

SYNCHRONICITY



GA no:	732240
Action full title:	SynchroniCity: Delivering an IoT enabled Digital Single Market for Europe and Beyond
Call/Topic:	Large Scale Pilots
Type of action:	Innovation Action (IA)
Starting date of action:	01.01.2017
Project duration:	36 months
Project end date:	31.12.2019
Deliverable number:	D2.10
Deliverable title:	Reference Architecture for IoT Enabled Smart Cities, Update
Document version:	1.0
WP number:	WP2
Lead beneficiary:	5-ENG
Main author(s):	Martino Maggio (ENG), Francesco Arigliano (ENG), Ömer Özdemir (ATOS), José Manuel Cantera (FF), Eunah Kim (UDG), Ignacio EliceGUI Maestro (UC), Andrea Gaglione, Angelo Capossole (DigiCat)
Internal reviewers:	Thomas Gilbert (AI), Jan-Joost van Kan (ATOS), Martin Brynskov (AU)
Type of deliverable:	Other
Dissemination level:	PU
Delivery date from Annex 1:	M14
Actual delivery date:	M20 (30.08.2018)



Co-funded by the
European Union

This deliverable is part of a project that has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no 732240.



5-CH (partner)



5-KR (partner)



5-IT (partner)

Executive Summary

This deliverable D2.10 "Reference Architecture for IoT Enabled Smart Cities, Update" represents an update of the deliverable D2.1 [1] and contains the final report about the specifications, architecture, reference implementation and API of the SynchroniCity technical framework.

In the following, we provide a high-level summary of the key points addressed in the report. For more details, please consult the respective sections of the deliverable.

SynchroniCity architecture baseline and final requirements

The SynchroniCity Large Scale Pilot is established to demonstrate the foundations of a global market for IoT-enabled urban services based on the OASC [2] common technical ground for standards-based innovation and procurement. In order to achieve this, the SynchroniCity partners have identified and summarized the common aspects of the main Smart City reference architectures, either standard or coming from research initiatives. The previous deliverable D2.1 included a detailed analysis about the most relevant Smart City architectures defined in research projects or related activities in Europe and in international contexts; in the section 1 some of these initiatives (ITU-T FG-SCC, ITU-T Y.2060, ISO/IEC JTC 1, oneM2M, BIG-IoT, OrganiCity, Espresso, FIWARE, AIOTI, EIP-SCC) that influenced or inspired, in different aspects, the design of SynchroniCity architecture are briefly listed and summarized.

Section 1 presents also updated system requirements, data management and service requirements as well as the security and privacy requirements of SynchroniCity that drove the design of its architecture and support the development of the digital single market.

The requirements have been derived by combining a set of requirements, principles and guidelines for the initial SynchroniCity high-level architecture captured in deliverable D1.3 [3], data protection and privacy concerns highlighted in deliverable D1.4 [4], a set of considerations elaborated from the analysis of relevant standards and technologies and a set of requirements emerged from the reference zone technical baseline analysis in deliverable D2.1.

Logical reference architecture

Section 2 introduces the logical components of the SynchroniCity IoT Enabled Smart Cities reference architecture. The main aim of this architecture is to define a set of logical components and functionalities that can enable different cities to be actively part of IoT Smart City Digital Single Market.

The SynchroniCity logical reference architecture, is the composition of different logical modules that are summarized below:

- **Context Data Management:** *it manages the context information coming from IoT devices and other public and private data sources, providing a context data access through a uniform 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 to interact, through specific IoT Agents, with the devices that use different standards or protocols, making them compatible and available to the SynchroniCity framework;*
- **Data Storage Management** *provides functionalities related to the data storage and access in the specific context of IoT systems and smart city platform, interacting with heterogeneous sources.*
- **IoT Data Marketplace** *supports business interactions between data suppliers 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,*

revenue management. These functions will support the creation of innovative business models.

- **Security, Privacy and Governance:** this module covers all the security aspects related to three main pillars: data, IoT infrastructure and the platform services, which underpin the applications and services of the cities. Around these pillars, security functionalities provide crucial security properties such as confidentiality, authentication, authorization, integrity, non-repudiation, access control, etc.
- **Monitoring and Platform management services:** it provides functionalities to manage platform configuration and to monitor activities of the platform services. It supports specific KPI definition to evaluate the status of the platform in relation to different aspects (e.g. performance, usage, reliability, quality of service etc.)

In the SynchroniCity architecture, a key role is played by the interoperability points that represents the main interfaces to access to the functionalities of the framework and in particular the way to consume and provide data. The concept of interoperability points is based on the Minimal Interoperability Mechanisms (MIMs) defined in OASC initiative [2]. MIMs are simple and transparent mechanisms, ready to use in any city, regardless of size or capacity. The interoperability points assure the replicability of the solutions built on top of the SynchroniCity framework that being compliant to them, are completely decoupled from the specific technological implementation and deployment of the architectural components.

Sections 2 describe the following categories of interoperability points:

- Context Management interface
- SynchroniCity Data Models
- Data Storage Interface
- IoT Data Marketplace Interface
- Security Interface

SynchroniCity framework reference implementation

One of the key objectives of the WP2 was to build an overall logical architecture to define the needed functionalities of the SynchroniCity IoT Enabled Framework for Smart Cities; furthermore, SynchroniCity proposes a reference implementation of the logical architecture in order to provide, to RZs open-source, ready to use components:

- that can speed-up the development of SynchroniCity API
- directly support the implementation new functionalities not already present in the existent RZ technical platform
- simplify the integration between the current RZ IoT infrastructure and the SynchroniCity technical framework.

Section 3 describes the technological solution proposed for the implementation of the SynchroniCity reference architecture introducing the functionalities and roles of the single components: these components are not mandatory, both in their adoption or usage, but should be considered only one of the possible implementations of the SynchroniCity architecture. The RZs can choose to adopt or integrate single components, among the proposed ones in the reference implementation, with their existent infrastructure or use their functionalities in “as a service approach” deploying them in specific cloud instances. The flexibility of this technical solution allows Reference Zones to choose the best approach related to their technical and business requirements.

SynchroniCity Reference Implementation API

SynchroniCity defined a concrete implementation of the logical interfaces (interoperability points): The SynchroniCity API based on HTTP RESTful approach and widely used standards.

The implementation of the SynchroniCity API is a basic requirement for the Reference Zones that want to be

compliant with SynchroniCity. The section 4 presented an overview of the API:

- *Context Management API the way to communicate with the Context Management module in order to manage the context entities. The API are based on NGSIv2*
- *Data Storage API provide access to historical data and Open Data. The definition of this API is inspired to the NGSI-LD Temporal Query language*
- *IoT data marketplace API allows the monetization of digital assets during the whole service life cycle. It is an extension of the Business API Ecosystem*
- *Security API provide authorisation functionalities to access to the SynchroniCity services. The API are based on OAUTH2 protocol*

Abbreviations

AIOTI	Alliance for the Internet of Things Innovation
API	Application Programming Interface
CDB	Context Data Broker
CEP	Context Event Processing
CKAN	Comprehensive Knowledge Archive Network
CSE	Common Service Entity
DCAT	Data Catalogue Vocabulary
DCAT-AP	DCAT Application profile
EIP-SCC	European Innovation Partnership on Smart Cities and Communities
FG-SSC	Focus Group on Smart Sustainable Cities
GA	Grant Agreement
GE	Generic Enabler
GPRS	General Packet Radio Service
HTTP	HyperText Transfer Protocol
ICT	Information and communication technology
IEC	International Electrotechnical Commission
IoT	Internet of things
ISO	International Standard Organization
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
JSON	JavaScript Object Notation
KPI	Key Performance Indicators
LOD	Linked Open Data
LoRa	Long Range
LTE	Long Term Evolution
LXM2M	Lightweight M2M
MQTT	Message Queue Telemetry Transport
NB-IoT	Narrowband IoT
NGSI	Next Generation Service Interfaces
OASC	Open & Agile Smart Cities
ODMS	Open Data Management System
PEP	Policy Enforcement Point
PDP	Policy Decision Point
RDF	Resource Description Framework

REST	Representational state transfer
RZ	Reference Zone
SAML	Security Assertion Markup Language
SDO	Standards Developing Organizations
SLA	Service Level Agreement
SOAP	Simple Object Access Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SR	System Requirement
STH	Short Term History
UL	Ultra Light
W3C	World Wide Web Consortium
WP	Work Package
XML	eXtensible Markup Language
XACML	eXtensible Access Control Markup Language

Contents

Introduction	11
1 SynchroniCity architecture baseline and final requirements	12
1.1 Relation with existing initiatives and standards	14
1.2 SynchroniCity architecture requirements	17
1.2.1 Architectural system requirements	17
1.2.2 Architectural data management and service requirements	21
1.2.3 Architectural security and privacy requirements	24
2 Logical reference architecture	26
2.1 Architecture overview	26
2.2 Components description	29
2.2.1 Context Data Management	29
2.2.2 IoT Management	33
2.2.3 Data Storage Management	36
2.2.4 IoT Data Marketplace	37
2.2.5 Security, Privacy and Governance	45
2.2.6 Monitoring and Platform management service	49
2.3 Interoperability points logical interfaces	51
2.3.1 Context Management Interface	51
2.3.2 Data Storage Interface	53
2.3.3 SynchroniCity Data Models	53
2.3.4 IoT Data Marketplace Interface	54
2.3.5 Security Interface	57
2.3.6 SynchroniCity compliance levels	59
3 SynchroniCity framework reference implementation	62
3.1 Context Data Management	63
3.1.1 Context Data Broker	63
3.1.2 Data Connector	64
3.2 Data storage	65
3.2.1 STH-Comet	65
3.2.2 Idra - Open Data Federation Platform	65
3.3 IoT Management	66
3.3.1 IoT Agent Manager (IDAS)	66

3.3.2	IoT Agents	67
3.4	Security layer	68
3.4.1	Identity and Authentication Manager - Keyrock	69
3.4.2	Pep Proxy - Wilma	70
3.5	IoT Data Marketplace	72
4	SynchroniCity API	74
4.1	Context Management API	74
4.2	Data Storage API	76
4.2.1	Historical	76
4.2.2	Open Data	76
4.3	IoT Data Marketplace API	77
4.4	Security API	78
5	Conformance tests	79
	Conclusions	81
	References	82

List of Figures

Figure 1 - D2.1 and D2.10 relationship.....	12
Figure 2 - WP2 tasks relationships.....	13
Figure 3: SynchroniCity Reference Architecture.....	28
Figure 4 - Conceptual model of the SynchroniCity IoT data marketplace [7]	37
Figure 5: Security Technological pillars.....	45
Figure 6: NGSI Context Information example [26]	51
Figure 7: Context Management API	52
Figure 8: Data Storage API	53
Figure 9: Different cases to enable Data Models Interoperability.....	54
Figure 10: IoT Data Marketplace APIs	54
Figure 11: Security Components Diagram.....	57
Figure 12: SynchroniCity compliance levels.....	59
Figure 13: Reference implementation architecture overview	63
Figure 14: Dataset search results in the Idra portal	66
Figure 15: Security Layer overview	69
Figure 16: Home page of the IdM Web portal.....	70
Figure 17: API access authorization flow	71

List of Tables

Table 1: Architectural system requirements	20
Table 2: Data Management and Service requirements	23
Table 3: Security and Privacy requirements	25
Table 4: Context Data Broker module description	30
Table 5: Context Event Processing module description.....	31
Table 6: Common Data Models Adapter module description.....	32
Table 7: Data Connector module description	33
Table 8: IoT Agents module description	34
Table 9: Device Management module description.....	35
Table 10: Catalogue Management module description	38
Table 11: Offers/Orders Management module description	39
Table 12: Peering Management module description	40
Table 13: Revenue Management module description	41
Table 14: Feedback and reputation module description	42
Table 15: Customer Management module description	43
Table 16: SLA and license Management module description	44
Table 17: Transparency and accountability module description	44
Table 18: Data protection and Privacy module description.....	46
Table 19: Identity and Authentication Management module description	47
Table 20: Authorization and Accounting module description	48
Table 21: Policy Management module description	48
Table 22: Administration & Configuration module description	49
Table 23: Platform Monitoring module description.....	50

Introduction

This document represents the final report about the architecture of the SynchroniCity technical framework. This deliverable is an update of the deliverable D2.1 [1], the first version of the SynchroniCity architecture. In particular, D2.10 includes content (i.e. requirements and architecture) that supersedes the ones provided in the D2.1 and completely new sections (related to SynchroniCity framework reference implementation and API). The initial analysis related to the architectures and the reference zones infrastructure, used as initial input for the first version of the architecture is contained in D2.1.

The main updates of this last version can be summarized in the following points:

- Consolidation of requirements and architectural component description and interactions (sections 1.2, 2): following the feedbacks and the discussion among technical partners and RZs inside *WP2 – “IoT Enabled Framework for Smart Cities”* and *WP3 - “Initial Applications and Services”*, the list of requirements has been revised in order to be in line also with the pilot applications needs. This revision consists in minor changes directly reflected in the architectural components definition and description. The overall architecture overview has also been updated because new relationships and dependencies, not present in the first version, are introduced.
- Interoperability points definition (section 2.3): interoperability points represents the main interfaces among SynchroniCity framework, Reference Zones and external stakeholders. The concept of interoperability points is based on the Minimal Interoperability Mechanisms (MIMs) defined in OASC initiative [2]. MIMs are simple and transparent mechanisms, ready to use in any city, regardless of size or capacity. In this new version of the deliverable is presented a clearer and more detailed description of these framework “entry points” that are the logical view of the SynchroniCity API.
- Reference implementation architecture: in D2.10 is included the overall description of the technical components proposed for the reference implementation of the reference architecture. These components, represent one of the possible ways to concretely implement the architecture being compliant with SynchroniCity specifications and API. The components described in section 0 are provided to the RZ and interested stakeholder as open software components ready to use.
- SynchroniCity API specification: in section 4 are included the RESTful APIs that represent the concrete implementation of the SynchroniCity interoperability points (MIMs), based on international well-know and open standards. SynchroniCity APIs provide the access for third party applications and new RZs to interact with the SynchroniCity framework, to provide and consume data, and to take advantage from the basic set of functionalities (such as security).

1 SynchroniCity architecture baseline and final requirements

Deliverable D2.10 represents the final version of the SynchroniCity architecture, the main output of Task 2.1. Starting from the initial version, D2.1, delivered in August 2017, this new version represents on one side a revision of the existing content in terms of requirements and logical specification and on the other side a finalization of the logical architecture with the concrete API definition and reference implementation description. Figure 1 shows the relationship between the two deliverables and the inputs that contributed to its definition. Deliverable D2.1 [1] provided a logical view of the SynchroniCity architecture: starting from a set of initial use case requirements and a detailed analysis of the existing technical infrastructure of the Reference Zones involved in the project, a set of requirements has been identified; these requirements together with the results of the state related to Smart Cities architectures drove the definition of the first release of the logical architecture: for each logical module have been described the functionalities and the relations among them.

D2.10 includes the main content of D2.1 revised using the new inputs collected in the following months of project activities: in particular, new specific needs coming from RZ and the design of the pilot application have been used to validate and revise the original architecture. The main improvement of the final version of the SynchroniCity architecture, reported in this document is the definition of a reference implementation for the SynchroniCity framework and the concrete API that will allow external system and applications to interact with the it. It is important to emphasise that the refinement process and in particular the definition of a reference implementation has been guided by new and improved input, feedback and collaboration with project and initiatives with a specific focus to standard adoption.

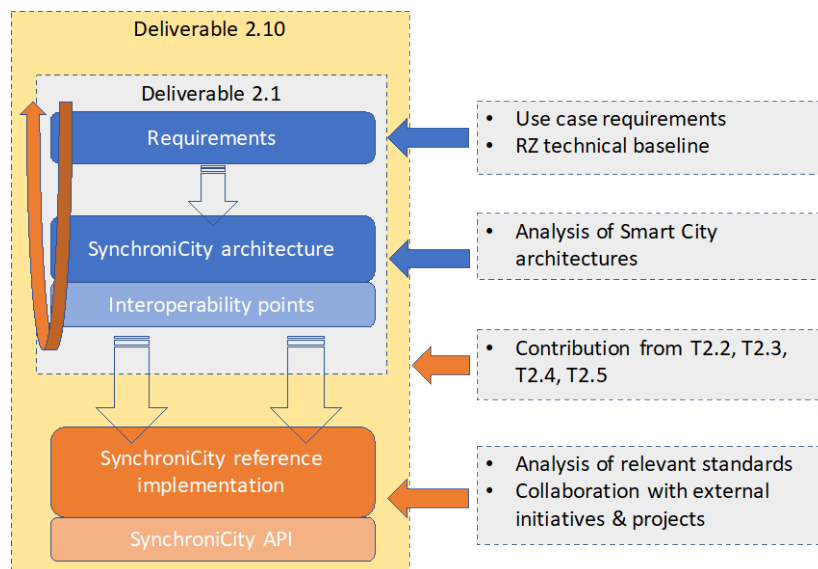


Figure 1 - D2.1 and D2.10 relationship

The reference architecture definition is the central and key activity of WP2 because it establishes the guidelines and the constraints for all the other technical tasks. The different components defined in the architecture are then further analysed and implemented in the others WP2 tasks. Task 2.2 focuses on the specification of the tools and process to manage the lifecycle of Data Models for the different domains in the Smart City adopting NGSI as agnostic information model and API in order to provide an interoperability layer. First results have been included in the D2.2 deliverable [5] to specify guidelines and compliance for SynchroniCity Information Models explaining and helping to understand how to develop a harmonised set of information models in the context of the

SynchroniCity architecture. Finally, the deliverable D2.3 [6] will define data models for verticals in the city (e.g. traffic, public transport, waste management, air quality, etc.) in order to provide a true interoperability mechanism for systems and applications in the city and among cities; this task is closely tied to WP3 and open calls pilots that will provide the main inputs for the data models definitions.

Task T2.3 investigates marketplace mechanisms to enable sustainable commercial viability of data by developing a considerable added value that goes beyond traditional rights-based licensing models of data sets. The SynchroniCity architecture described the basic marketplace module and functionalities while the D2.4 deliverable [7] detailed them, providing also the first proof of concept integrated with the IoT infrastructure in Santander RZ. D2.5 [8] will continue the development of the Marketplace extending it with advanced features.

Task 2.4 focuses on the integration of IoT devices and solutions with the SynchroniCity framework. It addresses the southbound interface of IoT-based smart cities. The D2.6 deliverable [9] analysed the existing solutions in RZs to provide guidelines for southbound integration, so that IoT products and solutions can easily be deployed into the SynchroniCity framework.

Finally, task 2.5 describes the operations of deploying concrete architectures that conform to the previous results in the Reference Zones. The D2.8 deliverable [10] provided a description of the specific reference zone infrastructure and platform elements to be deployed and operated in each city based on the initial architecture specification and the proposed technologies and components that have been documented in the D2.1.

Figure 2 depicts the relationships (black lines) among the WP2 tasks; as previously described such tasks are delegated to detail particular aspects related to the SynchroniCity architecture; some feedbacks directly influenced the final version of the reference architecture and reported in this document. For instance, from the 2.3 and 2.4 tasks, information was gathered to better understand the security components integration requirements; from the task 2.5 feedbacks on the modularization approach to adopt was collected to drive the reference implementation definition. In addition, task 2.2 was useful for understanding the requirements for data model management and also acted as a communication channel with WP3 and WP5 on applications and open calls definition.

