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:	D2.4
Deliverable title:	Basic data market place enablers
Document version:	1.0
WP number:	WP2
Lead beneficiary:	14-DigiCat
Main author(s):	Andrea Gaglione, Angelo Capossele, Daniel Puschmann, Alex Gluhak (DigiCat), Ignacio Elicegui Maestro (UC), Martino Maggio (ENG), Kasper Damgård (AI), Francisco Monsanto (Ubiwhere), Eunah Kim (UDG)
Internal reviewers:	Luis Muñoz (UC), Ömer Özdemir (ATOS), Geoffrey Stevens (FCC), Martin Brynskov (AU)
Type of deliverable:	Other
Dissemination level:	Public
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.

Executive Summary

We are still in the early days of the Internet of Things (IoT). Many of the IoT deployments to date have been focused on satisfying specific use cases, with resultant IoT data streams locked down to a single IoT platform and/or application. This inability to exploit existing IoT data streams for other services and/or applications has led to a sector-wide suppression in RoI due to the inefficient use of deployed infrastructure and unnecessarily reduplication of work. It has also stifled opportunities for innovation that could arise from combining data streams across applications and verticals in new and interesting ways.

IoT data marketplaces provide a means of unlocking existing IoT data silos and offer opportunities for generating new revenue streams and new value for all parties. The SynchroniCity vision is to scale these opportunities by envisioning a vibrant IoT data marketplace as a key enabler for the creation of a global market for IoT-enabled urban services with standardized interfaces and common information models. This vision is also shared with the Open & Agile Smart Cities (OASC) community, which already now encompasses more than 100 cities across the world.

This deliverable presents the basic marketplace enablers that are necessary to make this vision a reality. After analyzing the current approaches and key challenges for realizing data marketplace platforms, we extract the requirements and the value proposition of the SynchroniCity IoT data marketplace by conducting a joint analysis with the cities and some SMEs partners of the consortium. The analysis followed the Value Proposition Design (VPD) methodology and underpinned the design of the marketplace and its API specification. Eventually, an initial Proof-of Concept (PoC) prototype of the marketplace is presented. We integrate the prototype with the IoT infrastructure and available data deployed in Santander, one of the SynchroniCity RZs, define a typical use case scenario, and demonstrate the value that the marketplace can bring towards the various stakeholders.

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

Explored current landscape of IoT data marketplaces
Our initial analysis in Section 2 reveals that IoT data marketplaces are unlike traditional forms of e-Commerce, as the data is not so much 'sold' as 'leased', and data owners wish to retain a degree of control over the rights and restrictions regarding access and usage. On the purchase side of the transaction, data consumers need some guarantee of the trustworthiness of the data and its providers through independent verification mechanisms. Different approaches for creating IoT data marketplaces can be characterized by:

- *Architecture: centralized vs decentralized;*
- *Data dynamicity: streams or static datasets;*
- *Data access: external or internal API, download, blockchain based;*
- *Trustworthiness: user feedback vs data completeness vs data provenance;*
- *Openness: open vs closed.*

Desired properties and building blocks of the SynchroniCity IoT data marketplace
Our analysis in Section 3 maps barriers and opportunities, and extract desired properties for the

marketplace by presenting a joint analysis with several project partners, including Antwerp, Carouge, Eindhoven, Helsinki, Manchester, Porto, Santander, TST Sistemas, Ubiwhere, and BronzeLabs. Our approach was to look initially at the main customer segments for the marketplace:

- *Data providers;*
- *Data consumers;*
- *Marketplace providers;*
- *City councils as marketplace providers.*

We follow the VPD methodology to profile our customer segments by describing them in a more structured and detailed way in terms of jobs, pains, and gains. The value (proposition) map describes how we create value for a specific customer segment, by breaking the value proposition down into products and services, pain relievers, and gain creators. We then merge the different value propositions to derive the main characteristics of the SynchroniCity IoT data marketplace. Our analysis revealed that while there are a variety of factors that contribute to the successful adoption of an IoT data marketplace by the broader ecosystem, however there is evidence to suggest that the following characteristics are essentials:

- *Data catalog with lookup functions;*
- *Access control;*
- *Tools to create suitable agreements;*
- *Seamless monetization for transactions;*
- *SLA adherence monitoring and dispute resolution;*
- *Tools to assess the quality and trustworthiness of IoT data and its providers.*

System Architecture

By considering the outcome from the current landscape and the VPD, in Section 4 we present the design of the system architecture of the marketplace, specifically geared towards the needs and requirements of sharing and monetizing data within the Smart City IoT domain.
We define the basic marketplace enablers as a common set of APIs and provide a more extensive and detailed API specification in the Appendix of this document. For the specification of the basic marketplace enablers we have built on top of FIWARE/TMForum Business API Ecosystem and expanded the components of the reference implementation to fit the purpose and needs of the SynchroniCity project. We define and describe the core abstractions of the components and model the interactions among them. We also describe how security aspects such as the identity management, authentication, and authorization are integrated into the marketplace.

PoC

We have developed a PoC that includes the basic marketplace enablers described in this document. In Section 5 we describe a typical smart city use case involving environmental sensor systems deployed in Santander, and a weather forecast service. We show how the marketplace can be exploited in this use case and highlight the benefits it can bring to the involved stakeholders (in this case, the City of Santander, the service provider, the customers of the service and third party service providers).

We expect this initial effort in defining the marketplace architecture, providing the basic enablers, and analyzing the security aspects will support the integration activities of the SynchroniCity framework in WP2, as well as the development of IoT-based services and applications in WP3. Eventually, by going into finer details, some concepts outlined here are going to be reshaped in the next deliverable D2.5 (Advanced market place enablers).

Abbreviations

AE	Application Entity
AIOTI	Alliance for the Internet of Things Innovation
API	Application Programming Interface
AWS	Amazon Web Services
BSI	British Standards Institution
CB	Context Broker
CDB	Context Data Broker
CDMI	Cloud Data Management Interface
CEP	Complex Event Processing
CKAN	Comprehensive Knowledge Archive Network
CSE	Common Service Entity
D	Deliverable
DCAT	Data Catalogue Vocabulary
DMG	Device Management
DIS	Discovery
DMR	Data Management and Repository
EC	European Commission
FG-SSC	Focus Group on Smart Sustainable Cities
GA	Grant Agreement
GE	Generic Enabler
GPS	Global Positioning System
GPRS	General Packet Radio Service
GMG	Group Management
HART	Highway Addressable Remote Transducer
HDSF	Hadoop Distributed File System
HTTP	Hypertext Transfer Protocol

IA	Innovation Action
ICT	Information and communication technology
IDE	Integrated Development Environment
IOT	Internet of things
ISO	International Standard Organization
ITU	International Telecommunication Union
ITU-T	ITU Telecommunication Standardization Sector
JSON	JavaScript Object Notation
JWT	JSON Web Token
LTE	Long Term Evolution
Mca	a reference point for M2M communication between CSE and AE
Mcc	a reference point for M2M communication between CSE and CSE
Mcn	a reference point for M2M communication between CSE and NSE
MQTT	Message Queue Telemetry Transport
NGSI	Next Generation Service Interfaces
NSE	Network Service Entity
OASC	Open & Agile Smart Cities
ODF	Open Data Federation
ODMS	Open Data Management System
PPM	Privacy Policy Manager
PEP	Policy Enforcement Point
PDP	Policy Decision Point
RDF	Resource Description Framework
REST	Representational state transfer
RFID	Radio Frequency Identification
RPS	Receive Packet Steering
RZ	Reference Zone

SAML	Security Assertion Markup Language
SCADA	Supervisory Control And Data Acquisition
SDO	Standards Developing Organizations
SLA	Service Level Agreement
SNS	Simple Notification Service
SOAP	Simple Object Access Protocol
SQS	Simple Queue Service
SR	System Requirement
SSL	Secure Sockets Layer
SSO	Single Sign-On
TLS	Transport Layer Security
UC	Use Case
W3C	World Wide Web Consortium
WP	Work Package
WT	Work Task
XMPP	Extensible Messaging and Presence Protocol
XML	eXtensible Markup Language
XACML	eXtensible Access Control Markup Language

Table of Content

List of Figures

1 Introduction

SynchroniCity aims to foster the development of a digital single market for IoT-enabled smart cities. This will provide an environment for service providers, device manufacturers and solution/infrastructure providers to collaborate and compete towards smart city innovations. One of the main goals of the SynchroniCity project is to provide a set of marketplace enablers that help bring this vision to life. This deliverable is describing the first set of marketplace enablers (basic marketplace enablers). We derive the requirements based on the needs of actors on the marketplace to design the basic functionalities and architecture of the data marketplace.

Cities have the opportunity to become catalyzers for a multitude of data originating from smart devices connected to a gathering infrastructure, as envisioned by the SynchroniCity project. Relevant data about a city may lie in many places and could deliver value to different stakeholders. Data owners range from businesses that have information that is valuable to local authorities, to citizens or volunteer groups that have information that is valuable to businesses. Recently, many cities have been embracing the open data initiative, starting to open up public data in order to increase the transparency and provide better services to citizens. However, initial efforts of opening up data (i.e., first generation open data portals) fell short in delivering the full potential of providing cost savings to cities and better service to citizens. Such initiatives failed to produce the expected results, although they helped to identify the most challenging barriers to be overcome, such as the lack of standardization of data formats and APIs, the lack of trust in the platform/portal itself, and the limited overall data offering. Data strategies in the public sector should continue to evolve, as cities such as Copenhagen and London are proposing. For instance, Copenhagen worked along with Hitachi Consulting to deliver a data marketplace offering multiple sources of information to meet the challenges of sustainability and quality of life. The goal of the initiative is to collect data from public and private data owners and make it readily available for consumption by public and private entities that are working on solutions to improve their city.

A similar vision is shared by London in its data strategy, which aims at establishing the conditions under which both public and proprietary data can be published, shared, managed and disseminated. To set out the potential for data sharing and establish activities to encourage its exploitation, the Greater London Authority plans to leverage both technical and non-technological elements to promote the exploitation of city data across the whole range of city policy settings and activities. More specifically, the plan envisions the development of a coordinated data infrastructure, a data marketplace, data toolset, alongside with business and financial models, feedback loops, and clear licensing agreements to facilitate deep knowledge discovery, as well as the creation of new valuable integrated services. Practical features such as payments and advanced search across different platforms will enable data subscription models and financial transactions between providers and consumers. These enhancements will enable a secure sharing of new forms of city data, generated by the Internet of Things.

To better understand how to successfully approach opportunities of data monetization, we first analyze the lessons learnt from the private sector that have explored similar opportunities. Data marketplaces have already emerged since 2008 with the goal of making high-quality data accessible to consumers.

Data demand and interest is not always clear unless data is able to capture real value by focusing on specific sector or verticals and is aggregated from multiple sources. Many acquisitions showed the importance of exploiting huge data collections by selling services over the top. Moreover, data marketplaces boost data value, which is often underestimated due to the lack of knowledge of

individual data providers and data owners about how their data can be exploited.
Data monetization might come in different shapes and forms, and put in place in several ways:

- Through filtered or unfiltered access to raw data;
- Through processing and analyzing raw data;
- Deriving business insights from data mining and predictive modeling;
- Presenting insights in an actionable context for building new businesses.

By analyzing also instances of very successful data marketplaces such as Bloomberg, Reuters for primarily investment banking, Equifax, Experian, Call Credit for retail banking, loans and insurance, we can observe that key to their success is to rely on the combination of the following factors:

- Data fusion. Combining data from multiple sources increases its ability to produce valuable insights, increasing its value;
- Data quality. Data value is strictly dependent on its accuracy, reliability, timeliness and trustworthiness;
- Data suitability. Data has potential added value when it is exploitable for users in their operations, decision making and planning.

These qualities, in turn, transform data into a valuable product that customers would like to buy. However, selling data requires careful compliance with multiple regulations related to data privacy and data ownership as well as ensuring that a proper data license is delivered to protect data owners against data abuses. It also requires the commitment of providers to deliver high-quality.
This document presents the enablers and design of the marketplace component of the SynchroniCity framework, as well as the first Proof-of-Concept (PoC) integrated with the IoT infrastructure in Santander RZ. We analyze recent data marketplace platforms by examining their main characteristics in order to shape our vision for a platform aiming to cope with urban IoT data. We derive the value proposition of the marketplace by conducting interviews with several stakeholders (including representatives of seven RZs and three project partner SMEs). The value proposition underpins the design of the marketplace, for which we adapt and extend the FIWARE/TM Forum Business API Ecosystem Generic Enabler. We finally present a first PoC prototype of the marketplace, including its basic features to show its integration with the IoT infrastructure in Santander RZ, using the existing data streams deployed there in a use case scenario involving weather forecast.

2 Survey of data marketplace platforms

Data marketplaces are unlike traditional forms of e-Commerce, as the data is not so much ‘sold’ as ‘leased’, and data owners wish to retain a degree of control over the rights and restrictions regarding access and usage [1, 2]. More specifically, IoT data marketplaces are platforms to integrate IoT data sources with financial transactions and brokerage concepts for IoT data and services, including marketing and pricing, so that IoT sensor data, access to actuators, and relevant data processing/analytics functions (through APIs) can be efficiently listed, searched, and traded among users (i.e., data providers and data consumers) [3]. Within these platforms, description of IoT data sources typically includes physical and technical elements such as type, size, location, generation time, access method, ownership, etc., as well as their contents. Data providers use this description to specify what to sell and how it is being offered (price and licensing), e.g., public, semi-public, private, access right (read, write, modify, resale, etc.). On the other hand, data consumers can discover what to buy and how it can be bought (price and licensing). The IoT data marketplace provides a standardized way to handle the above data [4].
A number of centralized data marketplaces have recently emerged, however they require that participants trust the marketplace provider for data trade, payments and platform-related storage (e.g., feedback, rating, license agreements, transaction history).

The City Data Exchange (CDE) [5] is a software-as-a-service solution that allows users to sell, purchase, and share a wide variety of data from multiple sources among different users in the city of Copenhagen such as citizens, city government and businesses. The exchange enables large companies, small and medium enterprises, startup companies, as well as the academic and public sectors, to come together and integrate multiple sources of information to meet the challenges of sustainability and quality of life by allowing developers to build data-driven applications without investing resources in infrastructure development and data gathering. Potential customers can search for data sources and filter them by charge type (free vs. priced ones), update frequency (from one time up to annual update frequency) and category (e.g., infrastructure, demographics, utilities usage). Datasets that contain geographical information can be displayed on a map when the user selects a specific map search option. CDE offers several payment models (price plans), for example, one-time payment, subscription based model, and pay-per-use model. The first option is typically used when purchasing access to static datasets, while the subscription and pay-per-use options are used when purchasing access to data streams.

Thingful [6] is building a technical infrastructure for making distributed urban IoT data accessible in real-time by third parties in a trusted environment with data owners retaining full control over their data. Their approach aims at leaving data where it is created and owned, and not centralizing it in a single data hub or repository.
Similarly, the European project *BIG IoT* [7], funded by the Horizon 2020 program, focuses on bridging the interoperability gap between the vertically-oriented IoT platforms through the development of a common API. On top of this mutual API, the project is developing a data marketplace, acting as a Business to Business (B2B) broker for trading access to IoT information and functions. The BIG IoT marketplace [8] allows data providers to register their offerings (based on semantic descriptions) and data consumers to discover relevant offerings (based on semantic queries) at run-time. It also provides accounting support for consumers and providers to track the amount of resources accessed, as well as a web portal for developers and administrators to support

the implementation and management of their applications, services, and platforms. All entities registered in the marketplace are semantically annotated to enable better discoverability and orchestration. Recently, the BIG IoT team announced a collaboration with Thingful to build a software bridge between the two platforms aiming to provide very powerful search and access capabilities in both directions (from Thingful to BIG IoT and vice versa) [9] based on a common semantic vocabulary.

The *Business API Ecosystem (BAE) GE* [10] is a joint task force between *TM Forum* and *FIWARE* aiming to provide a standard set of APIs for enabling the management and monetization of different kinds of assets. Although the BAE supports both physical and digital assets, its features are well suited for monetizing IoT data sources. Data price can be based on both the basic cost to produce and curate data and the enterprise’s policy on revenue targets. Revenue sharing can be set to allow monetization of data sources related to multiple parties and to pay the marketplace provider for transaction fees. Pricing plans support one-time payment, subscription, and pay-per-use payment models. However, data can be offered for free as well. Additionally, BAE offers an endpoint to receive accounting information in order to charge customers periodically based on their consumption for those offerings who have defined a pay-per-use price plan.

