International Telecommunication Union

ITU-T

TELECOMMUNICATION
STANDARDIZATION SECTOR
OF ITU

Series Y

Supplement 61

(07/2020)

SERIES Y: GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES

Features of application programming interfaces for Internet of things data in smart cities and communities

ITU-T Y-series Recommendations – Supplement 61

ITU-T Y-SERIES RECOMMENDATIONS

GLOBAL INFORMATION INFRASTRUCTURE, INTERNET PROTOCOL ASPECTS, NEXT-GENERATION NETWORKS, INTERNET OF THINGS AND SMART CITIES

GLOBAL INFORMATION INFRASTRUCTURE	
General	Y.100–Y.199
Services, applications and middleware	Y.200–Y.299
Network aspects	Y.300–Y.399
Interfaces and protocols	Y.400–Y.499
Numbering, addressing and naming	Y.500–Y.599
Operation, administration and maintenance	Y.600–Y.699
Security	Y.700–Y.799
Performances	Y.800–Y.899
INTERNET PROTOCOL ASPECTS	
General	Y.1000–Y.1099
Services and applications	Y.1100–Y.1199
Architecture, access, network capabilities and resource management	Y.1200–Y.1299
Transport	Y.1300–Y.1399
Interworking	Y.1400–Y.1499
Quality of service and network performance	Y.1500–Y.1599
Signalling	Y.1600–Y.1699
Operation, administration and maintenance	Y.1700–Y.1799
Charging	Y.1800–Y.1899
IPTV over NGN	Y.1900–Y.1999
NEXT GENERATION NETWORKS	
Frameworks and functional architecture models	Y.2000–Y.2099
Quality of Service and performance	Y.2100–Y.2199
Service aspects: Service capabilities and service architecture	Y.2200–Y.2249
Service aspects: Interoperability of services and networks in NGN	Y.2250–Y.2299
Enhancements to NGN	Y.2300–Y.2399
Network management	Y.2400–Y.2499
Network control architectures and protocols	Y.2500–Y.2599
Packet-based Networks	Y.2600–Y.2699
Security	Y.2700–Y.2799
Generalized mobility	Y.2800–Y.2899
Carrier grade open environment	Y.2900–Y.2999
FUTURE NETWORKS	Y.3000–Y.3499
CLOUD COMPUTING	Y.3500–Y.3599
BIG DATA	Y.3600–Y.3799
QUANTUM KEY DISTRIBUTION NETWORKS	Y.3800–Y.3999
INTERNET OF THINGS AND SMART CITIES AND COMMUNITIES	
General	Y.4000–Y.4049
Definitions and terminologies	Y.4050–Y.4099
Requirements and use cases	Y.4100–Y.4249
Infrastructure, connectivity and networks	Y.4250–Y.4399
Frameworks, architectures and protocols	Y.4400–Y.4549
Services, applications, computation and data processing	Y.4550–Y.4699
Management, control and performance	Y.4700–Y.4799
Identification and security	Y.4800–Y.4899
Evaluation and assessment	Y.4900–Y.4999

For further details, please refer to the list of ITU-T Recommendations.

Supplement 61 to ITU-T Y-series Recommendations

Features of application programming interfaces for Internet of things data in smart cities and communities

Summary

A growing number of smart cities and administrations are inclined to collaborate and mutualize their efforts and resources for Internet of things (IoT) deployments and open data sharing. This Supplement 61 to ITU-T Y-series Recommendations studies the concept and potential of developing secured open and interoperable application programming interfaces (APIs) in the context of IoT deployment and open data management in smart cities. Supplement 61 to ITU-T Y-series Recommendations analyses current solutions implemented by administrations around the world, where applicable, including those adopted by smart cities, to share their data through open and interoperable interfaces. Supplement 61 to ITU-T Y-series Recommendations subsequently specifies open and interoperable APIs for secured open data architecture and to support IoT data interoperability for smart cities. Supplement 61 to ITU-T Y-series Recommendations concludes by mapping the specified APIs with relevant work performed by other international standards development organizations (SDOs) and alliances, to help consolidate the standards developed on the topic.

History

Edition	Recommendation	Approval	Study Group	Unique ID[*]
1.0	ITU-T Y Suppl. 61	2020-07-13	20	11.1002/1000/14368

Keywords

API, interoperable, IoT, open data.

[*] To access the Recommendation, type the URL http://handle.itu.int/n in the address field of your web browser, followed by the Recommendation's unique ID. For example, http://handle.itu.int/11.1002/1000/11830-en.

Table of Contents

Supplement 61 to ITU-T Y-series Recommendations

Features of application programming interfaces for Internet of things data in smart cities and communities

1 Scope

This Supplement complements [b-ITU-T Y.4472] and:

– addresses the concept and potential of APIs, their common characteristics and high-level requirements in the context of IoT deployment and open data in smart cities;

– addresses the current solutions implemented around the world, where applicable, including those adopted by smart cities, to share their data through open and interoperable interfaces;

– maps the developed APIs with relevant work performed by international standards development organizations (SDOs) and alliances.

2 References

None.

3 Definitions

3.1 Terms defined elsewhere

This Supplement uses the following terms defined elsewhere:

3.1.1 application programming interface (API) [b-ITU-T T.170]: Boundary across which a software application uses facilities of programming languages to invoke software services. These facilities may include procedures or operations, shared data objects and resolution of identifiers.

3.1.2 Internet of things (IoT) [b-ITU-T Y.4000]: A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies.

NOTE 1 – Through the exploitation of identification, data capture, processing and communication capabilities, the IoT makes full use of things to offer services to all kinds of applications, whilst ensuring that security and privacy requirements are fulfilled.

NOTE 2 – From a broader perspective, the IoT can be perceived as a vision with technological and societal implications.

3.2 Terms defined in this Supplement

This Supplement defines the following terms:

3.2.1 data monetization: Generation of economic benefits from different data sources.

3.2.2 open application programming interface: An application programming interface (API) that is publicly available for developers allowing programmatic access to web services or applications. An open API permits the exchange of data between different organizations.

3.2.3 open data: Data that can be publicly accessible to everybody through open standards and protocols or through other means. Use and redistribution of open data can be subject to rules.

3.2.4 smart city: Urban area where Internet of things devices are deployed to collect data. The data collection enables improvements to management of and quality of life in a city.

3.2.5 southbound interface: Interface enabling a given component to communicate with other components located at a lower level.

4 Abbreviations and acronyms

This Supplement uses the following abbreviations and acronyms:

3D	three Dimensional
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
6TiSCH	IPv6 over the Time-Slotted Channel Hopping mode of IEEE 802.15.4e
ABAC	Attribute-Based Access Control
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
APM	Application Performance Management
AR	Augmented Reality
B2B	Business to Business
B2C	Business to Consumer
CDB	Context Data Broker
CDM	Context Data Management
CoAP	Constrained Application Protocol
CSP	Communication Service Provider
DCAT	Data Catalogue Vocabulary
DCAT-AP	Data Catalogue Vocabulary-Application Profile
DISCES	Distributed Smart City Engine Scheduler
DSE	Domain-Specific Enabler
ESO	European Standardization Organization
ETL	Extract, Transform and Load
GDPR	General Data Protection Regulation
GE	Generic Enabler
GSMA	GSM Association
GTFS	General Transit Feed Specification
HTTP	Hypertext Transfer Protocol
HVAC	Heating, Ventilation and Air Conditioning
I2ND	Interface to Networks and Devices
ID	Identifier
IdM	Identity Management
IoT	Internet of Things
IPv6	Internet Protocol v6
ITS	Intelligent Transport System
JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation for Linked Data
LoRaWAN	Long Range Wide Area Network

LWM2M	Lightweight Machine to Machine
MIM	Minimal Interoperability Mechanism
MQTT	Message Queuing Telemetry Transport
NFV	Network Function Virtualization
NGSI	Next Generation Service Interface
NGSI-LD	Next Generation Service Interface Linked Data
NoSQL	Non-structured Query Language
ODMS	Open Data Management System
OPC-UA	Open Platform Communications Unified Architecture
PDP	Policy Decision Point
PEP	Policy Enforcement Point
RDF	Resource Description Framework
REST	Representational State Transfer
RTZ	Restricted Traffic Zone
RZ	Reference Zone
SAML	Security Assertion Markup Language
SAREF	Smart Appliances Reference
SI	Support of Integrated Interoperability
SDN	Software-Defined Networking
SDO	Standards Development Organization
SLA	Service Level Agreement
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Sockets Layer
TSCH	Time-Slotted Channel Hopping
UDG	Universal Device Gateway
UML	Unified Modelling Language
URI	Uniform Resource Identifier
XACML	Extensible Access Control Markup Language
XML	Extensible Markup Language
XMPP	Extensible Messaging and Presence Protocol

5 Conventions

None.

6 Introduction to open data and open APIs

This clause elaborates the terminology and concepts related to open data.

6.1 Nature of open data

In general, the concept of *open data* is premised on the fact that some data sets should be made accessible beyond the limits of copyright, patentsor censorship.

It is envisioned that open data platforms will increase the openness of operations and will promote transparency, boost civic engagement and facilitate development of new services. Before initiating the creation of an open data platform, it is important to understand the difference between the terms *public data* and *open data*. The main distinction between the two is that, although public data are made freely available, they need not be open. Open data often have a particular licence related to their use and distribution. The access to open data can be restricted through security measures like authentication and authorization in certain cases.

Open data platforms are often centred on governments initiating the collection of big data from multiple sources. However, it is important to realize that not all government data can be published as open data. Certain datasets may continue to remain restricted for reasons including national security.

Linked data involves publishing and connecting unstructured data to make a structured dataset. Such linked data will allow data providers to link their data with other different sources. This will assist in the creation of a web of data, providing more opportunities for the use of the data for innovative purposes. A web of data also enables large amounts of data to be processed and analysed efficiently.

Unstructured streams of data would not be usable for *data analysis* because the main value lies in efficient processing of these data to convert them into highly valued insights needed to drive decision-making processes and provide services. In effect, data analytics consists of the continuous large-scale analysis and processing of information (as more data get collected).

The main benefits of data analysis can bring, for instance:

– fraud detection;
– definition of safety rules;
– establishment of more appropriate and proportionate security mechanisms.

Data analytics is becoming reality in governments due the quantity of data available and the evolution of technologies and techniques used to analyse data. It also helps create a customer-centric approach when providing services and improves responsiveness in order identify key institutional mechanisms for further value creation.

6.1.1 Excerpt of some key open data definitions

The definitions listed in Table 1 are extracted from multiple literature sources. This is by no means an exhaustive list. More work is still needed to develop the concept of open data (towards a standardized definition and concept). However, this clause serves as a primary objective kernel to analyse the concept and explore some of its key facets.

The following areas are addressed in this Supplement.

1) The difference between open data, public data, and public sector information (PSI) reuse.
2) Whether open data is freely accessible by everyone. Is it limited to the citizens of the country? Alternatively, should it be made available to city inhabitants regardless of their nationality?
3) The risks associated with including all types of data sets in the open data definition. What data sets should be excluded from being open to protect users' privacy, and mitigate any security-related risks (e.g., critical infrastructure information)?

Table 1 – Open data definitions

Source	Definition
[b-OKF-a]	"Open data is data that can be freely used, re-used and redistributed by anyone – subject only, at most, to the requirement to attribute and share alike."
[b-OKF-b]	"Knowledge is open if anyone is free to access, use, modify, and share it – subject, at most, to measures that preserve provenance and openness." The following text appears on [b-OKF-a] to introduce the link to [b-OKF-b]. "To summarize the most important: **Availability and access**: The data must be available as a whole and at no more than a reasonable reproduction cost, preferably by downloading over the internet. The data must also be available in a convenient and modifiable form. **Re-use and redistribution**: The data must be provided under terms that permit re-use and redistribution including the intermixing with other datasets. **Universal participation**: Everyone must be able to use, re-use and redistribute – there should be no discrimination against fields of endeavour or against persons or groups. For example, 'non-commercial' restrictions that would prevent 'commercial' use, or restrictions of use for certain purposes (e.g., only in education), are not allowed. The following text appears on [b-OKF-b]. "The term *work* will be used to denote the item or piece of knowledge being transferred. The term *license* refers to the legal conditions under which the work is provided. The term *public domain* denotes the absence of copyright and similar restrictions, whether by default or waiver of all such conditions."
[b-Buhr]	"Public data" can be defined as all the information that public bodies produce, collect or pay for. Public data is quite varied and can include geographical data, statistics, meteorological data, data from publicly funded research projects, and digitized books from libraries. Generally, this data is collected by the public bodies in the framework of their public service missions, or as a consequence of them. It is not produced – nor paid for – with a view to be re-used by the public. 'Open data' is data that is readily accessible to be easily consulted and re-used by anyone with access to a computer. It is a raw material that can be analysed and exploited. 'Readily accessible' means much more than the mere absence of a restriction of access to the public and requires an active access and re-use policy of the data provider. Re-use of public sector information (PSI) means any creative use of the data, for instance by exploiting public data in a novel manner in order to give it a commercial value, or by combining different data sets from different sources to produce new results and new applications."
[b-Guillaume-Gentil]	"Open data is data that is freely available for everyone to use and share."

Considerations on open data

The publication of data could violate individual privacy constraints and copyright or jeopardize critical or classified information, which may also have some security implications. Accordingly, data should be filtered out or processed in a way that these risks are mitigated. Methods could include anonymization techniques, data smoothing, data classification and access control or any other technique meant to protect the data and its source.

6.1.2 Concept of open data

Data is the oil of the knowledge economy. It enables the creation of new applications and services, with an opportunity to generate new economic activity. Data analysis and processing potentially lead to new insights that can be exploited to create value. Data has to be produced and made available in a way that makes its re-use possible by third parties while adhering to a sound legal framework that secures such new usages without infringing copyright and data protection.

Open data refers to the concept that data should be available in a machine readable form, accessible, reusable and redistributable by a predefined set of actors. The private sector can also make its data open, however, since many data sets currently available belong to governments, (public bodies produce, collect or pay for the gathered data), it is important to analyse and address the extent to which administrations can open up data sets while meeting security, privacy, and economic aspects of all stakeholders affected. The different stakeholders involved in smart cities are the citizens, the city authorities and the service providers.

In practice, various socioeconomic, security, and technical barriers are associated with the supply and use of open data. These barriers are related to the data availability, quality, sensitivity, impact, compatibility, ease of access, usability, ownership and custodianship.

6.2 Concept of open APIs

Open APIs can be designed in many different ways, but the main priority of any open API architecture is that the API itself should be easy to consume and can be called by clients.

An open API (often referred to as a public API) is a publicly available API that provides developers with programmatic access to a proprietary software application or web service. APIs are interfaces, implicitly programmed functions, with a set of functions that define how one application can communicate and interact with another.

The concept of open APIs is even less developed and understood than open source. While open source describes the available code base as "open", the API links the code functions that are to be used by third parties. There is currently a discussion about the API economy that conceptualizes the benefits of interoperability achieved by connectivity through APIs to promote the data economy.

6.2.1 Open data API

An advanced concept of handling open data is facilitated by the Open Knowledge Foundation (OKF), provided by the CKAN framework.

CKAN is in use by numerous governments, organizations and communities around the world, currently the majority of open datasets globally are available and can be exposed via CKAN standards [b-CKAN], that originate from the OKF [b-OKF-c].

Data management via CKAN is described in the data management API (REF). In using the API, hypertext transfer protocol (HTTP) requests to URLs are performed like: /api/rest/dataset with this returning data in JavaScript object notation (JSON) format.

The appropriate syntax is described in Table 2.

Table 2 –Syntax appropriate for data management

Interface	Operation	Parameters

Search	Get the results of a search.
	For example, a search with the query "`q=open+street+map`" returns these results:
	
	`{`
	`    "count": 5,`
	`    "results": ["open-street-map-sample-file",`
	`"transport-the-18th-century-cassini-roads-and-cities-`
	`dataset", "transport-exports-by-mode-of-transport-1966",`
	`"transport-traffic-commissioners-local-bus-service-`
	`registration", "food-hygiene-rating-scheme-camden"]`
	`}`
	
	[b-CKAN]
		The query composed by words separated by the "+" character.

Get	Get the data from a specific data set.
	For example, the results of the operation are:
	`{`
	`    "count": 32,`
	`     "display_name": "test-date",`
	`     "name": "test-date"`
	`}`
	
	[b-CKAN]
		The name of the dataset.

Update	Update the data of a specific dataset.
	For example, if a modification is done through this operation, the results are:
	`{`
	`    "count": 32,`
	`     "display_name": "test-date-updated",`
	`     "name": "test-date-updated"`
	`}`
	
	[b-CKAN]
		The name of the dataset.

View	View the content of a specific dataset.
	For example, this operation displays these results:
	`{`
	`    "count": 32,`
	`     "display_name": "test-date-updated",`
	`     "name": "test-date-updated",`
	`     "tags": "education"`
	`}`
	
	[b-CKAN]
		The name of the dataset.

6.2.2 Private access data API

If datasets are *private* and not open by default, a proper decision should be considered with relation to an authentication, to maintain access control to authorized users or instead using open data (see clause 6.2.1).

When a dataset requires an authentication, users get a response to notify them that they are unauthenticated, like "403 Unauthenticated", which clarifies that the data are "Not Public".

Authentication is handled typically by an API proxy using OAuth standards. Generally, OAuth in its current version 2.0 [b- IETF RFC 6749] provides a "secure delegated access" to server resources to clients on behalf of a resource owner.

In the standard case, the server exchanges the authorization code for an access token by making a POST request to the authorization server's token endpoint: grant_type=authorization_code.

6.3 Open data APIs

The way to access and search for open data is very important to simplify and foster re-use by both end-users (e.g., citizens) and third-party stakeholders, who want, for example, to build applications based on the available data. In this context, the adoption of APIs is considered the best approach to simplify the access to open data from external systems in order to visualize, process and analyse it in different ways. The open data APIs can be analysed from the perspective of different interfaces:

i) *southbound interfaces* are more appropriate (but not limited) to data providers that need a standard way to exchange the data produced by their systems (e.g., city IoT platforms);

ii) *northbound interfaces*, from the point of view of the data consumers, represent the entry point to access to open data for further reuse.

iii) *east-west interface*, most often focuses on logically distributed software-defined networking (SDN) control planes.

Further details are provided in clauses 6.3.1 to 6.3.3.