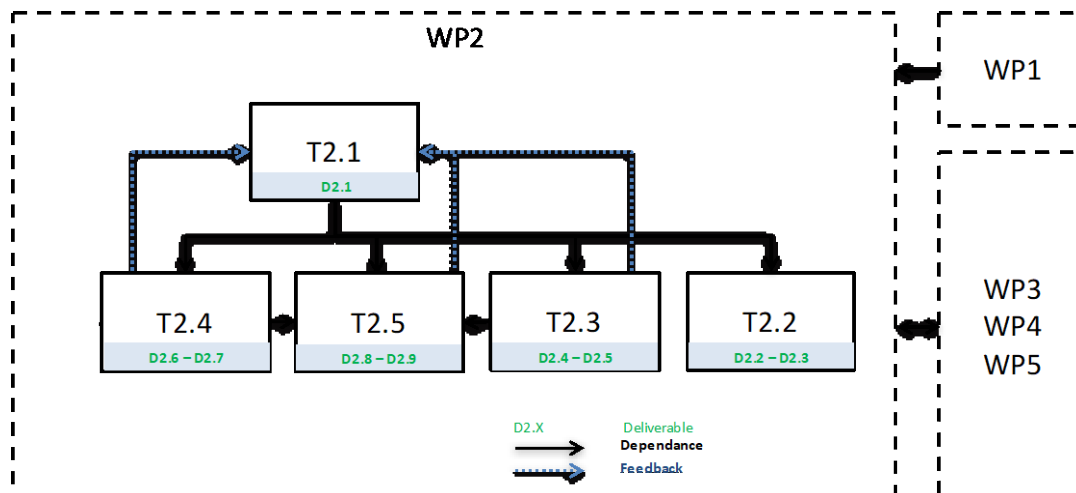


Figure 2 - WP2 tasks relationships

The following section summarizes the different inputs describing how they influenced the definition and consolidation of the SynchroniCity architecture: with a specific focus on the relationships with existing standards and initiatives connected to SynchroniCity.

1.1 Relation with existing initiatives and standards

Smart City is a key driver of large scale IoT markets where a complex set of different technologies, consisting of heterogeneous IoT devices, network infrastructure, cloud systems, multiple IoT platforms, go together to provide heterogeneous smart city applications and services. The scale of smart city applications varies from a single local network based system to a large scaled cross-platform deployment. Various technologies are used to support the use case requirements and they may include diverse IoT protocols, security and privacy, data storage, data analytics, etc. City management, central government, and markets are moving towards support of smart cities. Such active markets driving brings dynamic technology developments, but this process can lead to a lack of usable standards and common framework for smart cities. As a result, there are several existing and on-going studies in large scale IoT platforms and common frameworks for Smart Cities in various standard bodies, EU programs and initiatives.

SynchroniCity aims to build a Digital Single Market for Europe by piloting its foundations at scale in reference zones across 8 European cities, involving also other cities globally, for this reason it was crucial to start from the analysis (provided in [1]) of existing reference models and architectures suggested by some relevant initiatives: ITU-T FG-SCC, ITU-T Y.2060, ISO/IEC JTC 1, oneM2M, BIG-IoT, OrganiCity, Espresso, FIWARE, AIOTI, EIP-SCC.

The solutions proposed by these initiatives are different, but large commonalities are found related to basic concepts and functionalities provided. Particularly, main logical layers are relatively similar in many architectures.

SynchroniCity studied, collected and adopted the most suitable models and approaches with the aim of maximizing interoperability without precluding, but fostering the integration with existing local solutions and technical infrastructures in any city. SynchroniCity proposes an architecture framework that collects the most common capabilities and technologies needed by cities and that is easily extendable for the cities who want to extend their existing framework.

The SynchroniCity Large Scale Pilot is established to demonstrate the foundations of a global market for IoT-enabled urban services based on the OASC common technical ground for standards-based innovation and procurement. In order to achieve this, the SynchroniCity partners have identified and summarized the common aspects of the main Smart city Reference architectures, either standard or coming from research initiatives. The previous deliverable D2.1 included a detailed analysis about the most relevant Smart City architectures defined in research projects or related activities in Europe and in international contexts; in the following paragraph these initiatives that influenced or inspired, in different aspects, the design of SynchroniCity architecture, are briefly listed and summarized:

- **ITU-T FG-SSC:** the International Telecommunication Union (ITU) [11] is the United Nations specialised agency in the field of telecommunications, information and communication technologies (ICTs). The ITU Telecommunication Standardization Sector (ITU-T) is a permanent organ of ITU. ITU-T Study Group 5 set up the ITU-T Focus Group on Smart Sustainable Cities (FG-SSC), which aims to act as an open platform for smart-city stakeholders. The SSC ICT reference architecture model consists of four layers: Application layer, Data and Support layer, Network layer and Sensing layer.
- **ITU-T SG13 Y.2060** [12] provides a global overview of IoT that includes functional characteristics, high-level requirements and IoT reference model. It sees IoT as a global infrastructure and provides a reference model that can be applied to large-scale IoT based systems and applications. Smart City is a good example of using IoT as an infrastructure, and this Recommendation can be viewed as a conceptual analysis of functionalities in the SynchroniCity architecture. The IoTs reference model consists of four layers: application layer, service support and application support layer, network layer and device layer. Furthermore, two vertical capabilities are defined: management capabilities and security

capabilities.

- **ISO/IEC JTC1** established a Study Group on Smart Cities [13] and has investigated the value and scope of a Smart Cities' model, including a Smart City Reference Model from an ICT perspective, a domain knowledge model, and a data and service model. The model consists of business layer, data layer, cloud and network resources, sensing layer and security system.
- **AIOTI**: the Alliance for Internet of Things Innovation (AIOTI) [14] aims to create and master sustainable innovative European IoT ecosystems in the global context to address the challenges of IoT technology and applications deployment. This includes standardisation, interoperability and policy issues, in order to accelerate sustainable economic development and growth in the new emerging European and global digital markets. The High Level Architecture functional model, defined by AIOTI, consists of application layer, network layer IoT layer, and set of interfaces among the logical components.
- The **ESPRESSO** project [15] is a Horizon 2020 founded project with a consortium mainly including partners from Smarts Cities, public administration, European standardisation organizations (ESOs), national standardisation bodies (NSBs), standard development organizations (SDOs), industries, and research facilities. The main target of the ESPRESSO project is to ensure the interoperability of Smart City solutions. This will help cities avoiding entry barriers or vendor lock-in through promoting common meta-data structures and interoperable (open) interfaces instead of proprietary ones. In this sense, SynchroniCity has very similar objectives to ESPRESSO and for this reason the interoperability and openness aspects have been analysed and reflected in the architecture
- **EIP-SCC**: The European Innovation Partnership on Smart Cities and Communities (EIP-SCC) [16] is an initiative supported by the European Commission bringing together cities, industry, SMEs, banks, research and other smart city actors. It has 6 action clusters related to smart cities. The reference architecture for the common urban platform is carried out based on the high-level concept map and the capability map, in close collaboration with ESPRESSO project.
- **BIG-IOT**: The aim of the BIG IoT project [17] is to establish interoperability by defining a unified Web API for IoT platforms: the BIG IoT API. This Web API is aligned with the standards currently developed by the W3C Web of Things group the BIG IoT API has a broad range of functionalities to run IoT platforms and offer their resources. Once the API is implemented by an IoT platform or service, it can be registered at the BIG IoT Marketplace.

The SynchroniCity architecture (presented in section 2) has several relationships with the above-mentioned initiatives: the layered structure and most of the components can be easily mapped with some of the relevant architectures presented. The design of uniform access to the data, the IoT and virtual entities management and marketplace features have taken the approach proposed by some of the analysed projects in consideration, introducing the necessary modification needed to address Reference Zones and overall project requirements.

In addition to the state of art analysis in terms of Smart Cities architecture, the design of SynchroniCity architecture and, in particular, its reference implementation, have been based on standards and widely used technologies. Some of these are introduced here, but more information on their specific role in the architecture can be found in the next sections of this deliverable.

- **OneM2M** [18] is the global partnership project formed by global standard development

organizations (i.e. ATIS, TIA, ETSI, TSDSI, CCSA, TTA, TTC and ARIB) to bring a horizontal IoT/M2M middleware platform. The horizontal platform provides common services functions of different vertical service domains so that application developers can focus on application logics since oneM2M provides abstracted and common APIs. A oneM2M based IoT system can be composed of both non-oneM2M devices and oneM2M devices together. OneM2M can support smart city systems from the device to the applications in a holistic way. The SynchroniCity architecture is compliant with OneM2M approach and a specific pilot in South Korea (Seongnam) is going to demonstrate the integration of the SynchroniCity framework with IoT oneM2M standard based implementation.

- **FIWARE** [19] is an open source platform intended to provide the software infrastructure needed to support smart applications in multiple sectors. For smart cities in particular, FIWARE provides components that allow acquiring and harmonizing data coming from different IoT devices or external systems owned by the city. Such data is published to a data hub (context broker) which contains a uniform view, in real time, of the status of a city. Furthermore, the FIWARE Context Broker offers a modern REST API which allows applications and other services to manage, consume and subscribe to all the data generated by the city. Around the context broker different components can be plugged in. Part of the SynchroniCity architecture is inspired by the logical decomposition of that FIWARE and the SynchroniCity framework reference implementation is based on different FIWARE components (Generic Enablers)
- **OMA NGSI/ETSI NGSI-LD**: the meta-model used to represent context information has been based on OMA/FIWARE NGSI specifications. SynchroniCity is also following the definition of the new NGSI-LD specification [20], an extension of the original information model oriented to the semantical interoperability, defined by (CIM) ETSI Industry Specification Group (ISG). The NGSI-LD Information Model defines the structure of context information that shall be provided by an NGSI-LD system. Moreover, it defines the Context Information Management vocabularies to be used together with management operations. The NGSI LD Information Model is defined at two levels: the foundation classes which correspond to the Core Meta-model and the Cross-Domain Ontology. The NGSI-LD specifications, that at time of writing are in draft status, inspired the definition of the SynchroniCity Data storage historical API (see section 4.2.1). SynchroniCity will follow the evolution of this standardization process during the project execution becoming an early adopter and validator of the NGSI-LD API.

Finally, in the shared data models definition activities SynchroniCity adopted well-known information models as a key interoperability point for enabling a Digital Single Market in smart cities and to facilitate the replicability and portability of smart city solutions: the starting point was the initial adoption of FIWARE data models [21], that they will be extended to fill the gap with the specific needs of the Reference Zones. In the case of FIWARE data models, the modifications and extensions developed for SynchroniCity will be also proposed as an update of the existing data models through an active collaboration with the FIWARE community. SynchroniCity will also investigate and adopt other ontologies such as SAREF (Smart Appliances REference) [22] a shared model of consensus that facilitates the matching of existing assets (standards/protocols/datamodels/etc.) in the smart appliances domain. The SAREF ontology provides building blocks that allow separation and recombination of different parts of the ontology depending on specific needs. SAREF is one of the ontologies that can be adopted by the SynchroniCity being easily mapped to the NGSI-LD Information Model.

1.2 SynchroniCity architecture requirements

This section presents the system requirements, data management and service requirements as well as the security and privacy requirements of SynchroniCity that drove the design of its architecture and support the development of the digital single market.

The requirements have been derived by combining a set of requirements, principles and guidelines for the initial SynchroniCity high-level architecture captured in deliverable D1.3 [3], data protection and privacy concerns highlighted in deliverable D1.4 [4], a set of considerations elaborated from the analysis of relevant standards and technologies and a set of requirements emerged from the reference zone technical baseline analysis in deliverable D2.1 [1]. More specifically, a set of generic architectural system requirements highlights the importance of being prepared for technological changes by having an agile system able to be adapted and transformed to accommodate multiple emerging standards in an efficient way. Data management and service requirements primarily focus on addressing the use cases needs related to cities, marketplace providers, IoT device operators, service component providers, data providers, data consumers, infrastructure providers and citizen/end users. Finally, a set of security and privacy requirements provide measures in order to achieve a security layer compliant with data protection regulations (e.g., GDPR).

The set of requirements presented in the following section has also been also validated in a second iteration of the architectural design following the inputs and feedbacks coming from the technical activities carried out in the other tasks of work package 2 and in particular from more a specific interaction with the reference zones. This validation confirmed that, with minor modifications, the set of requirements remain valid and can be considered the base for the SynchroniCity architecture definition.

It is important to highlight that the following requirements are related to the SynchroniCity core services and not to the pilot applications that will be defined and implemented in WP3.

At the end of every subsection, a table summarizes the SynchroniCity architecture requirements, describing for each requirement its unique identifier (ID), title, high level category and typology of the requirement e.g.:

- **Functional:** it is a requirement that expresses a functionality of the platform that will be directly used by a user (human or external system);
- **Non-Functional:** this type of requirement is related with platform features that are not specific behaviours or functions, such as performance, security and interoperability.

1.2.1 Architectural system requirements

1.2.1.1 Decoupled & distributed components

The increasing investment in IoT technology results in a fast and dynamic advancement of solutions available in the market. Current IoT technologies can quickly become old and be replaced by better candidates. For this reason, the system should support deployment through a modular and flexible approach, thus every component can be replaced easily and with a very limited impact on other components and infrastructure. In turn this will increase the chances of services being adopted by cities by reducing the risk associated with deployment of monolithical or turn-key systems while also improving the development lifecycle. Moreover, designing distributed components ensures that they can run on multiple machines thus easily scaling up the running environment of a component.

1.2.1.2 Interoperability & Openness

To facilitate interoperability, the system must use as many publicly accepted standards as possible for communication and exchanging data; e.g., gateways and APIs might act as glue between those architectural components. The data and information in the platform must be provided and consumed by open protocols, standard technologies and clear agreements, so new components can easily access information already available. This also means that the APIs should be discovered and understood so that new application integrators can easily use them.

1.2.1.3 Scalability

The system can be expanded when we foresee more users of “things” and/or streams of data scaling both horizontally and vertically. In horizontal scaling other nodes could be added, where copies of the software will run on, ideally in a dynamic fashion so that nodes are added automatically when the need arises. In vertical scaling the system can store more data or have more memory to perform advanced computing.

1.2.1.4 Legacy Compatibility & heterogeneous landscape

In order to cope with the dynamic technological change, the architecture must be able to support both new and legacy components, while handling different versions of the components. Cities need to maximize the use of legacy wired/wireless infrastructures; thus the system has to support IoT based services by efficiently (re)using already available assets. Clearly, understanding the protocols used by the different RZs is a necessity, and the impact of adding new protocols needs to be minimized. The system needs to facilitate accessing and managing heterogeneous devices through a single common framework. It must offer a uniform and extendible way to access to the different devices accessible on the marketplace in order to overcome interoperability problems and reduce the friction in dealing with heterogeneous technologies.

1.2.1.5 Resilience to failure & Robustness

The architecture must be resilient to failure. Considering that components might fail and communications be affected, it should provide a self-healing system, including redundant links that cover breakdowns. We should consider that most of the IoT technologies have not yet reached a maturity level free from issues. Moreover, the interaction among many different types of components (e.g., sensors, network, wireless technology, data store, servers) from different actors could generate problems.

1.2.1.6 Performance

The system should guarantee a real-time user experience. Users should be able to responsively interact with the system, discovering new available assets at run time. The system should support assets availability and fruition in compliance with their SLA. Moreover, a continuous integration and delivery possible for each element in the architecture, automated testing to reduce regression and guarantee quality should support the system to be 24/7 operational and has a close to zero maintenance windows (software upgrades, firmware upgrades).

1.2.1.7 Communication

Communication in IoT can happen between the sensor/actuator and the gateway or between the gateway towards the platform or in some case (e.g., NB-IoT, LTE, etc.) directly from sensor/actuator to the platform. Communication with the sensor to the gateway (when wireless) is possible in numerous ways. At this moment, a variety of standards are available, thus, the platform should be able to handle different protocols (e.g., LoRa, 802.15.4, NB-IoT, WiFi, LTE, GPRS, etc.) and be flexible to incorporate future changes. When new components are selected, they should comply with communication patterns such as:

- Telemetry where communication flow is one-way from IoT device to gateway;
- Inquiries, where requests from devices looking to gather required information or asking to initiate activities, for example devices having their own business logic need input from a central server;
- Commands, where system provide execution commands to a device or a set of devices to perform specific activities;
- Notifications where information flows from other systems to a device or a group of devices by sending a broadcast message such as a time-sync message.

Table 1 summarizes the architectural system requirements by listing for each requirement its identifier, title, category and type (e.g., functional, non-functional):

ID	Title	Category	Type
SR-MODULARITY-01	Container technology	Decoupled & distributed components	Non-Functional
SR-INT-OPEN-01	Use of publicly accepted and open standards	Interoperability & Openness	Non-Functional
SR-SCALABILITY-01	Horizontal and vertical scaling	Scalability	Non-Functional
SR-LEGACY-01	Flexible support of new and legacy components	Legacy Compatibility & heterogeneous landscape	Non-Functional
SR-LEGACY-02	IoT Devices management	Legacy Compatibility & heterogeneous landscape	Functional
SR-ROBUSTNESS-01	Self-healing and robust system	Resilience to failure & Robustness	Non-Functional
SR-PERF-01	Real-time user experience	Performance	Non-Functional
SR-PERF-02	Operational	Performance	Non-Functional
SR-COMM-01	IoT communication patterns	Communication	Non-Functional
SR-COMM-02	IoT integration	Communication	Non-Functional

Table 1: Architectural system requirements

1.2.2 Architectural data management and service requirements

1.2.2.1 Data Management APIs

Access and consumption of data and services through standard and open APIs facilitate the reuse of solutions thus avoiding vendor lock-in. Moreover, by providing data publish/subscribe functionality sending and receiving data in the system can be simplified and improved. The system should provide a set of standard and open APIs to support also track changes and version update and notification and asset search functions among others capabilities. As a result, the system will be able to avoid problems and inconsistencies in the access to the resources, while simplifying the access to the APIs in the marketplace.

1.2.2.2 Data Storage Management

The architecture should address the storage of data from both platform and usage perspectives. From a platform point of view, data could be stored into a on premise system, in a cloud service or in a hybrid system. Several factors can drive the decision of where to store data. For instance, data embedding sensitive information may call for on premise systems in order to be compliant with data protection and privacy regulations. On the other hand, whenever there are no restrictions on the physical data storage location and depending on the expected amount of data, cloud service can be considered a flexible and viable solution. In this latter case, the system should take into account the latency of the network. A hybrid between both solutions where some data will be saved in locally owned systems and some data in cloud services could be also possible. From a usage perspective, applications and services may require to process data in various formats. For instance, structured data carries specific information that may fulfil the immediate needs of an application or a service, whereas raw data can embed information that may be used in the near future. Thus, the architecture should consider different data formats, providing storage support for both unstructured (e.g., raw data) and structured data and API to access historical data in a uniform manner. In order to guarantee that data access is performed in accordance with their license, policies of distribution and/or charging, the system should support different data categories based on restriction of their usage such as public or open data, private data and commercial data.

1.2.2.3 Data Models

The adoption of standard and open data models facilitates the reuse of assets and solutions, avoiding vendor lock-in. The system has to support open and standard data models and metadata by providing pre-built taxonomies to describe assets (e.g., data, services, applications, devices), to simplify the definition of the assets description and to allow reuse of existing data models.

1.2.2.4 IoT Data Marketplace

The architecture includes an IoT Data Marketplace in which data can be exchanged among users. IoT Data Marketplace providers can define different governance policies. The system should thus support a fine grain management in terms of validation procedures to be followed. Cities should be able to decide how to regulate the access to their data - either by vetting registration requests from both data providers and data consumers or by allowing an open access - how to be federated with other cities, what type of data should be accessible (e.g., personal data, anonymized data). Quality of published resources and providers (e.g., in terms of documentation, availability, completeness and reputation) as well as easy asset discovery should be supported to facilitate better interaction between consumers and providers. Ultimately, as cities consider citizens' trust as a key success factor, the system should provide tools that foster transparency on data usage and sharing, by tracking SLA agreements and by providing tools for auditing.

1.2.2.5 License

When sharing data with external parties, either by selling it or by offering it free of charge, it is fundamental to ensure that the data provider will keep control over their data. To support a dynamic ecosystem in which providers can establish various business models the system should provide several license models for their data, including both commercial and open licences. Furthermore, to better match the expectations of the stakeholders, the platform should offer data license templates with variable content, configurable based on the terms decided by the data provider. This will enable to define: (i) exclusivity of the data licence, (ii) business sectors which the data may be used for; (iii) geographical restrictions for the usage of data (iv) the period of validity of the authorization/right to access data, (v) the intended purpose that the data is used for; (vi) the authorization to resell data;

1.2.2.6 SLA

Many different stakeholders are part of the digital single market and different level of services may be required. The system should provide functionalities to define and manage extensible SLA for data access as well as provide common metadata to define SLA so that the management and the comprehension of the SLA descriptions can be simplified.

1.2.2.7 Feedback and Monitoring

Feedback, rating and reputation mechanisms are useful in order to provide a source of suggestions to improve data, services and applications deployed within the city, to facilitate the asset selection by the end users, and to build a reputation for the providers which can be exploited among different city marketplaces. Thus, the system has to provide a user feedback management for the different assets published on the marketplace, able to describe improvements and/or use experience and rate their quality. Moreover, the system has to provide advanced usage monitoring functions necessary in order to enable other services (e.g., usage statistics, revenue models, technical management).