MK Data Hub [11] is the key technical infrastructure component of the MK:Smart project, dedicated to the acquisition, management and processing of data relevant to the city so that they can be used in analytics and software applications. Both APIs and developer tools are provided to enable developers to find relevant data amongst the one offered within the data catalog, and data providers/owners to share and distribute their data through the data hub. The MK Data Hub provides an interface for reading streams of data that are aggregated from a multitude of datasets registered within the data catalog. The aggregation logic revolves around presenting all the data that the user has access to. Data access can be granted/denied according to several attributes (e.g., everyone, logged-in user, exceptions, specific users) while its license can be selected from a predefined list (e.g., CCA 4.0 International, CCA-ShareAlike 4.0 International, Flickr APIs terms of use, ODC Database Contents License DbCL 1.0).

Dawex [12] is a global data marketplace where organizations meet, buy and sell data, directly and securely. Acting as a trusted third-party, Dawex operates on a transactional-based revenue model. On Dawex, each organization is rigorously vetted with a strict process to ensure its identity. Companies can monetize datasets and APIs, raw data, refined data, and insights. Any kind of data can be traded: customer and product-related data, financial data, IoT data, licensed as one-off deals or subscriptions. Data license definition is very flexible and can be expressed by data providers in terms of exclusivity, territories, business sectors, uses, duration and right to sub-license. Dawex is also testing blockchain technology by means of a signed-contract-vault, a smart contract running on the Ethereum blockchain which stores GPG signatures of the agreed data license between seller and consumer. The Blockchain properties ensure that nobody can modify those signatures, therefore both sellers and consumers have the guarantee that the terms of the contract they agreed on are immutable [13].

Terbine [14] is designed to connect with prevailing IoT platforms and analytics engines, as an ultimate collector and source of both raw and processed sensor information. The system is intended to offer not only curation for publicly available sensor data, but a means for organizations to monetize their investments in IoT technologies on a widely accessible basis. Currently, Terbine focuses on Government, Financial and Energy sectors. Data quality is based on a grade mechanism related to how the provenance is tracked. Terbine automatically generates rich metadata for datasets or streams submitted into the system, containing attributes including provenance, ownership, legal and regulatory, context and geolocation.

More recently, decentralized approaches are being announced, according to which the data marketplace plays a more neutral role with respect to a centralized one. Marketplace operations are backed by Blockchain and Distributed Ledger Technologies (DLTs), and are governed by the entire set (or in some cases a predefined subset) of participants through consensus algorithms and independent verification mechanisms. However, their maturity level is affected by the initial development status of such technologies.

Streamr [15] is a real-time data backbone based on a decentralized network for scalable, low-latency and untamperable data delivery, operated by the DATAcoin token. The marketplace is open to everyone and users can publish new data streams and subscribe to the available ones. Access to data is purchased by means of license contracts. In return for DATAcoin, the contract grants access to an associated set of streams by registering the purchaser’s public key with the streams. The data license can be valid for a certain period. After the license expires the buyer will no longer have access to new data published on the stream(s). The terms of use may be stored in the data license contract either directly or as a link to content-addressed storage such as the Inter-Planetary File System (IPFS) [16]. The contract may also contain proof about fulfilled legal requirements such as the result of a Know-Your-Customer (KYC) process.
DataBroker DAO [17] is a data marketplace for IoT data that will connect sensor owners with purchasers of the data directly, utilizing existing infrastructure from telecommunication providers operating sensor connectivity networks based on GSM, LoRaWAN, SigFox or via a proprietary gateway of the sensor owner. Currently, DataBroker DAO runs on a private Ethereum blockchain network, planning to support data storage through IPFS or traditional cloud solutions.

DEX [18] is a marketplace based on the Ocean protocol, a decentralized backbone that incentivizes the supply of large volumes of high-quality data by using mechanisms to mitigate bad behavior. The network uses *BigchainDB* [19], a decentralized database network, for metadata capture. DEX reference data marketplace will have a permissive open source license to allow other startups to use the code as a starting point. DEX approach is not to store the data in the marketplace, rather to keep the data stored in their original premises while providing means to access them through a decentralized access control mechanism. Although free data exchanges are supported, multiple payment models enable monetization in the forms of fixed price, auction, and royalties.

IOTA [20] data marketplace proposes a slightly alternative approach, based again on a decentralized architecture based on a *directed acyclic graph*, called “tangle”. Every transaction validates two previous ones, which in turn validated two previous ones before them, and so on. This means that validation of the network is an intrinsic property of the network and no longer decoupled from the usage of the network, as in the case of regular Blockchain architectures. As a result, fees for transactions are entirely removed. Currently, the entire prototype runs in real-time on the IOTA test network, with full end-to-end data verifiability and security. Future plans include more emphasis focused on the impact of the EU GDPR.

The main characteristics of the IoT data marketplaces analyzed in this Section are summarized in Table 1.

Table 1: Summary of IoT data marketplaces.

Marketplace	**Architecture**	**Data dynamicity**	**Data Access**	**Trustworthiness**	**Source code**
Thingful	Centralized	Strems	External API	Feedback	Partially open
Big IoT	Centralized	Streams	External API	Feedback	Open*
Fiware BAE	Centralized	Streams / static datasets	External API / download	-	Open
Dawex	Centralized*	Streams / static datasets	Internal API / download	Completeness	Closed
Terbine	Centralized*	Streams / static datasets	Internal API	Provenance	Closed
CDE	Centralized	Streams / static datasets	Internal API / download	Feedback	Closed
MK Data Hub	Centralized	Streams / static datasets	Internal API / download	-	Open
Streamr	Decentralized	Streams	Streamr Network	Feedback	Open*
DataBroker DAO	Decentralized	Streams / static datasets	dAPIs	Feedback	Open*
DEX	Decentralized	Streams / static datasets	BigchainDB	Feedback	Open*
IOTA	Decentralized	Streams	-	-	Open*

The *architecture* of data marketplaces is either centralized or decentralized. In the former case, the functionalities of the platform such as data trading, payments, and platform related storage (e.g., feedback, rating, license agreements, transaction history) are controlled by the marketplace provider, whilst in the latter case, they are managed by the entire community of participants through consensus algorithms. Some centralized marketplaces such as Dawex and Terbine have started to integrate decentralized functionalities to store license agreements and transactions history, respectively. This is represented in the table by the term *centralized**.
Data dynamicity classifies the types of data with respect to the time dimension. They can be streams, updated with a certain frequency, or static datasets. In some cases, the latter can be periodically amended with up-to-date information and also provide APIs that enable querying and filtering functionalities.
Data access shows the different ways data consumers access purchased data sources. In centralized marketplaces consumers access data via either an internal API provided by the marketplace or an external API provided by the data provider. In the latter case, data consumption happens outside the scope of the marketplace. Differently, decentralized marketplaces provide data access by leveraging decentralized APIs (dAPIs) or on blockchain based storage protocols such as BigchainDB or the Streamr Network.

Trustworthiness shows the mechanism provided by the marketplace platforms to have a clear understanding around the trustworthiness of data, data providers, and data consumers. Most platforms have a feedback mechanism, which is based on either comments or rating scores provided by data consumers. Alternatively, Dawex mechanism is based on completeness metrics (e.g., percentage of missing entries in the dataset, precise description), whilst Terbine provides information of data provenance expressed in terms of data source reliability.

Source code indicates whether the source code of the platform is open or not. FIWARE/TMForum BAE, and MK Data Hub have released their source code on GitHub so that the open source community can contribute to improve their projects, while Dawex, Terbine, and CDE followed a closed source approach. Thingful has only released its API specification and various plugins and connectors, while Big IoT, Streamr, DataBroker DAO, DEX, and IOTA, intend to release their source code when they reach a higher maturity level.

3 Value proposition design

Value proposition design (VPD) is a methodology to shape products and services, and map them against customer requirements. Broadly speaking, the goal of VPD is to create new value propositions, which are promises of value to be delivered by a company to a customer or customer segments. In other words, a value proposition is a business or marketing statement that a company uses to summarize why a consumer should buy or use a service.
In this Section, we present the VPD to nail down the value propositions for the IoT data marketplace. As value propositions target the customers/stakeholders that will benefit from using a specific product, we first identified the main customer segments for the marketplace.
They represent a subset of all the stakeholders participating in the SynchroniCity platform, as discussed in D1.3, and are reported in the following:

- *Data providers*: they can be SMEs that might decide to share/sell their data for secondary use, such as taxi companies; public/private organizations related to tourism, environmental monitoring, utilities, etc.; service providers that might sell aggregate information; live connected IoT devices from testbeds, pilots, and functional real-world deployments.
- *Data consumers*: they purchase data on the marketplace to use them directly or to build new services on top of them, which in turn can be resold. Therefore, data consumers can be citizens, service providers, and city councils.
- *Marketplace provider*: it can be a single entity or a consortium which provides the basis for the provisioning of the underlying platform and its governance.
- as a fourth customer segment, we also consider *city councils* when playing the role of marketplace providers.

In the following, we present the *value proposition canvases (VPC)* for each of the above customer segments and then merge the insights obtained to derive comprehensive value propositions for the IoT data marketplace.
The VPC tool allows to make value propositions visible and tangible and thus easier to discuss, manage, and revise. A VPC has two sides:

- the *customer (segment) profile* allows to describe a specific customer in a more structured and detailed way, by breaking it down into its *jobs*, *pains*, and *gains*.
- the *value (proposition) map* describes how we create value for a specific customer, by breaking the value proposition down into *products and services*, *pain relievers*, and *gain creators*.

To shape the profiles of our customers, we got in touch with several organizations representing the customer segments targeted by our marketplace.

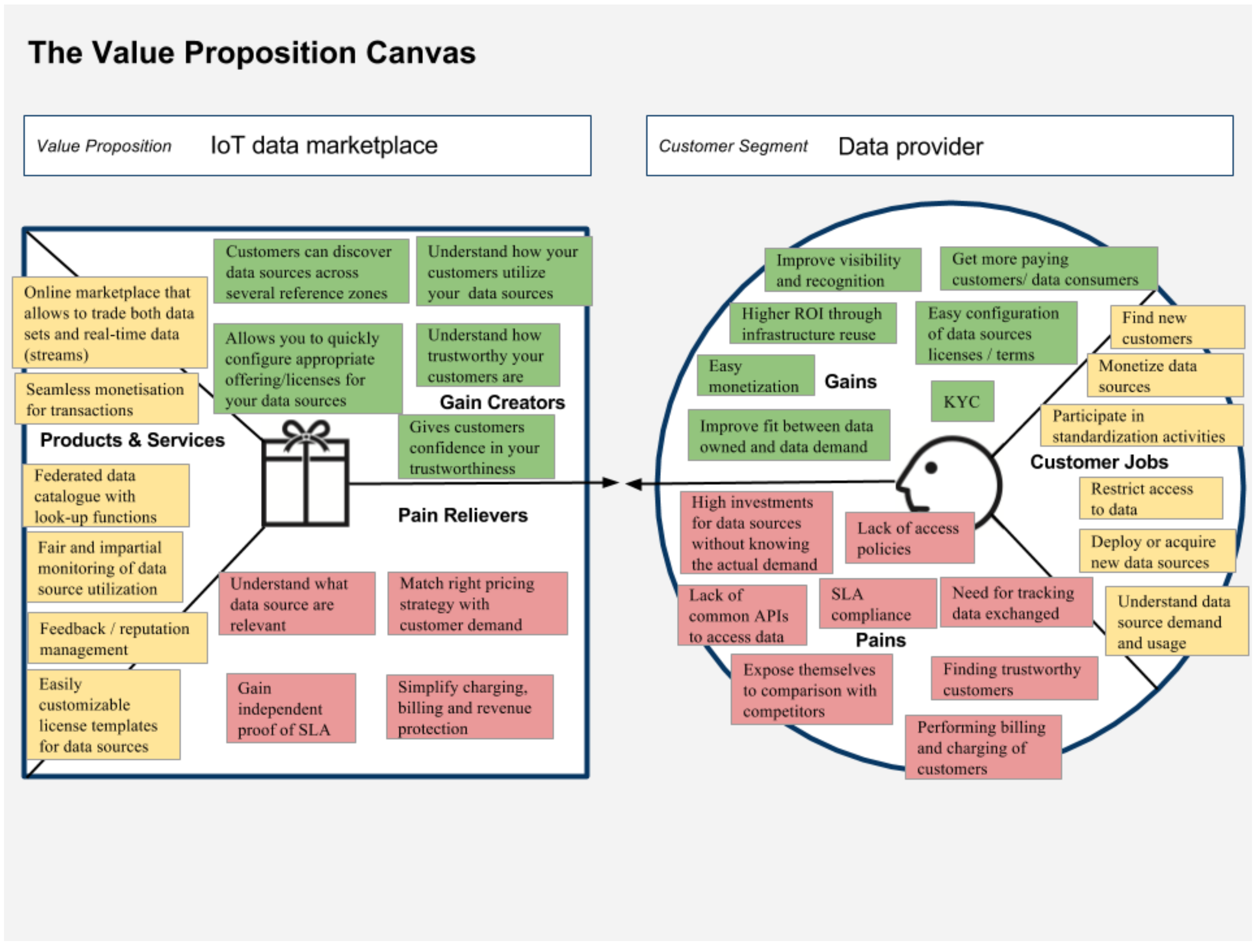
All the organizations reached out are involved in the SynchroniCity project. Specifically, we interviewed three representatives of IoT companies (TST Sistemas, Bronze Labs, and Ubiwhere) acting as data providers and data consumers, as well as seven representatives of city councils (Antwerp, Carouge, Copenhagen, Helsinki, Manchester, Santander and Porto).
In the following sections, for each customer segment, we first present the VPC, then describe its parts, namely the customer profile and the value map, and finally derive the overall value propositions for the marketplace. We analyze each customer segment separately by assembling the

information elicited by the organizations belonging to the same customer segment.
To shape the customer profiles, for each customer segment, we identify: (i) customer jobs, which are the things customers are trying to get done and the tasks they are trying to perform and complete; (ii) pains, describing negative emotions, undesired costs/situations and risks that customers will or can experience before, during, and after trying to get their jobs done; (iii) gains, describing the outcomes and benefits required, expected, or desired by the customers, or the ones that would surprise them.
To build our value maps, for each customer segment we specify: (i) products and services, which the value proposition is built on and which we offer to help customers to get their jobs done; (ii) pain relievers, describing how exactly our products and services alleviate specific customer pains; (iii) gain creators, outlining how our products and services create customer gains, that is how we intend to produce outcomes and benefits that our customers expect.
We conclude our VPD by deriving and presenting overall value propositions, which are composed of various products and services specified for all the customer segments targeted by the marketplace.

3.1 Data providers

3.1.1 Value proposition canvas

The VPC for the data provider customer segment is illustrated in Figure 1. In the following sections, we describe its parts, namely the customer profile and the value map.



The Value Proposition Canvas
Value Proposition
IoT data marketplace
Customer Segment
Data provider
Online marketplace that allows to trade both data sets and real-time data (streams)
Seamless monetisation for transactions
Products & Services
Federated data catalogue with look-up functions
Fair and impartial monitoring of data source utilization
Feedback / reputation management
Easily customizable license templates for data sources
Customers can discover data sources across several reference zones
Understand how your customers utilize your data sources
Allows you to quickly configure appropriate offering/licenses for your data sources
Understand how trustworthy your customers are
Gain Creators
Gives customers confidence in your trustworthiness
Pain Relievers
Understand what data source are relevant
Match right pricing strategy with customer demand
Gain independent proof of SLA
Simplify charging, billing and revenue protection
Improve visibility and recognition
Get more paying customers/ data consumers
Higher ROI through infrastructure reuse
Easy configuration of data sources licenses / terms
Easy monetization
Gains
KYC
Improve fit between data owned and data demand
Find new customers
Monetize data sources
Participate in standardization activities
Customer Jobs
Restrict access to data
Deploy or acquire new data sources
Understand data source demand and usage
High investments for data sources without knowing the actual demand
Lack of access policies
Lack of common APIs to access data
SLA compliance
Need for tracking data exchanged
Pains
Expose themselves to comparison with competitors
Finding trustworthy customers
Performing billing and charging of customers

Figure 1: Value proposition canvas for the data provider customer segment.

3.1.2 Customer profile

Customer jobs

- Data providers have a niche customer base. The primary job they want to get done is widening the scope of their customers and get in touch with several municipalities to put their solutions into value.
- The enablement of secondary revenue streams for IoT data as an opportunity for improving the ROI is also something data providers are looking at in order to recoup some of their investments in developing IoT sensors and platforms.