6.3.1 Common northbound interface

Generally, northbound interfaces can be considered as those that provide harmonized access to higher levels of an IT system hiding the functionalities or services of the lower layers. Translating this consideration into the domain of open data APIs, northbound interfaces represent the pathway to access the open data provided by a generic system so that the main focus is on the data consumer. The major issues related to northbound interfaces have been represented by the standardization of communication protocols (e.g., HTTP) and data models for both open data and metadata. Hence, a common northbound interface highlights the need to identify shared and standard ways to access open data systems. Despite the fact that the specification of unique interfaces is not an easy task (and might probably not be achievable due to large differences in data domain typology), it is necessary to establish some overall characteristics. Common northbound interfaces are also pathways for communication between different systems. Therefore, the concept is also relevant to standardization in the field of common security and privacy technologies.

In recent years, several national and international initiatives have worked on common northbound interfaces in the IoT and smart city domain in order to identify some common suggested approaches and technologies. Some of the most relevant ones include:

1) the use of a representational state transfer (REST) API based on presentational state transfer technology that explicitly takes advantage of HTTP methodologies specified in [b-IETF RFC 2616] protocol: one of the basic principles of REST is that each request from client to server must include the information needed to understand the request, and cannot take advantage of any stored context on the server;

2) context management: in a smart city, data (particularly in real time) about real-world entities represent the context information needed to understand complex phenomena; open data APIs need to provide functionalities to read and update the context information.

6.3.2 Common southbound interface

The concept of the southbound interface, like that of the northbound, is directly related to the architectural approach applied to a specific IT system and to the application domain. In this case, it is possible to relate this concept to the IoT and smart city domain in which the southbound interfaces can be considered to be connections to IoT platforms in lower layers. Specifically, southbound interfaces can be referred to different scenarios:

i) southbound interfaces are based on specific IoT protocols that interact with devices, e.g., well-known IoT protocols like the message queuing telemetry transport (MQTT), constrained application protocol (CoAP), extensible messaging and presence protocol (XMPP) and lightweight machine to machine (LWM2M);

ii) southbound interfaces can also be provided by middleware or an IoT platform API, which can have a gateway function exposing different device interfaces through a uniform interface (e.g., HTTP/REST) hiding the complexity of the underlying protocols.

In both cases, for an open data system dealing with potentially very different and heterogeneous technologies, the interoperability aspect represents a key issue in common southbound interface establishment: it can be established by different approaches to achieve interoperability, e.g., developing adapters for dedicated middleware that can harmonize both communication protocols and data format.

6.3.3 Common east-west interface

Smart city platforms are envisioned to be developed using virtualization technologies. SDN and network function virtualization (NFV) are the current trend for the creation of a city-wide operation, amusement and control platform. In the context of SDNs, removing the decision part or the control plane from network devices and shifting them towards a single point of control, the SDN control plane manages all underlying equipment in a unified manner. The SDN control plane and devices communicate via the southbound interface. In this context, the city platform should include an SDN controller. However, using a single controller risks the occurrence of scalability and performance issues. Accordingly, distributed architecture is advisable and for that an East-West interface is needed to communicate and align multiple controllers logically managing the same set of underlying southbound infrastructure.

7 Common characteristics and high-level requirements of open data APIs

The concept of smart cities has swept the world with expanding urban areas massively requiring interconnected software applications. In this scenario, the use of APIs has largely replaced technologies such as electronic data interchange and custom-written integration programs for development of new system interactions. Thus, over the years, APIs have become the *de facto* industry mechanism for integrating data and functionality across diverse application ecosystems within the complex and growing IoT and smart city domains.

The main technical issues regarding smart city solutions are related to data gathering, aggregation, strategy, data analytics, access and service delivery via smart city APIs [b-Badii]. Different types of smart city APIs enable smart city services and applications, while their effectiveness depends on the architectural solutions to pass from data to services for city users and operators, exploiting data analytics, and presenting services via APIs. Therefore, there is a strong activity on designing smart city architectures to cope with this complexity, putting in place a significant range of different kinds of services and processes.

In the urban sphere, both business to business (B2B) and business to consumer (B2C) interactions are supported by API connections. This usage has been growing annually. Additionally, the growth of public cloud and Internet of things (IoT), as well as the increasing use of mobile devices, containers and micro-services has accelerated the use of APIs in production environments. In line with this, industry standards or technologies such as REST, simple object access protocol (SOAP) and hypertext transfer protocol – secure (HTTPS) facilitated the process. HTTPS permits secure connections between the components using the API, guaranteeing the confidentiality and the integrity of the data exchanged through the API.

APIs built over the protocols standardize the integration process. They reduce the need for the integrations of the past, which were required to support exotic protocols and proprietary operational systems. In short, APIs are the standard currency of exchange connecting applications, devices and companies.

There are two sides to APIs:

1) provision: a large number of companies act as API providers, exposing their systems to those of customers, partners, and suppliers;

2) consumption: growing numbers of companies consume APIs to access data and functionality exposed by other entities and organizations.

Most participants in the API economy are doing both. Some are also monetizing access to data or internal systems as part of revenue generation.

The speed and breadth with which standards-based APIs have proliferated is impacting application performance management (APM) in a big way. Applications relying on APIs to provide data or functions necessary to complete a transaction – an Internet sale, for example – can be slowed or stalled by many of the same factors as tiered, distributed transactions. At the same time, however, APIs are supported by new protocols, connection methodologies and architectures that are largely unsupported by many traditional APM solutions.

7.1 Common characteristics

The API is designed to adhere to the following broad characteristics.

Consistency: All objects (basic or composite) share a consistent interface composed of a limited set of methods. This interface is documented in a consistent manner for all objects.

Inspection: Constructor parameters and parameter values determined by learning algorithms are stored and exposed as public attributes.

Non-proliferation of classes: Learning algorithms are the only objects to be represented using custom classes. Datasets are represented as NumPy arrays or SciPy sparse matrices. Hyper-parameter names and values are represented as standard Python strings or numbers whenever possible.

Composition: Many machine-learning tasks are expressible as sequences or combinations of transformations to data. Some learning algorithms are also naturally viewed as meta-algorithms parametrized on other algorithms. Whenever feasible, such algorithms are implemented and composed from existing building blocks.

Sensible defaults: Whenever an operation requires a user-defined parameter, an appropriate default value is defined by the library. The default value should cause the operation to be performed in a sensible way (giving a baseline solution for the task at hand).

7.2 High-level requirements

The scenarios that can be supported by the implementation of open data APIs can employ different approaches to a list of various requirements. From a high-level point of view, it is possible to identify a set of requirements to be satisfied by any API specification in the IoT and smart city domain as follows.

Context awareness and management: Context information is intended as structured data that contains status information about entities in the real world (e.g., sensor information in a smart city). The API has to provide support for (real-time) context management, e.g., enabling publication of, consumption of and subscription to context information.

Enable semantic interoperability: Semantic interoperability is enabled through the support, availability and future extension of data models, ontologies and controlled vocabularies to cover different needs of heterogeneous applicative domains.

Enabling the deployment of the corresponding data marketplace: Deployment fosters data transactions and monetization.

Interoperability among heterogeneous technology devices feeding information to the data repository: APIs have to be designed to simplify interoperability among existing smart city and IoT platforms.

Openness: APIs have to be available to be accessed without (or with very limited) restriction by external users (e.g., developers) in order to foster their adoption and the reuse of data.

Open standards: APIs have to be based on open standards that are publicly available and, preferably, promoted by international organizations. This avoids the technological fragmentation of the API specification, supporting, at the same time, the no-vendor lock-in principle.

Privacy by design: APIs should comply with the privacy by design and personal data minimization principles. APIs should ensure compliance with applicable data protection regulations and preserve data subject rights.

Reliable and resilient accessibility: APIs need to consume exposed contextual data while guaranteeing that data are consistent in all their dimensions (time and content).

Security by design and easy integration with encryption, authentication and authorization frameworks: APIs have to comply with security by design and to be compliant to the most relevant standard protocols and techniques to address critical security, integrity and privacy issues. This could include, for instance, end-to-end encryption, exclusive usage of secure sockets layer (SSL) and adoption of strong authentication and authorization mechanisms (e.g., OAuth 2.0).

Support for machine readable data: APIs are, by definition, based on a machine readable format, but it is also necessary that they support the exchange of machine readable data and metadata formats (e.g., JSON and Extensible Markup Language (XML)). In this way, APIs enable data processing by third party applications and systems.

Semantic web support: APIs have to be designed to satisfy the needs of semantic data, supporting semantic languages (e.g., resource description framework (RDF)), linked-data formats (e.g., JavaScript object notation for linked data (JSON-LD)) and specific query languages, etc.

Support for intensive and real-time data production: The IoT and smart city domain can be characterized by intensive (near) real-time data production that requires the use of specific technologies to access data in the correct way, in terms of performance and accuracy. The open data APIs have to support these types of technologies.

8 APIs for sharing data from IoT deployments in smart cities

8.1 Context data management API

The context data management API provides a standard way to communicate with the corresponding component, which manages and publishes context information at large scale coming from IoT devices and other open and closed or private data sources. Context information is intended as structured data that contains status information about context entities and their related attributes and metadata. A context entity can represent everything in the real world, such as users, places and devices that can be abstracted and represented using a predefined data model.

The approach followed by the context management interface is intended to cover the entire lifecycle of context information, including updates, queries, registrations and subscriptions.

More specifically, the context management interface consists of the following set of APIs.

– **Manage context API** provides methods for the creation, modification and deletion of context entities. Input entities will be issued with their attributes, in either a normalized way, by issuing structures, attributes, values and related data types, or in a compact way by issuing attributes directly as key-value pairs. In addition, it will be possible either to update or remove one or more entities as well as only specific attributes from each entity.

– **Query context API** provides methods for the discovery and retrieval of entities in the context data management component. Filters can be applied to refine the queries and only access entities that match specific metadata (e.g., identifier (ID), entity type) or attribute values. A query is composed of a list of statements, each of them expressing a matching condition, and return all the entities that match all conditions.

– **Subscription API** provides methods to manage subscriptions to asynchronous notification events about entity updates. It allows subscription to specific context entities, so when any updates occur to them, the subscriber application or service gets an asynchronous notification. When creating a new subscription, an update condition is issued for one or more attributes of each entity along with the expiration time.

8.2 Data storage interface

The data storage interface provides a standard way to access the data storage management component, which allows for storing historical raw and aggregated time series information about the evolution of context data.

The data storage management component provides functionalities that allow access to different storage technologies (e.g., different databases) and exposes a uniform data storage interface. Data managed by this component can be classified into two types:

– public or open data: provided without any restriction or with an open licence that explicitly defines the rights to access, use and share the data;

– private data: provided with strict restrictions (e.g., allow only access and use for specific purposes, sharing is typically not allowed); this category may include personal data, telco data, and energy supplier data.

Moreover, a distinction can be made between:

– actual data: with current values;

– historical data: including data over a period of historical time.

To access these two types of data, the data storage interface often provides two different kinds of API, as follows.

– **Private access data API** enables authorized users to access data through provided methods to retrieve both historical raw and aggregated time series information. These series can be provided according to specific temporal aggregation methods, such as mean, maximum and minimum values.

– **Open data API** provides a unique access point to search and discover open data sets, including historical data where applicable, coming from different open data management system (ODMS) portals.

8.3 IoT data transaction management API

The IoT data transaction management API provides visibility and accessibility to data sets and streams gathered in urban environments and published in a data marketplace, which acts as a catalogue for easy sharing and trading. Data available in the marketplace adheres to standardized data formats, thus making it easy for application developers – which are data consumers – to move their services across different cities. On top of that, data providers can specify data licence and service level agreements (SLAs) when offering their data, which gives data consumers a guarantee of what their rights to use and redistribute data are, as well as what they can expect from the quality of data delivery, improving their trust and confidence to commit to subscribe to data sources.

The conceptual model of a marketplace for IoT data is shown in Figure 1 [b-EU SynchroniCity D2.5]. Figure 1 captures the core abstractions of its architecture, which provide the key concepts enabling interaction with the platform. Figure 1 also includes optional conceptual elements represented by dashed lines; even if not compulsory, their implementation would enrich the basic functionality of the marketplace with features that may be required by a subset of users or stakeholders. At the core of the model is the notion of data offering, which is related to a single data source, a digital asset registered into the marketplace by a data provider. A data source has an endpoint used to access it. When a data offering is published on the marketplace by a data provider, it becomes discoverable by data consumers, who may acquire access to the endpoint of the data source that a data offering is related to. Optionally, external data sources (e.g., data sets already published in open data portals) can be imported into the marketplace by data providers, which can then publish related offerings. Several data offerings may refer to the same data source and provide different data licence agreements and price plans. Optionally, a SLA could be defined which gives data consumers a guarantee of what they can expect from data delivery (e.g., number of data messages over a given period, data completeness). The marketplace could optionally provide a reputation feature that allows data consumers to set a rating score to a data offering they acquired access to or update a previous rating score.

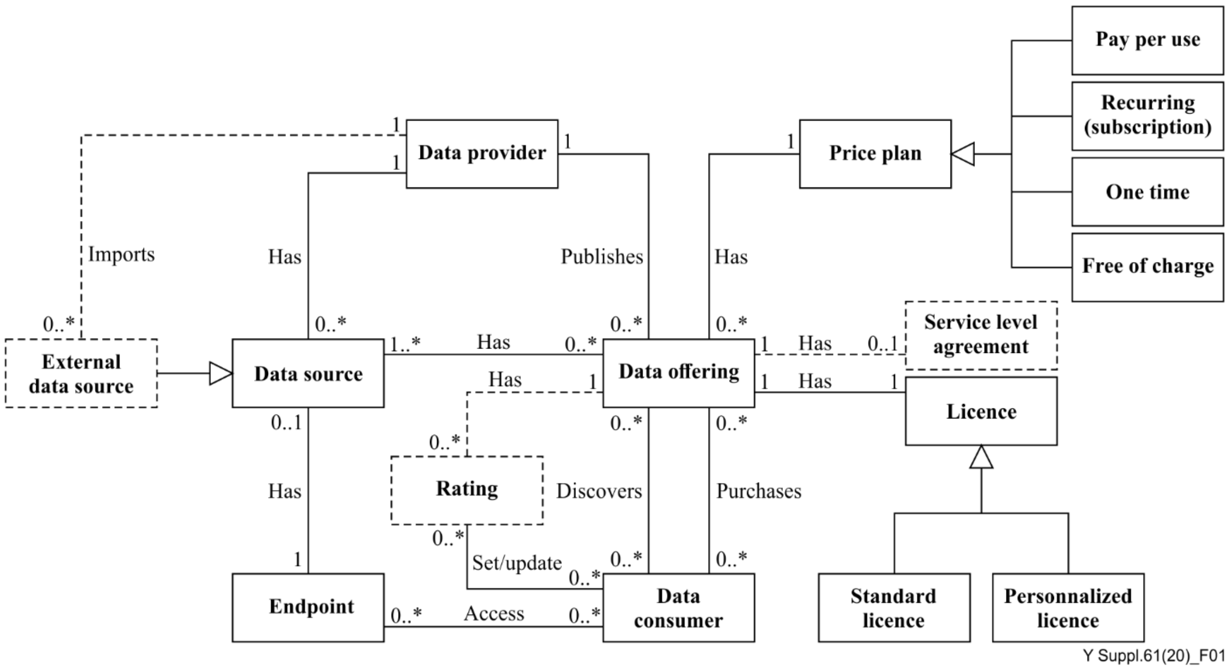


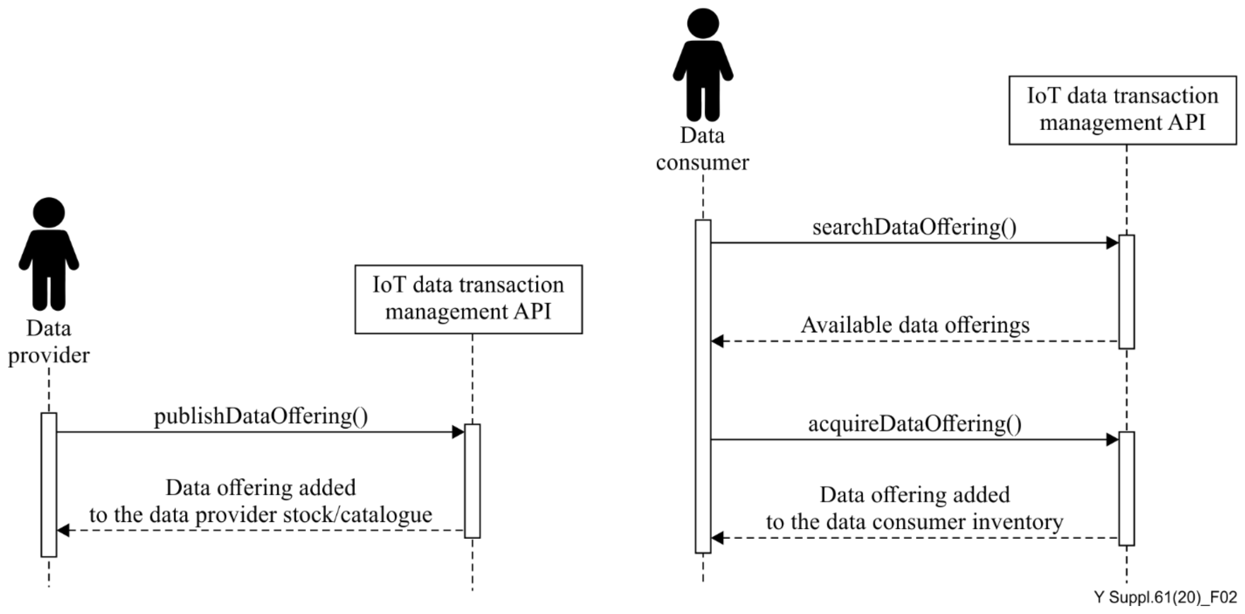
Figure 1 – Conceptual model of the IoT data marketplace Optional conceptual elements are represented by dashed lines (based on Figure 2 of [b-EU SynchroniCity D2.5])

The marketplace may be accessible through the IoT data transaction management API that can also expose a web portal where data providers and data consumers can interact with the platform (e.g., publish, discover and acquire offerings). Identity management (IdM), authentication, authorization and accounting functionalities are ensured by a security component integrated with the marketplace and discussed in clause 8.4. The main functionalities exposed by the IoT data transaction management API are listed in Table 3.