Table 2 summarizes the Data Management and Service requirements by listing for each requirement its identifier, title, category and type (e.g., functional, non-functional):

ID	TITLE	CATEGORY	TYPE
SR-API-01	Standard and Open API	Data Management APIs	Non-Functional
SR-API-02	Publish/subscribe data channels	Data Management APIs	Functional
SR-API-03	Asset version management	Data Management APIs	Functional
SR-API-04	Resources status notification	Data Management APIs	Functional
SR-API-05	Lookup asset	Data Management APIs	Functional

SR-API-06	The system has to allow to access/consume data through RESTful API	Data Management APIs	Functional
SR-POLICY-01	Data and service access policy	Data Management APIs	Functional
SR-STORAGE-01	Physical data storage location	Data Storage Management	Non-Functional
SR-STORAGE-02	Categorization	Data Storage Management	Non-Functional
SR-LICENSE-01	Data licenses definition	Licence	Functional
SR-LICENSE-02	Customisable Licenses	Licence	Functional
SR-LICENSE-03	Pre-built Licenses	Licence	Functional
SR-SLA-01	SLA management	SLA	Functional
SR-SLA-02	SLA common metadata	SLA	Functional
SR-MODELS-01	Asset description taxonomies	Models	Functional
SR-MODELS-02	Standard and open data models	Models	Non-Functional
SR-MKTPLACE-01	IoT Data Marketplace access	Marketplace	Functional
SR-MKTPLACE-02	Asset publication procedure	Marketplace	Functional
SR-MKTPLACE-03	Flexible revenue and pricing models	Marketplace	Functional
SR-MKTPLACE-04	Asset catalogue	Marketplace	Functional
SR-MKTPLACE-05	SynchroniCity compliance policy validation	Marketplace	Functional
SR-MKTPLACE-06	IoT Data Marketplace transparency	Marketplace	Non-Functional
SR-MKTPLACE-07	IoT Data Marketplace federation	Marketplace	Functional
SR-FEEDBACK-01	User Feedback collection	Feedback and monitoring	Functional
SR-MONITORING-01	Data usage monitoring	Feedback and monitoring	Functional

Table 2: Data Management and Service requirements

1.2.3 Architectural security and privacy requirements

1.2.3.1 Platform security

Data and services can have different security requirements based on their scope. The platform which is going to support the services of the city should provide flexible security capabilities in order to accommodate the different needs of specific target scenarios, by providing support for confidentiality, integrity, authentication, authorisation, immutability, trust and non-repudiation when needed.

1.2.3.2 Data protection and privacy

Data protection and privacy issues should be addressed at several levels, from the low-level platforms to specific end user applications. Anyway, the SynchroniCity architecture has to provide basic primitives to assure data protection and privacy that will be enforced by the application level security.

The system should use encryption and technology to authenticate and secure data in transit as well as mitigate the risk of data theft by encrypting physical storage/media to protect data at rest. It is necessary to provide systems for monitoring against any attacks and if a breach occurs (e.g. data are accessed by unauthorized entities) the system should be able to properly react with defined procedures.

As data providers have the need to restrict the access of data source(s) to third parties, the system has to allow defining and managing policies for data and service access control. Both the data provider and the data consumer must comply with the privacy and data protection policy, thus the system should provide procedures and guidelines in order to ensure compliance with respect to data protection rules. In addition, the system should provide both data anonymization and aggregation functions in order to delete personal or restricted information.

1.2.3.3 IoT infrastructure security

The huge heterogeneity in the IoT devices capability (in terms of memory, computational, or energy requirements) makes it impossible to identify a “unique” or “common” security solution set, whereas they call for a large spectrum of security level versus resource consumption trade-offs. In order to support both new and legacy IoT devices, the system should provide end-to-end security at the API level rather than supporting and coping with how different solutions (e.g., LoRa, 802.15.4, NB-IoT, WiFi, LTE, GPRS, etc.) handle security measures such as key management, authentication, integrity and confidentiality. More specifically, the system should define adaptation policies of these mechanisms in the boundary points while assuring that security remains independent from low level IoT components.

Table 3 summarizes the Security and Privacy requirements by listing for each requirement its identifier, title, category and type (e.g., functional, non-functional):

ID	TITLE	CATEGORY	TYPE
SR-PRIVACY-01	Privacy policies guidelines	Data protection and Privacy	Functional
SR-PRIVACY-02	Data protection	Data protection and Privacy	Non-Functional
SR-PRIVACY-03	Anonymization	Data protection and Privacy	Non-Functional
SR-PRIVACY-04	Personal Data usage	Data protection and Privacy	Functional
SR-SECURITY-01	End-to-end secure communication	IoT infrastructure	Non-Functional
SR-SECURITY-02	IoT adaptation policies	IoT infrastructure	Non-Functional
SR-SECURITY-03	Access policy	Platform	Functional
SR-SECURITY-04	Flexible security capabilities	Platform	Non-Functional

Table 3: Security and Privacy requirements

2 Logical reference architecture

This section describes the logical architecture of the SynchroniCity framework. It defines logical components that address the requirements described in section 1.2 highlighting in more detail their functionalities and relationships. The last part of the section is focused on the concept of interoperability points, the main set of logical interfaces that have to be implemented by the different Reference Zones, in order to be compliant with the SynchroniCity framework and to benefit from its functionalities.

2.1 Architecture overview

This section will introduce the logical components of the SynchroniCity IoT Enabled Smart Cities reference architecture. The main aim of this architecture is to define a set of logical components and functionalities that can enable different cities to be actively part of IoT Smart City Digital Single Market. The reference architecture that will be presented in the next sections is the outcomes of different inputs and analysis included here and in other SynchroniCity project documents:

- **Architecture guidelines and use case analysis:** deliverable D1.3 [3] and D2.1 described a series of guidelines, use cases and requirements related to SynchroniCity scenarios. The list of requirements has been further improved in this final architecture deliverable taking in consideration also the new needs arisen by the analysis of RZs infrastructure and the worked carried in the WP3 to support the application design. Every architectural component has a direct relationship with one or more requirements that is explicitly indicated in its description.
- **Reference zones compliance:** D2.1 (see annex [1] for further details) studied the current Reference Zone technical architectures and infrastructure, matching every single city asset to specific logical layer (see annex [1] for further details). The SynchroniCity reference architecture recalls the layered approach proposed in the analysis of the RZ technical architectures performed in D2.1 (see annex [1] for further details) stressing the aspects of southbound integration, context management, security and privacy and northbound API. These main layers, further extended in the global reference architecture picture, can be considered the central pillars of all the RZs IoT architectures and for this reason are directly reflected into the SynchroniCity architecture.
- **Reuse of existent approaches:** SynchroniCity logical components try to cover the common functionalities identified in the most relevant smart city architectures, already analysed in D2.1 (see annex [1] for further details), maintaining a very generic approach that will allow cities to implement the architectural blocks in different ways reusing standard approaches.
- **Open & Agile Smart Cities (OASC) [2] principles:** the architecture has been designed following the OASC principles and the definitions of Minimal Interoperability Mechanisms (MIMs), simple and transparent mechanisms, ready to use in any city, regardless of size or capacity. MIMs, in terms of concrete implementation, are:
 - o a common standard API for **context information management**: the context data manager is a key component of the SynchroniCity architecture and the implementation of its NGSI API and the emerging NGSI-LD API is considered an “interoperability point” to enable cities to participate to the SynchroniCity framework.
 - o a common set of information models: semantic interoperability achieved through the adoption of **common data models** is introduced as a basic requirement to enable re-use of the data for building applications in different cities and domains.

- o a set of common standards data publication platforms: the role of data is crucial in SynchroniCity and for this reason the reference architecture includes **specific data management components** that aim to provide, through standard interfaces, all the functionalities related to data life cycle management.

The SynchroniCity logical reference architecture, depicted in the Figure 3 is the composition of different logical modules that are summarized below:

- **Context Data Management:** it manages the context information coming from IoT devices and other public and private data sources, providing a context data access through a uniform 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 to interact, through specific IoT Agents, with the devices that use different standards or protocols, making them compatible and available to the SynchroniCity framework;
- **Data Storage Management** provides functionalities related to the data storage and access in the specific context of IoT systems and smart city platform, interacting with heterogeneous sources.
- **IoT Data Marketplace** supports business interactions between data suppliers 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, revenue management. These functions will support the creation of innovative business models.
- **Security, Privacy and Governance:** this module covers all the security aspects related to three main pillars: data, IoT infrastructure and the platform services, which underpin the applications and services of the cities. Around these pillars, security functionalities provide crucial security properties such as confidentiality, authentication, authorization, integrity, non-repudiation, access control, etc.
- **Monitoring and Platform management services:** it provides functionalities to manage platform configuration and to monitor activities of the platform services. It supports specific KPI definition to evaluate the status of the platform in relation to different aspects (e.g. performance, usage, reliability, quality of service etc.)

Figure 3 shows also two other layers connected with the overall platform:

- **Southbound interfaces:** represents the set of interfaces defined by SynchroniCity used to connect the overall architecture to heterogeneous data sources and IoT devices. The southbound interfaces in SynchroniCity correspond to the Context Management API, and represents one of the relevant “interoperability points” that should be implemented by the Reference Zones to be part of the SynchroniCity ecosystem
- **Northbound interfaces:** is the set of APIs that provides all the 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 the end-users Smart City applications and services (examples based on some of the WP3 and open calls application domains) that will rely on the SynchroniCity functionalities and data: in particular, using same API and data models, the applications will be able to work and interact with the different Reference Zones, part of SynchroniCity ecosystem, with minimum customisation, simplifying the replications process. Specific details about the initial applications that will be piloted in the different RZ of SynchroniCity are included in D3.1 [23].

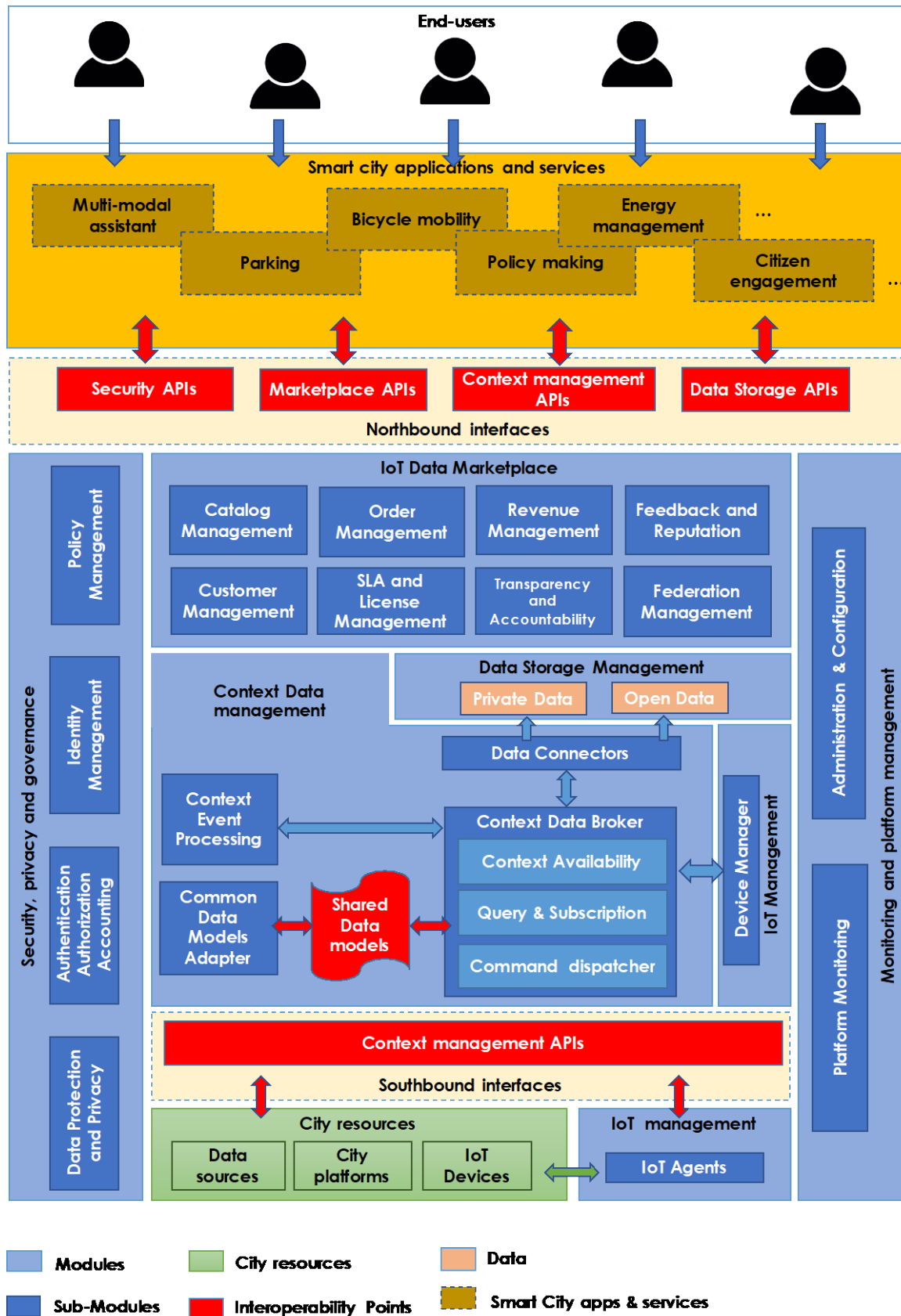


Figure 3: SynchroniCity Reference Architecture

2.2 Components description

The architectural components above mentioned will be described in details in the following sections.

2.2.1 Context Data Management

Context Data Management (CDM) is the component in charge of managing context information at large scale coming from IoT devices and other public and private data sources, providing a uniform approach and interface. This component plays a central role in the architecture as it represents a middleware that makes data from different data providers accessible to data consumers.

Context information are intended as structured data that contains status information of context entities and their related attributes and metadata. A context entity can represent everything in the real world; for instance, users, places, devices that can be abstracted and represented using a common data model. CDM provides functionalities to discover and register the context sources, in order to establish which context information is available and usable in the ecosystem.

CDM offers different ways to interact with data sources and external applications by implementing read and write access for context information. More specifically, for an application that plays a role of context data consumer, CDM implements a query mechanism, which allows building simple or complex queries, and a subscription mechanism to receive notifications when specific data is updated. At the same time an application could act on context data sources, playing the role of context data producer, because CDM provides functionalities to update context information changing attributes value of the entities.

CDM can support real-time event processing of context events by analysing event streams; in this case CDM analyses context information, consuming data as input events, in order to identify specific patterns and triggering related actions to generate responses to changing conditions. These responses can be considered output events that CDM will use to dispatch commands and update context information.

CDM exposes as interface a set of APIs to provide access to all the functionalities to manage context information. These APIs define operations and data structures to enable communication and exchanging of information between CDM and the other architectural components. CDM also provides additional features to adapt existing or legacy context data models to a set of common data models aligned with data structure defined by APIs, but enriched with specific attributes and metadata.

To enable data storing CDM provides connections to third-party storage solutions. Data Storage Management functional block allows view of context historical data.

Context Data Management is composed by the following functional modules, described in details in the next tables:

- Context Data Broker (Table 4)
- Context Event Processing (Table 5)
- Common Data Models Adapter (Table 6)
- Data Connector (Table 7)

Module Name	Context Data Broker
Module Description	Context Data Broker (CDB) enables discovering, gathering and publishing of context information through APIs. CDB, through its standard interface, makes available the context information regardless data source and using different type of interactions.
Main Functionalities	<i>Context Availability</i>: represents the operations to identify which context data sources are connected to SynchroniCity framework. <i>Query and Subscription</i>: represents the synchronous and asynchronous interactions with context data source. Synchronous interactions are performed using a query mechanism to obtain context information; the component allows building powerful queries, using different types of filters, in order to retrieve information with high level of precision. The asynchronous interaction is performed by publish-subscribe mechanism: a notification is generated when published data meets the subscription conditions; this feature is really useful to avoid the implementation of a polling process on data sources of interest, allowing to be notified when the context information changes. <i>Command Dispatcher</i>: through this function the Context Data Broker acts as an input channel for an IoT device that is able to receive commands from external system.
Requirements Mapping	SR-API-01 SR-API-02 SR-API-06 SR-INT-OPEN-01 SR-PERF-02 SR-ROBUSTNESS-01 SR-SCALABILITY-01
Interaction with other modules	This module interacts with: • <i>Context Event Processing</i>, exchanging the context information to detect specific events and managing the corresponding actions. • <i>Common Data Models Adapter</i>, exchanging the context information to obtain a translation in common data formats defined to be compliant with smart city applications. • <i>Data Connector</i>, providing context information to enable data storage.

Table 4: Context Data Broker module description

Module Name	Context Event Processing
Module Description	In a large scale IoT system, devices can generate several simple events; these events contain semantic information that could be quite limited to detect complex situations, so an additional processing logic is required. The Context Event Processing module processes a huge number of events and get valuable information from them. It is in charge of analysing context information in real-time in order to recognize specific event patterns and to enable adequate responses. It works by reacting not only to single events but also responding to a combination of events, in sequence or in parallel, triggering meaningful actions for applications or context update events.
Main Functionalities	<i>Pattern Management</i>: defines the rules to detect the types of possible patterns supported. Examples of pattern typologies are single event, sequence of events for a given time window, using aggregation function on one or more events, etc. <i>Event Processing Management</i>: defines the rules to analyse a set of events under specific conditions. Example of these conditions could be detection of events in a specific time interval, recognition of events that satisfy the same criteria about their attributes, etc. <i>Actions Management</i>: defines the rules to establish the actions responding to events, including the execution configuration.
Requirements Mapping	SR-API-06 SR-API-01
Interaction with other modules	This module interacts with the <i>Context Data Broker</i> receiving and analysing context information in order to find patterns and execute the actions to update the context information.

Table 5: Context Event Processing module description

Module Name	Common Data Models Adapter
Module Description	This is the module in charge of adapting heterogeneous context data to a specific data model belonging to specific set classified by application domains. The approach is to define a group of harmonized data models that cover the typical application domains of smart city applications. The module gets information stored in a specific context of Context Data Broker, or directly from southbound resources, publishing in the CDB a new copy based on a harmonized data model. This module plays an important role in terms of interoperability aspects, providing a way for application providers to define their applications using these models and ensuring a greater level of replicability of them in SynchroniCity Reference zones.
Main Functionalities	<i>Data Models Configuration</i>: allows defining the structure of the new data model to use in the mapping phase. <i>Data Models Mapping</i>: allows data manipulation (e.g. transformation, elaboration, filtering, merging) in order to adapt them to the specific data model. <i>Publishing</i>: allows publishing on specific context of the Context Data Broker the result of harmonisation of an entity with a supported data model.
Requirements Mapping	SR-MODELS-02 SR-INT-OPEN-01
Interaction with other modules	This module interacts with the Context Data Broker to receive the context information and to provide it the translated version.

Table 6: Common Data Models Adapter module description

Module Name	Data Connector
Module Description	The Data connector module is responsible to enable storage of context data. Context aware approach makes available information about the current status of relevant entities, but the Context Data Broker itself isn't able to provide consumers with historical data when such data are required. The Data connector module should provide a pluggable system that allows to feed external storage systems or other data analysis tools.
Main Functionalities	<i>Plugin management:</i> this functionality allows simultaneous use of multiple plugins. <i>Plugin:</i> in charge to enable storage functionalities supporting a specific storage system technology
Requirements Mapping	SR-PERF-02
Interaction with other modules	This module interacts with the Context Data Broker to receive the context information.

Table 7: Data Connector module description

2.2.2 IoT Management

IoT Management is the component responsible to interact with the devices that use different standards or protocols making them compatible and available to the SynchroniCity architecture; through its interface this component provides to the Context Management the data coming from the RZ IoT devices and platforms, handling the creation and the updates of Context Entities and their attributes.

The core components of the IoT Management are the IoT Agents, representing the software modules that implement the concrete interface with the devices; each agent handles a specific protocol and provides to Context Management the context entities with related information, by creating an entity for each device connected to the platform. IoT Management allows managing the IoT Agents by using centralized administration functionalities; in this way, each operation related to creation, configuration, monitoring of agents leverages a single administration point simplifying the overall IoT Agents management.