- Data providers are also interested in taking part in common standardization initiatives. In fact, to get some benefits out of their systems, it is crucial that data generated by them are compliant with well-known and accepted formats.
- Restrict access to data and make it visible only to their customers. Improving privacy and having licenses to give the right to access the data for a specific period of time and on certain conditions (and for a fee).
- New devices and data sources are required and need to be deployed to enrich their offer, thus covering a wider spectrum of application scenarios for their clients.
- Data providers need to have a clear understanding of what data is of particular interest and for what it is used, so that they can focus their investment targeting demanded and useful data.

Pains

- Data providers are worried to perform high investments for data sources without knowing whether there is actual demand for the data generated by their devices. Understanding the market value of data would be helpful to motivate such an investment.
- Lack of access policies is actually preventing providers from sharing their data as privacy requirement would be infringed. Also, sensitive data must be compliant with GDPR and ePrivacy directives.
- Lack of common APIs to access IoT data and lack of a standard data formats make it challenging for data providers to deploy and operate their devices in different environments/contexts.
- SLA compliance is a critical issue for data providers. Whenever the behavior of a specific solution breaks SLAs, they go on the spot, evaluate the issue and fix it. They can also pay money back (penalties) and lose reputation. Tracking data delivery of periodic data exchanges could help early detection of issues in devices and edge networks.
- Tracking data exchanged with the data consumers should be a must as it will help resolve possible disputes and support audit check.
- Data providers are generally worried about exposing themselves in a way that facilitates an easy comparison with their competitors.
- Trust is reciprocally built with loyal customers over several iterations, while reaching new trustworthy customers is not easy in a fragmented market such as the IoT one.
- Setting up a charging and billing infrastructure can be challenging.

Gains

- Be part of a platform ecosystem widely accepted at a European level, which also aims at standardizing both data formats and their access, entails a general improvement of visibility and recognition for data providers. This will eventually attract more customers to their businesses.
- A clear benefit data providers will get from such a platform is an increased exposure to consumers, as their data sources become more easily discoverable.
- Easier ways for trading and monetizing data can enable secondary revenue stream for data providers, as well as contribute to faster CAPEX recovery and improve the ROI through infrastructure reuse.

- Ability to reconfigure the license and terms of use of data sources provided.
- Obtain insights from information about data exchanged within the platform.
- Visible and measurable market demand, thus improving fit between data offered and real data demand.
- Integration of customer acceptance policies, identification procedures and monitoring of transaction to fulfil Know Your Customer (KYC) process requirements.

3.1.3 Value map

Products and services

- An online marketplace that allows trading of static datasets and real-time data streams. In the latter case data consumers can be charged according to a recurring price plan and pay a flat periodic subscription fee, or they can be charged periodically according to the amount of data they have actually “consumed”.
- A seamless monetization mechanism for transactions that supports traditional electronic payment channel (e.g., Paypal, Stripe).
- The platform allows data providers to expose their data sources in a federated catalogue of IoT data generated by solutions deployed in several cities.
- Fair and impartial mechanism to monitor and track IoT data flows from data providers’ devices/gateways to data consumers’ applications/services.
- A reputation mechanism allows the parties involved in transactions to provide feedbacks in order to rate and rank both data providers and consumers.
- Data providers can leverage easily customizable license templates to define access and usage policies to their data.

Gain creators

- Data sources can be discovered and accessed by data consumers across several European reference zones.
- Means to let data providers understand how their customers (data consumers) utilize data sources.
- Data providers are allowed to quickly configure appropriate offering/licenses for data sources.
- Means to allow data providers understand how trustworthy data consumers are.
- Give data consumers confidence in the trustworthiness of data providers.

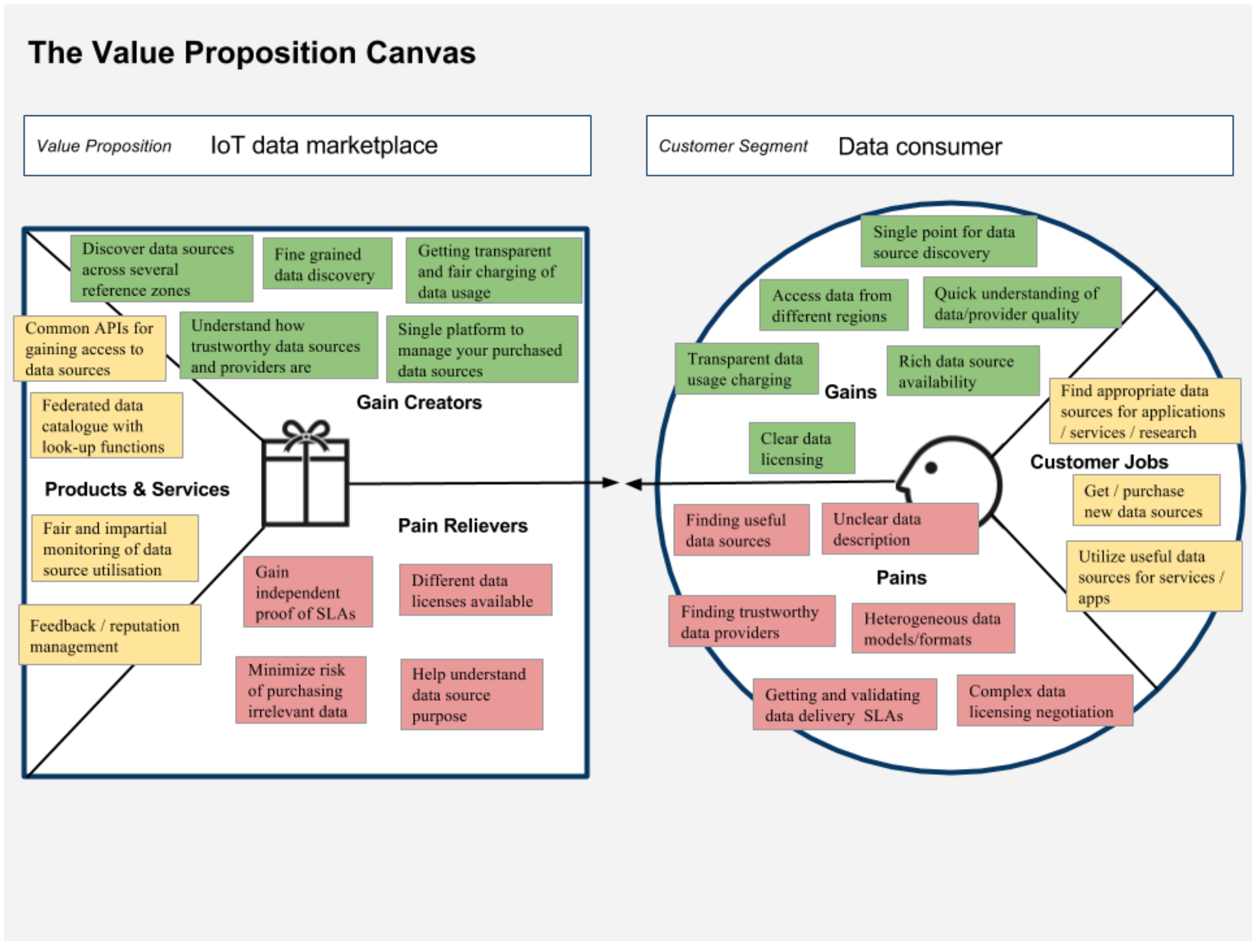
Pain relievers

- Means to understand what data sources are relevant in order to minimize the risk of providing irrelevant data and then improve sales.
- Capabilities to match right pricing strategy with the customer demand.
- Gain independent proof of SLA.
- Simplify charging, billing and revenue protection.

3.2 Data consumers

3.2.1 Value proposition canvas

The resulting VPC for the data consumer customer segment is illustrated in Figure 2. In the following sections, we describe its parts, namely the customer profile and the value map.



The Value Proposition Canvas
Value Proposition
IoT data marketplace
Customer Segment
Data consumer
Discover data sources across several reference zones
Fine grained data discovery
Getting transparent and fair charging of data usage
Common APIs for gaining access to data sources
Understand how trustworthy data sources and providers are
Single platform to manage your purchased data sources
Gain Creators
Federated data catalogue with look-up functions
Products & Services
Fair and impartial monitoring of data source utilisation
Pain Relievers
Gain independent proof of SLAs
Different data licenses available
Feedback / reputation management
Minimize risk of purchasing irrelevant data
Help understand data source purpose
Single point for data source discovery
Access data from different regions
Quick understanding of data/provider quality
Transparent data usage charging
Gains
Rich data source availability
Find appropriate data sources for applications / services / research
Clear data licensing
Customer Jobs
Get / purchase new data sources
Finding useful data sources
Unclear data description
Utilize useful data sources for services / apps
Pains
Finding trustworthy data providers
Heterogeneous data models/formats
Getting and validating data delivery SLAs
Complex data licensing negotiation

Figure 2: Value proposition canvas for the data consumer customer segment.

3.2.2 Customer profile

Customer jobs

- The primary job data consumers want to get done is geared towards having a platform where they can easily search for datasets or streams without being locked in the data silos of data providers.
- Some data consumers want data for free, others are keen to purchase data, which is used to feed their services and which in turn they aim to sell.
- Generally, they want to find useful data at the right quality and price.

Pains

- Data consumers struggle to find useful data sources and sometimes data description is either poor or missing.
- They are generally worried about the quality of data and related trustworthiness of data providers.
- Heterogeneous data models and lack of standard data formats make data consumers spend time in adaptation/translation tasks.
- Monitoring SLA compliance is a challenging task since whenever two parties (i.e., data provider and data consumer) agree on a SLA, valid and impartial facts to determine whether a SLA is breached are needed by both.
- Complex data licensing negotiation can discourage potential data exchange.

Gains

- Leveraging a platform where to look up for specific data sources brings in the clear benefit for data consumers of having a single point for discovering and accessing data.
- Data sources availability is expected to be rich as data consumers could access data generated by systems deployed in several geographical areas.
- Have a transparent mechanism according to which data consumers can quickly understand the trustworthiness of data providers.
- Have a transparent mechanism that allows data consumers to be charged on the basis of the actual delivery of data when coming from real-time stream channels.
- Data consumers have the possibility to choose a clear license model among those offered along with a specific data source. The availability of a specific license model for a given data source can help data consumers choose the right source according to their requirements.

3.2.3 Value map

Product and services

- Common APIs that allow data consumers to create replicable services on top of possibly heterogeneous data sources.
- An online platform that allows data consumers to look up data sources in a federated catalogue of IoT data generated by solutions deployed in several cities. Main characteristics of data sources are provided in a human readable standard format for an easy discovery.
- Fair and impartial mechanism to monitor and track IoT data flows from data providers' devices/gateways to data consumers' applications/services.
- A reputation mechanism allows the parties involved in transactions to provide feedbacks in order to rate and rank both data providers and consumers.

Gain creators

- Data sources can be discovered and accessed by data consumers across several European reference zones.

- A fine grained data discovery mechanism allows data consumers to easily discover data sources by using specific tags in their searches (e.g., type of application, city of deployment, type of data source).
- When purchasing real-time data streams, data consumers can choose to get charged transparently and fairly according to the actual delivery/usage of data.
- Means to understand how trustworthy data sources and providers are.
- Having a single platform to manage purchased data sources (e.g., keep track of them, renew subscriptions, update the API keys).

Pain relievers

- Gain independent proof of SLAs.
- Have the possibility to choose among different data licenses.
- Rich description of data sources minimizes the risk of purchasing irrelevant data and helps clarifying which purpose data source could serve.

3.3 Marketplace provider

3.3.1 Value proposition canvas

The VPC for the marketplace provider customer segment is illustrated in Figure 3. In the following sections, we describe its parts, namely the customer profile and the value map.

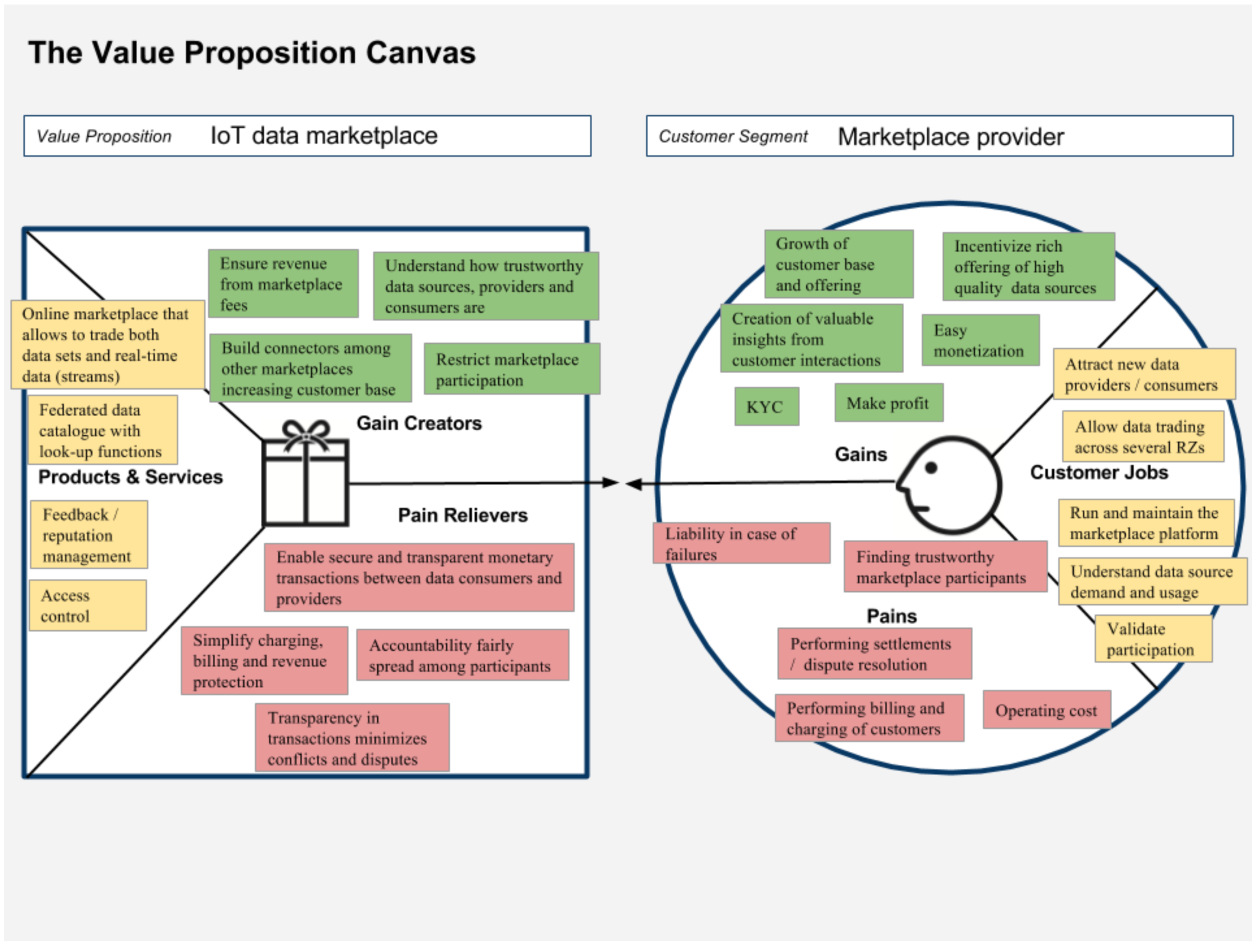

The Value Proposition Canvas
Value Proposition
IoT data marketplace
Customer Segment
Marketplace provider
Ensure revenue from marketplace fees
Understand how trustworthy data sources, providers and consumers are
Online marketplace that allows to trade both data sets and real-time data (streams)
Build connectors among other marketplaces increasing customer base
Restrict marketplace participation
Federated data catalogue with look-up functions
Gain Creators
Products & Services
Feedback / reputation management
Pain Relievers
Access control
Enable secure and transparent monetary transactions between data consumers and providers
Simplify charging, billing and revenue protection
Accountability fairly spread among participants
Transparency in transactions minimizes conflicts and disputes
Growth of customer base and offering
Incentivize rich offering of high quality data sources
Creation of valuable insights from customer interactions
Easy monetization
Attract new data providers / consumers
KYC
Make profit
Allow data trading across several RZs
Gains
Customer Jobs
Run and maintain the marketplace platform
Liability in case of failures
Finding trustworthy marketplace participants
Understand data source demand and usage
Pains
Validate participation
Performing settlements / dispute resolution
Performing billing and charging of customers
Operating cost

Figure 3: Value proposition canvas for the marketplace provider customer segment.