Table 3 – Main functionalities exposed by the IoT data transaction management API

API	Functionality exposed	Description
IoT data transaction management API	publishDataOffering()	Create a new data offering in the marketplace catalogue. Licence agreement and price plan is attached to the offering. Optionally, SLA can be set.
	searchDataOffering()	Look up data offerings in the catalogue. Filters (e.g., data type, location) can be applied.
	acquireDataOffering()	Acquire access to a specific data offering by receiving an access token.

Figure 2 shows two sequence diagrams that illustrate the main interactions of two different types of user, data providers and data consumers, with the IoT data transaction management API.



Data provider
IoT data transaction management API
publishDataOffering()
Data offering added to the data provider stock/catalogue
Data consumer
IoT data transaction management API
searchDataOffering()
Available data offerings
acquireDataOffering()
Data offering added to the data consumer inventory
Y Suppl.61(20)_F02

Figure 2 – Sequence diagrams showing the interactions of a data provider (left) and a data consumer (right) with the IoT data transaction management API

8.4 Security API

The security interface provides a unified approach for the management of security policies as a viable and scalable means to establish and enforce security rules consistently among the large variety of accessible resources (e.g., IoT devices, data and services).

With respect to standard IdM, authentication, authorization and accounting components that reflect the Organization for the Advancement of Structured Information Standards (OASIS) security standard (e.g., Extensible Access Control Markup Language (XACML), oneM2M, GSM Association (GSMA) and European Union Agency for Cybersecurity (ENISA)) policy management is decoupled from devices and services. Policy can be managed independently, thus focusing on providing business value and compliance to data protection regulations. Key management, encryption, digital signature and data anonymization functionalities are directly linked to resources and governed by policy management so that implementation of changes and enforcement are simplified by deploying policies on the fly affecting each point of use immediately. The main APIs that constitute the security interface are described in the following.

Identity and authentication management API: registers, imports and manages users and roles, and performs authentication. Upon successful registration, an ID is assigned to users. Users can also be registered by importing information from an external source. The API performs authentication by receiving user credentials, interacting with the authorization component and issuing authentication tokens. Users with administration privileges can retrieve, update and delete users required to pass their user ID to the API, specifying which operation has to be performed and the respective additional information. In this case. the API returns the outcome of the operation (e.g., success, fail).

Authorization and accounting management API: grants or denies permission to access resources and to log access request by communicating with the identity and authentication management and the policy management APIs.

Policy management API: creates, retrieves, updates and deletes authorization policies. Policies might be simply based on the roles that users have in the system or embed more complex rules specified in a standard mark-up language (e.g. XACML).

Data protection and privacy management API: provides functionality to ensure confidentiality, authentication, integrity, non-repudiation, data minimization, data retention and consent capabilities around the collection and dissemination of data. It provides methods to encrypt or decrypt data, digitally sign data (and verify signatures), and anonymize data using specific algorithms opportunely selected. Figure 3 is a sequence diagram of the interactions among the APIs of the security interface in a data access request. A description of such interactions which highlights specific components of the APIs follows.

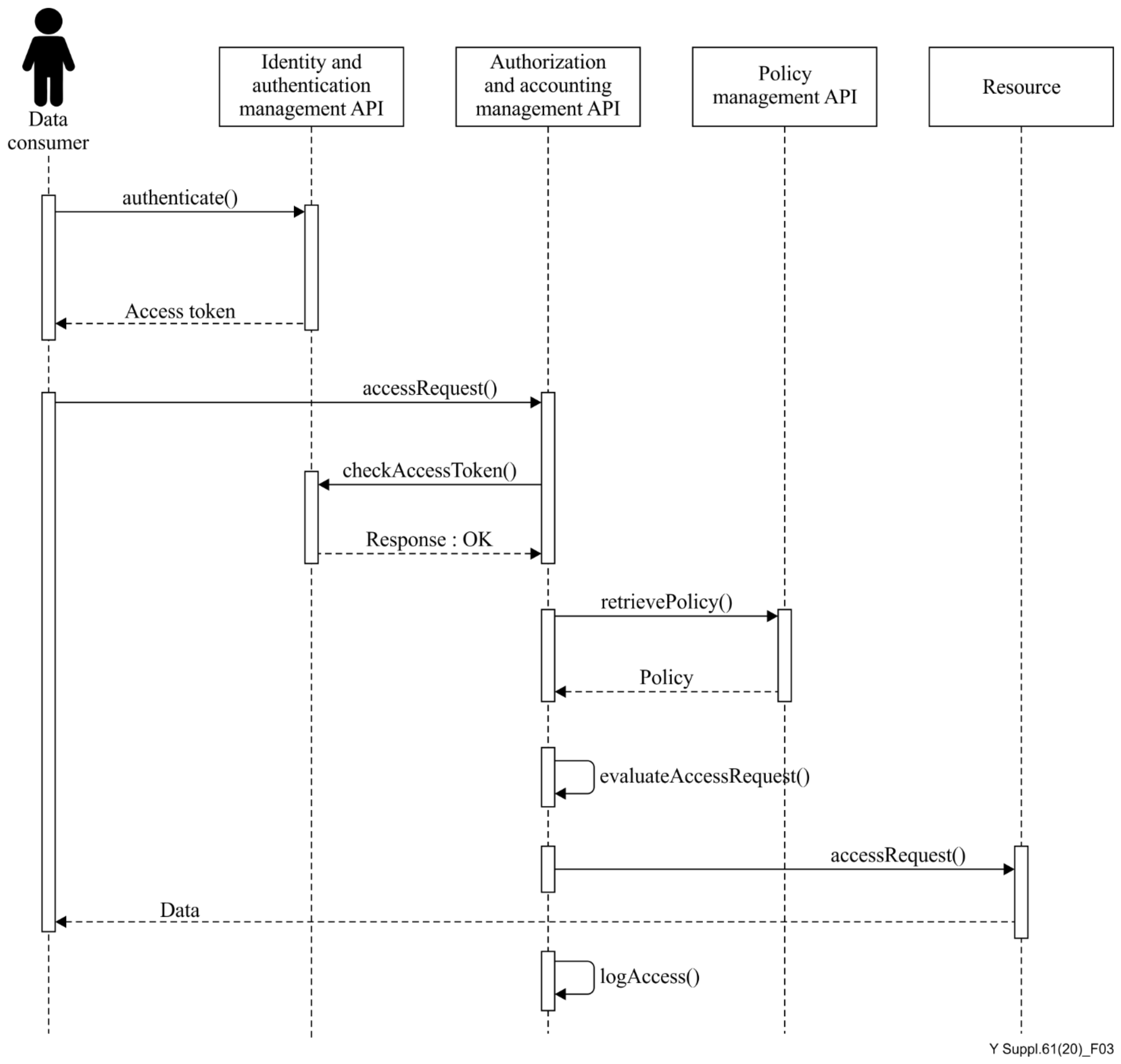


Figure 3 – Sequence diagram showing the interactions among the APIs of the security interface in a data access request

The data consumer (e.g., a service developer that has acquired access to a specific data source via the marketplace) sends an authentication request (login) to the identity and authentication management API, which then respond with an access token that the user will need to use to access the resource (data source). When users want to access a data source, they send an access request to the authorization and accounting management API. If XACML is used, the request is intercepted by the authorization policy enforcement point (PEP) proxy subcomponent that gets authorization information from the token (i.e., the user role) and checks the validity and status of the token through the identity and authentication management API. Eventually, the authorization policy decision point (PDP) subcomponent evaluates the access request after retrieving the authorization policy from the policy management API. Depending on this decision, the PEP proxy blocks or forwards access to the resource (through the context data management API). To track the amount of resources accessed, access information is logged into the accounting subcomponent of the authorization and accounting API.

Figure 4 is a sequence diagram showing the interactions among the APIs of the security interface for creating and retrieving a security and privacy policy. The data protection and privacy management

API enforces security and privacy measures previously established, such as encryption and anonymization, administered by the policy management API, directly on the resource (e.g., data).

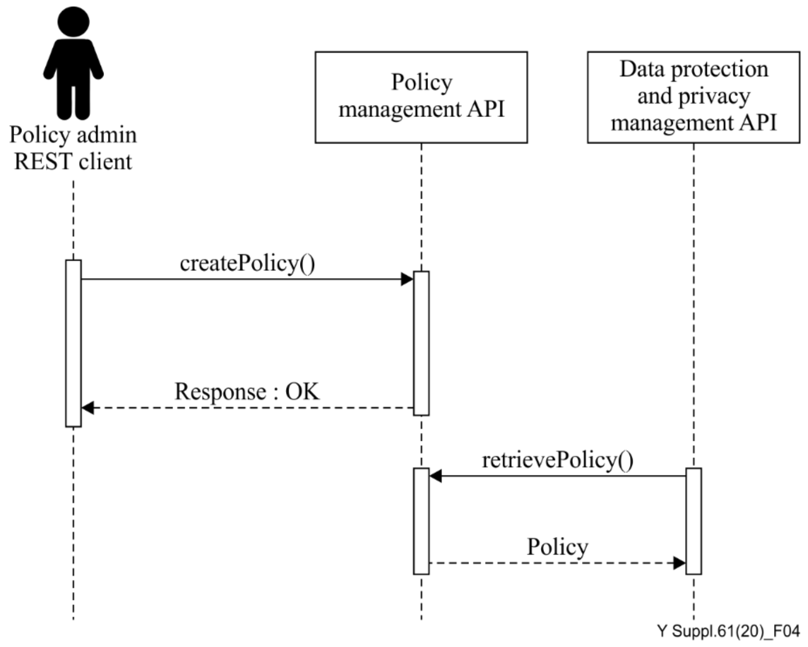


Figure 4 – Sequence diagram showing the interactions among the APIs of the security interface for creating and retrieving a security policy

The main functionalities exposed by the APIs of the security interface are listed in Table 4.

Table 4 – Main functionalities exposed by the APIs of the security interface

API	Functionality exposed	Description
Identity and authentication management	register()	Sign up new user
	authenticate()	Sign in user
	checkAccessToken()	Check the validity and status of the access token
Authorization and accounting management	accessRequest()	Handle resource (data) access requests from users
	logAccess()	Log access information
	evaluateAccessRequest()	Evaluate the access request against a specified policy
Policy management	createPolicy()	Define authorization policies (e.g., role based) as well as security and privacy measures
	updatePolicy()	Update a given authorization policy
	retrievePolicy()	Get a particular authorization policy
	deletePolicy()	Erase an authorization policy

Table 4 – Main functionalities exposed by the APIs of the security interface

API	Functionality exposed	Description
Data protection and privacy management	encrypt() / decrypt()	Encrypt / decrypt data
	sign()	Digitally sign data
	verify()	Verify signature
	anonymize()	Anonymize data using a specified algorithm

9 Common data models

One of the core elements in this Supplement is the adoption of common data models for representing the information linked to the observations. When considering interoperability as a means for breaking the vertical silos that have been characterizing urban ecosystems, it is mandatory to adopt homogeneous data models agreed by the different stakeholders. In this way, synergies among services can be put into practice. As an example, a waste management company can share real time information about the positions of trucks along city streets. Sharing that information with the traffic management department allows car drivers to have access, in real time, to information related to those streets that should be avoided due to potential traffic congestion. Further to that, when data are provided with the corresponding ontologies, the closer the implementation of true autonomic cities becomes. This means that, thanks to data interoperability among services, knowledge acquisition is faster and more reliable, fostering decision-making without human intervention.

In terms of design and implementation, data models have to be conceived to gather most of the usual needs linked to urban services. Furthermore, they have to be easily scaled according to new requirements imposed by the evolution of the technology and the characteristics of particular urban ecosystems.

Last but not least, in the same way that infrastructure federation is becoming a reality, data federation will become a key enabler in optimizing intercity interactions, thus providing an optimization opportunity that will go beyond the legacy local approaches.

See Figure 5.

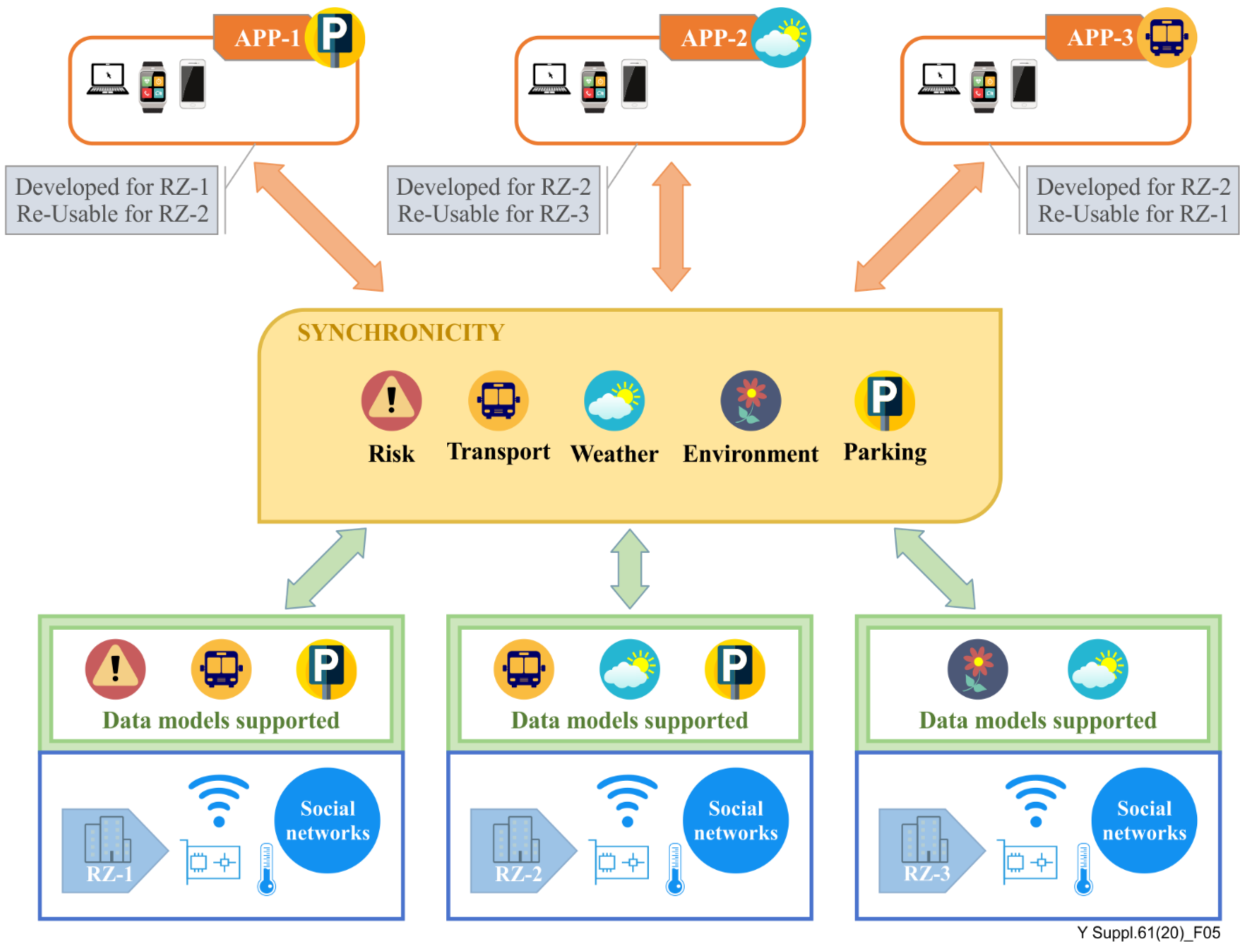


Figure 5 – Different cases to enable data model interoperability
(Figure 33 of [b-EU SynchroniCity D2.1])

10 Examples of specific data model instantiations

This clause describes some examples of data model instantiations proposed by different projects, organizations and initiatives aimed at providing a solid foundation as an enabler for the development of API-based interoperable frameworks.

10.1 SynchroniCity

In the SynchroniCity architecture, data models play a crucial role as they define the harmonized representation formats and semantics that will be used by both city services (through the northbound interface) to consume data and by data sources (IoT sensors, existing information systems, city databases) at the southbound to publish data. Furthermore, data models are one of the *interoperability points* identified in SynchroniCity, which enable participation in a digital single market and the deployment of interoperable IoT-enabled urban services.

To determine the baseline or reference meta-model for instantiating the data models in SynchroniCity, the project consortium has analysed several repositories of existing data models, including: the FIWARE data models, the IoT big data harmonized data models developed by GSMA, the schema.org data models, and the ontologies promoted by the European standards community, particularly the smart appliances reference (SAREF).

The project consortium then decided to select the FIWARE data models as a baseline for their data models compliant with the Open & Agile Smart Cities (OASC) alliance. In fact, they leverage the GSMA data models and reuse concepts coming from schema.org and SAREF, thus enabling interoperability and reusability. Also, they are based on the Open Mobile Alliance (OMA) next generation service interface (NGSI) information model and ready to be consumed by using the NGSI API, part of the context data management interface already adopted by SynchroniCity. The instantiations of these data models can easily be mapped or implemented using a wide variety of data stores including Non-structured Query Language (NoSQL), Structured Query Language (SQL) and Graph Databases. It is possible that they are established using a JSON schema and can be encoded by means of simple JSON structures (key-value structures) and, future-wise, they ensure a seamless transition to the ETSI standard on next generation service interface linked data (NGSI-LD) [b-ETSI WP 31], as it establishes an information model with direct mappings to OMA NGSI.

10.2 FIWARE

The FIWARE data models are based on the OMA NGSI information model. Such a meta-model is based on *entities*, *attributes* and *metadata*. An entity represents a thing, i.e., any physical or logical object (e.g., a sensor, a person, a car or an issue in a ticketing system). Each entity has an *ID* (e.g., "car-104") and a *type*, the latter indicating the specific data model that the entity is associated with. Attributes represent properties of entities or pointers to associated entities. For example, the current speed of a car could be modelled as attribute "speed" of entity "car-104". The *attribute value* contains the actual data and there is optional metadata describing properties of the attribute value, e.g., *accuracy, provider* or a *timestamp*.

An entity is represented by a JSON object that can be encoded using the NGSI normalized format or the key-value simplified format. The geospatial properties of an entity are represented using GeoJSON. The FIWARE data models are specified using a JSON schema.

FIWARE and TM Forum are collaborating on the frame of the FrontRunner project to establish a harmonized data model.

Table 5 summarizes the FIWARE data models currently available that target different applications, including, but not limited, to smart cities. The SynchroniCity consortium is actively contributing to the extension of these data models.

