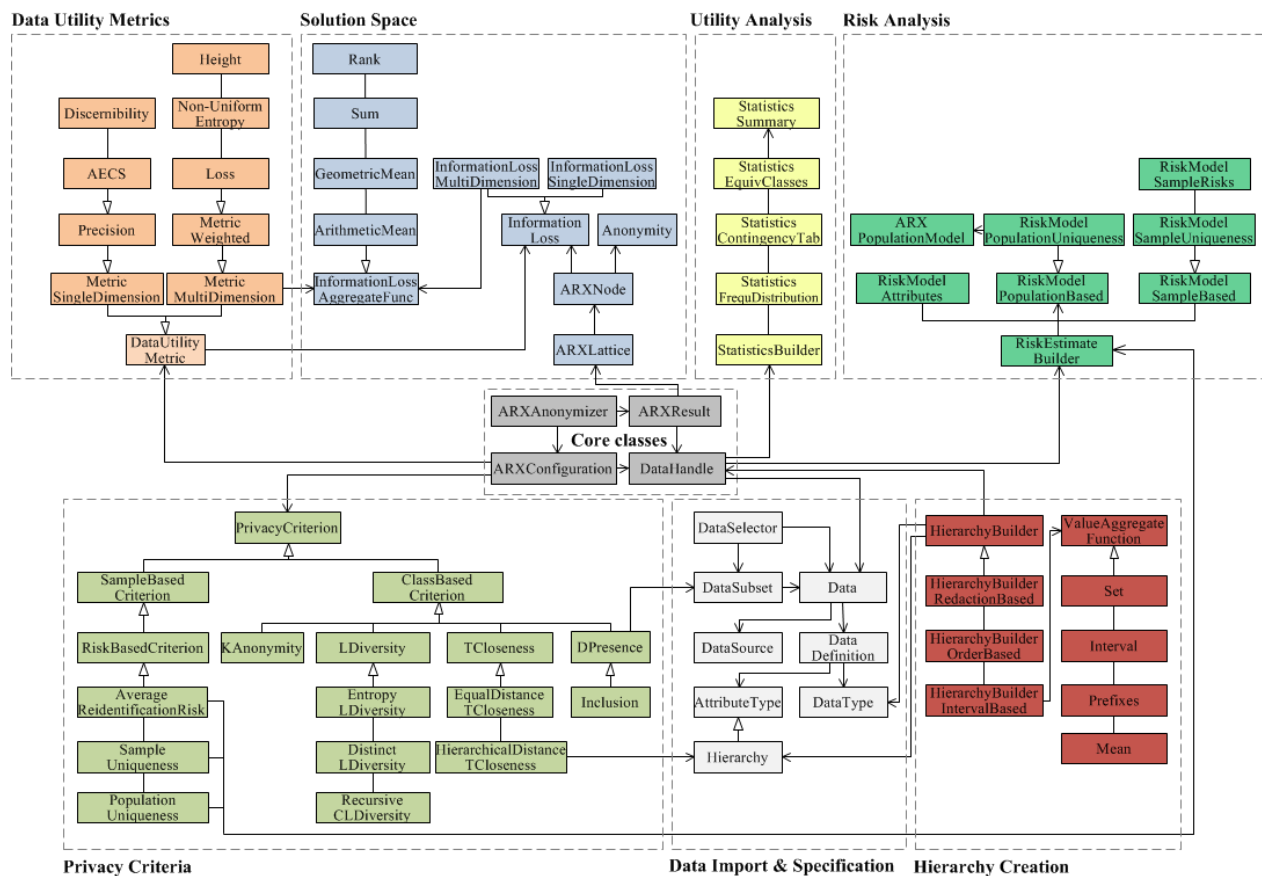


Figure 42 Overview ARX API