3.3.2 Customer profile

Customer jobs

- The primary job for a marketplace provider is to attract new customers, be it either data providers or consumers, in order to increase its revenue opportunities.
- Generally, the marketplace provider wants to scale up and facilitate data exchanging and trading across several reference zones.
- The marketplace provider will run and maintain the platform.
- The marketplace provider wants to understand and match data offer and demand in order

to opportunely stimulate the ecosystem. Some may want to be economically sustainable, others want to make money.
- The marketplace provider also wants to validate customers before letting them participate actively on the marketplace.

Pains

- The main concern for the marketplace provider is its liability in case of failures causing economic loss for any of the other stakeholders involved.
- Finding trustworthy customers is crucial for both the overall trust and credibility of the platform.
- Whenever a dispute arises between a data provider and a consumer, the marketplace provider is in charge of resolving the dispute and reaching a settlement.
- To be in charge of billing and charging the customers.
- Operating cost to keep the platform up and running.

Gains

- Having a platform for exchanging and trading IoT data stimulate the growth of customer base and offering.
- Means to incentivize rich offering of high quality data sources.
- Means to create valuable insights from customer interactions.
- The platform enable easy monetization mechanisms.
- Integration of customer acceptance policies, identification procedures and monitoring of transaction to fulfil Know Your Customer (KYC) process requirements.
- Make profit.

3.3.3 Value map

Product and services

- An online marketplace that enables to trade both datasets and real-time data streams. Marketplace also supports free of charge transactions to exchange open data.
- A federated catalogue that attracts customers from different geographical areas stimulates the ecosystem and allows the marketplace provider to gain scale and speed in the adoption of the platform.
- A reputation mechanism allows the parties involved in transactions to provide feedbacks in order to rate and rank both data providers and consumers, and to promote honest behavior among the customers.
- An access control mechanism allows the marketplace provider to validate customers before letting them use the platform.

Gain creators

- Ensure revenue by receiving a small recurring fee from data providers for putting their device on the platform and also a fee for each transaction on the platform.
- Understand how trustworthy data sources, providers and consumers are.

- Connectors with other instances of the federated marketplace increase the customer base.
- Restrict marketplace participation.

Pain relievers

- The marketplace enables secure and transparent monetary transactions between data consumers and providers which allows to resolve disputes that may arise between data providers and consumers (e.g., infringements of SLAs).
- Secure charging, billing and revenue protection.
- Accountability fairly spread among participants.

3.4 City councils as marketplace providers

3.4.1 Value proposition canvas

The VPC for the city council customer segment is illustrated in Figure 4. In the following sections, we describe its parts, namely the customer profile and the value map.

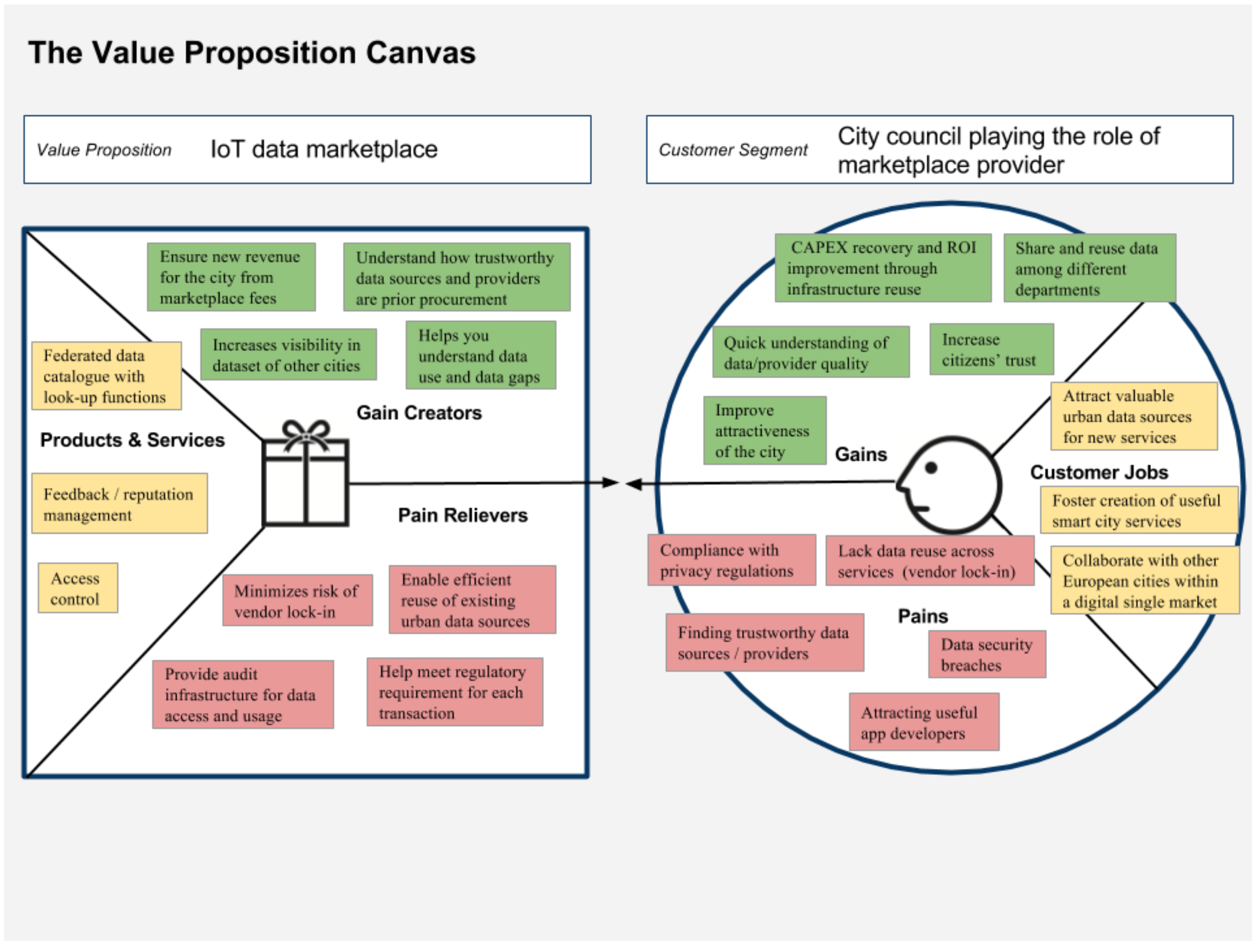

The Value Proposition Canvas
Value Proposition
IoT data marketplace
Customer Segment
City council playing the role of marketplace provider
Ensure new revenue for the city from marketplace fees
Understand how trustworthy data sources and providers are prior procurement
Federated data catalogue with look-up functions
Increases visibility in dataset of other cities
Helps you understand data use and data gaps
Gain Creators
Products & Services
Feedback / reputation management
Pain Relievers
Access control
Minimizes risk of vendor lock-in
Enable efficient reuse of existing urban data sources
Provide audit infrastructure for data access and usage
Help meet regulatory requirement for each transaction
CAPEX recovery and ROI improvement through infrastructure reuse
Share and reuse data among different departments
Quick understanding of data/provider quality
Increase citizens' trust
Improve attractiveness of the city
Gains
Attract valuable urban data sources for new services
Customer Jobs
Foster creation of useful smart city services
Compliance with privacy regulations
Lack data reuse across services (vendor lock-in)
Collaborate with other European cities within a digital single market
Pains
Finding trustworthy data sources / providers
Data security breaches
Attracting useful app developers

Figure 4: Value proposition canvas for the city council customer segment.

3.4.2 Customer profile

Customer jobs

- Cities need to attract valuable urban data sources from public and private data owners and make them readily available for consumption by public and private entities that are working on data-driven city services.
- Cities need to identify the opportunities for their data to provide answers to their challenges (e.g., provide better services to citizens, improve citizens' quality of life) and foster the development of smart city services.

- European cities are keen to collaborate with each other within a digital single market in order to collectively share ideas and be able to provide similar services and opportunities to their citizens.

Pains

- Be compliant with data privacy regulations when dealing with IoT data that can potentially be personal.
- Suffer vendor lock-in when trying to reuse data across different services.
- Find trustworthy data sources and providers able to match the high quality level required to confirm and strengthen citizens' trust.
- Have in place processes and technological elements to prevent and handle potential data security breaches.
- Attract application developers in order to build useful solutions and services for the city.

Gains

- Easier ways for monetizing data sources can enable cities to generate new revenue streams while contributing to faster CAPEX recovery and improving the ROI through infrastructure reuse.
- Promoting data sharing among different departments can boost their efficiency on reusing data for several purposes.
- Having an online platform providing information about data sources quality and provider trustworthiness can help city councils to understand what data/provider can better suit their needs.
- Sharing more data in a transparent way while including citizens' feedback in the loop can increase citizens' trust on public administration.
- By increasing availability of previously unavailable datasets and data sources will improve attractiveness of the city by stimulating private organizations to exploit data and provide better services.

3.4.3 Value map

Product and services

- A federated catalogue that gathers data sources from different geographical areas stimulates the ecosystem and allows cities to exploit and share data sources outside their boundaries.
- The quality of data sources can be reviewed and ranked so that other customers can determine whether the data provided by each organization complies with the quality specified, encouraging better practice. Moreover, a reputation mechanism allows the parties involved in transactions to provide feedbacks in order to rate and rank both data providers and data consumers, and to promote honest behavior among them.
- An access control mechanism allows the city to review registration requests from both data providers and data consumers before granting them access to the platform.

Gain creators

- Ensure revenue by receiving a small recurring fee from data providers for putting their device on the platform and also a fee for each transaction on the platform.
- Provide qualitative and quantitative metrics to understand how trustworthy data sources and providers are prior procurement.
- Connectors with other instances of the federated marketplace increase the visibility of a city's data sources and stimulates data sharing and collaboration with other cities.
- Help understand data use and data gaps in order for the cities to focus their resources on data sources which can bring a real value to their communities.

Pain relievers

- Minimize the risk of vendor lock-in by providing open and standard APIs.
- Enable an efficient reuse of existing urban data sources generated by legacy devices.
- Provide audit infrastructure for data access and usage.
- Help meet regulatory requirements for each transaction.

3.5 Combined value proposition analysis

SynchroniCity IoT data marketplace will help unlock thousands of datasets and data streams currently hidden within their organizations or held in silos by providing technical means to manage data licensing, access, quality, and a natural incentive to share data through monetization.
IoT data will be published in catalogs featuring discovery and trading functionalities. Some organizations will offer data for free such as local authorities offering open data, while other organizations might prefer to get paid for their data to generate additional revenues. Offering paid-for data from other sources alongside open data would allow cities to measure what data is of real value to their communities.
The quality of datasets and data providers will be reviewed and ranked, encouraging better practice and fair competition. By providing access to data, cities would be able to design better initiatives and policies based on real data rather than being restricted to estimates or models. Cities fostering the creation of new smart services will attract useful data source providers and increase their ROI through infrastructure reuse.

3.5.1 Data catalog with lookup functions

SynchroniCity IoT data marketplace will embed a set of APIs instrumenting an online platform that allows data providers to expose their data sources in a federated catalog of IoT data generated by solutions deployed in several cities. By federating data offers from different RZs, both data consumers and data providers will benefit from a richer set of available data and from a wider visibility, respectively. Such a platform will provide easy discovery of data sources published, as it will allow data consumers to refine search by using specific tags (e.g., type of application, city of deployment, type of data source).

3.5.2 Access control

SynchroniCity IoT data marketplace needs to be adaptive and suitable to municipalities' political change and different regulations and policies. In this way, cities, according to their governance and whether they want to play the role of marketplace provider, will be able to decide how to regulate the access to the platform - either by vetting registration requests from both data providers and data consumers or by allowing an open access - how to federate with other cities, what kind of data are allowed (e.g., personal data, anonymized data).

3.5.3 Tools to create suitable agreements

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. SynchroniCity IoT data marketplace will allow data providers to define SLA agreements and to choose among several license models for their data, including both commercial and open licences. Furthermore, to better match the expectations of the stakeholders, the platform will 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;

3.5.4 Seamless monetization for transactions

Traditional electronic payment channel (e.g., Paypal, Stripe) will be supported. Revenue shares will be easily configured and automatically processed whenever transactions involve multiple stakeholders (e.g., data offering owned by multiple parties). The marketplace provider will receive a fee for each transaction as incentive to run the platform.

3.5.5 SLA adherence monitoring and dispute resolution

Trust is a fundamental aspect for every marketplace. Customers have to rely on the marketplace provider, or a third party, to have proof of SLA compliance, especially in the case of data streams, and to perform dispute resolution. To overcome these issues, the adherence to the agreed SLA will be tracked by a metering tool regulated by smart contracts within a fair and transparent process. The latter will ensure that SLA tracking is done internally by continuous assessment of data flow, rather than be based on costly and potentially biased assessments carried out by external authorities. This translates into a transparent system in which trust is based on deterministic technology rather than on unpredictable behaviors and unclear relationships among parties. Moreover, data flow will be technically verifiable by auditing immutable records. Customers will not need to necessarily trust each other, while on the other hand, the marketplace provider will be relieved from dispute resolution duties.

3.5.6 Tools to assess the quality and trustworthiness of IoT data and its providers

To boost data consumers' confidence on data offerings and data providers, SynchroniCity IoT data marketplace will provide a rating and reputation mechanism. Dataset and streams will be reviewed by data consumer and ranked based on their overall quality obtained, considering metrics such as actual delivery, verifiable provenance, accuracy, response time, and timeliness. This will help potential buyers to evaluate whether data offered is well documented, how often it is refreshed, whether it reflects an accurate and verifiable reality. In addition, it will push competition among data providers as their reputation and trustworthiness will be directly linked to the quality of their offerings.

4 SynchroniCity IoT data marketplace architecture

In the SynchroniCity approach, the marketplace platform will encourage sustainable commercial viability of data by developing a considerable added value that goes beyond traditional rights-based licensing models of datasets.
IoT sensor data and access to actuators are published within a catalogue featuring discovery and search functionalities so that assets can be easily and directly traded among users (e.g., data providers, application developers, researchers, etc.). By having an IoT data marketplace, application developers could obtain access to a multitude of sensors and/or actuators made available by assets owners to build new applications, thus enabling the creation of new smart city services. On the other hand, asset providers could be incentivized to deploy IoT infrastructures having the opportunity to generate revenue by selling their data on the market, which every user can benefit from. Marketplace providers could benefit from running such a platform by attracting new customers (e.g., data consumers and providers) making profit out of marketplace transactions. Cities fostering the creation of new smart services could attract useful data source providers and increase their ROI through infrastructure reuse.
In this section, we provide an overview of the marketplace architecture. We first present the underpinning core abstractions of the platform and describe the main components and their APIs. Eventually, we model the communication patterns among the components, between the users and the platform, as well as the interaction with the security component.

4.1 Core abstractions of the marketplace architecture

This section presents the *core abstractions* of the marketplace architecture which provide the key concepts enabling interaction with the platform. The core abstractions derive from the analysis performed in the previous sections and are effectively captured in the conceptual model of the SynchroniCity IoT data marketplace shown in Figure 5. At the core of the model is the notion of *offering*. An offering is related to a single *data source*, a digital asset registered into the marketplace by a *data provider*. A data source has an *endpoint* used to access it. When an offering is published into the marketplace by a data provider, it becomes discoverable by *data consumers*. These latter may purchase the access to the endpoint of the data source that an offering is related to or obtain it for free if that refers to an open data digital asset. *External data sources* (e.g., datasets already published in open data portals) can be imported into the marketplace by data providers which can then publish related offerings.

Several offerings may refer to the same data source as they can provide different licensing models (*licenses*) and *price plans*. Licenses are separated into: (i) *open schema*, according to which data providers can write and set the terms and conditions associated with their offerings, (ii) *standard license* that can be chosen from a predefined set (e.g., Open Data Commons, Creative Commons), and *personalized license*, according to which data providers can set additional restrictions to the access rights of their data, such as exclusivity, geography, duration, and purpose, along with the permissions regarding redistribution, adaptation, and resale.

Price plans are separated into the following options. *Pay per use* plan enables to charge the data consumers periodically according to the amount of data they have actually "consumed". With the *recurring* plan, data consumers pay a flat periodic subscription fee, while the *one-time payment* plan only charges data consumers at the time of purchasing the offering. Finally, some offerings might be *free of charge*, which is mostly the case of offerings referring to open data digital assets.