Table 5 – FIWARE data models overview

Vertical	Entity type	Original source
Alert	Alert	FIWARE
Building	Building	GSMA
	BuildingOperation	GSMA
Civic Issue Tracking	Open311:ServiceType	FIWARE / Open311
	Open311:ServiceRequest	FIWARE / Open311
Devices	Device	GSMA – SAREF Extended by FIWARE
	DeviceModel	GSMA – Extended by FIWARE
Energy	ThreePhaseAcMeasurement	FIWARE
Environment	AeroAllergenObserved	GSMA
	AirQualityObserved	GSMA

Table 5 – FIWARE data models overview

Vertical	Entity type	Original source
	WaterQualityObserved	GSMA
	NoiseLevelObserved	FIWARE
Indicators	KeyPerformanceIndicator	FIWARE
Parking	OffStreetParking	FIWARE
	OnStreetParking	FIWARE
	ParkingGroup	FIWARE
	ParkingAccess	FIWARE
	ParkingSpot	FIWARE
Parks & Gardens	Garden	FIWARE
	GreenspaceRecord	FIWARE
	Flowerbed	FIWARE
PointOfInterest	PointOfInterest	GSMA
	Museum	FIWARE
	Beach	FIWARE
	TouristInformationCenter	schema.org
PointOfInteraction	SmartPointOfInteraction	FIWARE
	SmartSpot	FIWARE
Street Lighting	Streetlight	FIWARE
	StreetlightModel	FIWARE
	StreetlightGroup	FIWARE
	StreetlightControlCabinet	FIWARE
Transportation	BikeHireDockingStation	FIWARE
	Road	GSMA
	RoadSegment	GSMA
	TrafficFlowObserved	FIWARE
	CrowdFlowObserved	FIWARE
	Vehicle	GSMA – Extended by FIWARE

Table 5 – FIWARE data models overview

Vertical	Entity type	Original source
	VehicleModel	FIWARE
	EVChargingStation	FIWARE
Urban Mobility	GtfsAgency	FIWARE
	GtfsStop	FIWARE
	GtfsStation	FIWARE
	GtfsAccessPoint	FIWARE
	GtfsRoute	FIWARE
	GtfsTrip	FIWARE
	GtfdStopTime	FIWARE
	GtfsService	FIWARE
	GtfsCalendarRule	FIWARE
	GtfsCalendarDateRule	FIWARE
	GtfsFrequency	FIWARE
	GtfsTransferRule	FIWARE
	GtfsShape	FIWARE
	ArrivalEstimation	FIWARE
Weather	WeatherObserved	GSMA – Extended by FIWARE
	WeatherForecast	GSMA – Extended by FIWARE
	WeatherAlert	GSMA – Extended by FIWARE
Waste Management	WasteContainerIsle	FIWARE
	WasteContainerModel	FIWARE
	WasteContainer	FIWARE

Data model specification

An excerpt of a domain specific data model for weather is represented in Figure 6. The vocabulary of attributes used is specified by the FIWARE data models for weather and related to the entity "WeatherObserved". Such a schema only represents the key-value pairs of the data model.

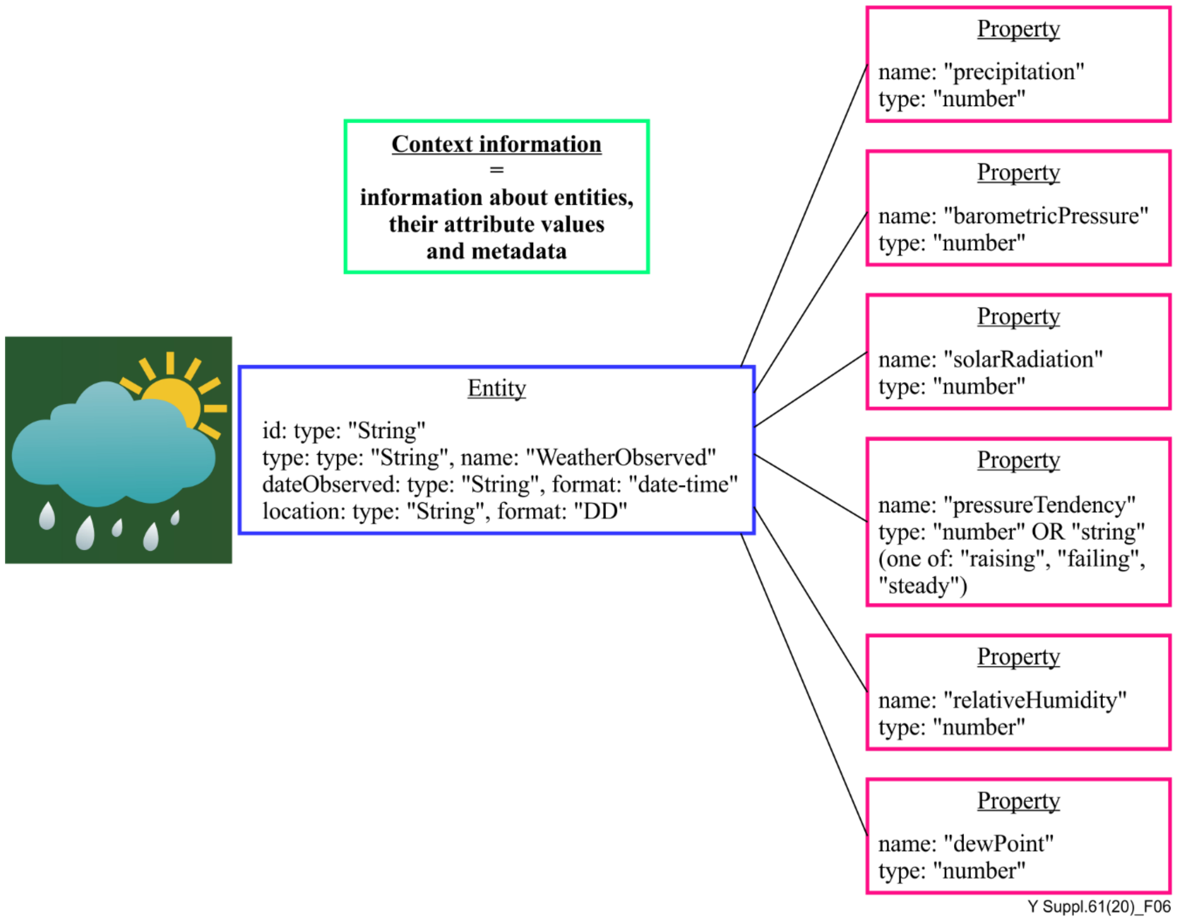


Figure 6 – Representation of an excerpt of a domain-specific data model for weather (FIWARE data model "WeatherObserved")

The same model specified using JSON Schema is reported in Figure 7. Similarly, a schema for the NGSI normalized model could be devised.

First of all, the different NGSI attributes are designated as properties of the JSON object. In addition, there are four elements required in this data model:

– *id*: corresponding to the entity ID (automatically generated by data providers);
– *type*: corresponding to the entity type which shall be equal to "WeatherObserved";
– *dateObserved*: which is the date and time at which the observation was made;
– *location*: a GeoJSON geometry which determines the location associated with the weather observation.

```
{
    "properties": {
      "type": {
        "type": "string",
        "enum": [
          "WeatherObserved"
        ],
        "description": "NGSI Entity type"
      },
      "dateObserved": {
        "type": "string",
        "format": "date-time"
      },
      "precipitation": {
          "type": "number",
          "minimum": 0
        },
        "barometricPressure": {
          "type": "number",
          "minimum": 0
        },
        "solarRadiation": {
          "type": "number",
          "minimum": 0
        },
        "pressureTendency": {
          "oneOf": [
            {
              "type": "string",
              "enum": [
                "raising",
                "falling",
                "steady"
              ]
            },
            {
              "type": "number"
            }
          ]
        },
        "dewPoint": {
          "type": "number"
        }
       },
  "required": [
    "id",
    "type",
    "dateObserved",
    "location"
  ]
}
```

Figure 7 – JSON Schema excerpt (FIWARE data model "WeatherObserved")
[b-dataModel.Weather]

Data model instantiation

An example of instantiations of the data model described above is reported in Figure 8. Apart from the mandatory entity type and ID, there are different attributes and values in accordance with the JSON schema described previously. The format shown is very compact and can be consumed very easily by different applications or services.

```
{
  "id": "Spain-WeatherObserved-Valladolid",
  "type": "WeatherObserved",
  "barometricPressure": 938.9,
  "dataProvider": "SynchroniCity",
  "dateObserved": "2018-11-30T07:00:00.00Z",
  "location": {
    "type": "Point",
    "coordinates": [
      -4.754444444,
      41.640833333
    ]
  },
  "precipitation": 0,
  "pressureTendency": 0.5,
  "relativeHumidity": 1,
  "source": http://www.aemet.es ,
  "temperature": 3.3,
  "windDirection": -45,
  "windSpeed": 2
}
```

Figure 8 – "WeatherObserved" entity instance [b-dataModel.Weatherexample]

10.3 SAREF

The SAREF ontology is conceived as a shared data model to bring together semantics and data from different smart appliances in buildings and households. The appliances covered are: home and building sensors and actuators; white goods (refrigerators, freezers, cookers, washing machines, etc.); heating, ventilation and air conditioning (HVAC); lighting; and micro renewable home solutions (solar panels and other energy-harvesting devices).

SAREF focuses on the concept of device, which is defined as "a tangible object designed to accomplish a particular task in households, common public buildings or offices. In order to accomplish this task, the device performs one or more functions". Examples of devices are a light switch, a temperature sensor, an energy meter and a washing machine. The ontology provides a list of basic functions that can be considered as building blocks to be combined, in order to have more complex functions in a single device. The device, when connected to a network, offers a service, which is the representation of one or more functions to a network, making them discoverable, registerable and remotely controllable by other devices in the network. Figure 9 depicts an overview of the SAREF: Device class, the main class of the ontology and its properties. [b-ETSI SAREF].

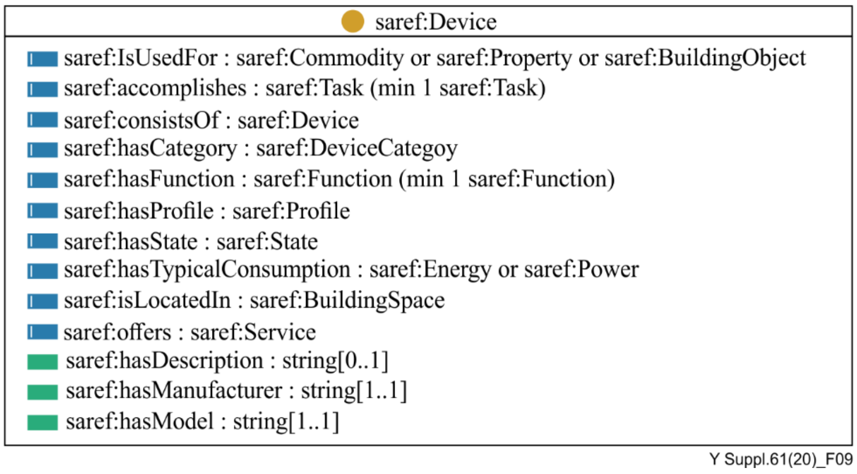

saref:Device
saref:IsUsedFor : saref:Commodity or saref:Property or saref:BuildingObject
saref:accomplishes : saref:Task (min 1 saref:Task)
saref:consistsOf : saref:Device
saref:hasCategory : saref:DeviceCategoy
saref:hasFunction : saref:Function (min 1 saref:Function)
saref:hasProfile : saref:Profile
saref:hasState : saref:State
saref:hasTypicalConsumption : saref:Energy or saref:Power
saref:isLocatedIn : saref:BuildingSpace
saref:offers : saref:Service
saref:hasDescription : string[0..1]
saref:hasManufacturer : string[1..1]
saref:hasModel : string[1..1]
Y Suppl.61(20)_F09

Figure 9 – Device class and its properties

Data model instantiation

The ontology defined by SAREF can be mapped to other ontology and data models and then the instantiation rules of the data model that the SAREF ontology is mapped to can be applied. For instance, to use the SAREF ontology in SynchroniCity, an initial mapping or conversion is needed to flatten the intrinsic triple structure of the SAREF ontology to a structured representation through a JSON schema. In particular, the relationship between concepts is to be mapped as entity properties, with the right semantics and cardinalities defined in the original SAREF ontology. On the other side, SynchroniCity data models can be extended with a new model able to represent the SAREF semantic coverage regarding devices, sensors and their specification in terms of functions and states. This consideration leads to the extension of the Device model (from FIWARE), to include concepts coming from SAREF.

Similarly, SAREF ontology can be mapped to the oneM2M base ontology and used to model oneM2M resources. A two-step approach for the mapping of SAREF instances to oneM2M resources is used. In the first step, key SAREF classes are mapped to oneM2M base ontology classes by defining an "is-a" relation between the SAREF and the oneM2M class. Therefore, instances modelled according to those SAREF classes for which such a definition exists, are also automatically modelled according to the corresponding oneM2M base ontology classes. In the second step, the oneM2M instantiation rules are applied to those instances of SAREF classes that are derived from oneM2M classes.

10.4 INSPIRE

INSPIRE (INfrastructure for Spatial Information in Europe), is the European Union directive that aims to create an infrastructure for sharing environmental spatial data between public authorities of the member States. Each state produces and uses geospatial data, especially in the environment sector, e.g., groundwater, pollution, land-use and population. Each country has its own specific convention to describe data. However, often data might have sources across country borders (e.g., data regarding river flows or mountains) or actions have to be taken very quickly in coordination among different European countries (e.g., to react to critical events like natural disasters). Hence, there is a need to create a common standard representation.

Common data models established in the initiative are called INSPIRE objects and apply to specific environmental applications, with a total number of 338 spatial objects grouped in 34 application data themes. Some themes regarding contexts not strictly related to urban aspects, e.g., geology, elevation and agriculture facilities, will not primarily be taken into consideration in the context of the SynchroniCity project.

Table 6 lists the relevant INSPIRE themes for the objects for each theme. The data models are defined through Unified Modelling Language (UML) class diagrams with also related XML schemas, representing relationships between the different themes, object types, data type classes and their properties.

Table 6 – Relevant INSPIRE themes and objects (data models)

Theme	Description	Relevant objects
Addresses	Fixed location of a property and its components	Address, AddressComponent, GeographicPosition, PostalDescriptor
Administrative Units	Units of administration with jurisdictional rights, separated by administrative boundaries.	AdministrativeUnit, AdministrativeBoundary, Condominium, MaritimeUnits
Cadastral Parcels	Areas defined by cadastral registers or equivalent.	CadastralParcels
Geographical Names	Names of areas, regions, localities, cities, suburbs, or any geographical or topographical feature	GeographicalName
Hydrography	Marine areas and all other related water bodies	Watercourse, Shore, Falls
Transport Networks	Road, rail, air and water transport networks and related infrastructure. Includes links between different networks.	Road, Railway, Air, Cable and Water Transport Network Common Transport Elements (e.g., TransportNode, Link, Area)

Table 6 – Relevant INSPIRE themes and objects (data models)

Theme	Description	Relevant objects
Area Management Restriction and Regulation Zones	Areas managed and regulated under specific restrictions: dumping sites and areas, regulated fairways at sea, noise restriction zones, etc.	ManagementRestrictionOrRegulationZone
Energy Resources	Energy resources including hydrocarbons, hydropower, bio-energy, solar, wind, etc.	FossilFuelValue, RenewableAndWasteValue, VerticalReferenceValue
Environmental Monitoring Facilities	Location and operation of environmental monitoring e.g., observation and measurement of emissions.	EnvironmentalMonitoringActivity, EnvironmentalMonitoringFacility, EnvironmentalMonitoringNetwork, EnvironmentalMonitoringProgramme
Atmospheric and Meteorological Conditions	Physical conditions in the atmosphere: spatial data based on measurements	Observation (Point, MultiPoint, Grid), ObservedProperty
Buildings	Characteristics of buildings and their position	BuildingGeometry2D, BuildingPart
Human Health and Safety	Geographical distribution of dominance of pathologies, the effect on health linked to the quality of the environment.	Disease, NoiseMeasure, Biomarker, Safety (e.g., TrafficRelatedEvent, NaturalHazardRelatedEvent)
Natural Risk Zones	Vulnerable areas characterized according to natural hazards	HazardArea, RiskZone, NaturalHazardClassification
Utility and Governmental Services	Facilities such as sewage, waste management, energy supply and water supply, administrative and social governmental services	GovernmentalService, Utility Networks (e.g., Water, Oil-Gas, Thermal, Sewer, Electricity)

Data model instantiation

Similarly to SAREF ontology, INSPIRE data models can also be mapped to another data model collection and then the instantiation rules of such a collection can be applied.

For instance, INSPIRE data models can be used in SynchroniCity to extend some of its data models with more detailed information: the INSPIRE data models under the "Administrative Unit" theme can be leveraged to define SynchroniCity reference zones (RZs), i.e., municipalities, in a more specialized way than simply using "Points of Interest".

Another relevant example is related to the "Transport Networks" theme: the INSPIRE objects may collide with the SynchroniCity "Urban Mobility" data models. In this case, for colliding concepts, a deeper analysis of new and existing properties would help to identify which properties or relationships can be integrated into the existing models; e.g., *speedLimit* property, in the "Road Transport Network" schema of the "Transport Networks" theme, could be conceptually added to the SynchroniCity GTFS:Route data model (where GTFS is the general transit feed specification).

Possibly, "Atmospheric Conditions and Meteorological Geographical Features" and "Human Health and Safety" themes could extend the SynchroniCity AirQualityObserved and WeatherObserved data models.

10.5 Eurocities

The Eurocities initiative, founded in 1986, is a network of local governments of the major European cities. It brings together the local governments of over 140 cities across 39 countries, offering a platform for sharing knowledge and exchanging ideas in the economic, political, social and cultural domains. Through six thematic forums, a wide range of working groups, projects, activities and events, Eurocities collaborates with EU institutions to respond to common issues that affect the lives of citizens. Therefore, the main focus is to reinforce the important role that local governments should play in a multilevel governance structure; connecting cities enables to exchange common issues and solutions, in order to learn from one another.

In particular, the Data Working Group deals with topics like data management, collection and storage. One of its main objectives is to find equitable solutions between public data and data from private sources, to create "shared urban data schemes". The Standards and Interoperability Working Group instead, is more focused on developing common standards in order to improve the interoperability of smart city infrastructures and maximize the accessibility and usability of data.