Table 8 and Table 9 reports description about the related sub modules (IoT Agents and Device Management).

Module Name	IoT Agents
Module Description	An IoT Agent is a software module that enables sensors and actuators to send their data using their own native protocols, and translates them into Context Data Management API to be managed by a Context Data Broker. IoT Agents should also be able to deal with security aspects (authentication and authorization of the channel) and provide other common services to the device programmer; this is the module that represents the whole set of IoT Agents covering all the protocols and standard communication interfaces exposed by devices of Data Sources layer.
Main Functionalities	Managing of information exchanged with devices (e.g. measure reading). The agent is in charge to mediate between the raw data coming from the device and the virtual entity representation at the context level. Execution of commands or actions by an actuator device. Mapping of device and its features to a context entity with related attributes and metadata Manage entities lifecycles at the Context Management level
Requirements Mapping	SR-COMM-01 SR-COMM-02 SR-LEGACY-01 SR-SECURITY-02
Interaction with other modules	This module interacts with the Context Data Broker to provide context entity information and manage queries, subscriptions and commands executed on devices.

Table 8: IoT Agents module description

Module Name	Device Management
Module Description	This module is in charge of managing configuration of the IoT Agents to set parameters needed in southbound connection to the devices, depending on the specific protocol used, and managing publication of data observations to specific contexts associating the measurements to the related context entities; this module is used only for connections to devices that not support the Context Management APIs.
Main Functionalities	<i>Device Registration/Creation:</i> before sending observations, the devices have to be registered. This registry can include several information such as the Device ID, Entity ID, Entity type, attributes, static attributes and command parameters related to the device. <i>Device Observation:</i> once a device is registered, observations might be sent with a simple HTTP request. <i>Monitoring:</i> this functionality includes all the activities about the device status checking.
Requirements Mapping	SR-COMM-02 SR-LEGACY-01 SR-LEGACY-02
Interaction with other modules	This module interacts with the Context Management in order to provide the endpoints where the data are published and to manage the appropriate mapping to context elements.

Table 9: Device Management module description

2.2.3 Data Storage Management

The Data Storage Management aims to solve the issues related to the data storing and homogeneous access inside the SynchroniCity architecture.

Data coming from various heterogeneous sources can differ in terms of types, characteristics and constraints imposed by their owners about their exploitation.

The SynchroniCity project wants to address these challenges by enabling the usage of data in smart city applications through a secure and trusted environment according to a defined set of management policies. In particular, the Data Storage Management provides the functionalities related to the uniform access to different storage technologies (e.g. different databases) exposing a uniform API, the Data Storage API, to access to historical data and open data.

To make this possible, a set of functionalities to configure, provision and report storage activities have to be made available (through the security component), also to ensure data security and quality.

In terms of data security, several different aspects can be considered such as data anonymization, by classifying types of data that request such treatment.

In addition to these aspects, it is important to consider that the promotion of data exploitation and use requires a certain quality level of the data to be guaranteed. Hence, Data Storage Management could also provide functionalities related to data cleansing and in general tools to checks data quality.

For the purpose of this module, two main categories of data managed can be identified:

- Public or Open Data: data provided without restrictions or provided with open licence on access, use and share.
- Private Data: data provided with restrictions about their usage; this category includes for instance personal data.

The Data Storage Management interacts with the IoT Data Marketplace providing the access to the specific data managed in the Marketplace catalogue.

2.2.4 IoT Data Marketplace

A Marketplace is a brokerage site that favours the meeting between demand and supply of goods and services. Different kinds of marketplaces exist: e-commerce, B2B, C2C, B2C, m-commerce. The SynchroniCity architecture includes an IoT Data Marketplace component that will encourage sustainable commercial viability of data by developing a considerable added value that goes beyond traditional rights-based licensing models of data sets. It will also be responsible for the revenue, billing and charging management and enabling the creation of innovative business models. Eventually, the marketplace will implement mechanisms to gather user rating and feedback on the assets, and to support trust and accountability of the publishers. Detailed information about IoT Data Marketplace are included in [7].

It is important to underline that SynchroniCity identifies also another marketplace, the *IoT Product Marketplace* that can be considered an e-commerce platform supporting B2B and B2C for IoT devices and solutions for the Smart City infrastructure. Not only providing e-commerce on the IoT products used in Smart City, the IoT product Marketplace provides Smart City stakeholders to share the knowledge of Smart City infrastructure information and to exchange implementation experiences. Device manufactures, vendors and service providers can register and provide their Smart City products and the municipalities can share their infrastructure information per service to be referred to the other cities who have the plan to develop similar cases. As the IoT Product Marketplace is a type of service that SynchroniCity offers to the stakeholders and the inputs are not directly from the sensing data but from external source (e.g., from IoT device manufacturers, vendors, etc.), it is not included in the technical architecture like the IoT Data Marketplace, which is created from the Smart City data being processed in the SynchroniCity framework. All the details related to the IoT Product Marketplace will be included in deliverable D2.7 [24].

It is possible to define nine functional modules for the IoT Data Marketplace:

- Catalogue management (Table 10)
- Offers/Orders management (Table 11)
- Federation management (Table 12)
- Revenue management (Table 13)
- Feedback and reputation service (Table 14)
- Customer management (Table 15)
- SLA and license management (Table 16)
- Transparency and accountability service (Table 17)

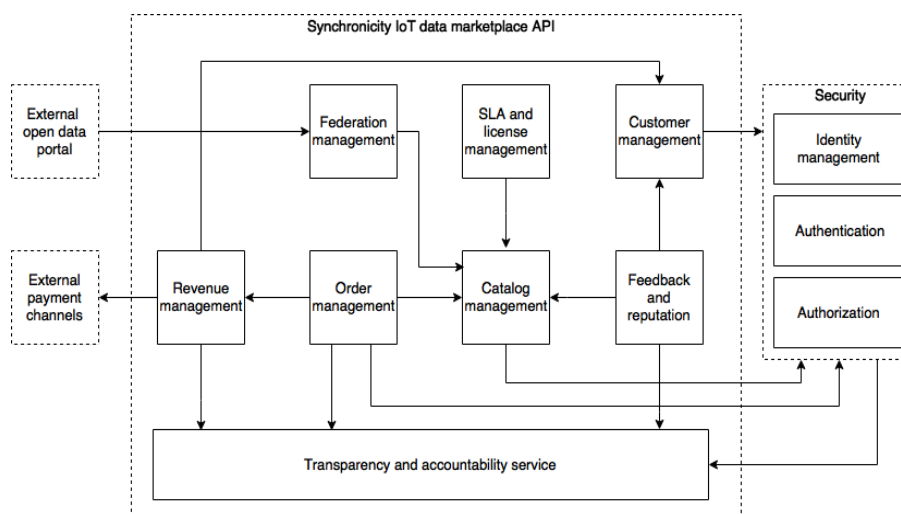


Figure 4 - Conceptual model of the SynchroniCity IoT data marketplace [7]

Module Name	Catalogue Management
Module Description	This module provides functionalities to publish and search for different data offerings. Data offerings can be organized into groups/categories - in a hierarchical fashion when possible - to allow for an easy navigation and discovery of them. Attributes define characteristics and properties of data offerings. They may also be inherited from a higher level in a category hierarchy. The module allows data providers to define the technical description of the data offerings they own as well as information related to the offering terms such as price, SLA, license, etc.
Main functionalities	Data catalogue curation: allows the marketplace provider to create and update both catalogues and categories in order to better organize and ease the discovery process of data offerings. Data source specification: allows data providers to register a new data source specification (prior validation to ensure the data provider ownership of the published resource) by detailing its description (e.g., version, data model, endpoint URL). Data offering publication: allows data providers to register a new data offering by linking a previously specified data source specification to pricing information, Service Level Agreements (SLAs) and license terms as well as defining the associated catalogue and categories. Data offering discovery: a list of available data offerings can be retrieved and refined by specifying keywords and filters that match description, characteristics and properties of the desired digital asset. As a result, data consumers can easily discover what kind of data offerings are available in the marketplace.
Requirements Mapping	SR-MKTPLACE-02, SR-MKTPLACE-04, SR-MKTPLACE-05 SR-MKTPLACE-06 SR-PERF-01, SR-MODELS-01 SR-API-05, SR-API-04, SR-API-03
Interaction with other modules	The <i>catalogue management</i> interacts with the <i>SLA and license management</i> to bind an SLA and a license respectively to a new data offering. It interacts with the order management module when registering new orders. Information is also exchanged between this component and the <i>feedback and reputation</i> one to retrieve feedback, rating and reputation score for every digital asset. The <i>catalogue management</i> also interacts with the <i>federation management</i> to retrieve metadata information from external open data catalogues or other data marketplaces, so that different data offerings can be published within a single federated marketplace. Whenever a data provider specifies a data offering for a protected data source, this component interacts with the <i>security</i> components to check data provider's permission for that specific data source.

Table 10: Catalogue Management module description

Module Name	Orders Management
Module Description	This module allows to order and acquire data offerings and managing acquired data sources. More specifically, a data consumer interested in purchasing a data offering available in the catalogue can place an order to finalize the purchase of that digital asset. To create a non-repudiable and clear proof regarding the terms of the agreement (i.e., SLAs and license terms) between a data consumer and a data provider, the latter are required to provide a digital signature of the order. This API allows to perform operations such as unsubscription, activation, deactivation, renew, by updating the data consumer role in the identity management according to the order status and the data consumer preferences.
Main functionalities	Ordering of data offerings: allows data consumers to purchase a specific data source offered through an offering available on the data marketplace, which can be a static batch of data or a real-time data streamed by one or more data sources. Management of purchased data offerings: allows data consumers to keep track of the assets purchased through the marketplace.
Requirements Mapping	SR-PERF-01 SR-MKTPLACE-03
Interaction with other modules	The <i>order management</i> interacts with the revenue management component to enable monetization mechanisms, with the <i>transparency and accountability service</i> to track data usage information, and with the <i>security</i> components to update permissions of data consumers when purchasing data offerings.

Table 11: Offers/Orders Management module description

Module Name	Federation Management
Module Description	This module is in charge of managing a set of federation capabilities in accordance with the marketplace governance. Federation capabilities allow different marketplaces to interact with each other and access their resources indistinctly to provide access to data offerings across them and enable the development of aggregated services.
Main functionalities	Federation configuration: this function allows to define all the parameters related to the data offerings federation among marketplaces. For each data offering, a dissemination level parameter can be specified to enable the federation/inclusion of the data offering into other marketplaces (e.g. private, public, restricted to a specific group/network of marketplaces) and the set of information that can be shared for the related data source. Other technical parameters can be also configured to enable federation (e.g., security). Federation discovery: allows to specify all the information needed to be federated. Such information includes number/list of data offerings that can be federated, level of dissemination of the data source (e.g., specific groups or networks), last update date, security and technical access information. Every marketplace can discover the other ones by calling the federation discovery API. Data offering federation: this function provides the metadata of the data offerings exposed by a marketplace in order to be federated by other ones. The metadata exposed can be a subset of the complete metadata information of the data offering included in the original marketplace. The metadata retrieved using this function will be stored by the recipient marketplace that asked for it. Import external data catalogues: this function provides the support for retrieving metadata information related to external data catalogues. It can be used as a tool to automatically gather and expose all the datasets already published and available on different data portals directly through the marketplace.
Requirements Mapping	SR-MKTPLACE-07
Interaction with other modules	The <i>federation management</i> interacts with the <i>catalogue management</i> to retrieve metadata information from external open data catalogues or other data marketplaces, so that different data offerings can be published within a single federated marketplace.

Table 12: Peering Management module description

Module Name	Revenue Management
Module Description	This module allows data providers to generate revenue for their offerings by charging data consumers for purchasing them. It provides tools to manage data usage information in order to enable usage-based business models. It exposes an interface to interact with external charging platforms such as PayPal. It collects all the information required for the charging process (price, data usage, consumer identifier, etc.), which may differ according to the pricing model associated with the data offering and the outcome received by the external charging platform.
Main Components	Charging management: provides the charging functionality to the system by interacting with one or multiple charging platforms (e.g., PayPal) and performing the required actions to charge the data consumers for purchasing data offerings provided by different data providers. Management of data usage specification: allows the marketplace provider to support different data usage pricing model (e.g., Mbytes, seconds, number of calls, etc.). Revenue sharing management: allows to define revenue sharing models to distribute revenues between the involved stakeholders (e.g., revenue shared between data provider and data marketplace provider as the transaction fee). Billing management: is in charge of sending invoices to asset consumers for their purchases. The invoicing process starts when a purchasing order is completed. In case of static batch of data or services, a single invoice is sent to the consumer. Whereas, in case of real-time data, invoicing can be done through time-triggered transactions.
Requirements Mapping	SR-MKTPLACE-03
Interaction with other modules	The <i>revenue management</i> interacts with the <i>order management</i> as well as with external payment channels to enable monetization mechanisms by exchanging transactions outcome and charging information. It also interacts with the <i>customer management</i> to retrieve users' billing information as well as with the <i>transparency and accountability service</i> to record transactions information.

Table 13: Revenue Management module description

Module Name	Feedback and reputation
Module Description	This module provides user feedback management for the different data offerings published on the marketplace. It also provides rating and reputation mechanisms to support data consumers in selecting the data offerings and to promote an honest behaviour among users.
Main Functionalities	User feedback: allows users to provide feedbacks on data offerings they have purchased. Feedbacks will be based on the quality and reliability of data sources as well as on their compliance to the related SLAs. In case of data streaming or services running for extensive periods, data consumers will be allowed to adjust their feedbacks periodically according to up-to-date levels of service. Data offerings rating: is in charge of building and maintaining a ranking of data offerings with respect to feedbacks received by the data consumers. Data provider reputation: is in charge of building overall reputations of data providers according to the rating scores of their data offerings.
Requirements Mapping	SR-FEEDBACK-01
Interaction with other modules	The <i>feedback and reputation</i> module interacts with the <i>catalogue management</i> and the <i>customer management</i> to exchange information regarding feedback, rate and reputation of data offerings and customers. It also interacts with the <i>transparency and accountability service</i> to track information on feedback, rate and reputation.

Table 14: Feedback and reputation module description

Module Name	Customer Management
Module Description	This module is in charge of identifying and gathering information about the users of the marketplace. It provides tools to manage customer information and related parties, which are the legal entities associated with the customer accounts. Depending on the access restrictions for the marketplace defined by the marketplace provider (e.g., city council, consortium, 3rd party), customers can be created and linked to specific roles (e.g., data provider, data consumer, administrator, etc.).
Main functionalities	Management of customer information and accounts: provides methods for the creation, retrieval, update and deletion of customer information and accounts. Management of parties: provides methods for the creation, retrieval, update and deletion of parties.
Requirements Mapping	SR-PRIVACY-04
Interaction with other modules	The <i>customer management</i> interacts with the <i>identity management</i> component to retrieve users' information. It also interacts with the revenue management to retrieve users' billing information.

Table 15: Customer Management module description

Module Name	SLA and license Management
Module Description	This module allows data providers to set, define and customize different SLAs and licenses for data offering published on the data marketplace, thus enabling the creation of a dynamic ecosystem in which data providers can establish various business models. It provides an interface to retrieve pre-defined data license templates so that data providers can link a data usage license instance selected among the available templates to the related data offerings. If the license templates do not fulfil the data provider needs, this API allows to customize them or create new ones in order for the license to better reflect the business model requirements. For instance, customizable templates allow to define: (i) business activity sectors for which the data may be used, (ii) purposes for purchasing and using the data, (iii) authorization to resell the data, (iv) geographical territories in which the data may be used and, (v) the date after which the authorization period to use the data ends.
Main Functionalities	License definition and customization: allows data providers to define different licenses templates based on standard licenses (e.g., GPL, Apache, Creative Commons) or based on custom models according to the specific business models chosen by the data providers. SLA specification: allows to define and manage extensible SLA for published

	data offerings in order to satisfy different stakeholder requirements. It allows to define SLAs for data offerings published in the data marketplace (e.g., delivery, timeliness, completeness, etc.).
Requirements Mapping	SR-LICENSE-01 SR-LICENSE-02 SR-LICENSE-03 SR-SECURITY-03 SR-SLA-01 SR-SLA-02
Interaction with other modules	The <i>SLA and license management</i> interacts with the <i>catalogue management</i> to bind an SLA and a license respectively to a new data offering.

Table 16: SLA and license Management module description

Module Name	Transparency and accountability
Module Description	This module provides tools for auditing orders (including pricing model, license terms, SLAs) and tracking the parameters defined by SLAs.
Functional Components	The main functionalities provided by this module are: • Auditing orders • Tracking SLAs
Requirements Mapping	SR-MKTPLACE-06
Interaction with other modules	This module interacts with the revenue management to record transactions information, as well as with the <i>order management</i> Feedback and Reputation service, Offer/Order management and Customer management modules to exchange information on transactions and transparency within the marketplace.

Table 17: Transparency and accountability module description

2.2.5 Security, Privacy and Governance

Security in the SynchroniCity framework can be related to three main security pillars: Data, IoT infrastructure and the overall basic Platform. Alongside these pillars, security functionalities provide crucial security properties such as confidentiality, authentication, authorization, integrity, non-repudiation, access control, etc.

A flexible and manageable security configuration can be tailored to address specific application or service security requirements considering the large heterogeneity of IoT devices exposing data and control functionalities while being compliant with the different governance, policies and procedures of each city.

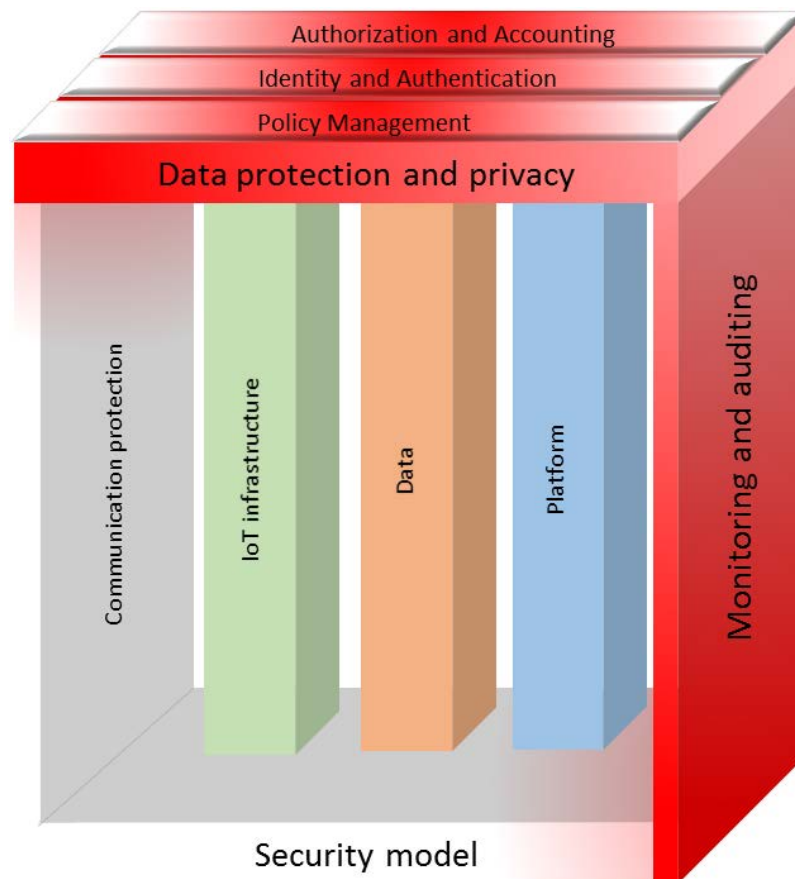


Figure 5: Security Technological pillars

Security layer contains a set of modules aimed to ensure data protection and privacy, manage identities, authenticate and authorize users accessing assets according to the RZ' governance:

- Data protection and Privacy (Table 18)
- Identity and Authentication Management (Table 19)
- Authorization and Accounting (Table 20)
- Policy Management (Table 21)

Module Name	Data protection and Privacy
Module Description	This module provides flexible security and privacy capabilities in order to accommodate the different needs of specific target scenarios, by providing support for confidentiality, integrity, authentication, immutability and non-repudiation. It provides cryptographic mechanisms to authenticate and secure data in transit as well as mitigating the risk of data theft by encrypting physical storage/media to protect data at rest.
Main functionalities	<i>Data protection:</i> provides encryption and key management functionalities to ensure that data are secure while being in transit and at rest. <i>Privacy:</i> provides functionalities in line with the privacy by design principle. It ensures that the system respects privacy during its operation. The actual level of privacy protection depends on the actual policy. Any personal data, and their interrelationships, are hidden from plain view, thus they cannot easily be abused. By separating the processing or storage of several sources of personal data that belong to the same person, complete profiles of one person cannot be made while also achieving purpose limitation. Moreover, data is processed at the highest level of aggregation and with the least possible detail in which it is (still) useful. Aggregation of information over groups of attributes or groups of individuals, differential privacy and/or noise injection will minimize the probability of identifying a specific subject, thus protecting its privacy. Moreover, this module manages the process to inform data subjects whenever personal data is processed, also providing the right to view, update and even ask the deletion of personal data collected about her. It also provides a tool to show how the privacy policy is effectively implemented within the system.
Requirements Mapping	SR-PRIVACY-01 SR-PRIVACY-02 SR-PRIVACY-03 SR-PRIVACY-04 SR-SECURITY-04 SR-SECURITY-01
Interaction with other modules	This module interacts with the Customer Management and with the Transparency and Accountability modules to exchange information regarding the access and process of personal data. It also interacts with the Data Storage Management to enforce data protection and anonymization according to the policies defined in the Policy Management module.