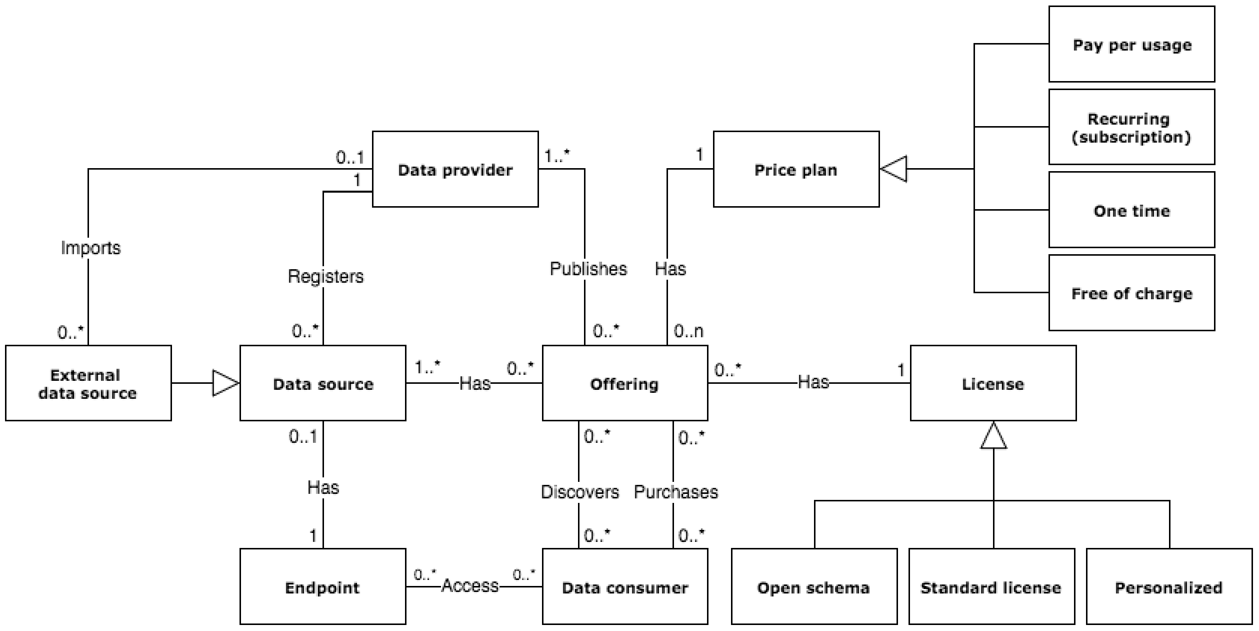


Figure 5: Conceptual model of the SynchroniCity IoT data marketplace.

4.2 Marketplace overview

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.
- **Security** includes three sub-components: (i) *Identity management* system to store and organize the identities of users; (ii) *Authentication* component that regulates the accesses to the platform; (iii) *Authorization* component that grants the access to the functionalities and digital assets according to predefined policies.

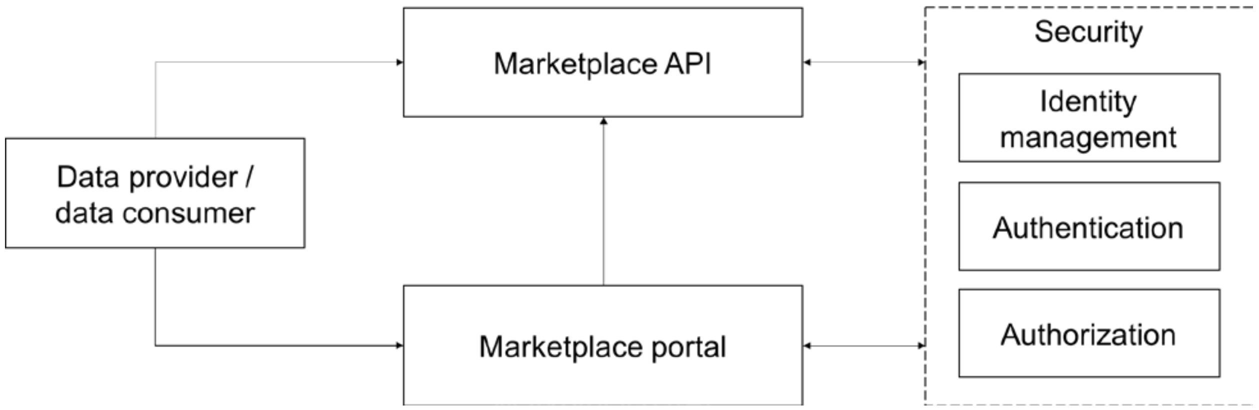


Figure 6: High-level view of the marketplace platform.

4.2.1 Marketplace API and portal

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. They can be accessed via the *marketplace portal*, which provides a user interface for data providers and data consumers to publish and purchase data offerings as well as manage other relevant information. An overview of the *marketplace API* and the *marketplace portal* is shown in Figure 7 and a description its components is provided next.

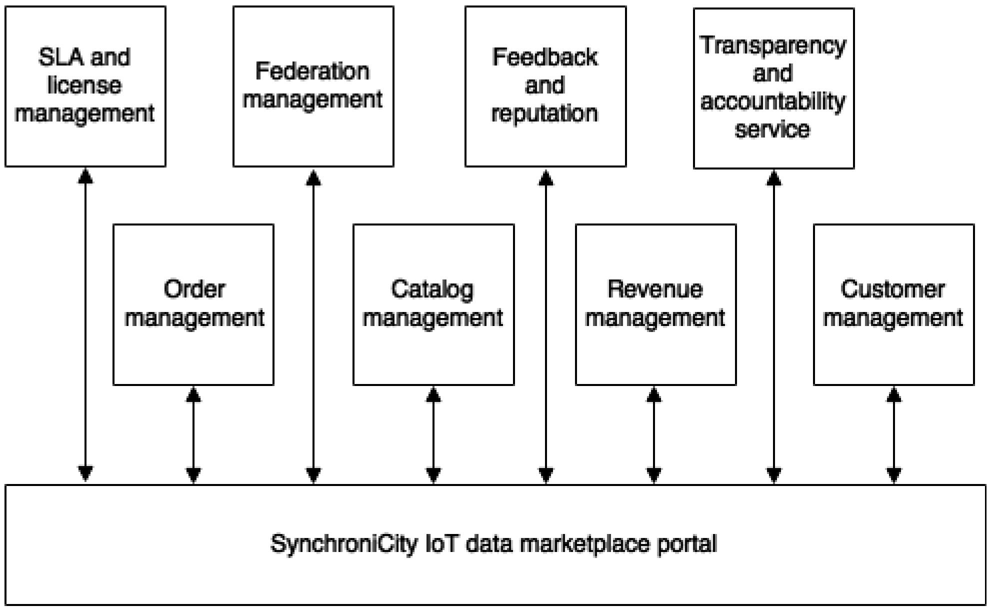


Figure 7: Marketplace API and portal.

4.2.1.1 Catalog management

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.
The main functionalities provided by this module are:

- Data catalog curation: allows the marketplace provider to create and update both catalogs 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 catalog 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.

Catalog management APIs are logically grouped into four sub-modules: category management, catalog management, data source specification management, and data offering management. More specifically:

- Category management. A Category is used to group catalogs and data offerings in logical containers. Categories can contain other categories being possible to create a tree of categories. It exposes the following interfaces:
    - Category collection: interfaces for retrieving a list of categories registered and for creating a new category.
    - Category entry: interfaces for retrieving and updating a specific category entry.
- Catalog management. These catalogs are collections of data offerings that are grouped together according to the data marketplace and data providers needs. It exposes the following interfaces:
    - Catalog collection: interfaces for retrieving a list of catalogs registered and for creating a new catalog.
    - Data source specification entry: interfaces for retrieving a specific catalog entry and for updating a specific catalog entry.
- 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. It exposes the following interfaces:

    - Data source specification collection: interfaces for retrieving a list of the registered data source specifications and for creating a new data source specification.
    - Data source specification entry: interfaces for retrieving and updating a specific data source specification entry.
- Data offering management: data offerings represent entities that are orderable from data providers and are published in the catalog. This resource includes pricing information and is linked to SLAs and license terms. It exposes the following interfaces:
    - Data offering collection: interfaces for retrieving a list of the data offerings available and for creating a new data offering.
    - Data offering entry: interfaces for retrieving and updating a specific data offering entry.

The complete data source specification management and data offering management submodules specification of this API, based on the FIWARE/TM Forum Business API Ecosystem specification [21] and adapted to best fit the SynchroniCity IoT Data marketplace requirements, is described in the Appendix and also available online at: https://synchronicityiotdm.docs.apiary.io/#, whereas the rest of the submodules (i.e., Category management and catalog management) is left unchanged with respect to the original FIWARE/TMForum specification.

4.2.1.2 Federation management

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.

The main functionalities provided by the federation management module are:

- 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 catalogs: this function provides the support for retrieving metadata information related to external data catalogs (e.g., CKAN [43], Hypercat [44]). 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.

The Federation management component, providing the above mentioned functions, allows to create a network of federated marketplaces that exposes a common catalog of data offerings. It is important to highlight that the federation should be considered at catalog 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.

Federation management APIs

- Federation management. API for the management of federation functionalities. It exposes the following interface:
    - Import data catalog collection: interface for retrieving a list of datasets from a specified (external) data catalog.

Currently, only the import data catalog collection interface specification has been fully specified and partially implemented (CKAN supported, Hypercat planned), and described in the Appendix. The rest of the federation management functionalities will be part of D2.5 (Advanced data marketplace enablers).

4.2.1.3 Order management

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 catalog 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[1]. 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. The main functionalities provided by this module are:

- 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.

[1] This feature is currently only specified in the API specification in the Appendix section but not integrated in the Proof-of-Concept prototype. Integration of this feature will be part of the next Deliverable: Advanced Data Marketplace Enablers.

- Management of purchased data offerings: allows data consumers to keep track of the assets purchased through the marketplace.

Order management APIs are logically grouped into four sub-modules: order management, inventory management, usage specification management, and usage management. More specifically:

- 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. It exposes the following interfaces:
    - Order collection: interfaces for retrieving a list of orders registered and for creating a new order.
    - Order entry: interfaces for retrieving and updating a specific order entry.
- 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. It exposes the following interfaces:
    - Product collection: interface for retrieving a list of the purchased data sources.
    - Product entry: interface for retrieving a specific data source purchased.

The order management submodules specification of this API, based on the BAE specification [21] and adapted to best fit the SynchroniCity IoT data marketplace requirements, is described in the Appendix (and also available online at: https://synchronicityiotdm.docs.apiary.io/#), whereas the inventory management, usage specification management and usage management specifications are left unchanged with respect to the original FIWARE/TMForum specification.

4.2.1.4 SLA and license management

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 template 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.
The main functionalities provided by this module are:

- 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.).

SLA and license management APIs are logically grouped into two sub-modules: license specification management and SLA specification management. More specifically:

- SLA specification management. A SLA specification is a detailed description of the SLA related to a particular data offering available in the marketplace. It exposes the following interfaces:
    - SLA specification collection: interfaces for retrieving a list of SLA specifications and for creating a new SLA specification.
    - SLA specification entry: interfaces for retrieving and updating a specific SLA specification entry.

- 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. It exposes the following interfaces:
    - License specification collection: interfaces for retrieving a list of license specifications and for creating a new license specification.
    - License specification entry: interfaces for retrieving and updating a specific license specification entry.

In the original reference implementation of the BAE, licenses were attached to the data source specifications. This approach is inflexible in respect to the distribution and monetization of IoT/smart city data. For example, a traffic dataset could be offered as open data for free, to be used for academic purposes, or for purchase, to be used for commercial use. In this case, the data provider would need to create two data source specifications for the same source but with different licenses, and then again two data offerings with different pricing models attached to the appropriate data source. For this reason, we have moved the attachment of data licenses to the data offering. Now, if a data provider wants to offer the same data under different monetization schemes based on different data licenses, only one data source specification needs to be provided for which multiple data offerings can now be created, each with their own data license. Furthermore, in the reference implementation of the BAE, the data license was simply provided as a free-form description. This approach leads to inflexible licenses that are not machine interpretable. Similar licenses could be provided in a way that although they are semantically equivalent, their description would differ from each other, making it harder for data consumers to understand the meaning of the data license and putting the pressure on the data provider side to be as specific in the data license description as possible. We are avoiding these pitfalls with a two-tiered approach. We offer two possibilities for the data providers to attach licenses that both decrease their effort and make it more understandable and transparent for the consumer side.
The complete SLA and license management API specification is described in the Appendix (as well as on https://synchronicityiotdm.docs.apiary.io/#). Currently, only the license specification management is integrated in the Proof-of-Concept prototype (see section 5.2). The integration of the SLA specification management, although defined in the Appendix, will be part of D2.5

(Advanced data marketplace enablers), which will also include improvements to the license specification management (see Conclusion).

4.2.1.5 Revenue management

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.
The main functionalities provided by this module are:

- 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.

Revenue management APIs are logically grouped into six sub-modules: usage specification management, usage management, revenue sharing model management, transaction management, billing charges management and billing account management. More specifically:

- Usage specification management. Usage specifications are a detailed description of a usage event which can then be used in an usage pricing model. Usage specifications define all the attributes known for a particular type of usage. It exposes the following interfaces:
    - Usage specification collection: interfaces for retrieving a list of the usage specifications and for creating a new usage specification.
    - Usage specification entry: interfaces for retrieving and updating a specific usage specification entry.

- Usage management. Usage documents contain the actual usage made of a purchased data offering, including the information defined in its usage specification. It exposes the following interfaces:
    - Usage collection: interfaces for retrieving a list of the usages and for creating a new usage.
    - Usage entry: interfaces for retrieving and updating a specific usage entry.

- 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. It exposes the following interfaces:
    - Revenue sharing model collection: interfaces for retrieving a list of the revenue sharing models and for creating a new revenue sharing model.

- 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. It exposes the following interfaces:
    - Transaction collection: interface for retrieving a list of the transactions.
    - Settlement collection: interface for launching a new settlement process.

- 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. It exposes the following interfaces:
    - Billing charge collection: interface for retrieving a list of billing charges and for creating a new billing charge.
    - Billing charge entry: interface for retrieving and updating a specific billing charge entry.

- Billing account management. A billing account is a description of a customer bill structure. It exposes the following interfaces:
    - Billing account collection: interface for retrieving a list of billing accounts and for creating a new billing account.
    - Billing account entry: interface for retrieving and updating a specific billing account entry.

The specification of this API is left unchanged with respect to the original FIWARE/TMForum specification.

4.2.1.6 Customer management

This module allows to identify and gather 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.). Note that this API is not responsible for managing security permissions of purchased data sources, which is performed by the order management API (see Section 4.2.1.3 and Section 4.4.3). The main functionalities provided by this module are:

- 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.

Customer management APIs are logically grouped into three sub-modules: customer management, customer account, and party management. More specifically:

- Customer management API. It is used for saving customer private information that cannot be included within the party resources. This API is used jointly with customer account and billing account to maintain different contact mediums (e.g., email, phone, and address) attached to different billing accounts. It exposes the following interfaces:
  - Customer collection: interfaces for retrieving a list of customers and for creating a new customer.
  - Customer entry: interfaces for retrieving and updating a specific customer entry.
- 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. It exposes the following interfaces:
  - Customer account collection: interfaces for retrieving a list of customer accounts and for creating a new customer account.
  - Customer account entry: interfaces for retrieving and updating a specific customer account entry.
- Party management API. It allows to create, retrieve and update the parties. It exposes the following interfaces:
  - Individual collection: interfaces for retrieving a list of individuals and for creating a new individual.
  - Individual entry: interfaces for retrieving and updating a specific individual entry.

The specification of this API is left unchanged with respect to the original FIWARE/TM Forum specification.

4.2.1.7 Feedback and reputation

This module will be developed and discussed in D2.5 (Advanced data marketplace enablers). It will provide user feedback management for the different data offerings published on the marketplace. It will also provide rating and reputation mechanisms to support data consumers in selecting the data offerings and to promote a honest behavior among users. The main functionalities provided by this module are:

- 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.

4.2.1.8 Transparency and accountability service

This module will be developed and discussed in D2.5 (Advanced data marketplace enablers). The aim is to have a tool for auditing orders (including pricing model, license terms, SLAs) and tracking the parameters defined by SLAs.
The main functionalities provided by this module will be:

- Auditing orders
- Tracking SLAs

4.2.1.9 Data marketplace portal

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 [22], which we adapted to reflect the different and additional features provided by our set of APIs.
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 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.