[b-Eurocities] highlights the importance of having well agreed common standards and indicates the three European Standardization Organizations (ESOs) responsible for European standards: CEN, CENELEC and ETSI.

10.6 TM Forum

Business API ecosystem

The business API ecosystem generic enabler (GE) is the result of the collaboration between FIWARE and the TM Forum. More specifically, the business API ecosystem GE is a joint component made up by integrating the FIWARE business framework with a set of standard APIs (and its reference implementations) provided by the TM Forum in its TMF API ecosystem. This enabler allows the monetization of assets during the whole service lifecycle, from offering creation to its charging, accounting and revenue settlement and sharing. In this way, the business API ecosystem provides data providers with the means to manage, publish and generate revenue from their data, applications and services.

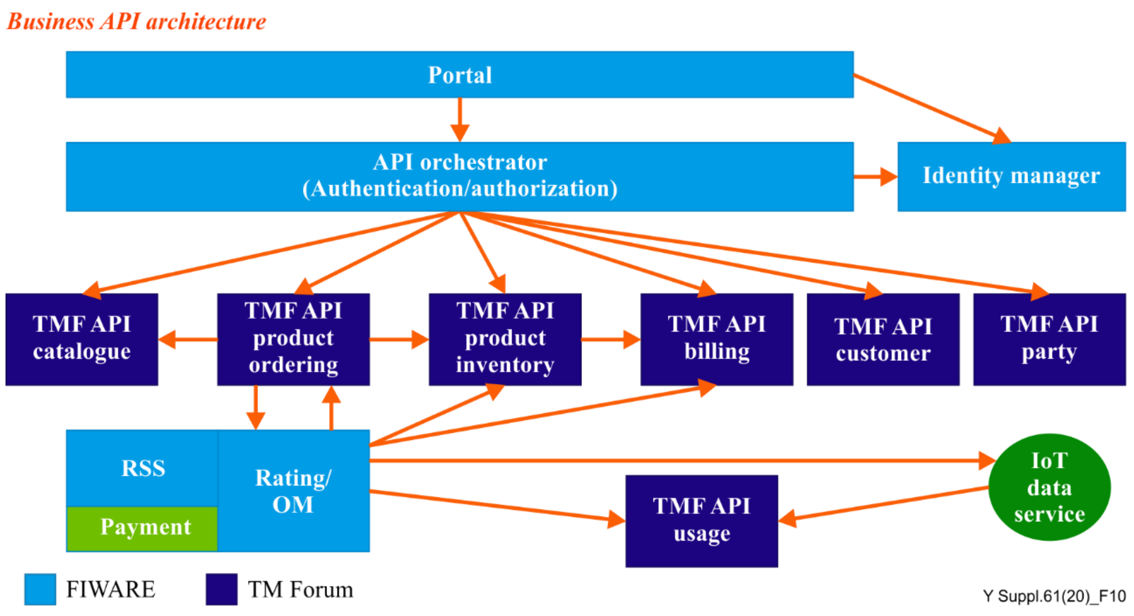


Figure 10 – Business API ecosystem architecture
(Figure 44 of [b-EU SynchroniCity D2.1])

IoT device management API

The IoT device management API is a TM Forum specification that can accommodate multiple protocols and data models including the FIWARE NGSI protocol.

In this context, the IoT device management API provides the mechanisms to manage IoT devices and to access their data. Several partners (TM Forum, IoT Lab, Axiata and Vodafone) have designed the IoT device management API to ease end-to-end IoT device management and so to speed up deployments of IoT devices for communication service providers (CSPs).

Figure 11 shows the global architecture where the IoT device management API is used within the IoT service management API.

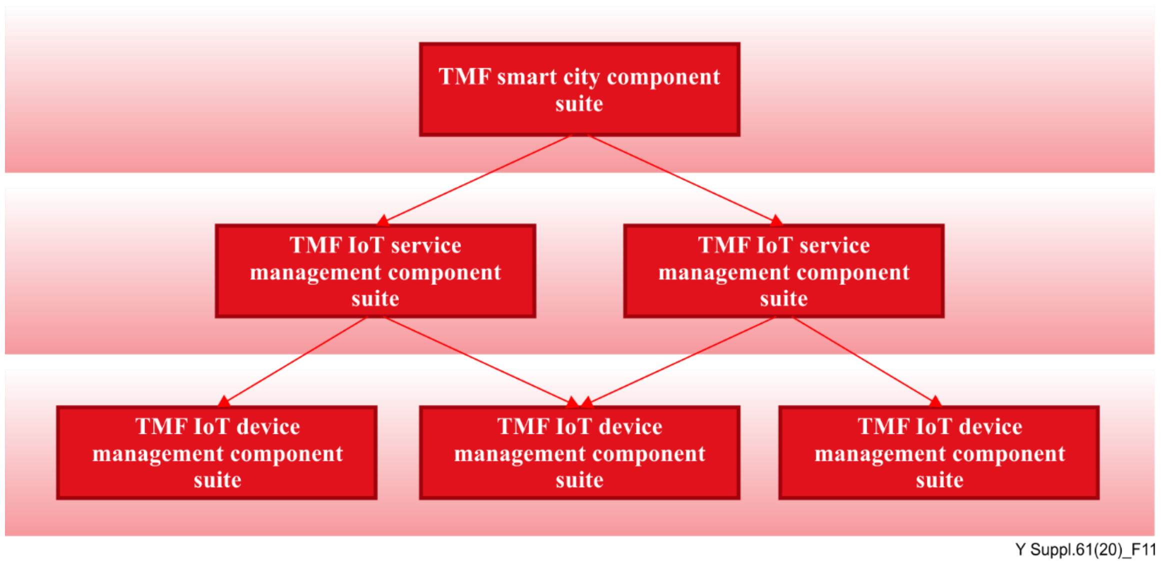

TMF smart city component suite
TMF IoT service management component suite
TMF IoT service management component suite
TMF IoT device management component suite
TMF IoT device management component suite
TMF IoT device management component suite
Y Suppl.61(20)_F11

Figure 11 – TM Forum smart city suite [b-TMForum]

The TMF IoT device management API handles not only IoT devices themselves, but also IoT agents accessing the data generated by the IoT devices. Through this API, the IoT devices can be configured and monitored. The API also permits the alarms coming from the devices to be obtained. The API encompasses the elements of the IoT data access endpoint API also created by TM Forum. The data access endpoint API allows the encapsulation of IoT protocols, like MQTT and NGSI, and then access to the data and events.

IoT Lab has taken part in the elaboration of the IoT device management API. At the same time, the universal device gateway (UDG) is using this open API to publish the IoT devices and their related data. The multiprotocol interoperability among communication protocols at the lower layers is provided by the UDG.

IoT service management API

The TMF IoT service management API builds services based on the data provided by the IoT devices through the IoT device management API. This API is currently in development by TM Forum and IoT Lab, and will be released soon.

FrontRunner – Unified data model

FIWARE and TM Forum are working together on an initiative named FrontRunner which is creating a unified data model. TM Forum and FIWARE are building a unified city data model that is used in the IoT device and service management APIs.

11 Current API solutions

11.1 SynchroniCity – Open and Agile Smart City Alliance API

OASC is a non-profit, international smart city network that has the goal of creating and shaping the nascent global smart city data and services market. OASC is driven by implementation and is focused on open platforms and citizen engagement. OASC involves more than 140 smart cities globally organized in national networks from 27 countries and regions.

OASC establishes and fosters the adoption of minimal interoperability mechanisms (MIMs), which are simple and transparent, ready to use in any city, regardless of size or capacity. By implementing MIMs, cities increase the speed and openness of innovation and development, while decreasing cost and inefficiency. In essence, MIMs allow cities to engage in global digital transformation.

In practice, the OASC MIMs are a set of common (real-time) APIs to access data, context information to structure data, and a common, but optional, data platform to store and serve data. In addition, a reference architecture and a reference implementation complete the set of MIMs.

The OASC MIMs [b-OASC] are an evolving set of technical mechanisms selected from a baseline of global best-practice, driven by implementation in the member cities of the network, and feeding into standardization activities such as ETSI, ISO, and ITU. See Figure 12.

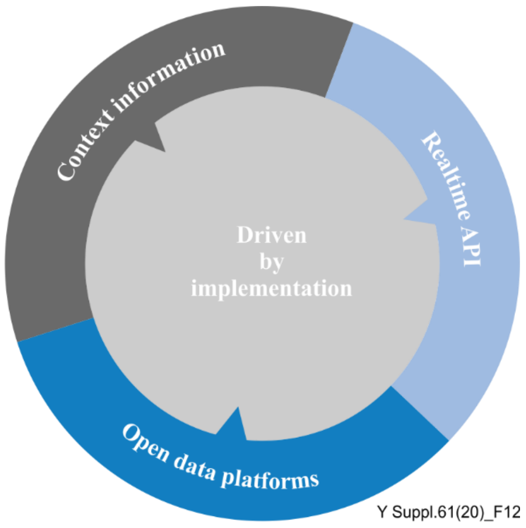


Figure 12 – OASC MIMs [b-OASC]

The first version of SynchroniCity logical reference architecture, depicted in Figure 13 [b-EU SynchroniCity D2.5], is the composition of different logical modules that are summarized as follows.

– **Context data management** (CDM) manages the context information coming from IoT devices and other public and private data sources, providing a uniform approach and interface. Context information contains status information about real world entities defined in a structured way. CDM provides functionalities to enable access to different data sources and analyse context information, e.g., for detecting events.

– **IoT management** is the module responsible for the interaction, through specific IoT agents, with devices that use different standards or protocols, making them compatible and available to the SynchroniCity platform.

– **Data storage management** provides functionalities related to data storage and security, as well as quality in the specific context of IoT systems and smart city platform interaction with heterogeneous sources.

– **Marketplace and asset management** supports business interactions between suppliers of valuable digital assets (i.e., IoT data or services) that are part of the SynchroniCity ecosystem and consumers. It will implement a hub to enable digital data exchange for urban data and IoT capabilities providing features in order to manage asset catalogues, orders and revenue management. These functions will support the creation of innovative business models.

– **Security, privacy and governance** is a module that covers all security aspects related to three main pillars: data; IoT infrastructure; and the platform services, which underpin the applications and services of cities. Around these pillars, security functionalities provide crucial security properties, e.g., confidentiality, authentication, authorization, integrity, non-repudiation and access control.

– **Monitoring and platform management services** provide functionalities to manage platform configuration and to monitor activities of platform services. They support specific key performance indicator definition to evaluate the status of the platform in relation to different aspects (e.g., performance, usage, reliability and quality of service)

Figure 13 also shows two layers connected with the overall platform as follows.

– **Southbound interfaces** represent the set of interfaces defined by SynchroniCity used to connect the overall platform to heterogeneous data sources and IoT devices. This represents one of the relevant interoperability points that should be implemented by the RZs to be part of the SynchroniCity ecosystem.

– **Northbound interfaces** make up the set of APIs that provides all platform functionalities that will be used by the final smart city end-user applications. Also, this layer can be considered an interoperability point, because it is the main way, for external applications, to interact with the platform and to be part of the digital single market that will be technically enabled by SynchroniCity.

In the upper part of the architecture are shown end-user smart city applications and services (examples based on some of the pilot and open call application domains) that will rely on SynchroniCity functionalities and data: in particular, using the same API and data models, the applications will be able to work and interact with the different RZs, part of the SynchroniCity ecosystem, with minimum customization, simplifying the replications process. These services and applications are also digital assets that can be managed inside the marketplace in order to be shared and monetized.

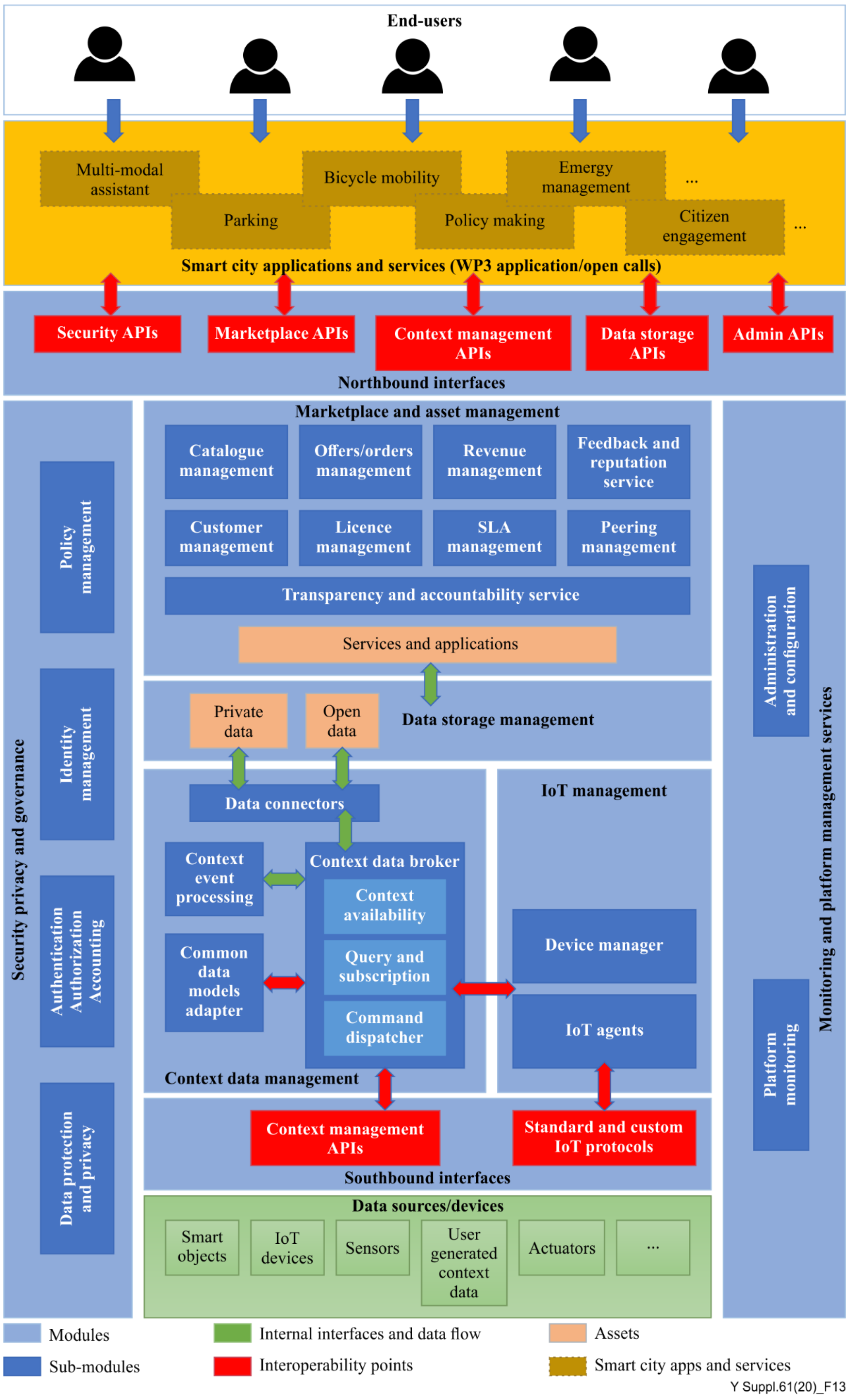


Figure 13 – SynchroniCity architecture (Figure 3 of [b-EU SynchroniCity D2.1])

SynchroniCity API

In the following text are presented some relevant APIs that the SynchroniCity platform will provide as southbound (for data sources, device and existing IoT platform connections) and northbound, to be accessed not only by end-user applications, but also other systems or stakeholders that want to access platform functionalities. A logical description of this API is included in [b-EU SynchroniCity D2.1] (reported in the following text).

Southbound: Context management API and IoT protocols

To ensure interoperability in this domain, SynchroniCity framework establishes context management APIs that represent a standard way to communicate with the context data management module, in particular with the context data broker (CDB).

A CDB enables the discovery, gathering and publication of context information through APIs. The CDB, through its standard interface, makes available the context information regardless of data source and using different type of interactions. Its main functionalities are as follows.

– *Context availability* represents the operations to identify which context data sources are connected to the SynchroniCity platform.

– *Query and subscription* represents the synchronous and asynchronous interactions with a context data source. Synchronous interactions are performed using a query mechanism to obtain context information; the component builds powerful queries, using different types of filters, in order to retrieve information with a high level of precision. The asynchronous interaction is performed by a publish-subscribe mechanism: a notification is generated when published data meets the subscription conditions; this feature is useful to avoid the implementation of a polling process on data sources of interest, allowing the user to be notified when the context information changes.

– *Command dispatcher* is the function through which the CDB acts as an input channel for an IoT device that is able to receive commands from an external system.

The main concept behind the management of data and information in SynchroniCity is the adoption of an abstract level that bases representation of information on the model of the context entity. Each data source is represented as an entity characterized by its attributes and metadata.

The logical model of these interfaces is inspired by the NGSI standard, which offers a powerful way to model the real world, in order to extract high quality information from observation of phenomena. In terms of the IoT system, each device or data source can be mapped to an NGSI model and analysed by using different modalities. Figure 14 depicts how real-life objects can be mapped using NGSI approach.

This approach is coherent with the interoperability requirements of the SynchroniCity platform and is not restricted to the use of a predetermined data model. In terms of exchanging of data or commands, it is possible to distinguish two interoperability layers, northbound and southbound. Considering the interoperability aspects from the southbound connection point of view, being compliant with SynchroniCity means that the RZs have to be able to communicate with the context management module using its APIs.

To achieve this objective RZs have different options (see Figure 14) [b-EU SynchroniCity D2.1] as follows.

– RZs implement directly the context management APIs: this is the case that ensures the highest level of compatibility because the RZ is, from a technical point of view, ready to interact with SynchroniCity.

– RZs adopt standards and well-known IoT protocols in their infrastructure: in this case, the presence of IoT agents supporting these standards, already provided by SynchroniCity, enables the integration between the RZ and the SynchroniCity platform.

– RZs adopt *ad hoc* solutions and proprietary protocols in its infrastructure: in this case, to achieve integration into the SynchroniCity platform, a custom IoT agent should be developed in order to adapt the custom RZ technologies and context management interfaces.