Table 18: Data protection and Privacy module description

Module Name	Identity and Authentication Management
Module Description	The Identity Management module is the first step for accessing data, services and applications, by providing secure and private identification and authentication of users, trust management, and Single Sign-On (SSO) to service domains and Identity Federation towards applications.
Main functionalities	Management of user life-cycle functions by providing account creation and management, and enforcement of policies and procedures for user registration, identification and authentication. Support to different authentication providers and several applications can be linked to this module, thus enabling single sign-on (SSO) to all these applications.
Requirements Mapping	SR-MKTPLACE-01 SR-SECURITY-01 SR-SECURITY-04
Interaction with other modules	This module interacts with the Authorization and Accounting component to exchange information on access request to assets.

Table 19: Identity and Authentication Management module description

Module Name	Authorization and Accounting
Module Description	This module provides authorization and accounting capabilities which are critical aspects to support Smart City services and applications. It enforces a set of conditions defining whether users have access granted to a particular resource, while also storing information regarding access for audit purposes.
Main Functionalities	<i>Authorization:</i> provides a Policy Enforcement Point (PEP) which intercepts resource access requests, makes access control decision requests, and enforces access control decisions. Moreover, a Policy Decision Point (PDP) evaluates access request by checking authorization policies for rendering an access control decision. It provides also a Policy Retrieval Point (PRP) that connects to the Policy Management component and a Policy Information Point (PIP) to obtain applicable authorization policies according to an access control decision request and attributes that are needed for evaluating authorization policies, for example the IP address of the requester, creation time of the resource, current time or location information of the requester. This information is then combined in order to get a final access control decision. <i>Accounting:</i> measures resource consumption performed by users during access. This includes the amount of system time or the amount of data a user has sent and/or received during a session. Accounting is carried out by

	logging of session statistics and usage information. It is used also for authorization control, billing, and resource utilization.
Requirements Mapping	SR-SECURITY-01 SR-SECURITY-04 SR-MONITORING-01
Interaction with other modules	This module interacts with the Identity and Authentication, and with the Policy Management component to exchange information related to access request to assets and to make authorization decision based on defined access policies. It also interacts with the Transparency and Accountability component to exchange information on accesses and usage so that the latter component can store and provide a way to show that data, application and services are accessed and used according to their privacy and SLA requirements.

Table 20: Authorization and Accounting module description

Module Name	Policy Management
Module Description	This module provides a unified policy management regarding access, privacy and governance of the SynchroniCity framework. By providing a unified and central policy management point, access policies updates have an immediate result on how the authorization component performs decision. In addition, this component allows to define adaptation policies of security functionalities in the boundary points thus being independent from low level IoT components.
Main functionalities	The Policy Management component allows to define policies related to privacy, access, data protection and federation.
Requirements Mapping	SR-SECURITY-02 SR-SECURITY-03 SR-SECURITY-04
Interaction with other modules	This module interacts with each security and privacy components as a point providing policies to be enforced. It also interacts with the Peering management to exchange information regarding assets shared among different RZs marketplace.

Table 21: Policy Management module description

2.2.6 Monitoring and Platform management service

SynchroniCity offers a set of functionalities to manage platform configuration and to monitor activities of the components.

In terms of platform configuration SynchroniCity allows setting of all the parameters related to components functionalities and interaction. In addition to this, another set of parameters is related to monitoring functionalities in particular needed for platform KPI calculation.

In terms of platform monitoring, SynchroniCity establishes a set of quantitative and qualitative metrics to extract meaningful performance indicators to have an overview of the status of the platform in relation to different aspects (e.g. performance, usage, reliability, quality of service, etc.).

Calculation of KPIs is based on collection of specific logs and execution of ad hoc algorithms; this process is related to parameters configured by an administrator, for instance, time interval, constants, threshold values and other variables used in the calculation.

In general, the monitoring of the platform aims to collect and elaborate the information to make it available for each type of analysis.

Table 22 and Table 23 describe the details about each module: Administration & Configuration and Platform Monitoring.

Module Name	Administration & Configuration
Module Description	This is the entry point to set parameters related to SynchroniCity framework and its architecture components, like Context Management, IoT Management, etc. defining their behaviour The configurations relate both to the components' functionalities, integration and all the parameters needed to setup and start the platform monitoring processes.
Main Functionalities	<i>Platform and Component configuration:</i> allows to setup server configuration, deployment configuration etc. It is necessary in order to run services and applications of the platform, or to enable/disable specific functionalities of the platform. <i>Monitoring Configuration:</i> allows to configure parameters such as time intervals, threshold values, etc. necessary to create the log files used in KPI calculation.
Requirements Mapping	SR-MODULARITY-01
Interaction with other modules	This module interacts with all the modules of the architecture in order to propagate the configuration parameters.

Table 22: Administration & Configuration module description

Module Name	Platform Monitoring
Module Description	This is the module in charge of running the monitoring processes of the platform in order to check the quality of services and applications and extract statistics and indicators that provide an overview on the performances and reliability of the platform.
Main Functionalities	<i>Log analysis and KPI calculation:</i> takes into account the logs generated at platform runtime and elaborates them by applying specific algorithms.
Requirements Mapping	SR-ROBUSTNESS-01 SR-MONITORING-01
Interaction with other modules	This module interacts with all the modules of the architecture that provide logs for KPI calculation.

Table 23: Platform Monitoring module description

2.3 Interoperability points logical interfaces

In the SynchroniCity architecture, a key role is played by the interoperability points that represents the main interfaces to access to the functionalities of the framework and in particular the way to consume and provide data. The interoperability points assure also the replicability of the solutions built on top of the SynchroniCity framework that being compliant to them, are completely decoupled from the specific technological implementation and deployment of the architectural components.

It is possible to identify the following categories of interoperability points:

- Context Management interface
- SynchroniCity Data Models
- Data Storage Interface
- IoT Data Marketplace Interface
- Security Interface

In the following sections are presented the logical definition of the interoperability points, but SynchroniCity defined also a concrete API specification, based on widely-adopted standards (see section 4). The compliance with these specifications represents the main requirement to enable the interoperability between RZ and SynchroniCity (e.g. communication between existent system and SynchroniCity framework): at the same time the adoption of the specifications at northbound level allow third party applications and services to interact with the platform using its capabilities and resources.

2.3.1 Context Management Interface

Context Management APIs that represent a standard way to communicate with the Context Management module, a key component of the architecture.

The main concept behind managing of data and information in SynchroniCity, is the adoption of an abstraction level that represents any information related to the real world (e.g. sensors data) as a “context entity”. characterised by specific attributes and metadata.

The logical model of these interfaces is based to NGSI meta model [25], which offers a powerful way to model the real world, in order to extract high quality information from observation of phenomena. For instance, modelling IoT systems, each device or data source can be mapped to NGSI model and analysed by using different ways. Figure 6 depicts how real-life objects can be mapped using NGSI approach:

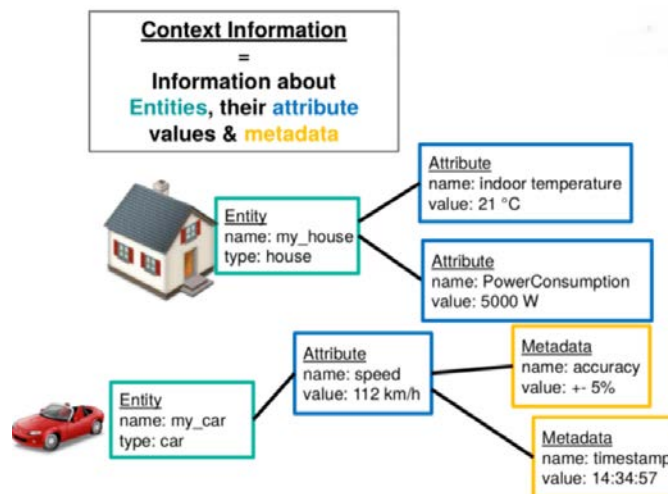


Figure 6: NGSI Context Information example [26]

This approach is coherent with the interoperability requirements of SynchroniCity framework and it doesn't bind to the use of predetermined data models. In relation with the SynchroniCity architecture, Context Management interfaces can be considered at both southbound and northbound levels.

Context Management interfaces at southbound level represents the way to feed SynchroniCity framework with the data coming from the city IoT infrastructure: considering the interoperability aspects from southbound connection point of view, be compliant with SynchroniCity means that the RZs (or other stakeholders that want to connect their IoT/southbound resources to SynchroniCity) have to be able to communicate with the Context Management module using its APIs. Specific guidelines about the IoT/southbound integration are included in the deliverable D2.6 [9]

From the northbound side, the Context Management interfaces mainly allow, end-user applications and services to consume data, manage context entities and their attributes, provided by SynchroniCity.

The approach followed by Context Management Interface is intended to manage the entire lifecycle of context information, including updates, queries, registrations, and subscriptions, according to the NGSI specification.

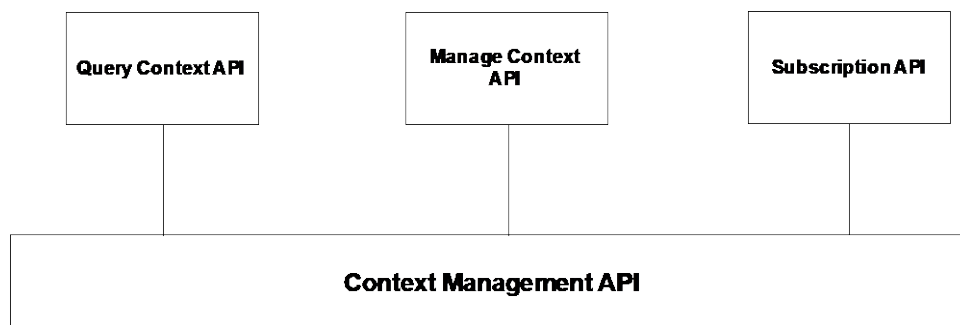


Figure 7: Context Management API

More specifically, the Context Management interface, as depicted in Figure 7, consists of the following set of APIs:

Manage Context API exposes methods for the synchronous 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-values pairs. In addition, it will be possible either to update or remove one or more entities, as like as only specific attributes from each entity.

Query Context API provides methods for the retrieval and discovery with different search filters of entities present in the Context Management module (Context Data Broker). A list of available entities can be retrieved and refined by specifying further filters that match entities metadata (e.g. id, entity type, etc.) or attributes values. A query is composed by a list of statements, each of them expresses a matching condition. The query will return all the entities that match all the matching conditions.

Subscription API provide methods to manage the subscriptions to asynchronous notifications about entity updates. It enables to subscribe to specific context entities, so when some update occurs on them, the client will get an asynchronous notification. In detail, when creating a new subscription, for each entity will be issued the update condition for one or more attributes and the expiration time.

2.3.2 Data Storage Interface

To ensure access to the data managed by the framework, the SynchroniCity defines the Data Storage API, depicted in Figure 8.

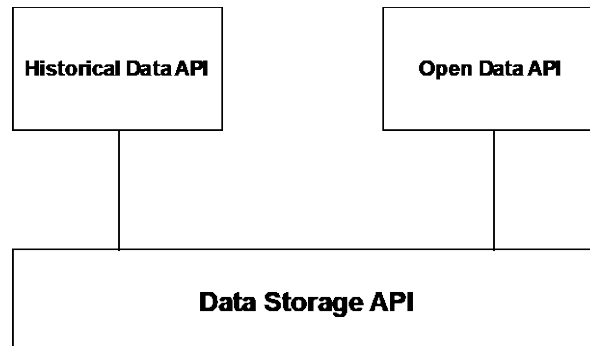


Figure 8: Data Storage API

This kinds of data API can be categorized in the following:

- Historical Data API
- Open Data API

Historical Data API provides methods to retrieve both historical raw and aggregated time series information about the evolution in time of context data (i.e., entity attribute values) registered in the Context Management module. These series can be provided according to specific temporal aggregation methods, such as mean, max, min values, etc.

Open Data API provides a unique access point to search and discover Open Data sets coming from different Open Data Management Systems (ODMS) portals, belonging to the RZs. The datasets are imported in the SynchroniCity framework and related metadata are uniformed to a standard format.

2.3.3 SynchroniCity Data Models

The adoption of common Data Models is a crucial interoperability point. SynchroniCity provides specifications and tools to enable data models interoperability: a set of guidelines to define common data models, as concrete extension of the OASC Minimal Interoperability Mechanisms (MIMs), and a catalogue of concrete data models for the verticals and Smart City application domains. Example of these domains can include, for instance, mobility, weather, environment, risks, parking etc. (Figure 9). Each of them defines a set of context entities and related attributes that can be mapped to generic context entities managed by Context Management. This approach allows a one-to-one mapping between generic entities and domain entities. Likewise, also is also possible a one-to-many mapping: from a generic entity definition, it is feasible to obtain its representation in a specific domain using two or more complementary domain entities.

The adoption of SynchroniCity data models is relevant for both southbound and northbound interactions. Figure 9 shows an example in which different applications, in different domains, uses the same data models to ensure interoperability: in this way, for instance a software developer can design and develop an application, for a specific use case (and city) with a high level of interoperability. If a new city (RZ-2) supports, the same SynchroniCity data models, the application initially designed for RZ-1 can be easily reused in RZ-2 with low customisation effort. The adoption of common data models and API gives so is also one of the technological enabler for a common

digital single market for services and applications.

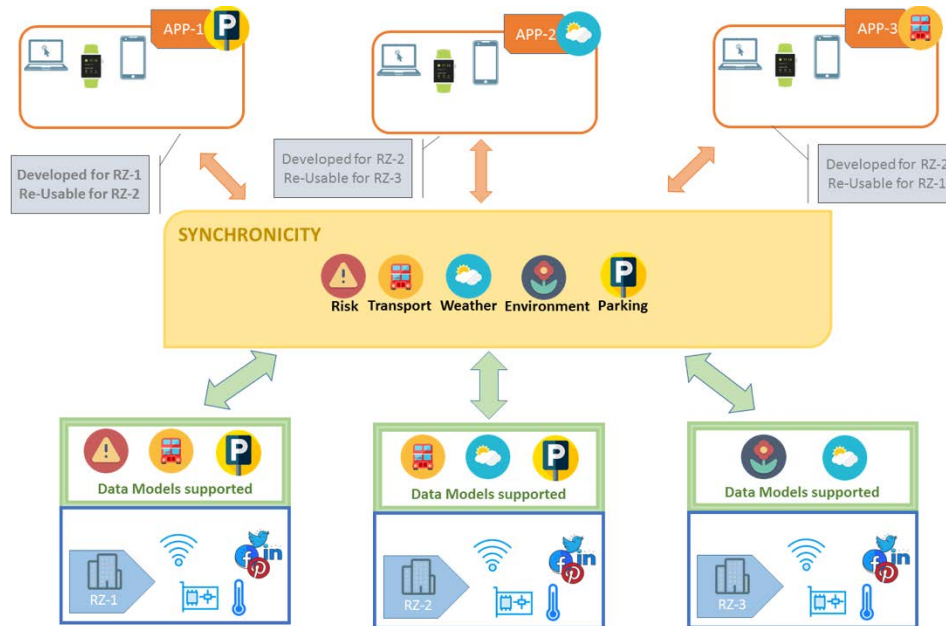


Figure 9: Different cases to enable Data Models Interoperability

Further details about data models guidelines can be found in the SynchroniCity deliverable D2.2 [5]

2.3.4 IoT Data Marketplace Interface

The *marketplace API* provides a hub to enable exchange and trading of digital asset within the SynchroniCity framework. It includes several modules that expose a set of interfaces and services for enabling the management and monetization of urban IoT data. An overview of the *marketplace API* is shown in Figure 10 and a description of its components is provided next.

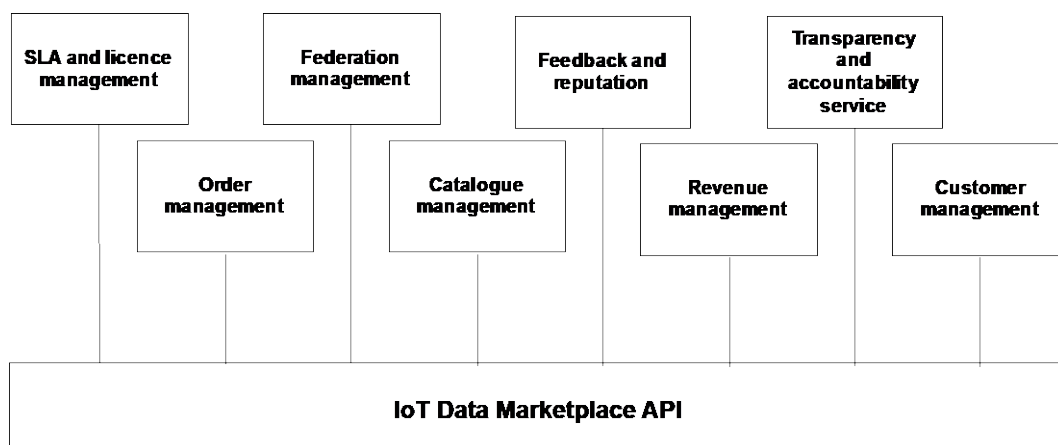


Figure 10: IoT Data Marketplace APIs

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

Catalogue management APIs are logically grouped into four sub-modules:

- Category management. A Category is used to group catalogues and data offerings in logical containers. Categories can contain other categories being possible to create a tree of categories.
- Catalogue management. These catalogues are collections of data offerings that are grouped together according to the data marketplace and data providers needs.
- Data source specification management. A data source specification is a detailed description of a data source made available in the form of a data offering to customers playing a data consumer role.
- Data offering management: data offerings represent entities that are orderable from data providers and are published in the catalogue. This resource includes pricing information and is linked to SLAs and license terms.

Federation management APIs allow to create a network of federated marketplaces that exposes a common catalogue of data offerings. It is important to highlight that the federation should be considered at catalogue level: all the other functionalities, in particular the ones related to the finalization of data offering purchases, are managed by the original marketplace that “owns” the original data offering: this approach implies that security and user profile information has to be shared among the different marketplaces that are part of the federation.

Order management APIs are logically grouped into order management and inventory management:

- Order management. Orders are made by data consumers, and include a set of order items, each specifying a data offering to be acquired. When creating an order, customers can select the value of the different pricing options (if available) to be applied and agree on SLA and license terms.
- Inventory management API. It allows data consumers to retrieve information of the data sources they have acquired, including the specific characteristics and pricing model selected.

SLA and license management APIs are logically grouped into two sub-modules:

- SLA specification management. A SLA specification is a detailed description of the SLA related to a particular data offering available in the marketplace.
- License specification management. A license specification is a detailed description of the terms and conditions by which the related data offering is made available through the marketplace.

Revenue management APIs are logically grouped into six sub-modules:

- Usage specification management. Usage specifications are a detailed description of a usage event which can then be used in a usage pricing model. Usage specifications define all the attributes known for a particular type of usage.