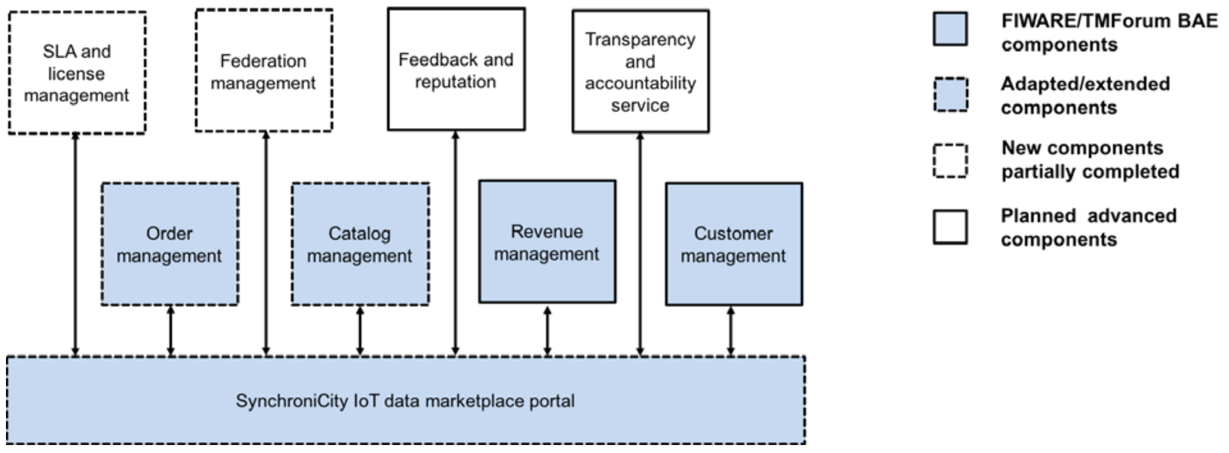

SLA and license management
Federation management
Feedback and reputation
Transparency and accountability service
Order management
Catalog management
Revenue management
Customer management
SynchroniCity IoT data marketplace portal
FIWARE/TMForum BAE components
Adapted/extended components
New components partially completed
Planned advanced components

Figure 8: Current implementation status of the marketplace APIs and marketplace portal.

Figure 8 shows the current implementation status of th*e 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 managemen*t 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 behavior is specified in the next Section. 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 (Advanced data market place enablers).

4.3 Marketplace interaction and components behavior

An overview of the data marketplace interactions among its internal components and other external ones is shown in Figure 9.

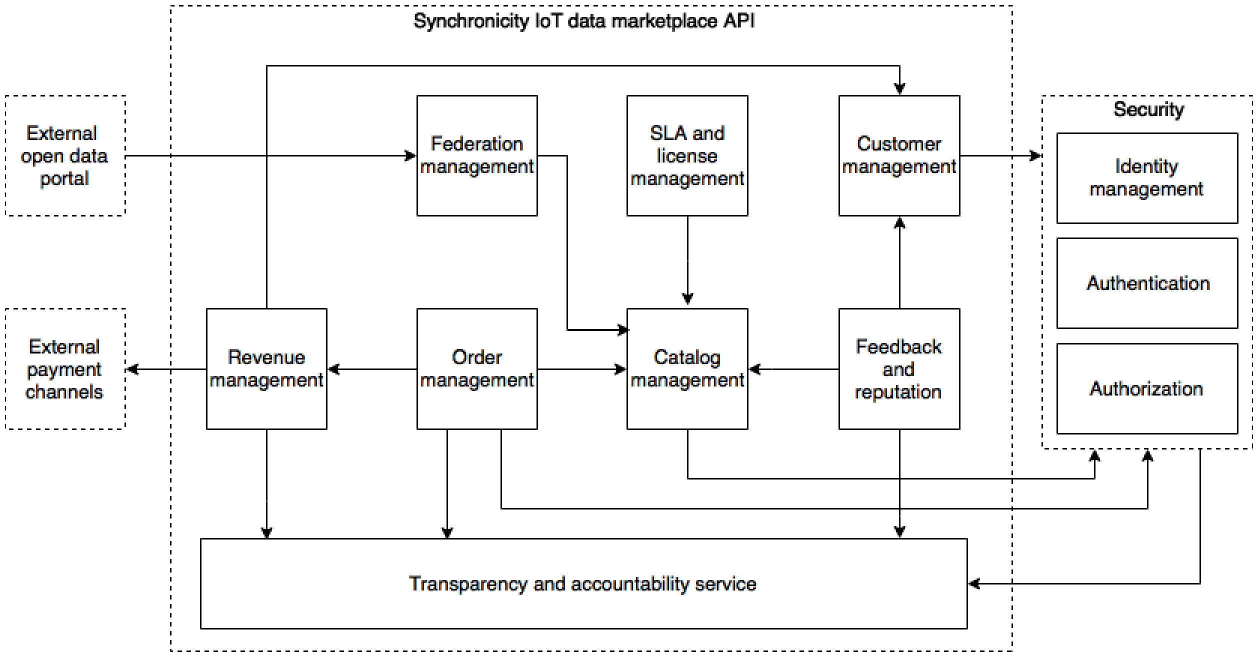


Figure 9: IoT data marketplace component interactions.

The *catalog management* interacts with the *SLA and license management* 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 *feedback and reputation* one to retrieve feedback, rating and reputation score for every digital asset. The *catalog management* also interacts with the *federation management* to retrieve metadata information from external open data catalogs 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 *security* components to check data provider's permission for that specific data source.
The *order management* interacts with the revenue management component to enable monetization mechanisms, with the *transparency and accountability service* to track data usage information, and with the *security* components to update permissions of data consumers when

purchasing data offerings.
The *revenue management* interacts with the *order management* as well as with external payment channels to enable monetization mechanisms by exchanging transactions outcome and charging information. It also interacts with the *customer management* to retrieve users' billing information as well as with the *transparency and accountability service* to record transactions information. The *customer management* also interacts with the *identity management* component to retrieve users information.
The *feedback and reputation* interacts with the *catalog management* and the *customer management* to exchange information regarding feedback, rate and reputation of data offerings and customers. It also interacts with the *transparency and accountability service* to track information on feedback, rate and reputation.

A description of typical end user interactions with the platform is provided next along with sequence diagrams.

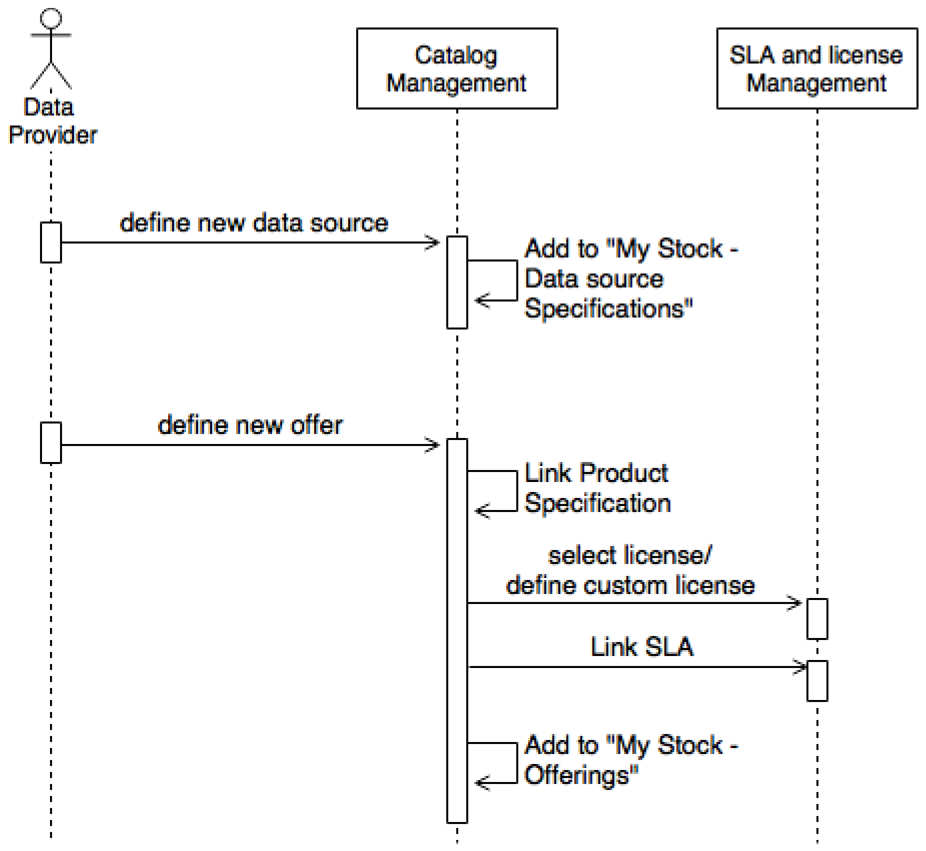

Data Provider
Catalog Management
SLA and license Management
define new data source
Add to "My Stock - Data source Specifications"
define new offer
Link Product Specification
select license/ define custom license
Link SLA
Add to "My Stock - Offerings"
Figure 10: New data offering specification sequence diagram.

Figure 10 shows the process of registering a new data source in the data marketplace by a data provider. The data provider interacts with the *catalog management* component by defining the specifications of the new data source (e.g., title, version, data model, endpoint). Once the data source has been defined, the *catalog management* creates a link to it and adds the reference to the data provider's stock. To create an offering related to the previously specified data source, the data provider interacts again with the *catalog management* by defining the new offering (e.g., title, data source, price, license, SLA). More specifically, the *catalog management* interacts with the *SLA and license management* component by linking the offering to the license and SLA selected/defined by the data consumer. Finally, it creates a new offering which is added to the data provider's stock.

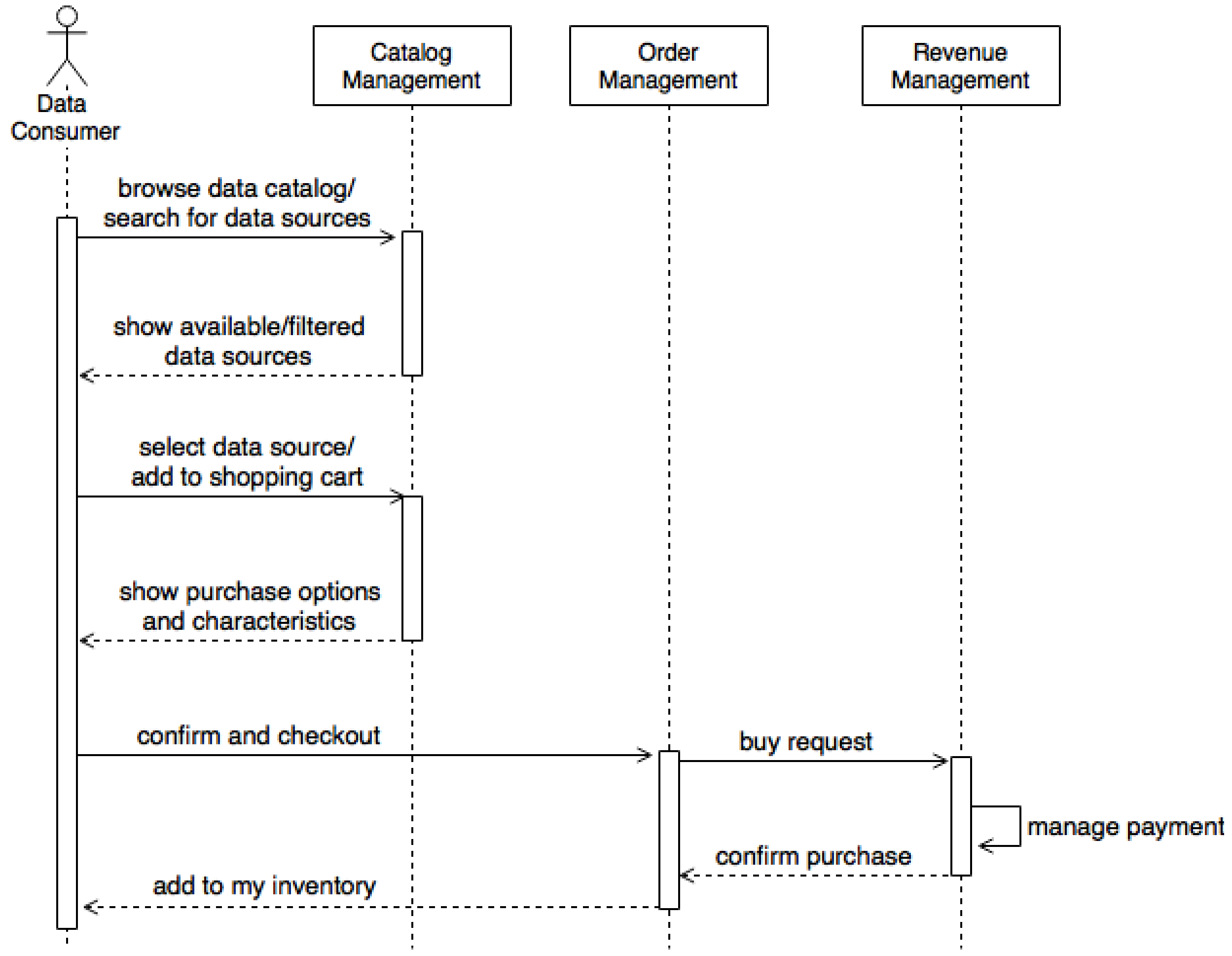

Data Consumer
Catalog Management
Order Management
Revenue Management
browse data catalog/ search for data sources
show available/filtered data sources
select data source/ add to shopping cart
show purchase options and characteristics
confirm and checkout
buy request
manage payment
confirm purchase
add to my inventory

Figure 11: Data offering purchase sequence diagram.

Figure 11 shows the interaction flow regarding the creation of a new order and the finalization of the related purchase. First, a data consumer searches for a particular data offering of interest. The *catalog management* returns to the data consumer a list of matching data offerings from which the data consumer select one for purchase. Thus, the *catalog management* shows to the data consumer available purchase options. After the data consumer confirms the the selected offering, the *order management* creates a new order. The *order management* interacts with the *revenue management* to manage the payment for the consumer's request. The *revenue management* finalizes the purchase, whose response is sent back to the data consumer, and adds the purchased data source to the data consumer's inventory.
The following use case diagrams describe the interaction between two different marketplaces in the federation process and in a generic data offering purchase task. All the interactions related to security are not shown in the diagrams:

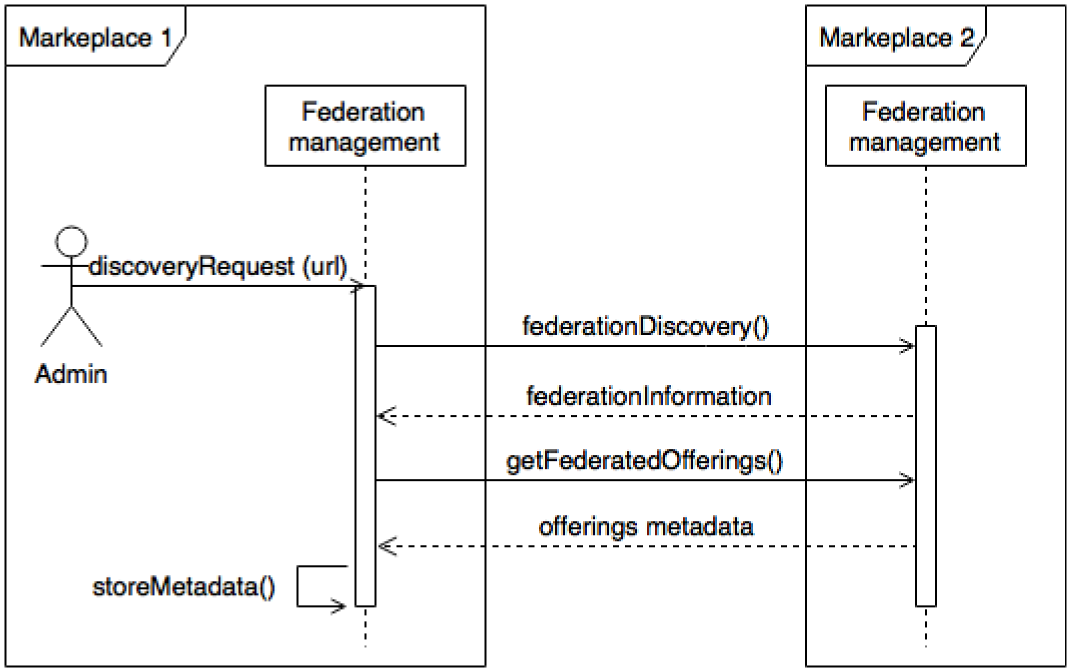


Figure 12: Federation process sequence diagram.