These three options allow communication with the context management component, which is the basic condition to be part of the SynchroniCity platform. Following these approaches, it will be possible to realize a southbound integration between the platform and the existing and legacy technologies of the RZs. Considering the interaction issues from the northbound side, not only do RZs have to be compliant with specifications, but also applications providers have to take into account the interoperability between the applications created and the SynchroniCity platform. Therefore, from this point of view the platform exposes, at northbound, context management APIs to allow external applications to manage context entities and their attributes by using the SynchroniCity framework.

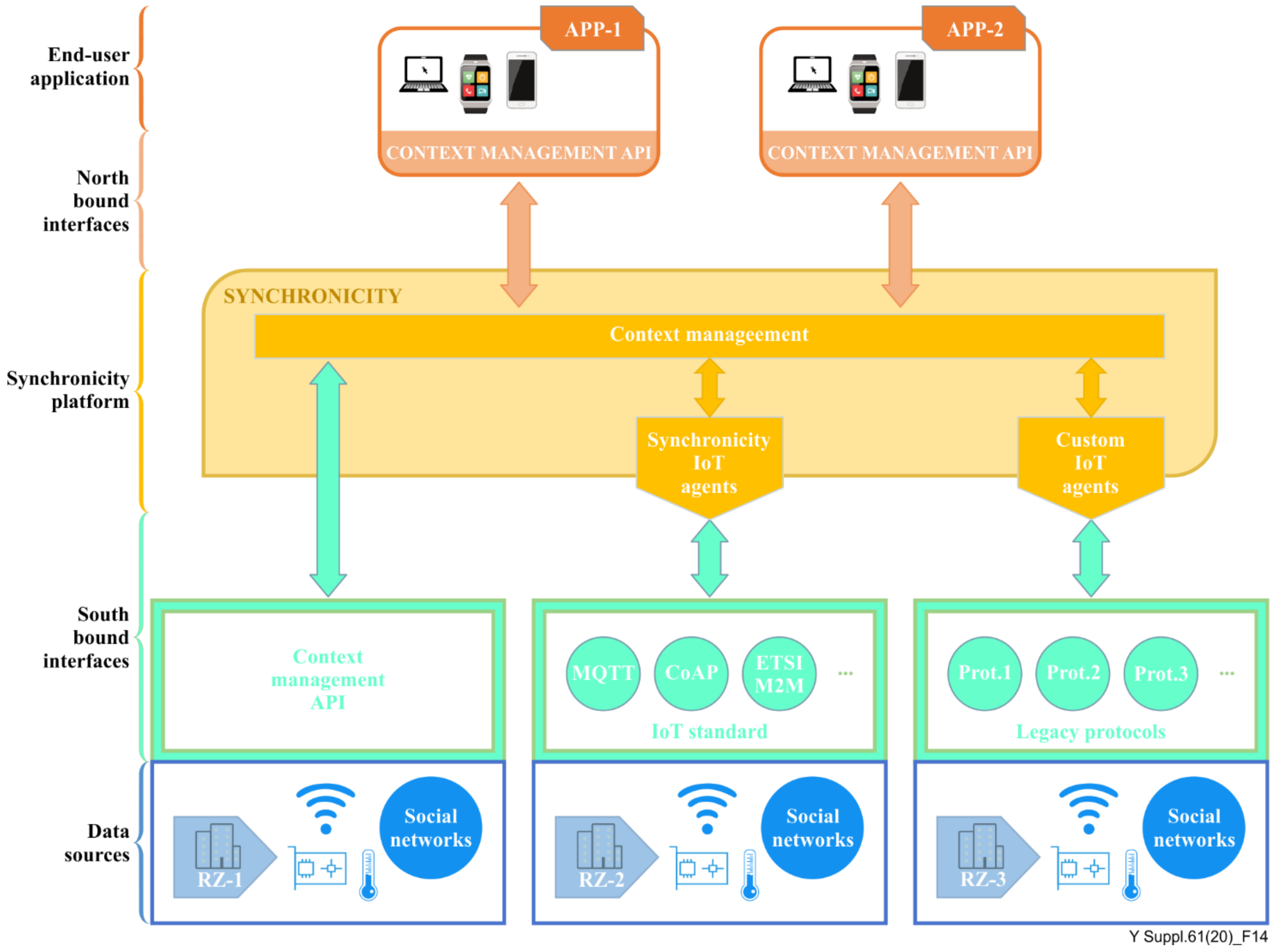


Figure 14 – Three different solutions to enable context management interoperability
(Figure 32 of [b-EU SynchroniCity D2.1])

For the first reference implementation of the SynchroniCity it was decided to use FIWARE-NGSI v2 [b-FIWARE-NGSI] specification as context management APIs.

Regarding the possibility to natively support IoT protocols to be directly connected with SynchroniCity during the project, some specific adapter will be provided or developed by technical partners based on the needs of the cities.

As mentioned before, the context management APIs are also provided at northbound level.

Northbound: Marketplace APIs

SynchroniCity provides a set of APIs aimed to lower the barriers for data sharing, while increasing the trust of data consumers and providers. Different stakeholders are incentivized to share valuable proprietary data. The marketplace APIs (see Figure 15) provide a hub to enable digital asset exchange for urban data and IoT capabilities.

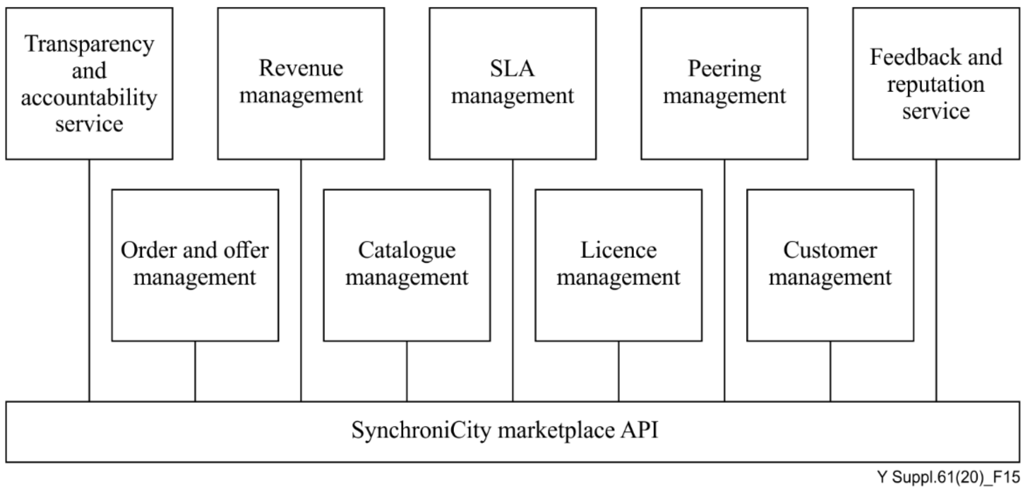

Transparency and accountability service
Revenue management
SLA management
Peering management
Feedback and reputation service
Order and offer management
Catalogue management
Licence management
Customer management
SynchroniCity marketplace API
Y Suppl.61(20)_F15

Figure 15 – SynchroniCity marketplace APIs
(Figure 34 of [b-EU SynchroniCity D2.1])

More specifically, the SynchroniCity marketplace interface consists of the following set of APIs.

Catalogue management API provides methods for the discovery of assets with different search filters, methods, and for publishing new asset offers and requests. A list of available assets can be retrieved and refined by specifying keywords and filters that match the description, characteristics and properties of the desired assets. As a result, asset consumers and cities can easily discover what kinds of assets are available in the market. The creation of a new offer for an asset requires an asset description, characteristics and properties (e.g., scope, purpose and metadata). Moreover, the asset will need to be linked to a usage licence and an SLA that may be selected among predefined ones or customized by the asset provider. Once a new asset offer has been completely determined, it will be embedded in the catalogue. On the other hand, assets not currently available in the catalogue can be requested by creating a new asset request. Similarly to the asset offering, a new asset request can be created by providing an asset description, characteristics and properties.

Offer and order management API provides methods to manage asset offerings and orders. The asset provider can use this API to perform several operations, e.g., update asset description, licence, SLA and price, or removal of an asset offer. From the asset consumer side, this API provides tools to place and manage the order. More specifically, an asset consumer interested in acquiring or purchasing an asset available in the catalogue can place an order to finalize the acquisition of that asset. If an asset consumer needs to manage an order, the API performs operations, e.g., un-subscribing, activation, deactivation and renewal.

Peering management API provides methods for the management of federation requests and for the management of federation offers. This API is accessible only to the marketplace administrator. It exposes specific assets that could potentially be part of a federated marketplace. By defining the set of assets to expose, according to the marketplace governance, and the set of market-place with which to federate, the federation offering handler will send federation requests to the specified marketplaces. On the other hand, the federation request handler manages the federation request. In this case, the API retrieves request embedding information regarding assets proposed to be part of a federation. The result of accepting a federation request will be the inclusion of federated assets into the local asset catalogue. Similarly, other functionalities, e.g., feedback and reputation, transparency and accountability or revenue management, can be federated so that different RZs can share the same information and be part of a digital single market.

Revenue management API provides methods to manage asset usage information, in order to enable usage-based business models, manage billing, charging and share revenues. It exposes an interface to interact with external charging platforms such as PayPal. It collects all the information required for charging (price, data usage, consumer ID, etc.), which may differ based on the business model associated with the asset order (e.g., pay per use or subscription) and based on the outcome received by the external charging platform, creates a transaction embedding funds transferred from a consumer to a provider or a set of providers if a revenue share mechanism has been specified for the specific asset. More specifically, the asset provider can enable a revenue-sharing mechanism among other stakeholders by specifying for which asset offer and with whom revenues are shared for that particular asset. The API also produces an invoice for each transaction.

Feedback and reputation service API provides methods for creating, updating and deleting user feedback to rate assets and manage customer reputation. User feedback can include a text form or a rate related to the asset being evaluated and can be issued only by asset consumers that placed an order for the evaluated asset. The outcome of rating is linked to the respective asset and is visible within the catalogue. Customer reputation is fairly and transparently updated based on several factors, e.g., proof of SLA and usage licence fulfilment, as well as overall rate of the assets provided in the marketplace.

Customer management API provides methods for the registration, retrieval update and deletion of customer accounts and for the management of user consent. Depending on the access restrictions to the marketplace specified by the marketplace provider (e.g., city council, consortium, third party), customer accounts can be created and linked to a specific role (e.g., asset provider, asset consumer or administrator). This API also provides an interface to create, retrieve, update and delete user consent so that the desired result can be enforced by the privacy management subcomponent.

Licence management provides methods for the definition and customization of data usage licences. More specifically, it provides an interface to retrieve a pre-defined data licence template so that asset providers can link a data usage licence instance selected among the available templates to the offered asset. If the licence template does not fulfil the asset provider needs, this API customizes available licence templates or creates new ones so that the new licence can better reflect the business model requirements.

SLA management API provides methods for the definition of SLAs. Similarly to the licence management API, it exposes an interface that creates, retrieves, updates and deletes an SLA.

Transparency and accountability service API provides methods for the audit of transactions and user consent. The API takes a transaction ID and related information, which it will process by creating and storing a digest. Similarly, whenever user consent is given to this API along with the user ID, a digest is created and stored.

Northbound: Security APIs

SynchroniCity provides a set of APIs aimed to fulfil the security and privacy requirements of each RZ (see Figure 16). The SynchroniCity security architecture is built on the OASIS security standard (e.g., XACML), oneM2M, GSMA and ENISA IoT security guidelines following a modular design by splitting it into several components and subcomponents to be consistent with SynchroniCity overall architectural and security requirements. It provides a unified approach to the management of security policies as a viable and scalable means to establish and enforce security rules consistently among the large variety of accessible resources (e.g., IoT devices, data and services). SynchroniCity access control is primarily based on an attribute-based access control (ABAC) system. Access rights are granted to users through policies combining attributes such as users, resources, actions and objects. However, different access control models can also be supported (e.g., role-based access control). With respect to the standard IdM, authentication, authorization and accounting components that reflect the security standard and guidelines previously mentioned, SynchroniCity policy management is decoupled from devices and services. Policy can be managed independently, thus focusing on providing business value and compliance to data protection regulations. Key management, encryption, digital signature and data anonymization functionalities are directly linked to resources and governed by the policy management so that implementation of changes and enforcement are simplified by deploying policies on the fly affecting each point of use immediately.

The access requester performs an authentication request by interacting with the identity and authentication management API, which will respond with a session token upon authorization thus concluding the login phase. The authorization PEP proxy subcomponent is a consumer of session tokens, therefore it validates tokens while getting authorization information from the token, such as the user role; it is also a client of the authorization PDP since it intercepts resource access requests that are then forwarded to the authorization PDP to get an authorization decision (permit or deny). The PDP evaluates the access request by checking authorization policies from the PRP, which is the connector between the authorization module and the policy management module. It also considers other information, such as the status of the session token by interacting with the PIP. Depending on this decision, the PEP proxy blocks or forwards the request and sends information into the accounting subcomponent. The data protection and privacy module enforces the security measures, such as encryption and anonymization, established by the policy management module, directly on the resource or asset (e.g., data, service or application).

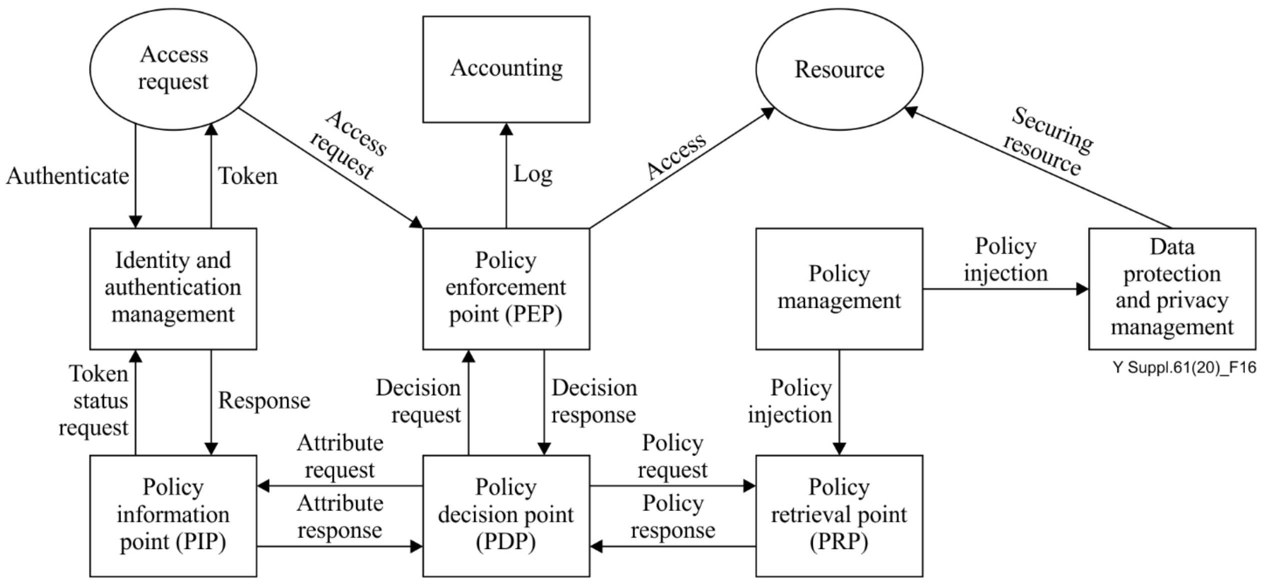


Figure 16 – Security components diagram
(Figure 35 of [b-EU SynchroniCity D2.1])

Identity and authentication management API exposes interfaces to create, import, retrieve, update, delete users and roles, and to perform authentication. More specifically, the API creates a user profile by taking as input username, optional information about the user (e.g., contact information, description and purpose), password and role, returning as output the user ID upon successful creation. Similarly, user import can parse user information from an external source. Retrieval, update and deletion require the user ID to be passed to the API, specifying which operation has to be performed and the respective additional information. In this case, the API will return the outcome of the operation (e.g., success, fail). Moreover, this API performs authentication by receiving user credentials, interacting with the authorization component and issuing authentication tokens.

Authorization and accounting management API exposes interfaces to grant or deny permission to access resources and to log access requests. It exposes interfaces to the authentication component and the policy management component. More specifically, it receives access requests from the authentication component and according to the policies received by the policy management component, it grants or denies access to a particular resource. The accounting interface receives access requests and store them.

Policy management API provides an interface to create, retrieve, update and delete policies. Each policy embeds a title, a description, rules and subjects to those rules. When a new policy is created, the API returns a policy ID, while when updated or deleted it returns the outcome of the operation (e.g., success or fail). This API retrieves policies by passing the policy ID or by passing the subject. In the former case, the API will return the policy associated to the policy ID passed, while in the latter case it will return a list of policies matching the subject passed.

Data protection and privacy API provides interfaces for confidentiality, authentication, integrity, non-repudiation and privacy capabilities. More specifically, it exposes interfaces for data encryption, which takes as input the plaintext data, the encryption key, the encryption algorithm and other information related to the specific algorithm selected (e.g., initialization vector, mode) and returns the encrypted data. Similarly, the decryption interface will take as input the encrypted data, the decryption key, the decryption algorithm and other information related to the specific algorithm selected (e.g., initialization vector, mode) returning the plaintext data. It also exposes an interface for digital signature, which takes as input the data to authenticate, the private key and returns the digital signature. Data anonymization is provided by specifying the data to be anonymized, the algorithm to use and other information related to the specific algorithm selected and returns an anonymized version of the original data. It also exposes an interface to allow update, deletion and notification regarding processing of personal data.

11.2 Sii-Mobility

The Sii-Mobility project started from the observation that terrestrial transportation systems are often affected by congestion since specific situations cannot be easily foreseen by using traditional intelligent transport systems (ITSs) that only cope with transport data and mobility aspects, disregarding events, energy, weather, people flow, etc. Local transportation systems present very high social costs related to the uneasiness of citizens with respect to the available mobility solutions, which are not sustainable and poorly efficient. This is mainly due to the low level of interoperability and intelligence among management and monitoring transport systems, services for mobility, services and systems for goods transportation, ordinances and public services (such as hospital, centres, museums), events, private transport, rail transport, car parks, and moving people, because of the limited capacity of the system to incorporate and react to changes in the city and citizens. With the aim of producing a smart city infrastructure for solving these problems, Sii-Mobility worked on enabling technologies for the smart city and mobility, to integrate and produce support of integrated interoperability (SII). SII aims to create a smart city big data framework for providing integrated data, supporting services such as a mobile app, decision support, integrated ticketing, personal assistants, participative portals, crowd sourcing and dashboarding, via a smart city API. SII also aims to support data analytics and data intelligence based on integrated data collected from public administration open data, private data from operators and personal data from social media and city users. See Figure 17.