- Usage management. Usage documents contain the actual usage made of a purchased data offering, including the information defined in its usage specification.
- Revenue sharing model management. A revenue sharing model specifies how the revenues must be distributed between the involved stakeholders. This API allows to retrieve, create, update, and delete revenue sharing models.
- Transaction management. API for the management of Charging Data Record (CDR) documents describing transactions. This API allows to register transactions. Additionally, it allows to launch the settlement process that aggregates the transactions and calculates the distribution of revenues.
- Billing charges management. A billing charge includes the information of a payment made by a data consumer for a specific data offering purchased in the marketplace.
- Billing account management. A billing account is a description of a customer bill structure.

Customer management APIs are logically grouped into three sub-modules:

- Customer management API. It is used for saving customer private information that cannot be included within the party resources.
- Customer account API. Customer accounts are used as the link the billing account included in the orders to the customer objects that contain the customer contacts.
- Party management API. It allows to create, retrieve and update the parties.

Feedback and reputation APIs provide the following main functionalities:

- User feedback: allows users to provide feedbacks on data offerings they have purchased. Feedbacks will be based on the quality and reliability of data sources as well as on their compliance to the related SLAs.
- Data offerings rating: is in charge of building and maintaining a ranking of data offerings with respect to feedbacks received by the data consumers.
- Data provider reputation: is in charge of building overall reputations of data providers according to the rating scores of their data offerings.

Transparency and accountability service provides tool for auditing orders (including pricing model, license terms, SLAs) and tracking the parameters defined by SLAs.

2.3.5 Security Interface

SynchroniCity provides a set of APIs aimed to fulfil the security and privacy requirements of each RZ (Figure 11). The SynchroniCity security architecture is built upon OASIS security standard (e.g., XACML [27]), oneM2M, GSMA [28] and ENISA [29] IoT security guidelines following a modular design by splitting it into several components and sub-components to be consistent to SynchroniCity overall architectural and security requirements. It provides a unified approach to the management of security policies as a viable and scalable means to define 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 system (ABAC). Access rights are granted to users through policies combining attributes such as users, resources, actions, objects, etc. However, different access control model can also be supported (e.g., Role-Based Access Control).

With respect to the standard identity management, authentication, authorization and accounting components which reflect the security standard and guidelines mentioned above, 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, which will respond with a session token upon authorization thus concluding the login phase. The Authorization PEP Proxy sub-component is a consumer of session tokens, therefore it validates tokens while getting authorization info from the token, such as the user role; it is also a client of the Authorization PDP since it intercepts resource access requests which are then forwarded to the Authorization PDP to get an authorization decision (Permit or Deny). The PDP evaluates 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 sub-component. The Data protection and Privacy module enforce the security measures such as encryption and anonymization, defined by the Policy Management module, directly on the resource/asset (e.g., data, service or application).

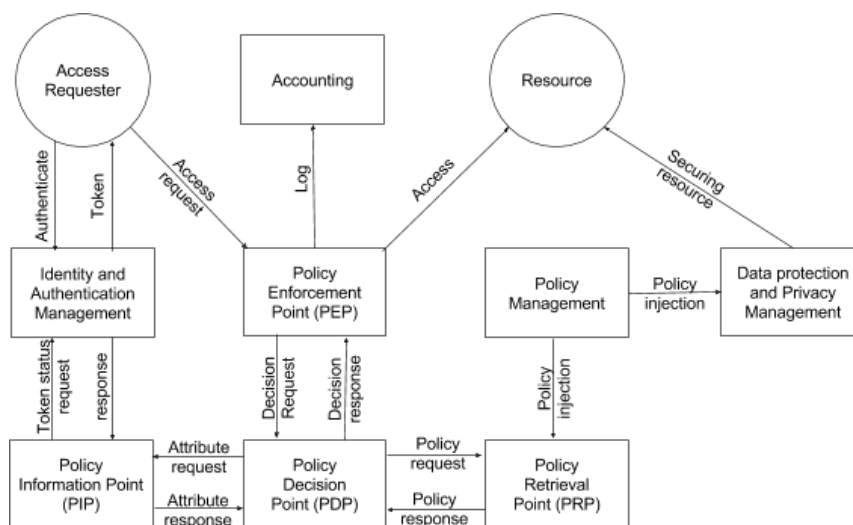


Figure 11: Security Components Diagram

Identity and Authentication Management API exposes interfaces to create, import, retrieve, update, delete users and roles and to perform authentication. More specifically, the API allows to create a user by taking as input username, optional information about the user (e.g., contact info, description, purpose, etc.), password and role, returning as output the user identifier upon successful creation. Similarly, user import can parse user information from an external source. Retrieve, update and delete require to pass the user identifier 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 allows to perform 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 request. It exposes interfaces to the authentication component and the policy management component. More specifically, it receives access request 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 identifier, while when updated or deleted it returns the outcome of the operation (e.g., success or fail). This API allows to retrieve policies by passing the policy identifier or by passing the subject. In the former case, the API will return the policy associated to the policy identifier passed while in the latter case it will return a list of policies matching with 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.

2.3.6 SynchroniCity compliance levels

The previous sections underlined the main components and interfaces that can be considered “interoperability points”, the ones that should be implemented by RZs or external systems and applications to be part of the SynchroniCity framework. An important aspect that should be stressed is that every RZ, based on its level of technical maturity, can join the SynchroniCity ecosystem with a specific level of compliance. The technical activities to reach the framework compliance can be various in terms of both complexity and duration but it is possible to define some milestones: the Figure 12 describes this approach presenting four progressive steps with different levels of compliance with SynchroniCity that can be reached by RZs indicating also different degrees of interoperability with the framework:



Figure 12: SynchroniCity compliance levels

In the following section are described in details the different phases and potential activities to perform in order to reach the specific compliance level.

Level 0 - Initial situation: proprietary/specific systems/API
The Reference Zone is out of SynchroniCity. The IoT infrastructure and software platforms are managed/owned by the municipality or by external providers. The IoT devices communicate with the infrastructure using specific/proprietary protocols. The data are accessible through specific/proprietary API. The data models are specific, not aligned with open standards. All the systems and applications are strictly tailored by for the specific RZ.
Level 1 - Entering SynchroniCity: asset identification
The Reference Zone wants to be part of SynchroniCity and defined a technical interoperable architecture in order to make the existent IoT infrastructure compliant with the SynchroniCity specifications (interoperability points). RZ identified which datasets and services can be brought into SynchroniCity framework.
Activities to be performed to achieve the level
The technical activities in this stage should be focused on two different tracks: the identification of the possible interactions between the RZ IoT infrastructure and the SynchroniCity framework, estimating the minimum technical requirements needed to implement the “interoperability points”. In this phase, it should be identified which components (hardware and software), of the existing city infrastructure, have to be adapted using specific connectors or data injectors to convert native interfaces (e.g. REST API) to the SynchroniCity ones. In parallel it is necessary to investigate and identify which datasets should be exposed following the SynchroniCity specifications: these datasets can include, for instance, real-time data coming from IoT devices, historical and open data, micro services data.
Mandatory requirements
Knowledge of the SynchroniCity framework architecture and interoperability points specifications
Optional requirements
N/A

Level 2 – Implementation of SynchroniCity open API
The RZ implemented the SynchroniCity open API: city datasets are accessible through open and standard interfaces that represents the main interoperability points with the SynchroniCity framework.
Activities to be performed to achieve the level
The activities inside the RZ are devoted to the provisioning of the data using open and standard API: the RZ should work together with the technical partners to adapt the existing interfaces to the SynchroniCity API in particular to the context management one. RZ can use existent software components provided by SynchroniCity or can develop specific software adapter to inject data into the Context Broker. This can be considered the first step to share the city data in SynchroniCity and be part of the ecosystem.
Mandatory requirements
• Implementation Context Management API for real time data provisioning (Interoperability point) • Implementation of Security API to be compliant with SynchroniCity authentication and authorisation (Interoperability point)
Optional requirements
• Implementation of Data Storage API to give access to RZ historical data (Interoperability point) and Open Data repositories
Level 3 – Adoption of SynchroniCity Data models
The RZ adopts SynchroniCity common data models, based on OASC principles, to represent the city data. The common data models are domain specific and their usage enable the replicability of the applications and services, based on them, among different RZs.
Activities to be performed to achieve the level
The RZ has to map its own legacy data models with the SynchroniCity data models in order to select the most suitable ones for the specific smart city domains. The SynchroniCity data model can be extended or new ones can be created to represent new entities or attributes not covered by the current ones. Once identified the correct data model and mapping, specific adapters should be implemented to convert the original data in the new format. SynchroniCity project provides tools and guidelines to speed up the mapping and conversion of the RZ data.
Mandatory requirements
• Adoption of SynchroniCity Data models or compliance with SynchroniCity guidelines for the definition of new common data models (Interoperability point)
Optional requirements
N/A

Level 4 – Marketplace participation
The RZ is fully part of the SynchroniCity ecosystem actively participating in the marketplace. RZ data is included in SynchroniCity marketplace catalogue and can be accessed by third party applications and external stakeholders. The marketplace supports also monetisation mechanisms allowing, for instance, to sell the access to real time or specific historical datasets. The active participation to the marketplace enables the concrete realization of a digital single market.
Activities to be performed to achieve the level
From technical point of view the RZ, being compliant with the previous levels, is ready to be part of the marketplace. RZ can also decide to access to a central SynchroniCity marketplace or to install a specific version customised for the RZ local ecosystem. If the RZ already has a marketplace it could be federated by the SynchroniCity Marketplace: in this case, limited technical activities will be necessary to implement the federation. In addition to that, RZ will be involved in other business and strategical decisions to define the terms and approaches to be part of the SynchroniCity marketplace (e.g. proper handling of ownership, service access terms, data licenses etc.). It will be possible to have different levels of integration starting from the simple participation to the marketplace catalogue, to promote specific RZ assets, up to supporting all the other functionalities, such as revenue sharing, feedback collection, SLA management etc.
Mandatory requirements
• Compliance with Marketplace API (Interoperability point)
Optional requirements
• Deployment of a local Marketplace instance

3 SynchroniCity framework reference implementation

One of the key objectives of the WP2 was to build an overall logical architecture to define the needed functionalities of the SynchroniCity IoT Enabled Framework for Smart Cities: furthermore, SynchroniCity proposes a reference implementation of the logical architecture in order to provide, to RZs open-source, ready to use components:

- that can speed-up the development of SynchroniCity API (see section 4)
- directly support the implementation new functionalities not already present in the existent RZ technical platform
- simplify the integration between the current RZ IoT infrastructure and the SynchroniCity technical framework.

However, it is important to underline that the proposed technical components, described in the following section, are not mandatory, both in their adoption or usage, but should be considered only one of the possible implementations of the SynchroniCity architecture. The RZs can choose to adopt or integrate single components, among the proposed ones in the reference implementation, with their existent infrastructure or use their functionalities in “as a service approach” deploying them in specific cloud instances. The flexibility of this technical solution allows Reference Zones to choose the best approach related to their technical and business requirements.

This section describes the technological solution proposed for the implementation of the SynchroniCity reference architecture introducing the functionalities and roles of the single components. Further information of these software components, that are independent from the SynchroniCity, can be found in the official documentation referenced in the specific sections.

The SynchroniCity reference implementation components are accessible in the SynchroniCity project GitLab repository [30]

In order to make the deployment in the RZs as easy and quick as possible, a container deployment architecture for SynchroniCity has been defined using Docker compose [31]. Each component will be deployed as a Docker container and linked each other. The containers will be able to both communicate internally and expose specific interfaces (port) to the outside. The approach with Docker compose will ease the setup and deployment of the whole architecture with a single command. The Figure 13 depicts the components and containers used to make the working prototype of the architecture.

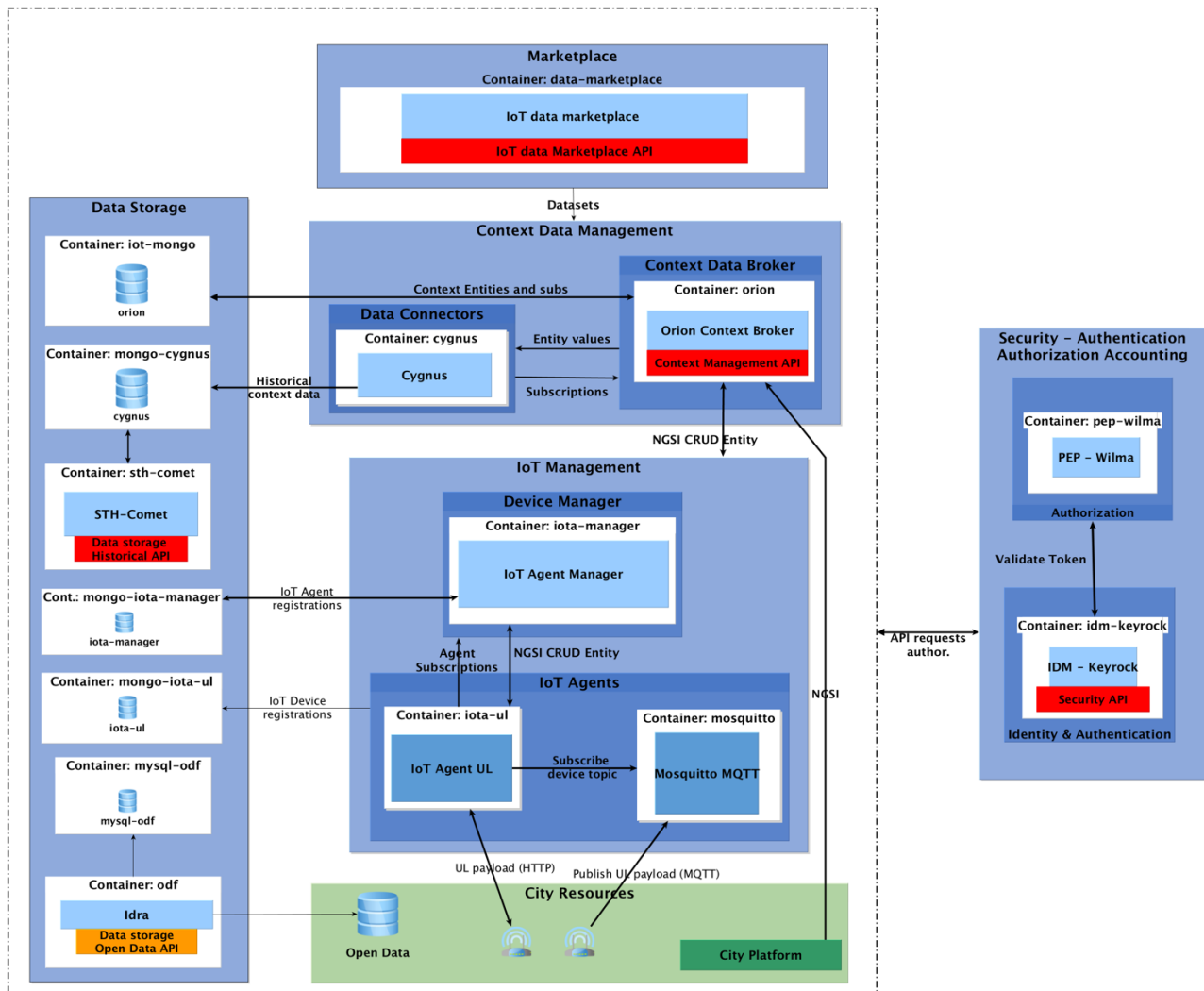


Figure 13: Reference implementation architecture overview

3.1 Context Data Management

The following sections describe the concrete implementation of the modules that are part of the Context Data Management component, described in Section 2.2.1

3.1.1 Context Data Broker

The implementation of the Context Data Broker module consists in the Orion Context Broker [32]. It represents the FIWARE reference implementation of the Publish/Subscribe Context Broker GE specification. Orion is a C++ implementation of the NGSIv2 REST API binding developed as a part of the FIWARE platform.

The Context Broker is in charge of managing the lifecycle of the Context information, including creation, updates, queries and subscriptions. The Context information consists of structured Context Entities, containing attributes and related metadata. Stored entities will represent the mapped southbound devices and their attributes (active, lazy, commands).

At the same time, the Broker provides capabilities to gather context data from external data

providers. Therefore, it can forward an incoming query to a registered provider, in order to search the requested entities on it. In a similar way, it can also update attributes or send commands, either to local entities or to external ones.

The module provides a subscription mechanism, which allows to subscribe to one or more entities, in order to get notified about the updates of related attributes, when published data meets the subscription conditions (e.g. a change in the value of temperature attribute). It is important to point out that the Orion Context Broker is not able to store and provide a historical view of entity/attributes values, this is performed by the Data Connector module.

Finally, it exposes its functionalities, such as discovering, gathering and publishing of context entities, through the NGSI APIs [33].

3.1.2 Data Connector

The data connector module is realized by using the Cygnus component, acting as a connector between the Context Broker and several third-party storage systems [34]. It is in charge of persisting the NGSI-like context data managed by Orion, in different third-party storages, creating a historical view of such data. In particular, it enables to create and store an historical view of the entities and related attributes issued by the Context Broker.

It supports several «sinks», which are plugins writing in different persistence systems:

- MongoDB
- MySQL
- CKAN
- PostgreSQL
- STH (Short Term History)

In order to retrieve the entities, Cygnus uses the subscription/notification feature of the Context Broker. A subscription is made in Orion on behalf of Cygnus, specifying which entities should be notified to Cygnus when an update occurs on their attributes and under certain conditions.

In particular, in the reference implementation are used the following sinks:

- **MongoDB**: The values of the entities notified by the Context Broker are persisted as historical raw context information in a MongoDB, the NOSQL document-oriented database.
- **STH** (Short Term History): The consecutive values of the context entities are persisted as historical aggregated time series information, in a MongoDB instance. This data will be accessed by the STH Comet component.

Both sinks store information in the MongoDB container (**mongo-cygnus**).

3.2 Data storage

The following sections describe the concrete implementation of the modules that are part of the Data Storage Management component, specifically for historical data retrieval and Open Data described in Section 0

3.2.1 STH-Comet

The Short Time Historic (STH, aka. Comet) [12] is the component in charge of managing (storing and retrieving) historical raw data and aggregated time series information about the context data (i.e., entity attribute values) registered in the Context Broker instance. This is an extended version that implements the SynchroniCity Historical API [13], by adding it to those already provided by the original Fiware STH-Comet component [14].

In this architecture, the **formal** way is used [15]. In this approach, is Cygnus that persists the raw and aggregated time series context information into a mongo container; then, this information is used by the STH component. The subscription of the entities is then still performed between Cygnus and Orion Context Broker.

STH exposes an HTTP REST API to let external clients query the available historical context information:

- **Raw:** the concrete values that an entity attribute took overtime, including the concrete timestamps when the values of the attribute changed.
- **Aggregated time series:** the evolution of the entity attribute values grouped by time, according to a time offset, several aggregation periods and functions, such as mean, min, max, etc.
- **SynchroniCity Historical API:** historical data related to an attribute of an entity, according to the SynchroniCity API definition [14].

3.2.2 Idra - Open Data Federation Platform

“Idra - Open Data Federation platform” is a Java EE Web platform able to federate existing Open Data Management Systems (ODMS), which belongs to the RZs and are based on different technologies. In this way Idra provides a unique access point to retrieve, search and visualize open data sets coming from the different federated ODMS catalogues. These functionalities are exposed by the Platform API, in order to be used by generic external client or to develop third party applications. These APIs represent the implementation of the Open Data API, as described in Section 2.3.2.

The platform is in charge of collecting metadata of Open Data from federated ODMS and then translating them into a uniform and standard representation: DCAT-AP [35]. Idra supports natively ODMS based on CKAN, DKAN and Socrata APIs and provides a set of REST interfaces to federate ODMSs not natively supported. In addition, it is possible to federate a generic Web Portal that does not expose any supported API, either by using the Web Scraping functionality or by uploading a dump of the datasets in the DCAT-AP serialization format. Moreover, Idra provides a SPARQL endpoint in order to perform queries on 5 stars RDF linked open data (LOD) collected from federated ODMSs.

Figure 14 shows a page of Idra portal, the web user interface of the Idra: the user, using this web portal, will be able to search, even in a multilingual way, through the Eurovoc thesaurus [36], among the collected datasets, according to multiple DCAT-AP metadata filters, such as keywords, title, description, dates, publisher, file formats, etc.