Figure 12 shows how the marketplace administrator discovers other marketplaces (e.g., instances of the marketplace running in different cities) to be federated: the *federation management* component of *Marketplace 1* connects to *Marketplace 2* to check whether it can be federated. Upon receiving federation information, *Marketplace 1* is able to retrieve and store the metadata related to the data offerings that *Marketplace 2* shares within the federation.

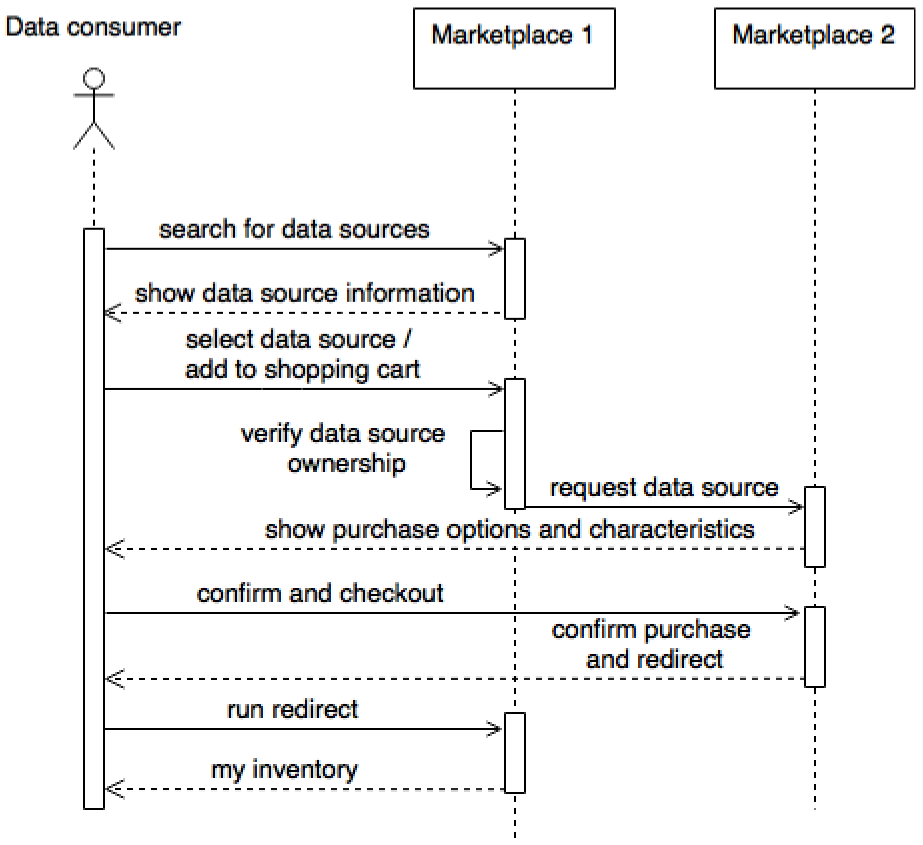

Data consumer
Marketplace 1
Marketplace 2
search for data sources
show data source information
select data source / add to shopping cart
verify data source ownership
request data source
show purchase options and characteristics
confirm and checkout
confirm purchase and redirect
run redirect
my inventory

Figure 13: Federated data offering purchase sequence diagram.

Figure 13 depicts the purchase process of one data offering related to a federated marketplace. A data consumer searches for a data source on *Marketplace 1* and wants to purchase it: the related offering belongs to *Marketplace 2*, so *Marketplace 1* redirects the data consumer to *Marketplace 2*. The data consumer can then complete the data offering purchase and when finished, she will return on the original *Marketplace 1*.

4.4 Security

SynchroniCity security mechanisms are based on modular components and standard protocols. As described in the Deliverable 2.1 Section 3.4.4, the main building blocks provide a specific set of security features such as *identity management*, *access control*, *authentication*, and *authorization*. More specifically, the *identity management* component aims at providing identity as a service (IDaaS), therefore it must comply with requirements such as multi-tenancy, scalability, and standard-compliant protocols and APIs. Together with the *authentication* component, it provides secure and private authentication from users to IoT devices, applications and services, user profile management, Single Sign-On (SSO) to service domains, and identity federation towards applications. It supports the OAuth2 [23] standard protocol to authenticate and authorize requests, as well as for delegation and access token generation.

The *authorization* component, hereafter referred as the Policy Decision Point (PDP), evaluates authorization decisions based on XACML [24] policies and attributes related to a given access request (e.g., requester identity, requested resource, requested action), following the policy evaluation logic defined in the XACML standard. This feature is provided to external clients through a REST API (e.g., PDP API). This component also includes an authorization Policy Administration Point (PAP) acting as policy management, for creation, retrieval, update and removal of XACML policies. This feature allows policy administrators (e.g., resource owners, system and network administrator) to configure the XACML policies to be evaluated when calling the PDP API.

Moreover, access control and filtering is performed by a *Policy Enforcement Point (PEP) proxy* which aims at protecting 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, and on the PDP to authorize it (deny of permit).

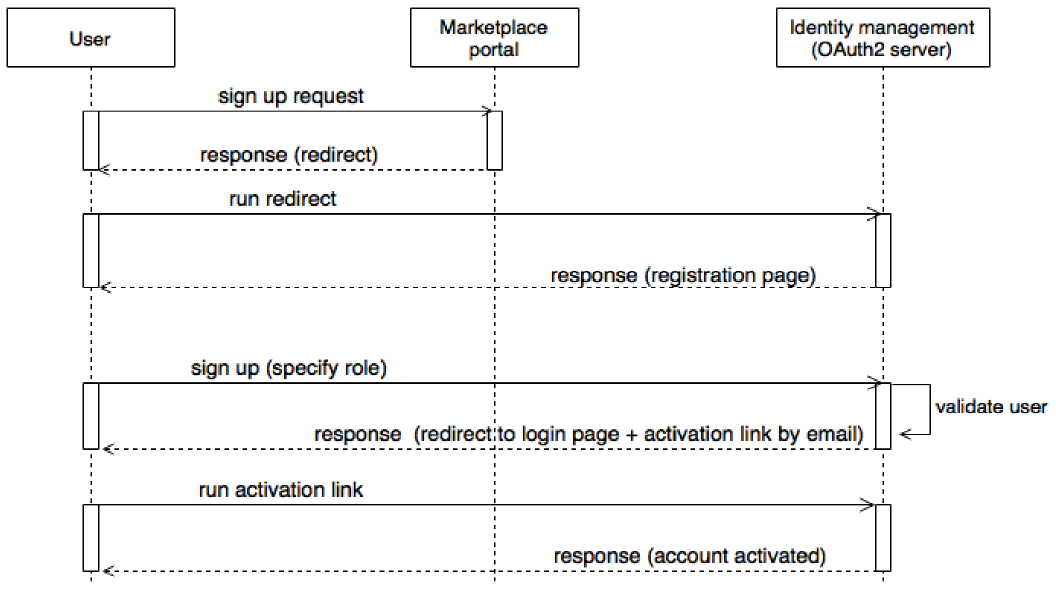
In order to secure communication and in particular the users access tokens, communication must happen using encrypted and authenticated channels. If this is not the case, an adversary could intercept the access tokens and act as the user. A third factor is integrity of the messages which is equally important as an adversary could otherwise use the access token to authenticate his own messages. We will therefore employ TLS which provides both confidentiality, authentication and integrity of the messages sent. With the help of a global Certificate Authority (CA) which we assume the system trusts, the system can be setup using standard implementations of TLS. We note that TLS 1.0 and TLS 1.1 should not be used if it is possible to avoid this as those versions have been deemed insecure by today's standards [25].

As the communication among components and APIs must be secured, the data marketplace APIs, PDP API, PAP API and *PEP proxy* must be configured to provide HTTPS URLs. Furthermore, data providers must configure the URL to the protected data source endpoint, so that the PEP Proxy knows where to forward the requests.

4.4.1 Customer Registration and Interaction

The *marketplace portal* uses the **OAuth2** protocol to authenticate end users by relying on the Identity management component acting as the OAuth2 authorization server.

Users' identities are centrally managed by the *identity management* component, thus registration into the marketplace translates to a registration into the *identity management*. The marketplace registration workflow is depicted by Figure 14:


User
Marketplace portal
Identity management (OAuth2 server)
sign up request
response (redirect)
run redirect
response (registration page)
sign up (specify role)
validate user
response (redirect to login page + activation link by email)
run activation link
response (account activated)
Figure 14: Sequence diagram of a user registration into the marketplace.

We assume that the marketplace has been registered into the *identity management* component as trusted application (i.e., it has obtained a valid client-id and client-secret as specified by OAuth2 RFC 6749). The user clicks on the sign up button of the *marketplace portal* homepage. This operation leads the user to be redirected to the *identity management* registration page. Here the user signs up by providing his username, chosen password, email and the role he would like to be registered with to the marketplace (e.g., data provider, data consumer). After the *identity management* receives the user registration request, a validation procedure is performed by the data marketplace provider/admin according to the policy defined by the marketplace governance (i.e., city councils consortium). In case of a positive validation outcome, the *identity management* sends back to the user a response on different channels: 1) a redirect request to the *identity management* login page; 2) an email including an activation link. The user runs the activation link, thus resulting in a successful account activation.

As in the typical OAuth2 delegation flow, the end-user (referred as the resource owner in the OAuth2 specification [23]) delegates the marketplace to perform operations and act on its behalf such as accessing user profile and roles information from the Identity management. More specifically, the sequence diagram showing the procedure to log into the marketplace login is depicted in Figure 15:

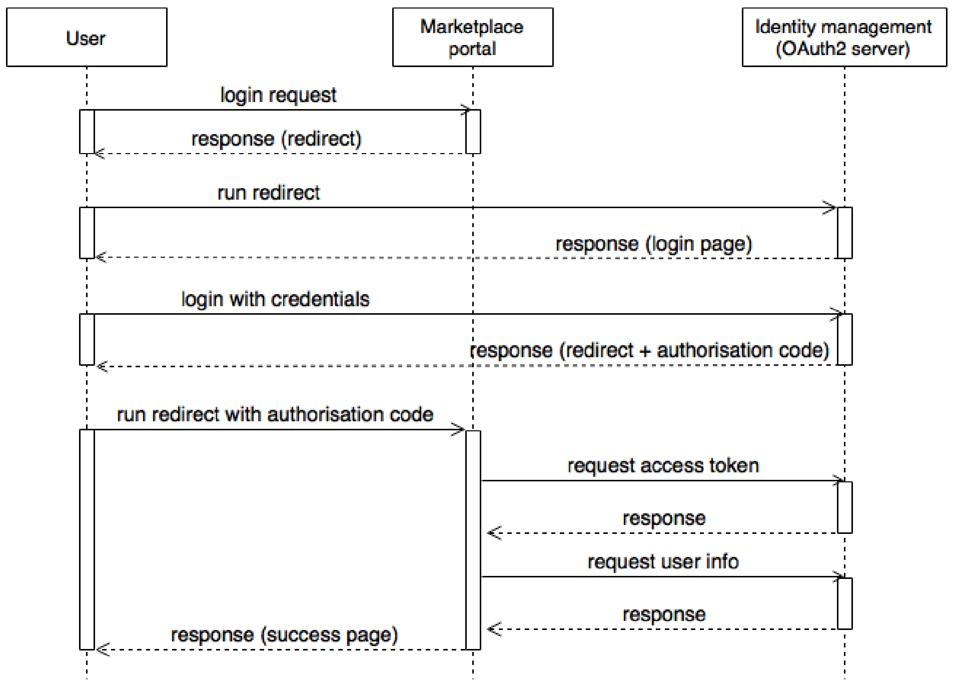


Figure 15: Sequence diagram of a user login.

We assume that the marketplace has been registered to the *identity management* component as trusted application. We also assume that the end user has already performed the access to the *marketplace portal* at least once, thus she has already granted the permission to the marketplace to access its user related information, create a user profile through the *customer management* API, and eventually act on its behalf.
The end user performs a login request on the *marketplace portal* which will redirect the end user to the *identity management* portal (note that if the end user's browser has a valid open session with the *identity management* the *marketplace portal* will "transparently" log the end user in). The end user inserts its credentials on the *identity management* portal thus performing its login. The *identity management* sends the authentication response to the end user which embeds a call back redirection to the *marketplace portal* along with the authorization code.
The end user performs the redirection thus sending the authorization code to the marketplace. The marketplace sends the authorization code just received from the end user to the *identity management* in order to receive an access token that it can use to act on the end user behalf. Note that the scope of the token needs to be present and it has to match the scope authorized by the end user. The marketplace embeds in the authorization header the access token obtained and queries the *identity management* to retrieve the information about the user (e.g., id, name, domain, roles). Finally, a success page is presented to the end user.
The *marketplace APIs* require an OAuth2 token that is used to check the caller's right (e.g., caller identity, role, etc.) to perform the specific API call. The sequence diagram showing the procedure to call a *marketplace API*, is depicted in Figure 16.

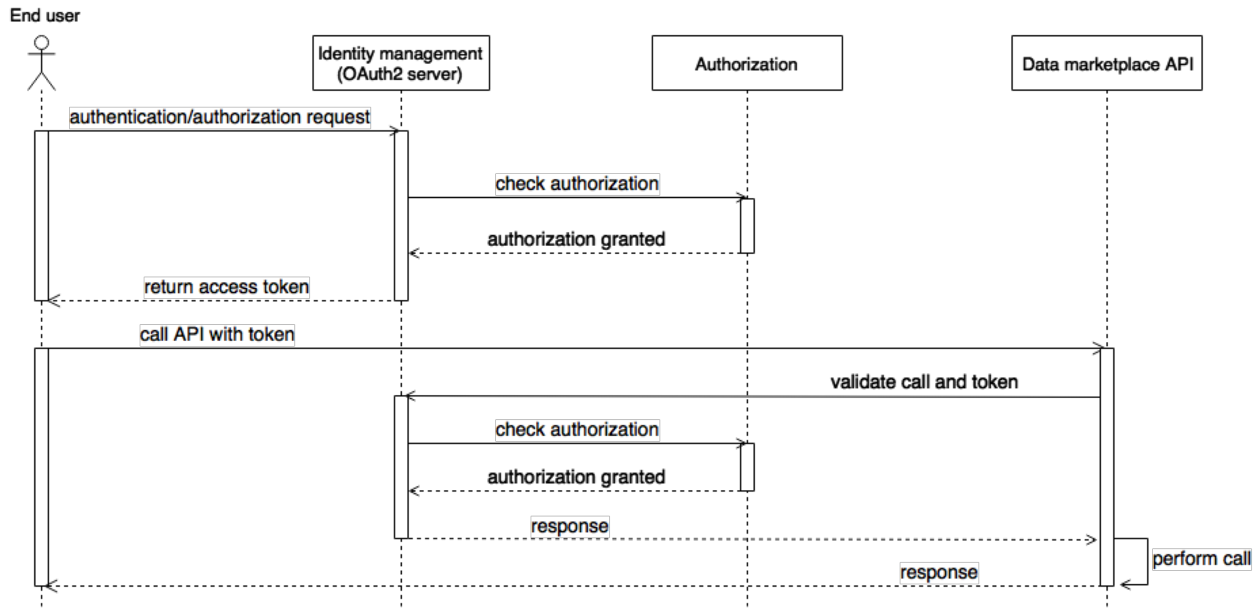


Figure 16: Sequence diagram of a call to a marketplace API.

First, the end user needs to obtain an OAuth2 token from the *identity management*. She sends an authentication/authorization request to the *identity management* along with his credentials or a refresh token. The *identity management* checks through the *authorization* component whether the end user is allowed to be granted for a token and, if granted, returns to the end user a valid token. The end user can now call a *marketplace API* sending along with the API call the access token just obtained to the marketplace. Before performing the requested call, the marketplace checks that the end user access token is valid and based on its role if she is allowed to perform the specific call. It sends the call and token validation request to the Identity management, which in turn queries the *authorization* component to obtain an authorization response (i.e., granted/denied). The validation response is then sent back to the *marketplace API*. Based on the validation response, the marketplace performs the requested call and send back to the end user the response call or in case of denied authorization it sends back an error code (i.e., HTTP code 403 forbidden).

4.4.2 Data offering

We assume that the data provider has registered his data source endpoint (e.g., orion context broker) in the *identity management* component, including an authorization policy for this resource through the PAP AP. The registration of a new data offering follows the workflow described by the sequence diagram in Figure 17.