The main objectives of Sii-Mobility are to:

1) reduce the social costs of mobility;
2) simplify the use of mobility systems;
3) provide solutions for assisting connected drivers or people exploiting intermodality (connect drive, smart drive or walk);
4) establish solutions for the interoperability with other smart city management systems;
5) establish solutions for city user engagement and participation and awareness, personalized management of access policies;
6) study and propose solutions for dynamic management of restricted traffic zone (RTZ) boundaries;
7) real-time monitoring of supply and demand of public transport.

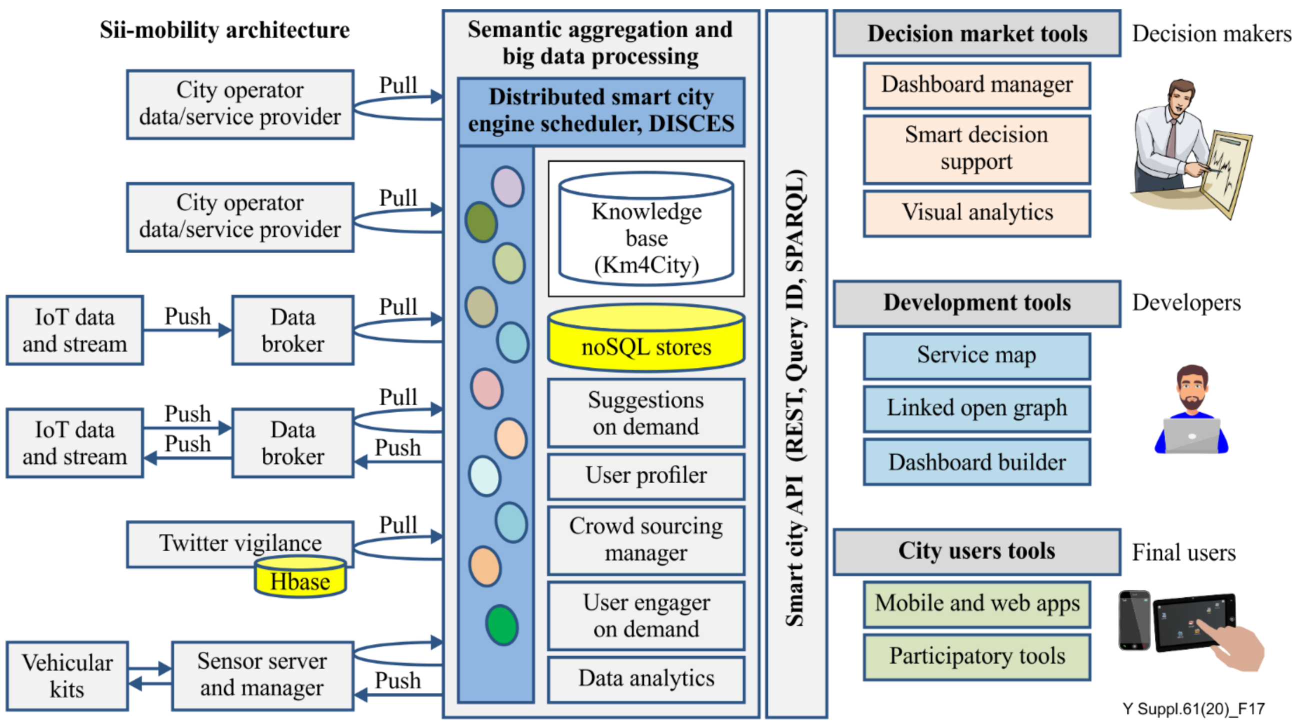


Figure 17 – Sii-Mobility architecture in line with the smart city concept [b-Badii]

The Sii-Mobility architecture collects data from the following.

– City operators and data brokers pull data by using extract, transform and load (ETL) processes that are scheduled on the big data processing back office based on the distributed smart city engine scheduler (DISCES) tool developed for Sii-Mobility and made open source. Among the data collected are those provided as open data from the municipalities, Tuscany region (Observatory of Mobility), the LaMMA weather agency, Agenzia regionale per la protezione ambientale della Toscana. [Tuscany Regional Environmental Protection Agency] (ARPAT), etc., and several sources of private data from city or regionals operators: mobility, energy, health, cultural heritage, services, tourism, wine and food services, education, wellness, environment, civil protection and weather forecasting.

– The sensor server and manager is a specific data broker to collect and manage data coming from vehicular kits developed in Sii-Mobility for monitoring and informing car, bus and bike drivers; as well as mobile devices or apps. They can be directly used with mobile devices to provide several functionalities of the Sii-Mobility solution, e.g., recommendations, navigation and assistance.

– The Twitter Vigilance platform regarding the monitoring of some Twitter.com channels and users related to city mobility and traffic.

– Mobile phones and other devices connected to Sii-Mobility servers: In this case, the data collected from mobiles relate to: (i) the position, velocity and acceleration of city users; (ii) requests to the smart city API such as search queries; and (iii) requests for recommendations intended for city users.

The smart city APIs provided by Sii-Mobility have the possibility of posing requests, by using different modalities:

– SPARQL query: calls are directly performed on the RDF store endpoint using the standard SPARQL query protocol (based on REST) [b-Ong] using GET or POST requests with the query parameter containing the SPARQL query;

– SPARQL query with inference: calls are directly performed on the RDF store endpoint, using the standard SPARQL query protocol (which uses RESTful GET or POST requests containing a SPARQL query), including inference aspects in the case of Virtuoso, or automatically exploiting the inference in the case of OWLIM.

– REST: calls are performed by using APIs using full text, keywords, service ID (uniform resource identifier (URI)) to get information, geolocation, service ID (URI) to get closer services, time, etc.;
– Query ID: calls are performed by using a QueryID (identification) assigned by the ServiceMap tool manager, after having performed a query by using the graphic user interface, as a visual query.

Idra

Idra is a web application able to federate existing ODMSs based on different technologies providing a unique access point to search and discover open datasets coming from heterogeneous sources. Idra aligns representation of collected open datasets, thanks to the adoption of international standards (Data Catalogue Vocabulary-Application Profile (DCAT-AP)) and provides a set of RESTful APIs to be used by third party applications [b-Klímek].

Idra supports natively ODMS based on CKAN, DKAN, Socrata, Orion Context Broker and many other technologies: Idra also provides a set of APIs to federate ODMS not natively supported. In addition, it is possible to federate generic open data portals, that do not expose API, using the web scraping functionality or providing a dump file of the datasets in DCAT-AP. Furthermore, Idra provides a SPARQL endpoint in order to perform queries on five-star RDF linked open data collected from federated ODMS and easily creates charts based on federated open datasets.

11.3 FIWARE

FIWARE provides an OpenStack-based cloud environment plus a set of open standard APIs to connect to the IoT, process and analyse big data and real-time media or incorporate advanced features for user interaction. FIWARE provides means to produce, gather, publish and consume context information on a large scale and exploit it make applications smart.

FIWARE introduces the concept of GEs to offer a number of general-purpose functions, offered through well-defined APIs, to facilitate development of smart applications in multiple sectors. These GEs set the foundations of the architecture associated with the application to be developed by smart city systems designers or developers. In addition to GEs, FIWARE also introduces the concept of domain-specific enablers (DSEs), which when combined with GEs provide applications focused on specific verticals in the domains of energy, creative media, smart manufacturing, health and wellbeing, and the agri-food sector.

Context information is represented through values assigned to attributes that characterize those entities relevant to the application. A context broker (Orion is an open source reference implementation) is able to handle context information at large scale by implementing standard REST APIs. See Figure 18.

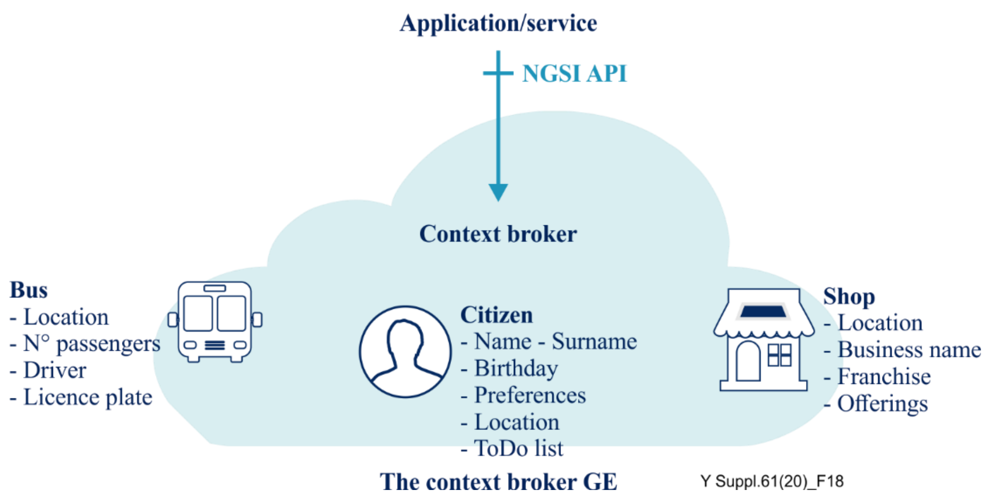


Figure 18 – Application of NGSI API
(Figure 37 of [b-EU SynchroniCity D2.1])

Context information may come from many different sources:

– already existing systems;

– users, through mobile apps;

– sensor networks.

The context broker GE models and gains access to context information in a way that is independent from the source of that information. As an example, an application may need to be aware about "Places" (identified by their specific geographical coordinates) and the "temperature" in them. Regardless of the way in which the temperature of a given place was obtained (e.g., measurements in a given street can be taken by stationary sensors, while in another they may be obtained from those deployed on circulating buses), using the REST API exported by the context broker GE, the application will always query the temperature of places or subscribe to changing values of that parameter in the same manner. See Figure 19.

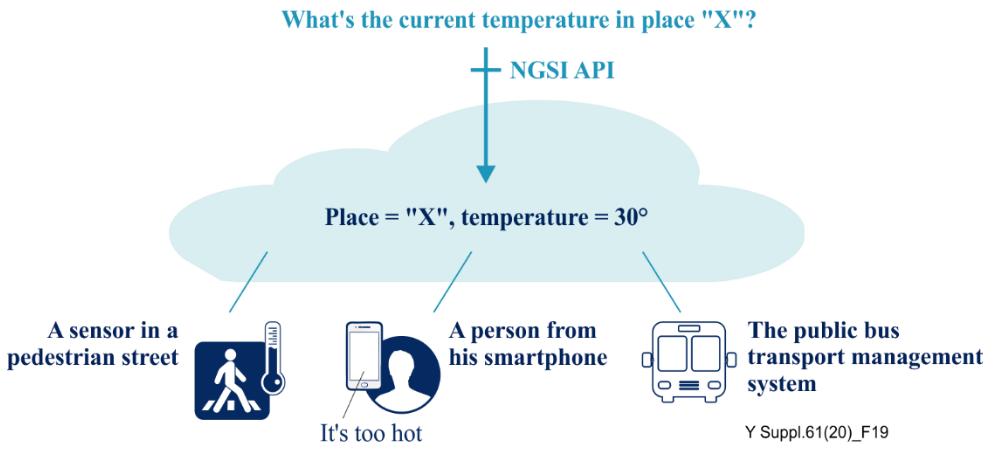

What's the current temperature in place "X"?
NGSI API
Place = "X", temperature = 30°
A sensor in a pedestrian street
A person from his smartphone
It's too hot
The public bus transport management system
Y Suppl.61(20)_F19

Figure 19 – Use-case of NGSI API
(Source: [b-FIWARE-Temperature])

Processes running as part of the application architecture that update context information using REST operations that the context broker GE exports, are said to play a "context producer" role. On the other hand, processes running as part of the application architecture that queries context information using REST operations exported by the context broker GE are said to play a "context consumer" role.

NGSI-LD

The NGSI-LD context information management API, specified by the ETSI Industry Specification Group for Context Information Management (ISG CIM), can be considered an evolution of the OMA/FIWARE-NGSI v2 for linked data. NGSI-LD provides a scalable solution to connect, publish and federate diverse data sources (not only IoT data sources). The specification determines how such an API enables applications to perform updates on context, register context providers that can be queried to get updates on context, query information on current and historical context information and subscribe to receive notifications of context changes. The adoption of NGSI-LD, context information exchange, has different advantages: the applications can flexibly discover and query relevant information; the NGSI-LD API determines the meaning of the most commonly needed terms and provides the tools to create domain-specific extensions to model any other type of information.

The context information management framework defined by NGSI-LD includes an information model that prescribes the structure of context information that shall be supported by an NGSI-LD system. It specifies the data representation mechanisms that shall be used by the NGSI-LD API itself. In addition, it specifies the structure of the context information management vocabularies to be used in conjunction with the API.

The NGSI-LD information model is specified at two levels: the foundation classes, which correspond to the core meta-model, and the cross-domain ontology. The former amounts to a formal specification of the property graph model. The latter is a set of generic, transversal classes that aim to avoid conflicting or redundant specifications of the same classes in each of the domain-specific ontologies.

See Figure 20.

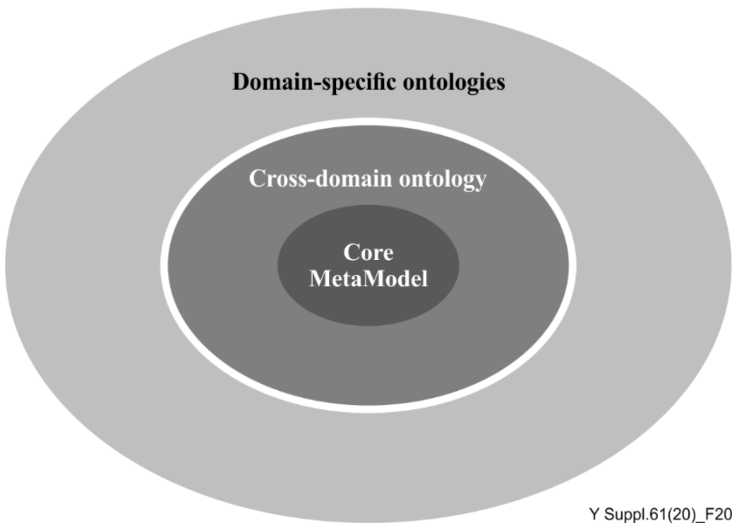


Figure 20 – NGSI-LD model
(adapted from Figure 5 of [b-ETSI WP 31])

11.3.1 Internet of things service enablement

Connecting objects or things involves the need to overcome a set of challenges arising in the different layers of the communication model. Using its data or acting upon them requires interaction with a heterogeneous environment of devices running different protocols (due to the lack of globally accepted standards), dispersed and accessible through multiple wireless technologies.

They are resource constrained and cannot use full standard protocol stacks: they cannot transmit information too frequently due to battery drainage; they are not always reachable since they are connected through heterogeneous wireless networks; their communication protocols are too specific and lack integrated approach; and they use different data encoding languages, so it is tricky to find a global deployment.

There are two IoT typical use-case scenarios (combination of IoT GEs) described in the FIWARE IoT architecture:

– **IoT backend** comprises the set of functions, logical resources and services hosted in a cloud data centre. Up north, it is connected to the data chapter context broker, so IoT resources are translated into NGSI context entities. South-wise, the IoT backend is connected to the IoT edge elements, which is all the physical IoT infrastructure.

– **IoT edge** is made up of all on-field IoT infrastructure elements needed to connect physical devices to FIWARE apps. Typically, it comprises IoT end-nodes, IoT gateways and IoT networks (connectivity). The IoT edge and its related APIs will facilitate the integration of new types of gateways and devices that are under development in many innovative research projects, and guarantee the openness of FIWARE IoT architecture.

Two IoT use case scenarios are provided in FIWARE, the more common and tested common simple scenario is depicted in Figure 21.

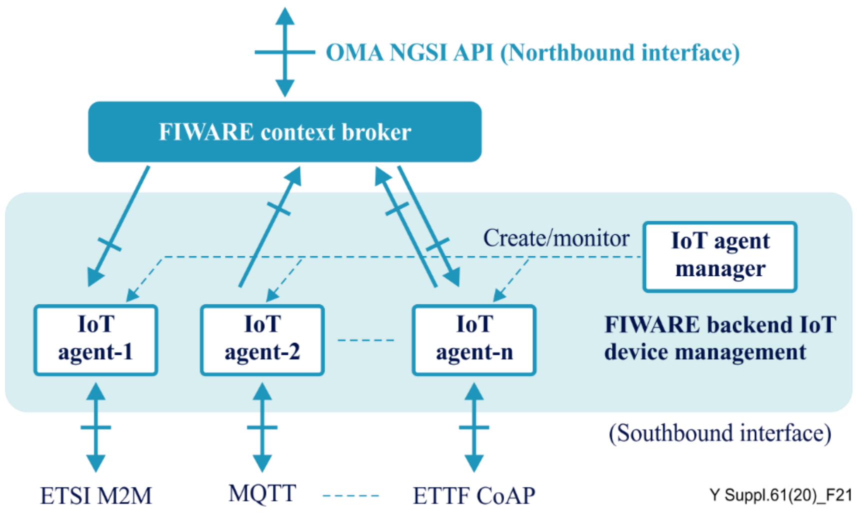


Figure 21 – FIWARE IoT device management GE architecture
(Source: [b-FIWARE-Device])

– **Context broker** remains as the main front-end for developers. Developers access IoT data as attributes of entities representing devices and may also send commands to devices by updating command-related attributes, providing they have access rights for that operation.

– **IoT agents** stay at the southbound of the context broker and are used by IoT integrators to connect devices in this scenario. IoT agents support several IoT protocols with a modular architecture. Therefore, integrators need to determine first which protocol they will be using to connect devices and then select the right IoT agent. The TM Forum IoT device management API is able to take this role.

11.3.2 Data or context management GE

The data or context management GE is used to access, gather, process, publish and analyse context information on a large scale.

Orion context broker

The Orion context broker is an implementation of the publish/subscribe context broker GE, which follows the NGSI 9 and NGSI 10 interface specifications in the first version. Using these interfaces, clients can do several operations:

– register context producer applications, e.g., a temperature sensor within a room;

– update context information, e.g., send updates of temperature;

– be notified when changes in context information take place (e.g., the temperature has changed) or with a given frequency (e.g., get the temperature each minute);

– query context information – the Orion context broker stores context information updated from applications, so queries are resolved based on that information.