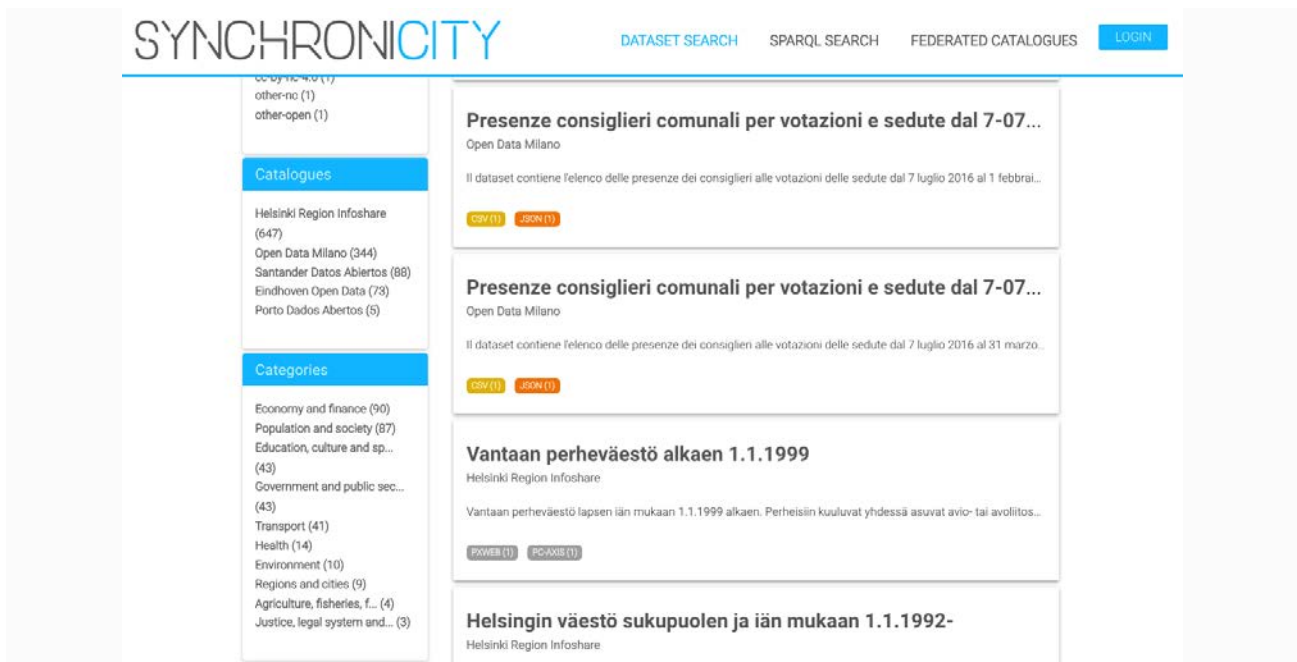


Figure 14: Dataset search results in the Idra portal

3.3 IoT Management

The following sections describe the concrete implementation of the modules that are part of the IoT Management component, described in Section 2.2.2

3.3.1 IoT Agent Manager (IDAS)

The IoT Agent Manager represents the implementation of the centralized administration functionalities provided by the IoT Management component. The module, also known as IDAS, is an implementation of the FIWARE Backend Device Management GE [37]. The module provides a single administration point, by implementing several IoT Agent management operations, such as registration, configuration and monitoring of agents with different protocols.

In addition, the module performs provisioning tasks, and for the scenarios where multiple IoT Agents offer different southbound protocols, it will work as a proxy, by redirecting NGSI requests to the appropriate IoTAgent, based on the declared protocol.

Each registered Agent is uniquely identified by the following two parameters:

- **Protocol:** Name of the protocol served by the IoTAgent.
- **Resource-APIKey:** Unique pair of strings used to identify different IoT Agents for the same protocol.

Finally, the module exposes the Subscription and Provision APIs [38], in order to manage agent provisioning and the services to be redirected to the appropriate Agent.

By using IoT Agents, the devices will be represented in the SynchroniCity platform as NGSI entities in the Context Data Broker. IoT Agents translate IoT-specific protocols (e.g. LWM2M, UL, JSON etc.) into the NGSI context information protocol.

3.3.2 IoT Agents

An IoT Agent is a component that lets a group of devices to send their data to and be managed from an NGSI Context Broker, using their own native protocols, (e.g. UltraLight protocol) for the measure payload. The module manages the device registration, by specifying its attributes and the related mapping to a NGSI context entity. Each Device is mapped in the Context Broker as an Entity associated to a Context Provider; hence, each measure obtained from a device will be mapped to a different entity attribute.

Device measures can have three different behaviours:

- Active – pushed from the device to the IoT agent.
- Lazy – passive device will wait for the IoT Agent to request for data.
- Commands – special attribute if the device can accept commands.

The provision process is meant to provide the IoTAgent with the following information:

- Entity translation information: information about how to convert the data coming from the South Bound device into a NGSI information. E.g. entity name, type and attributes that will be created in the entity.
- Southbound protocol identification: To identify a particular device when a new measure comes to the Southbound (**Device ID** and **API Key**).

In the following section is described an example of IoTAgent (UltraLight protocol) deployed in the SynchroniCity reference architecture. Further agents can be developed during the project.

3.3.2.1 IoT UL Agent

In the SynchroniCity reference implementation, an UltraLight IoT Agent is provided [39]. The agent acts as a bridge between devices using the Ultralight 2.0 protocol and NGSI Context Broker. Ultralight 2.0 is a lightweight text-based protocol aiming to keep the communications, coming from devices with limited bandwidth and memory, as small as possible. The agent will provide different transport protocol bindings for the same protocol: HTTP and MQTT

The UltraLight payload can be carried out through two transport protocols for the same Southbound protocol:

- **HTTP**: The agent itself exposes HTTP southbound APIs, in order to accept values coming from Devices.
- **MQTT**: The agent subscribes to a device-specific topic, by using the **Eclipse Mosquitto** MQTT broker [40]. This open-source message broker implements the MQTT protocol, according to a publish/subscribe pattern. In this case, it publishes messages coming from the devices to specific topics. Then, the IoT Agent will receive the notification messages sent by Mosquitto, containing the measures of a specific device.

For both transport protocols, the IoT Agent container provides a console client, in order to simulate and test the device measurements.

3.4 Security layer

SynchroniCity security mechanisms are based on modular components and standard protocols. The main building blocks provide a specific set of security features such as *identity management*, *access control*, *authentication*, and *authorization*. Backend APIs, exposed by the previously described components, are protected from unauthorized access by the OAuth2 [41] standard protocol. Its role is to authenticate and authorize requests, as well as to delegate and generate access token.

The *identity and authentication management* component aim at providing identity as a service (IDaaS), by means of standard-compliant protocols and APIs. It embeds *authentication* capability to provide secure and private authentication from users to IoT data sources, applications and services, user profile management, Single Sign-On (SSO) to service domains, and identity federation towards applications. Moreover, it provides an OAuth2 server implementation, supporting both the Authorization Code and Implicit Grants flows to issue bearer tokens to authenticated and users. Different *roles* and *permissions* can be issued to users/applications, thus, access tokens can be used to grant or deny access to the APIs exposed by the platform components based on provided roles and permissions.

To improve communication overhead due to the exchange of requests and responses among the different security sub-components as described in section 0 of this document, we decided to group together some sub-components (PEP, PDP, PIP, PRP and accounting) as internal modules of a single authorization component hereafter referred as Wilma PEP Proxy. This implementation choice improves communication efficiency while keeping the same benefit of the modular approach described in section 0. This component embeds an attribute based access control mechanism in which each role is defined as a set of attributes. A *Policy Enforcement Point (PEP) proxy* protects resources against *unauthorized* access. The *PEP proxy* intercepts each access request to a particular resource, it relies on the *identity management* to authenticate the request. It also provides a Policy Decision Point (PDP) which evaluates authorization decisions based on roles and/or attributes related to a given access request (e.g., requester identity, requested resource, requested action). Furthermore, it provides accounting functionalities by logging requests and responses from users and protected resources.

Figure 15 depicts a possible implementation of the Security Layer, as composition of two modules, (Docker containers) that interact with all the other modules in the platform and a reverse proxy.

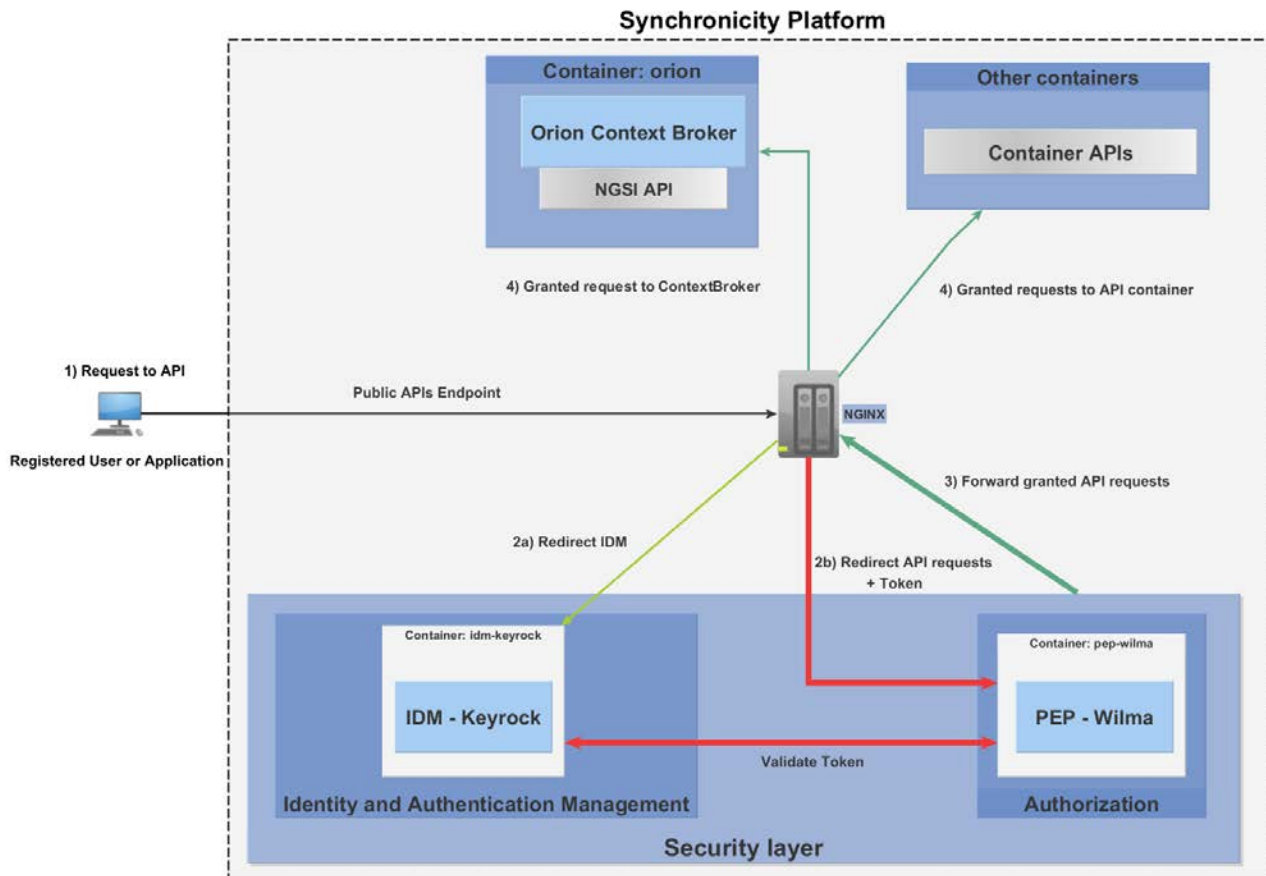


Figure 15: Security Layer overview

The modules are summarized below:

- **Identity and Authentication Management: - Keyrock** is the module responsible of the user management and authentication functionalities.
- **Authorization Management – Wilma PEP Proxy:** is the module responsible of intercepting all the requests directed to the platform components, enforcing the access policies.
- **NGINX:** is the reverse proxy server in charge of exposing publicly the platform endpoints and redirecting all the requests to the appropriate components.

As previously described, the PEP Proxy will intercept all the requests coming to the entry point of the architecture, represented by the Nginx reverse proxy, which, without any security components, would redirect them to the internal endpoints, corresponding to the different components. In this case, except for direct requests to IdM, all other requests will pass through the PEP Proxy.

3.4.1 Identity and Authentication Manager - Keyrock

The implementation of the Identity and Authentication Management module consists of the Identity Manager Keyrock [42]. It represents the reference implementation of the FIWARE Identity Management GE specification. Keyrock offers tools for handling user life-cycle functions, by realizing typical operations involved during account creation and management, as it supports the enforcement of policies and procedures for user registration, secure and private authentication and user profile management. In addition, it allows to link an application with the user account, in order to enable that application to authenticate on users' behalf, by interacting with its OAuth2 APIs [43].

The web portal, representing the frontend of the IdM, is depicted in Figure 16. The back-end is a REST service component that exposes both its own Identity Provider APIs and Oauth2 APIs.

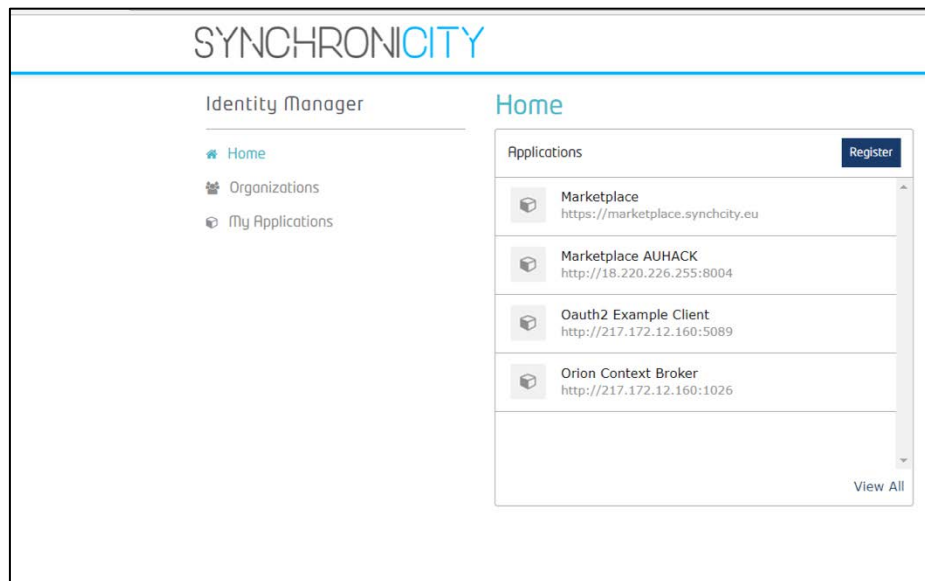


Figure 16: Home page of the IdM Web portal

3.4.2 Pep Proxy - Wilma

The implementation of the Authorization Management module consists of an enhanced version of Wilma PEP Proxy [44], representing the reference implementation of the FIWARE PEP Proxy Generic Enabler.

It is a backend component, without frontend interface. It is in charge of intercepting all the requests coming from users/applications to protected SynchroniCity APIs. Every request has to embed a valid OAuth2 token which is validated by the PEP against the Identity Management (IdM). Token validation requires PEP to interact with the IdM to retrieve requester information such as username and roles. If the token is valid, PEP checks that the requester owns the necessary roles required to access the specific requested resource. This task is performed by a Role-Based Access Control engine, which parses the url of the requested resource to retrieve the required roles and finally find a match with the requester roles. If the token validation succeeds, the proxy forwards the request to the requested resource.

In order to obtain an access token, the user/application must have an account on the IdM and one or more roles associated to the protected resource (e.g. the Orion Context Broker). When purchasing/acquiring access to data sources through the marketplace, the access token can be retrieved directly from the marketplace. However, users/applications can request an access token directly to the IdM by providing their credentials via one of the available Oauth2 flows.

Figure 5 depicts an interaction of a “basic use case” of Wilma Pep-Proxy. A web application, through an Oauth library, performs an Oauth2 Flow, in order to get an access token, which will be used in the header of an access request to a back-end app (e.g., Orion Context Broker), protected by the PEP proxy.

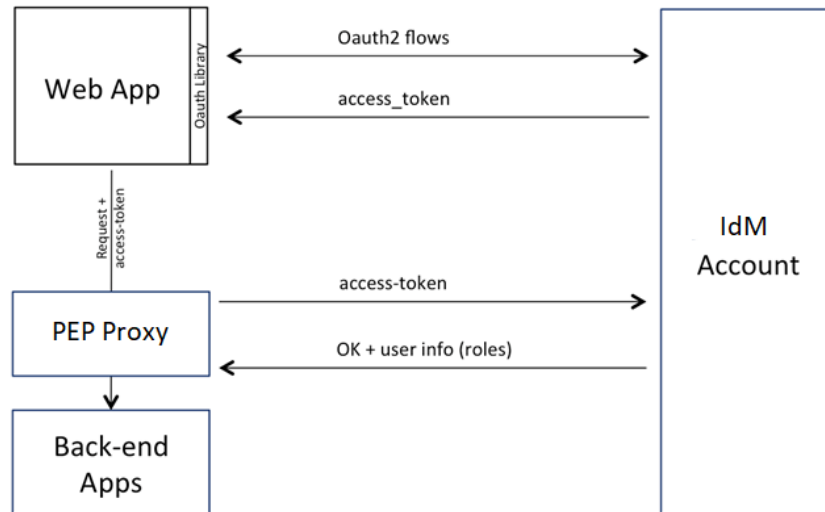


Figure 17: API access authorization flow

3.5 IoT Data Marketplace

The high-level view of the marketplace platform is shown in Figure 6 and consists of the following components:

Marketplace API is the core element of the platform and includes several sub-components that expose a set of functions to interact with that. Specifically, the Marketplace API allows data providers to register or import data sources into the platform, and publish offerings containing its description (e.g., version, data model, endpoint URL), and allows data consumers (e.g., service developers) to discover and purchase offerings.

Marketplace portal is an optional component, shown in dashed line in the figure, providing a user interface through which data providers and data consumers can interact with the platform, use the Marketplace API, and manage their accounts and information.

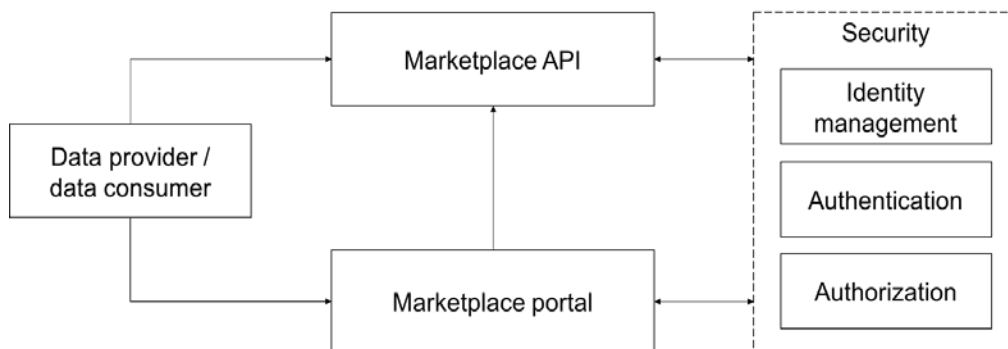


Figure 6: High-level view of the marketplace platform.

The *marketplace portal* acts as the endpoint for accessing the *marketplace APIs* and orchestrates them validating user requests, including authentication, authorization, and the content of the request from a business logic point of view. The back end is based on the Fiware Business Ecosystem Logic Proxy [45], which has been adapted to reflect the different and additional features provided by a specific set of APIs (see section 4.3).

The application logic is implemented by means of controllers driving the APIs provided by the different components. Its core is running on a Node.js instance, suitable for the event-driven nature of the SynchroniCity IoT data marketplace. The non-blocking I/O model of Node.js results in a very lightweight and efficient solution.

As for the front end, we provide a web portal that can be used to interact with the system. It provides functionalities and a graphical user interface to manage all the elements instantiated in the system.