Orion is a C++ implementation of the NGSI 9/10 REST API binding developed as a part of the FIWARE platform. A new version supporting FIWARE-NGSI v2 has now appeared.

The Orion context broker allows users to manage the entire lifecycle of context information including updates, queries, registrations and subscriptions. It is an FIWARE-NGSI v2 server implementation to manage context information and its availability. Using the Orion context broker, the user is able to register context elements and manage them through updates and queries. In addition, users can subscribe to context information so when some condition occurs (e.g., the context elements have changed) they receive a notification. Context information is represented through values assigned to attributes that characterize those entities relevant to your application. The context broker is able to handle context information on a large scale by implementing standard REST APIs.

In 2019, the Orion context broker evolved to be compliant with the new ETSI ISG CIM specification known as NGSI-LD.

11.3.3 Advanced web-based user interface

For web-based user interfaces, there are some possibilities to use software capable of handling three dimensional (3D) and augmented reality (AR). For example, WebTundra available in the FIWARE catalogue is basically a web client displaying 3D virtual worlds on a web browser. The rendering is made through WebGL. The information provided by the other FIWARE components permits the generatiojn of 3D applications through JavaScript source code.

Similarly, the data coming from an Orion context broker or from a database used for storage can be used in web user interfaces as inputs for AR. So, new interactions with the users can be built upon the graphical modules and the other FIWARE enablers.

11.3.4 Security

Important points to take into account when building a smart city platform are security and privacy. Indeed, such a platform should make delivery and usage of services trustworthy by meeting security and privacy requirements.

Two important generic enablers provided by the FIWARE community are KeyRock and Wilma. KeyRock is intended for IdM. Indeed, KeyRock permits the placement in an easy way authentication and authorization security through OAuth 2.0 tokens. So, the access to services and applications is limited to authenticated and authorized users. On the other hand, Wilma is a PEP proxy, whose role is to protect applications and services from unwanted access. Practically, if users wish to access, for example, a particular data set stored in an Orion context broker, they should authenticate themselves to KeyRock. The authorization to access the data stored in Orion is managed by Wilma through different policies. If users comply with the pre-defined policies, they can retrieve indirectly the data of Orion by the intermediary Wilma. Indeed, Wilma plays the role of proxy for Orion and avoids a direct contact between the Orion context broker and external users.

11.3.5 Interface to networks and devices

Interface to networks and devices (I2ND) enables the construction of communication-efficient distributed applications, exploitation of advanced network capabilities and easy management of robotic devices. This interface uses all the new features proposed by SDN. Access to different IoT devices is realized through the middleware. Furthermore, I2ND is also the interface to robots.

11.3.6 Architecture of applications/Services ecosystem and delivery framework

The system architecture allows the co-creation, publication, cross-selling and consummation of applications and services, by addressing all business aspects. This leads to an ecosystem composed of applications and services. It is in fact a kind of delivery framework dedicated to smart cities. The marketplace is the entry point of the delivery framework for the end-users.

11.3.7 Cloud hosting

The cloud hosting provides computation, storage and network resources to manage services. The network resources connects all sorts of IoT devices installed in smart cities. The computing resources handle the incoming data and forward them to the right place, notably to the storage resources composed of different databases for short and long terms storage.

12 General framework for data platforms

The key aspects of data platforms are described in clauses 12.1 to 12.6. These features have been derived from clause 8, which underscored some current API solutions.

12.1 Data catalogue

Data platforms will incorporate a set of APIs that will allow data providers to underscore their data sources in a centralized or federated catalogue of data collected from the IoT ecosystem. As data will be collected from a variety of sources, there will be an increase in the number of data sets. These platforms will also offer the possibility of refining the search by using specific search functions and tags relating to the data source, data type, application, etc.

12.2 Data access and control

The data platform should be adaptable such that access to the platform is thoroughly vetted depending on the type of data being accessed (e.g., personal or anonymized data). The governance and regulatory control of the platform will remain in the hands of the relevant public entities, in conformance with applicable data regulations.

12.3 Data licensing

Data sharing especially with external parties can be conducted either for free or be offered at a certain fee. It is essential that data providers retain control over the data and respect the regulation of data protection like the EU general data protection regulation (GDPR) when sharing data through licensing. With reference to any data platform, data providers can choose to adopt licence models. This model has the following benefits:

– specifies the role of each involved party, like data providers and end-users;
– sets the precise duration for data use through the licence issued;
– limits the activities and sectors for which the data are utilized;
– establishes geographical restrictions;
– demonstrates the purpose of data utilization;
– presents the authorization to resell data (with restrictions).

12.4 Dispute resolution

When handling large amounts of data, trust forms an integral part of any data platform. Nevertheless, in the event of any dispute, is it is essential to have a resolution mechanism. To ensure a fair and transparent process, metering tools regulated by smart contracts could be employed. This will help ensure that data flow is traceable through records and can be verified.

12.5 Quality assurance

The data sets and streams available on the platform will be given consumer feedback and ranked based on the accuracy and usability of the data. This will evaluate whether the data available is well documented and regularly updated.

12.6 Security

Security for data sharing platform should account for the following:

– IdM system to store and organize the identities of users;

– authentication component that regulates access to the platform;

– authorization component that grants access to functionalities and digital assets according to predefined policies.

13 Open data platform on the IoT reference model

In clauses 13.1 and 13.2, an initial set of characteristics and technologies suggested for the implementation of an API for an open data platform (based on an IoT reference model), is presented. The API is related to northbound and southbound layers, highlighting the most relevant features.

NOTE – This clause does not represent an exhaustive API specification. It presents a baseline for further detailed definition.

13.1 Northbound layer

In the northbound layer of an open data platform, the ones that interact with end-user applications and systems can be composed of the following components that provide a corresponding set of APIs.

Context management: This component is related to the management of the context information produced by the IoT open data platform. The context information represents data related to entities in the real world and is useful for the interpretation of complex phenomena. The context manager provides access to the data through a publish-subscribe-based interface in order to be notified by specific events (i.e., context changes). The context manager supports not only asynchronous communication (represented by the notification mechanism), but also synchronous interactions allowing data to be queried. Possible candidates for this API specification are as follows.

– FIWARE-NGSI v2 API is intended to manage the entire lifecycle of context information, including updates, queries, registrations and subscriptions. It establishes a data model for context information, a context data interface for information exchange through query and subscription and a context availability interface about how to obtain context information.

– ETSI NGSI-LD API, developed by the ETSI ISG CIM, can be considered an evolution of FIWARE-NGSI v2, extending the support to linked data.

– FIWARE and TMF are working together in the initiative named FrontRunner which is creating a unified data model. TM Forum and FIWARE are building a unified city data model which is used in the IoT device and service management APIs.

– The data access endpoint of the TM Forum IoT device management API that is able to convey the data using several protocols. It permits multiple interactions.

Open data management: This component is in charge of providing access to open data that can be exposed as traditional data catalogues with descriptions of datasets or linked data. The features of this API are as follows.

– DCAT-AP compliant API: The data catalogue vocabulary (DCAT) developed by the World Wide Web Consortium (W3C) is an RDF vocabulary designed to facilitate interoperability between data catalogues published on the web. DCAT-AP is an application profile that specifies mandatory, recommended and optional elements and establishes recommendations for controlled vocabularies. DCAT-AP represents an international standard for metadata publishing.

– SPARQL endpoint: API should also provide a way to queries linked open data using specific query languages. It is suggested an endpoint be provided that supports SPARQL, a query language that is able to retrieve data stored in the RDF format.

Security management: The security component manages all the main security and privacy functionalities. At the northbound level, the security should cover not only authentication and authorization issues, but also encryption and anonymization in order to delete sensitive or personal information that cannot publicity provided. Possible, candidate technologies for this API are as follows.

– **The Security Assertion Markup Language (SAML)** is an XML-based framework for authentication and authorization developed by the Security Services Committee of OASIS. It determines how to share information about user identities, grant or deny access to resources and request authentication.

– **XACML** establishes a declarative fine-grained, ABAC policy language, an architecture and a processing model describing how to evaluate access requests according to the rules defined in policies.

– **OAuth** is another open standard for authorization not authentication purposes. The OAuth specifications determine the following roles: 1) the end user or the entity that owns the resource in question; 2) the resource server (OAuth provider), which is the entity hosting the resource; 3) the client (OAuth consumer), which is the entity that is looking to consume the resource after getting authorization from the client. The evolution, OAuth 2.0, implements some improvements of the original protocol.

13.2 Southbound layer

The southbound layer is more related to the data provider side so the focus is mainly on the IoT and semantic interoperability. Some suggested approaches to the API specification are as follows.

Context management: The context management API already present in the northbound layer is also fundamental to the southbound, in particular from the data producer point of view. Data produced by the platform (e.g., generated for IoT devices) is published as context data to be provided (i.e., notified) to the upper layers. The suggested interfaces for context management have already been presented in clause 13.1 (FIWARE-NGSI and ETSI NGSI-LD).

Adoption of common data models and ontologies: A key element to enable interoperability among different entities and technologies is the adoption of shared common data models, ontologies and controlled vocabularies. There are several initiatives aimed at defining common data models and ontologies in the field of IoT and smart city. Among the most relevant ones we can mention the following.

– FIWARE data models: is a set of data models specified with the aim of enabling data portability in different applications with a specific focus on the smart city and those related to the IoT. They are fully compliant with FIWARE-NGSI v2.

– SAREF is a common ontology that facilitates the matching of existing assets (standards, protocols, data models, etc.) in the smart appliances domain. SAREF has different domain specific extensions, such as energy, environment and buildings. SAREF can be mapped to the NGSI-LD information model.

IoT protocol interoperability: Considering the large variety of IoT and network protocols, the provision is suggested of a uniform southbound API (e.g., context management) as well as harmonization of the different protocol interfaces specifying protocol adapters managed by dedicated middleware. This can be considered a strategy to achieve interoperability among heterogeneous technologies. There are several solutions that provide platforms for IoT interoperability-based modular protocol adapter, the following are highlighted to be the most suitable in relation to the aforementioned technologies.

– IDAS: IDAS (part of the FIWARE framework) provides different IoT agents that represent bridges between a specific IoT and NGSI. Currently the following protocols are supported: LWM2M over CoAP, JSON or UltraLight over HTTP/MQTT, long range wide area network (LoRaWAN), open platform communications unified architecture (OPC-UA).
– The UDG Alliance supports a set of IoT protocols in link connectivity, networking and data communication in diverse application domains: about 50 IoT protocols are supported including HTTP, MQTT, LWM2M, CoAP, Internet protocol v6 (IPv6) over Low-Power Wireless Personal Area Networks (6LoWPAN), IPv6 over the time-slotted channel hopping (TSCH) mode of IEEE 802.15.4e (6TiSCH), Advanced Message Queuing Protocol (AMQP)). It also supports context management being compliant with NGSI.

Bibliography

[b-ITU-T T.170]	Recommendation ITU-T T.170 (1998), *Framework of the T.170-Series of Recommendations.*
[b-ITU-T Y.4000]	Recommendation ITU-T Y.4000/Y.2060 (2012), *Overview of the Internet of things.*
[b-ITU-T Y.4472]	Recommendation ITU-T Y.4472 (2020), *Open data application programming interface for IoT data in smart cities and communities*.
[b-ETSI SAREF]	ETSI (2020), *Smart Applications REFerence ontology, and extensions*. Sophia Antipolis: European Telecommunications Standards Institute. Available [viewed 2020-09-13] at: https://saref.etsi.org/
[b-ETSI WP 31]	Bees, D., Frost, L., Bauer, M., Fisher, M. Li, W. (2019), *NGSI-LD API: For context information management*, *ETSI White Paper* No. 31. Sophia Antipolis: European Telecommunications Standards Institute. 19 pp. Available [viewed 2020-09-13] at: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp31_NGSI_API.pdf
[b-IEEE 802.15.4e]	IEEE 802.15.4e-2012,[1] *Local and metropolitan area networks – Part 15.4: Low-rate wireless personal area networks (LR-WPANs) Amendment 1: MAC sublayer*.
[b-IETF RFC 2616]	IETF RFC 2616 (1999), *Hypertext transfer protocol – HTTP/1.1*.
[b-IETF RFC 6749]	IETF RFC 6749 (2012), *The OAuth 2.0 authorization framework.*
[b-EU SynchroniCity D2.1]	SynchroniCity (2017), *Reference architecture for IoT enabled smart cities*, deliverable D2.1. Brussels: European Commission. 179 pp. Available [viewed 2020-09-13] at: https://synchronicity-iot.eu/wp-content/uploads/2018/05/synchronicity_d2_1_reference_architecture_for_iot_enabled_smart_cities.pdf .
[b-EU SynchroniCity D2.5]	SynchroniCity (2017). *Advanced data market place enablers*, deliverable D2.5. Brussels: European Commission. 34 pp. Available [viewed 2020-09-12] at: https://synchronicity-iot.eu/wp-content/uploads/2020/02/SynchroniCity_D2.5.pdf
[b-Badii]	Badii, C., Bellini, P., Cenni, D., Difino, A., Nesi, P., M Paolucci, M. (2017). Analysis and assessment of a knowledge based smart city architecture providing service APIs. *Future Gener. Comput. Sys.* **75**, pp. 14-29.
[b-Buhr]	Buhr, C.-C., Kleiner, T. (2012). European open data policy: Challenges and opportunities. *Z. Politikberatung (ZPB)/Policy Advice and Political Consulting*, **5**(3), pp. 141-146.
[b-CKAN]	CKAN (Internet). *CKAN, the world's leading open source data portal platform.* Available [viewed 2020-09-12] at: https://ckan.org/

[1] Superseded.

[b-dataModel.Weather] Github Inc. (2020), *dataModel.Weather*. Available [viewed 2020-09-13] at: https://github.com/smart-data-models/dataModel.Weather/blob/master/WeatherObserved/schema.json

[b-dataModel.Weatherexample] Github Inc. (2020), *dataModel.Weather: Example.* https://github.com/smart-data-models/dataModel.Weather/blob/master/WeatherObserved/examples/example.json

[b-Eurocities] Eurocities (2017), *Eurocities standards and interoperability management guide*. Brussels: Eurocities. 2 pp. Available [viewed 2020-09-13] at: http://nws.eurocities.eu/MediaShell/media/EUROCITIES_Standards_Interoperability_Guide_04-10-2017.pdf

[b-FIWARE-NGSI] FIWARE (2018), *FIWARE-NGSI v2 specification.* Available [viewed 2020-09-13] at: https://fiware.github.io/specifications/ngsiv2/stable/

[b-FIWARE-Device] FIWARE (Internet). *FIWARE IoT device management GE architecture.* Available [viewed 2020-09-14] at: https://tourguidemigration.readthedocs.io/en/latest/02_iot/Connection%20to%20the%20Internet%20of%20Things/

[b-FIWARE-Temperature] FIWARE (Internet), *Core context management.* Available [viewed 2020-09-14] at: https://fiwaretourguide.readthedocs.io/en/latest/core/introduction/

[b-Guillaume-Gentil] Guillaume-Gentil, A. (2015). Decision support systems: Open and big data. *Spore*, No. 175, pp. 13-17. Wageningen: Technical Centre for Agricultural and Rural Cooperation (CTA).

[b-Klímek] Klímek, J. (2019). DCAT-AP representation of Czech National Open Data Catalogue and its impact. *J. Web Semant.*, **55**, pp. 69-85.

[b-OASC] OASC (2020), (*The OASC story*. Brussels: Open & Agile Smart Cities. Available [viewed 2020-09-13] at:https://oascities.org/about-oasc/

[b-OKF-a] OKF (Internet-a). What is open data? In: *Open data handbook*. Open Knowledge Foundation. Available [viewed 2020-09-11] at: http://opendatahandbook.org/guide/en/what-is-open-data/

[b-OKF-b] OKF (Internet-b). *Open definition 2.1*. Open Knowledge Foundation. Available [2020-09-11] at: https://opendefinition.org/od/2.1/en/

[b-OKF-c] OKF (Internet-c). *A fair, free and open future*. Open Knowledge Foundation. Available [2020-09-12] at: https://okfn.org/

[b-Ong] Ong, S.P., Cholia, S., Jain, A., Brafman, M., Gunter, D., Ceder, G., Persson, K.A. (2015), The materials application programming interface (API): A simple, flexible and efficient API for materials data based on representational state transfer (REST) principles. *Comput. Mater. Sci.*, **97**.pp. 209-215.

[b-TMForum] Gauthier, P., TM Forum (2019), IoT Standards Trends and Convergence – Joint Workshop: Internet of Things for Smart Cities & Communities (IoT4SCC). 21 pp. Available [viewed 2020-09-17] at: https://iotweek.blob.core.windows.net/slides2019/4.%20THURSDAY%2020/IoT%20Standards%20Trends%20and%20Convergence/01%20Pierre%20Gauthier_IoT%20Standards%20Trends%20and%20Convergence.pdf.

SERIES OF ITU-T RECOMMENDATIONS

Series A	Organization of the work of ITU-T
Series D	Tariff and accounting principles and international telecommunication/ICT economic and policy issues
Series E	Overall network operation, telephone service, service operation and human factors
Series F	Non-telephone telecommunication services
Series G	Transmission systems and media, digital systems and networks
Series H	Audiovisual and multimedia systems
Series I	Integrated services digital network
Series J	Cable networks and transmission of television, sound programme and other multimedia signals
Series K	Protection against interference
Series L	Environment and ICTs, climate change, e-waste, energy efficiency; construction, installation and protection of cables and other elements of outside plant
Series M	Telecommunication management, including TMN and network maintenance
Series N	Maintenance: international sound programme and television transmission circuits
Series O	Specifications of measuring equipment
Series P	Telephone transmission quality, telephone installations, local line networks
Series Q	Switching and signalling, and associated measurements and tests
Series R	Telegraph transmission
Series S	Telegraph services terminal equipment
Series T	Terminals for telematic services
Series U	Telegraph switching
Series V	Data communication over the telephone network
Series X	Data networks, open system communications and security
Series Y	**Global information infrastructure, Internet protocol aspects, next-generation networks, Internet of Things and smart cities**
Series Z	Languages and general software aspects for telecommunication systems

Printed in Switzerland
Geneva, 2020