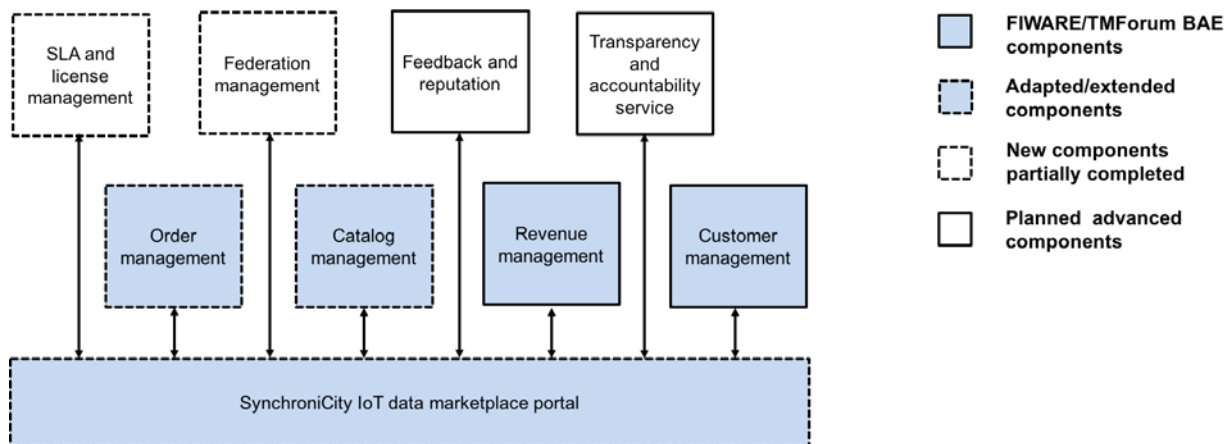


Figure 7: Current implementation status of the marketplace APIs and marketplace portal.

Figure 7 shows the current implementation status of the *marketplace APIs* and *marketplace portal*. The basic functionalities are provided by adapting/extending the FIWARE/TMForum BAE components as well as by developing new components. The *revenue management* and *customer management* components are left unchanged from the BAE specification/implementation whereas the *catalog management* and *order management* have been adapted and extended to better fit the SynchroniCity IoT data marketplace requirements. New components such as the *SLA and license management*, and the *federation management* have been partially developed. The set of features provided by the current implementation defines the basic enablers for the data marketplace and their behaviour is specified in the deliverable D2.4 (Basic data market place enablers). The advanced enablers, including the *feedback and reputation*, the *transparency and accountability service*, as well as the full implementation of the *SLA and license management* and the *federation management* will be discussed in D2.5 [8]

A detailed documentation as well as admin and user guides can be retrieved from [46]

4 SynchroniCity API

SynchroniCity defined a concrete implementation of the logical interfaces (interoperability points) introduced in section 2.3: the SynchroniCity API based on HTTP RESTful approach and widely used standards. The implementation of the SynchroniCity API is a basic requirement for the Reference Zones that want to be compliant with SynchroniCity. The following section presented an overview of the API with the main functionalities. The complete documentation can be found in [47].

SynchroniCity API are based on RESTful approach and widely used standards. There are four sets of API:

- **Context Management API** the way to communicate with the Context Management module in order to manage the context entities. The API are based on NGSIv2
- **Data Storage API** provide access to historical data and Open Data. The definition of this API is inspired to the NGSI-LD Temporal Query language
- **IoT data marketplace API** allows the monetization of digital assets during the whole service life cycle. It is an extension of the Business API Ecosystem
- **Security API** provide authorisation functionalities to access to the SynchroniCity services. The API are based on OAUTH2 protocol

4.1 Context Management API

Context Management API represent a standard way to communicate with the Context Management module, the key component of the SynchroniCity architecture.

The concrete implementation of this API is based on NGSIv2 http RESTful interface. This API allows to manage and query context entities, attributes and to create subscription to be notified for entity updates.

API set	Description
API Entry Point	This resource allows you to obtain a list of sub-resources published by the API: • entities to deal with your NGSI entities • types to deal with your entity types • subscriptions to deal with subscriptions to context changes • registrations to deal with context data by registering external providers
Entities	This collection allows to create new Entities and to retrieve a list of entities that match different criteria by id, type, pattern matching (either id or type) and/or those which match a query or geographical query.
Entity by ID	This collection allows to access and manage a specific Entity identified by its ID. The user would be able to: • retrieve the entity • retrieve, update or replace its attributes • delete the entity
Attribute by Entity ID	This collection allows to retrieve a specific entity attribute using both entity ID and the attribute name. The main functionalities

	provided by this collection are CRUD operations to be performed on the specific entity's attribute.
Attribute Value by Entity ID	Whit this collection the user would be able to retrieve or update the attribute data using the entity ID and the attribute name. These operations work on the <i>value</i> property of the specific attribute.
Entity types	This collection provides the list of entity types and their metadata registered in the platform. Results are ordered by entity <i>type</i> in alphabetical order.
Entity Type	This API allows to retrieve information about a specific entity type. Moreover, it provides the set of attribute names along with all the entities of such type and the number of entities belonging to that type.
Subscriptions	This collection allows to create new Subscriptions and to retrieve the list of all the subscriptions present in the system.
Subscription by ID	This collection allows to manage a specific subscription by using its identifier. The provided functionalities help to: • retrieve the subscription information • update the subscription • delete the subscription
Registrations	A Context Registration allows to bind external context information sources so that they can play the role of providers of certain subsets (entities, attributes) of the context information. This collection helps to create new Registrations and to list all the context provider registration present in the system.
Registration by ID	This collection provides functionalities to retrieve, update or delete a specific registration element by its identifier.
Batch Operations	This collection allows to create, update and/or delete several entities in a single batch operation. This operation is split in as many individual operations as entities provided in the request. Moreover, this collection allows to query several entities or entity types in a single batch operation.

4.2 Data Storage API

4.2.1 Historical

The SynchroniCity Historical API allows access to historical data related to an attribute of an entity using the same URL approach defined by the Context Broker NGSIv2 to access an attribute value.

The definition of this API is inspired to the NGSI-LD Temporal Query language included in the ETSI ISG CIM NGSI-LD draft specification [20]

API set	Description
Historical Data Retrieval	This API let the user query on historical data related to an attribute of an entity. The user will be able to specify the entity and the attribute he/she would query and the desired time interval. Moreover, pagination parameters can be specified in order to limit the results.

4.2.2 Open Data

Open Data APIs are a subset of SynchroniCity Data Storage API and allows to access to the Open Data provided by the different Reference Zones (i.e cities) of SynchroniCity project in a uniform and standard way, using the same metadata format (DCAT-AP). Currently the Open Data APIs are "suggested" and are not defined as an "interoperability point", so they are not considered mandatory to be compliant with SynchroniCity Framework

The Open Data APIs are based on the REST APIs of Idra Open Data Federation Platform developed by Engineering Ingegneria Informatica S.p.A. Idra is a web platform that allows to federate existing Open Data Management Systems (ODMS) based on different technologies.

Idra natively supports ODMS based on CKAN, DKAN and Socrata. In addition, it is possible to federate a generic Web Portal, either by using the Web Scraping functionality or by uploading a dump of the datasets in DCAT-AP format. Moreover, Idra provides a SPARQL endpoint in order to perform queries on 5 stars RDF linked open data collected from federated ODMSs.

API set	Description
Platform Info	This API provides information about the specific instance of the OpenData Federation. It allows the user to check the current version of the APIs and to perform sanity checks in order to verify the proper installation of the tool.
List Catalogues	This API provides the list of the federated Open Data Management Systems (ODMS) referred as Catalogues. The user would be able to investigate on every catalogues' base technology and additional information such as their countries or hosts.
Search Dataset	This API allows to query the federated datasets' metadata. Different types of query can be performed, starting to the full text search to a faceted search over several datasets' metadata.

	Moreover, the user can perform multilanguage searches taking advantage of the EuroVoc Thesaurus.
Search DCAT_AP	This API allows to serialize the results of a search operation in several formats and profiles. For instance, the supported formats are: TURTLE, NTRIPLES, RDFXML, N3, RDFJSON or JSONLD. This API allows to serialize data using the standard DCAT_AP profile and also using the DCAR_AP-IT Italian profile.
Sparql Query	This API allows to perform a Sparql query on the 5 stars RDF linked open data collected from the ODMS. The input Sparql query will be executed over the triple store containing the RDF contexts.

4.3 IoT Data Marketplace API

The SynchroniCity IoT data marketplace API is an extension of the Business API Ecosystem [48] made up of the FIWARE Business Framework and a set of APIs (and its reference implementations) provided by the TMForum [49]. This set of APIs allows the monetization of digital assets during the whole service life cycle, from offering creation to its charging, accounting and revenue settlement and sharing. It exposes functionalities such as catalogue management, ordering management, federation management, revenue management, customer management, SLA and license management, feedback and reputation, and transparency and accountability service.

API set	Description
Data source Specification Management API	This API is a subset of the Catalogue Management API and focuses on the management of Data Sources Specification. A Data Source Specification is a detailed description of a data source made available in the form of a Data Offering to customers playing a data consumer role.
Data offering Management API	API for the management of Data Offerings. Data Offerings represents entities that are orderable from data providers and are published in the catalogue. This resource includes pricing information and links (optional) Service Level Agreements and license terms.
Order Management API	API for the management of Orders. Orders are made by customers as data consumers, and include a set of order items, each specifying a data offering to be acquired. When creating an order, customers can select the value of the different pricing options (if available) to be applied and agree on SLA and license terms.
License Specification Management API	API for the management of License Specifications. A License Specification is a detailed description of the terms and conditions by which the related data is made available through the marketplace.
SLA Specification Management API	API for the management of SLA Specifications. A SLA Specification is a detailed description of the Service Level Agreement related to a particular data offering available in the marketplace.
Federation Management API	API for the management of federation functionalities.

4.4 Security API

SynchroniCity Security API takes advantage of the FIWARE IdM [50]. This section describes the OAuth2 Authentication processes. The FIWARE IdM complies with the OAuth2 standard described in RFC 6749 and supports all four grant types defined there.

API set	Description
Authorization Grant	This API verifies the provided credentials and grants specific authorization to resources. Two usage methods are supported by this API: • Authorization Code Grant: by setting the <code>response_type</code> parameter to <code>code</code> • Implicit Grant: by setting the <code>response_type</code> parameter to <code>token</code>
Authorization Token	This API return a proper authorization token to the user which provides the proper credentials. Several usage methods are supported: • Access Token Request: by setting the <code>grant_type</code> parameter to <code>authorization_code</code> • Resource Owner Password Credentials Grant: by setting the <code>grant_type</code> parameter to <code>password</code> • Client Credentials Grant: by setting the <code>grant_type</code> parameter to <code>client_credentials</code>
User information and roles	This API provides the user information and his/her roles. It accepts the <code>access_token</code> as input and, if valid, returns the requested information.

5 Conformance tests

Each SynchroniCity framework deployment will be verified and validated, according SynchroniCity specifications, to guarantee the proper fulfilment of identified technical and functional requirements, as well as to ensure the interoperability of the different solutions developed on top of it. This exhaustive evaluation process will be addressed within WP4, and the results will be shown in D4.2 [51].

The proper validation of a SynchroniCity framework instance will consist on a set of conformance tests that evaluate each one of the identified interoperability points in two steps:

- Step 1: it will check all the functionalities addressed to the selected component/interface, to guarantee that all functional requirements are properly covered by the instance.
- Step 2: a stress test will be executed against the selected component/interface in order to obtain a set of performance parameters that shows the capabilities and characteristics of the analysed instance.

According to this process, each SynchroniCity framework instance to be validated will be required to provide, at least:

- an NGSI interface endpoint, to support Context Management capabilities (northbound and southbound related)
- a Historical Data Access API endpoint, to provide historical data about context entities
- an OAuth 2.0 compliant identity management access, to provide the requested SynchroniCity authorisation management capabilities
- the list of datasets, with corresponding SynchroniCity data models, to be shared through the SynchroniCity instance
- for those instances that also integrate the IoT Data Marketplace component, the corresponding access to this component

All this information will be used within Task 4.2 to execute the whole defined validation process that covers:

- **Context Management and Context broker** conformance test: using the NGSI endpoint and the corresponding NGSI standard implementation, this test will check
 - NGSI interface availability, according NGSI specifications
 - Discover context entities capabilities
 - Query/Retrieve context information functionality
 - Subscriptions capabilities
 - Performance of NGSI operations
- **Standard IoT Backend API**: based on NGSI specifications, this test will evaluate the standard southbound capabilities
 - NGSI Update execution
 - create new context entities
 - update existing context entities
 - modify existing context entities
 - Southbound API performance
- **Identity Management** conformance test, that will evaluate the OAuth2.0 interface
 - OAuth 2.0 compliant access management functionalities
 - Identity Management performance
- **Historical Data** conformance test, focused on the defined SynchroniCity Historical Data Access API
 - Historical data access features
 - retrieving temporal series
 - Historical data access performance

- **Data sets and Data models** conformance test will validate the number of context entities available on the Context Broker instance according the corresponding SynchroniCity data models
 - Check, by each available entity type, the proper definition of attributes and schemas, according to NGSI specifications and SynchroniCity Data models definitions

The final objective of these conformance tests is to provide an overall picture about what is available on each SynchroniCity framework instance according to the defined SynchroniCity architecture and specification.

Conclusions

This deliverable presented the updated version of the SynchroniCity Reference Architecture for IoT Enabled Smart Cities: starting from the content reported in the deliverable D2.1, the original specifications, requirements and architectural components have been refined and improved taking in consideration the inputs coming from the other WP2 tasks and other work packages (in particular the WP3).

The major advancements included in D2.10 consist in the definition of the SynchroniCity framework reference implementation (Section 0) and the SynchroniCity APIs (Section 4): the reference implementation is a set of open software component that implement the basic SynchroniCity architectural components. These components have been adopted by RZs in order to deploy/integrate, in a simple and fast way, an instance of SynchroniCity framework.

The SynchroniCity APIs are a concrete implementation of the logical interfaces (interoperability points) based on HTTP RESTful approach and widely used standards. Through SynchroniCity APIs data consumer (e.g. third party applications) and data provider (e.g. IoT city platforms) can interact with SynchroniCity framework components in a standard and uniform way.

The deliverable D2.10 can be considered the final version of the overall SynchroniCity technical framework and the main reference for Reference Zones that are interested into SynchroniCity adoption and for external stakeholder (e.g. open call participants) that want to build their services on top of SynchroniCity. Anyway, the specifications described in this document can be subjected to minor changes due, for instance, to new use case requirements or technical issues: therefore, it is suggested to check the last version of the SynchroniCity API and Reference Implementation components in the official API documentation page [47] and software repository [30].

References

- [1] SynchroniCity, "D2.1 - Reference Architecture for IoT Enabled Smart Cities".
- [2] "Open & Agile Smart Cities," [Online]. Available: <http://oascities.org/>.
- [3] SynchroniCity, "D1.3 - Guidelines for SynchroniCity architecture".
- [4] SynchroniCity, "D1.4 - Privacy by design methodology & PIA".
- [5] SynchroniCity, "D2.2 - Guidelines for the definition of OASC Shared Data Models".
- [6] SynchroniCity, "D2.3 - Catalogue of OASC Shared Data Models for Smart City domains".
- [7] SynchroniCity, "D2.4 - Basic data market place enablers".
- [8] SynchroniCity, "D2.5 - Advanced data market place enablers".
- [9] SynchroniCity, "D2.6 - Guidelines for the integration of IoT devices in OASC compliant platforms".
- [10] SynchroniCity, "D2.8 - Report on Basic Reference Zones platform deployment and operational plan".
- [11] "ITU (International Telecommunication Union)," [Online]. Available: <http://www.itu.int/en/Pages/default.aspx>.
- [12] "Recommendation ITU-T Y.2060 Overview of Internet of Things," [Online]. Available: <https://www.itu.int/rec/T-REC-Y.2060-201206-I>.
- [13] ISO/IEC JTC1 Study Group on Smart Cities, "Resolutions 3, ISO/IEC JTC 1 N 11894, 2013-11-12," 2012.
- [14] "AIOTI The Alliance for the Internet of Things Innovation," [Online]. Available: <https://aioti-space.org/>.
- [15] "ESPRESSO Project," [Online]. Available: <http://espresso-project.eu/>.
- [16] "The European Innovation Partnership on Smart Cities and Communities EIP-SCC," [Online]. Available: <https://eu-smartcities.eu/about>.
- [17] "BIG IOT Project," [Online]. Available: <http://big-iot.eu/>.
- [18] "oneM2M," [Online]. Available: <http://www.onem2m.org/>.
- [19] "FIWARE," [Online]. Available: <https://www.fiware.org/>.
- [20] E. G. CIM, "Context Information Management (CIM) - Application Programming Interface (API)," [Online]. Available:

- https://www.etsi.org/deliver/etsi_gs/CIM/001_099/004/01.01.01_60/gs_CIM004v010101p.pdf.
- [21] "FIWARE DATA MODELS," [Online]. Available: <https://www.fiware.org/developers/data-models/>.
- [22] [Online]. Available: <http://ontology.tno.nl/saref/index.html>.
- [23] SynchroniCity, "D3.1 - Documentation of service designs".
- [24] SynchroniCity, "D2.7 - Catalogue of IoT devices ready for Smart City platforms integratio".
- [25] O. M. Alliance, "Next Generation Service Interfaces Architecture," 2012.
- [26] D. Gori, "Progetto e sviluppo di un servizio di lookup basato su hash table distribuite per la piattaforma FIWARE," 2015/2016.
- [27] "OASIS XAMCL version 3.0," [Online]. Available: <http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html>.
- [28] "GSMA IoT Security Guidelines," [Online]. Available: <https://www.gsma.com/iot/future-iot-networks/iot-security-guidelines/>.
- [29] "ENISA (European Union Agency for Network and Information Security) Smart Cities Architecture Model," [Online]. Available: <https://www.enisa.europa.eu/publications/smart-cities-architecture-model>.
- [30] "SynchroniCity GitLab repository," [Online]. Available: <https://gitlab.com/synchronicity-iot/>.
- [31] D. Inc., "Docker Compose," [Online]. Available: <https://docs.docker.com/compose/>.
- [32] "ORION CONTEXT BROKER," [Online]. Available: <https://fiware-orion.readthedocs.io/en/master/>.
- [33] "API Walkthrough," [Online]. Available: <https://github.com/telefonicaid/fiware-orion#api-walkthrough>.
- [34] C. C. Team, "Cygnus," [Online]. Available: <https://github.com/telefonicaid/fiware-cygnus>.
- [35] "DCAT Application profile for data portals in Europe (DCAT-AP)," [Online]. Available: https://joinup.ec.europa.eu/asset/dcat_application_profile/description.
- [36] E. Union, "EuroVoc, il thesaurus multilingue dell'Unione europea," [Online]. Available: <http://eurovoc.europa.eu/drupal/?q=it/node>.
- [37] "IoT Agent Manager," [Online]. Available: <https://github.com/telefonicaid/iotagent-manager>.
- [38] "IoT Agents," [Online]. Available: <https://telefonicaiotagents.docs.apiary.io>.
- [39] "IoT Agent for the Ultralight 2.0 protocol," [Online]. Available: <https://github.com/telefonicaid/iotagent-ul>.

- [40] E. Foundation, "Eclipse Mosquitto™," [Online]. Available: <https://mosquitto.org/>.
- [41] "OAuth 2.0 protocol," [Online].
- [42] FIWARE, "Identity Management - KeyRock," [Online]. Available: <https://catalogue.fiware.org/enablers/identity-management-keyrock>.
- [43] FIWARE, "OAuth2 Authentication," [Online]. Available: <http://fiware-idm.readthedocs.io/en/latest/api/#oauth2-authentication>.
- [44] FIWARE, "PEP Proxy - Wilma," [Online]. Available: <https://catalogue.fiware.org/enablers/pep-proxy-wilma>.
- [45] "FIWARE Business Ecosystem Logic Proxy," [Online]. Available: <https://github.com/FIWARE-TMForum/business-ecosystem-logic-proxy>.
- [46] "SynchroniCity IoT Data Marketplace guide," [Online]. Available: <http://synchronicity-iot-data-marketplace.readthedocs.io/en/latest/index.html>.
- [47] "SynchroniCity APIs," [Online]. Available: <https://synchronicityiot.docs.apiary.io/>.
- [48] "FIWARE Business API Ecosystem," [Online]. Available: <https://catalogue.fiware.org/enablers/business-api-ecosystem-biz-ecosystem-ri>.
- [49] "TM Forum," [Online]. Available: <https://www.tmforum.org/>.
- [50] "FIWARE Keyrock IDM GE," [Online]. Available: <https://catalogue.fiware.org/enablers/identity-management-keyrock>.
- [51] SynchroniCity, "D4.2 - Technical Validation (Phase 1)".