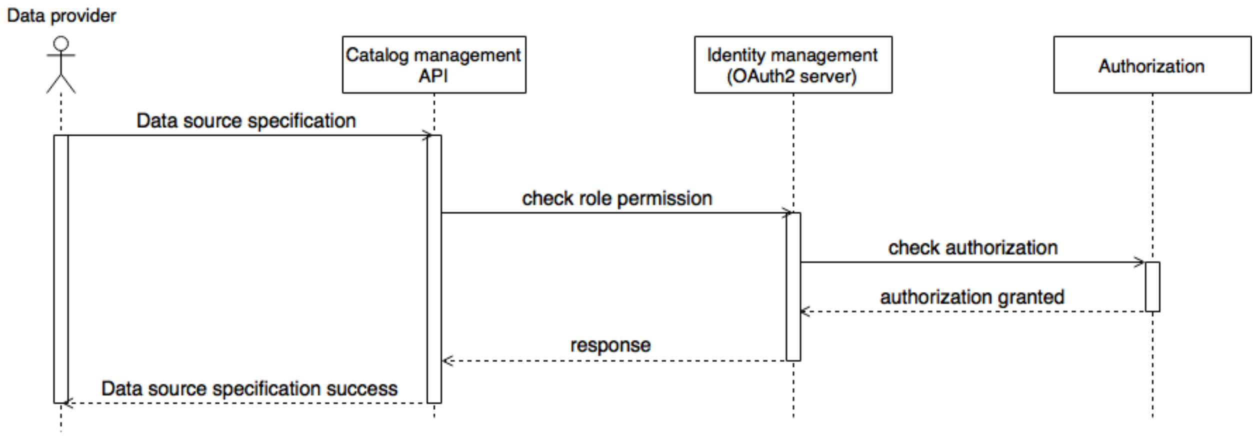


Figure 17: Data offering specification sequence diagram.

We assume that the data provider has been authenticated (through the *identity management* by using OAuth2 flow, and got an access token from it as a result) to access to the *marketplace API*, therefore she can perform operations such as registering a new data offering. The data provider then specifies a new data offering by interacting with the *catalog management* API. The data provider request will need to be validated to check whether the data provider has the right to create offering for that particular data source. To that aim, the *catalog management* API sends information about the data source endpoint (previously registered to the Identity management component by the data provider), and the data provider related information such as its role. The *identity management* checks the received request against the *authorization* component (PDP) and forwards the authorization response (e.g., granted/denied) back to the *catalog management* API. Finally, the response from the *catalog management* returns to the data provider as successful or unauthorized based on the decision of the security components.

4.4.3 Data offering purchase and consumption

The *PEP proxy* must intercept each incoming API call request to all protected data sources. The protected data source must be setup rigorously such that only the *PEP proxy* is allowed to call the API of the data source. This could include firewall rules restricting access from the IP address of the *PEP proxy* or, even better, configuration of the PEP proxy certificate as the only allowed incoming traffic. The *PEP proxy* performs the authentication, authorization and accounting workflow according to the sequence diagram shown in Figure 18.

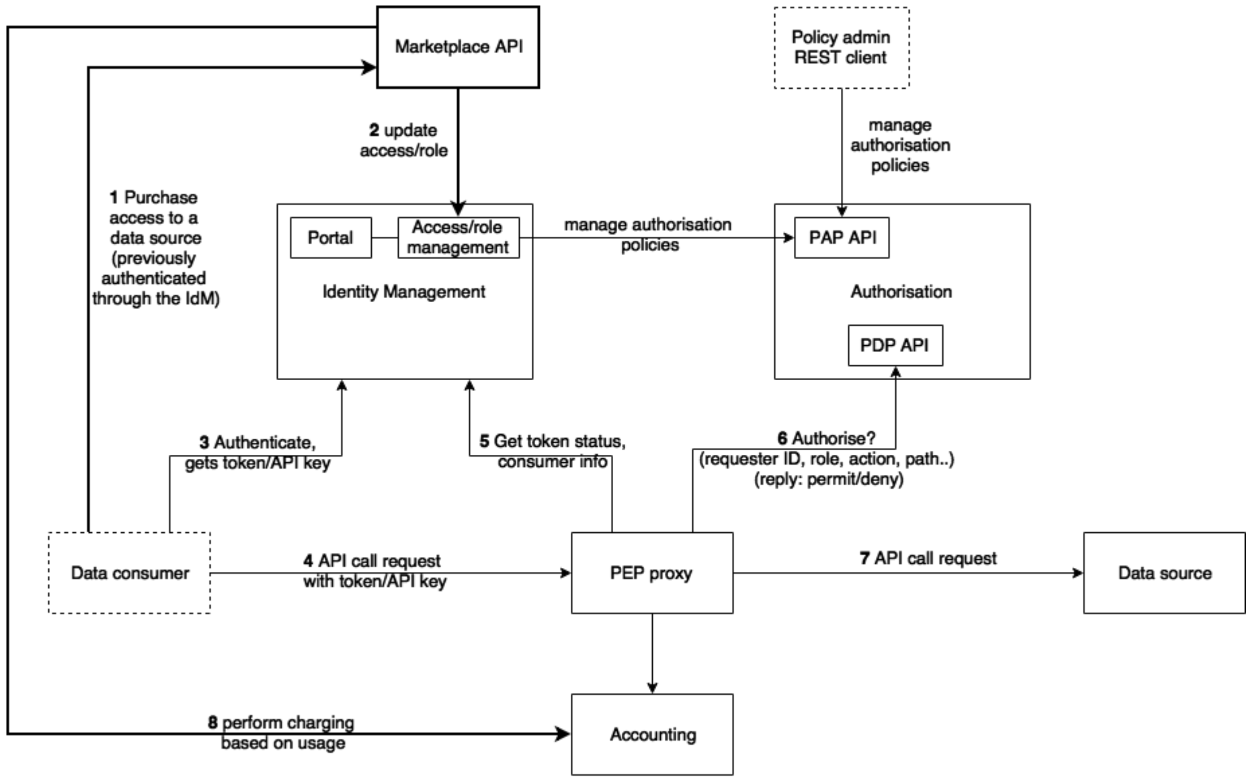

Marketplace API
Policy admin REST client
2 update access/role
manage authorisation policies
1 Purchase access to a data source (previously authenticated through the IdM)
Portal
Access/role management
manage authorisation policies
PAP API
Identity Management
Authorisation
PDP API
3 Authenticate, gets token/API key
5 Get token status, consumer info
6 Authorise? (requester ID, role, action, path..) (reply: permit/deny)
Data consumer
4 API call request with token/API key
PEP proxy
7 API call request
Data source
8 perform charging based on usage
Accounting

Figure 18: Security components interaction during purchase and consumption. Bold connectors highlight the interactions between the marketplace APIs and the security components.

1. We assume that the data consumer has been authenticated (through the *identity management* by using OAuth2 flow, and got an access token from it as a result) to access the *marketplace API*, therefore she can perform operations such as purchasing data offerings. The data consumer discovers a data offering published on the data marketplace and purchase the access to that.
2. The data *marketplace API* validates the consumer's purchase (through the *revenue management* component) and if the transaction is confirmed, the it updates the role and access permissions of the data consumer for the purchased resource through the *access/role management* module.
3. The consumer can now authenticate itself to the *identity management* using OAuth2 flow, and get an access token from it as a result. Note that if the token obtained by the data consumer at the step 1 is still valid (i.e., tokens expire and need to be refreshed) this step is not required. Moreover, if the purchased data source is protected by a *PEP proxy* supporting API keys, the data consumer will need to provide this API key (obtained as result of the purchase phase) instead of the OAuth2 token.
4. The data consumer sends the API call request with the token/API key included as the *PEP Proxy* expects.
5. When the *PEP Proxy* receives the consumer request, it extracts the access token and sends it to the *identity management* for validation. If it is valid, the *identity management* returns the validation result and other token-related information, such as information about the authenticated data consumer (e.g. user role). If the token turns out not valid, the request is

not authenticated therefore denied. A similar interaction happens if the API call request embeds an API key instead of a OAuth2 token.

6. The *PEP Proxy* sends an authorization decision request to the PDP (e.g., based on the XACML protocol) that contains this *identity management* related information with other information about the access request: requested resource ID, action ID (HTTP method), etc. The PDP computes the authorization decision – Permit or Deny – and returns it to the *PEP proxy*.
7. If PDP decision is Permit, the *PEP proxy* forwards the API call request to the protected data source, and forwards the response back to the data consumer. If the decision was Deny, the *PEP proxy* denies the API call request and replies with a HTTP response 403 (Forbidden).
8. If the protected data source has been offered in the marketplace with a pay-per-use pricing model, the *marketplace API* regularly queries the *accounting* component to perform charging of the data consumer based on its usage consumption.

More specifically, the interaction of the data marketplace components with the *security* components with respect to a data offering purchase follows the workflow shown by the sequence diagram depicted in Figure 19.

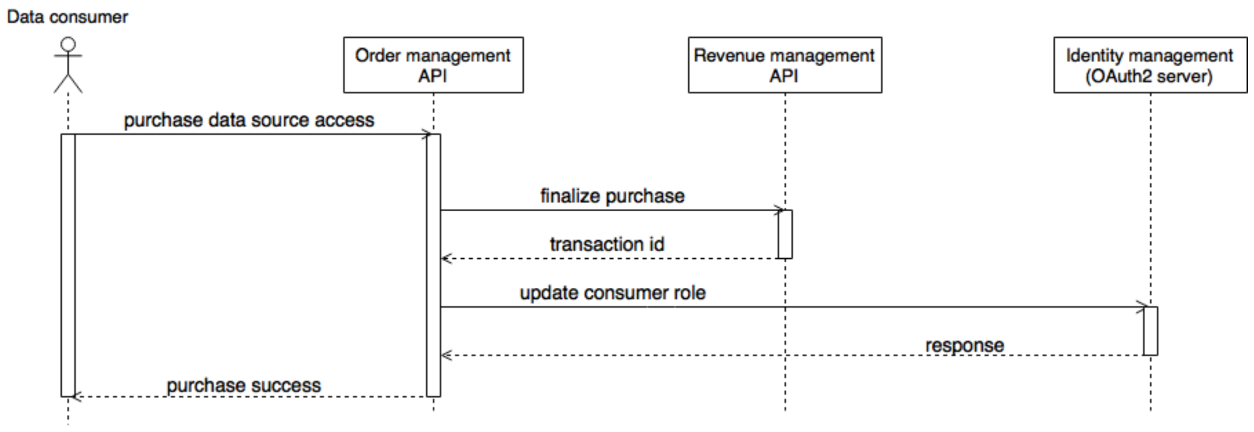


Figure 19: Sequence diagram of a data offering purchase.

We assume that the data consumer has been authenticated (through the Identity Management by using OAuth2 flow, and got an access token from it as a result) to access to the *marketplace APIs*, therefore she can perform operations such as purchasing data offering. The data consumer then purchases the access to a data source offering through the *order management API*. The latter validates the consumer purchase through the *revenue management* and if the transaction is successful, then it updates the role and access permission of the data consumer for the purchased resource through the *identity management*, which exposes an API for managing users' roles.

4.4.4 Privacy concerns

Data is the entire foundation of business in the SynchroniCity environment. If no data was present, the system would not be used. However, whenever data is flowing in any system of either business value or personal data, it should be considered how to protect it. We split data in two main

categories: personal data and business data. The reason being that personal data falls under the GDPR and has to be protected differently than normal data. Personal data includes for instance transaction data, personal email. Aggregated data which no longer can be traced back to a single person, however, can be considered anonymous, meaning the data does not fall under the GDPR. This means that any agreements about data confidentiality or security of the data should be subject to individual agreements between data providers and the owner of the running SynchroniCity system. These two entities may legally be the same in which case no agreement is needed.
If the data provided cannot be considered anonymous, it falls under the personal data category and should be treated as such. All parties within the system should be compliant with current regulations, both european and national. We refer to deliverable D1.4 for further information on this topic.
To be in line with data protection by design, we encrypt the databases to protect data at rest. As previously mentioned, TLS will be enabled for all communication to protect live data. This holds for both business and personal data
We assume that the system contains backup of the databases. When this is the case, GDPR is not yet clear on how to handle deletion requests. A protocol should be in place which deals with the problem of deletion requests and backup. We here present our suggestion for a best-effort protocol. Since backup might be located on an offline location and can therefore not be accessed immediately on request, a guarantee on immediate deletion from all sources is not feasible. Thus we propose to only store backups as long as is needed (e.g., have a running 3 month window where new backups are made before the older are deleted). A separate specific log should be in place to note the deletion requests. Be aware that the log should NOT contain any personal data. Then, upon restoring backup to the system, the log should be traversed and deletion requests must be run before the system goes online.

4.4.4.1 Personal data

Personal data must be protected in regulation to the GDPR, and furthermore we work from the principle of privacy by design and data protection by design. This means that personal data should be kept only as long as it is needed. It also means that owners of personal data should be able to retrieve, ratify and delete all their data that the system has gathered on them. Personal data owners can operate on their data as shown by the sequence diagram depicted in Figure 20.

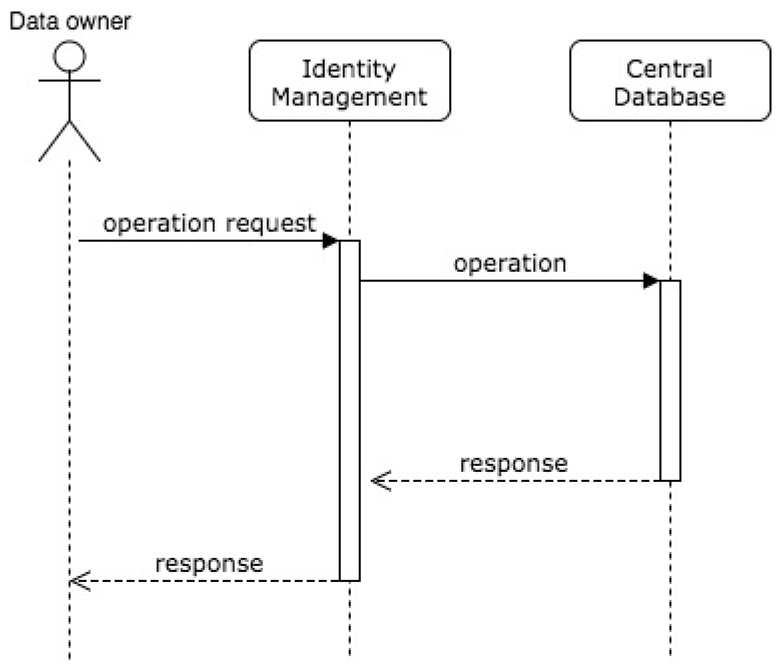


Figure 20: Sequence diagram showing how a data owner operates on her personal data.

We assume that the data owner is logged into the *identity management* and has obtained an access token allowing her to operate on her own data. The data owner can then send a request to the *identity management*, telling it to do one of the following operations on his data:

- *Fetch*: A GET request returning all entries about the user that the system knows about her (right of access).
- *Delete*: A DELETE request which enforces a deletion of all data the system knows about the user (right to be forgotten).
- *Modify*: A POST request which alters the attribute(s) contained in the request to the values the request contains (right to rectification).

The *identity management* forwards the request to a *central database* where information is stored and gets a response back which is processed and a final response is sent to the user.
The response should always be success, but if something fails, this should be logged by the system and the user needs to be informed about what measures are taken to ensure that his request will be processed.
Further non-technical measures should be taken to be fully compliant with the GDPR, and we refer readers to the deliverable D1.4 for a more thorough description.

5 Initial validation

In this section, we present an initial PoC prototype of the marketplace, including the basic enablers described in this deliverable, and validate it against a typical smart city use case by integrating it with the Santander RZ IoT facility. The use case involves environmental sensor systems deployed in Santander and a weather forecast service. We first outline the Santander RZ IoT facility, as well as the datasets, data streams, and data models exploited in the use case. We then validate the prototype by highlighting the benefits the marketplace can provide to the key stakeholders.

5.1 Target reference zone infrastructure

Santander RZ IoT facility is based on a real IoT deployment in an urban setting. The core of the facility is located in the city of Santander and surroundings, encompassing IoT deployments (Figure 21), in different key areas of the city infrastructure, ranging from public transport, key logistics facilities, public places and buildings, work places and residential areas, thus creating the basis for development of a Smart City. This deployment exhibits the diversity, dynamics and scale that are essential for the evaluation of advanced protocol solutions. The current Santander IoT infrastructure compiles the fruitful result of the assorted smart city and IoT projects carried out by the city since 2010 [27], in close collaboration with the University of Cantabria, involving also a wide set of SMEs and international companies and attracting European, national and local funding, both, public and private.

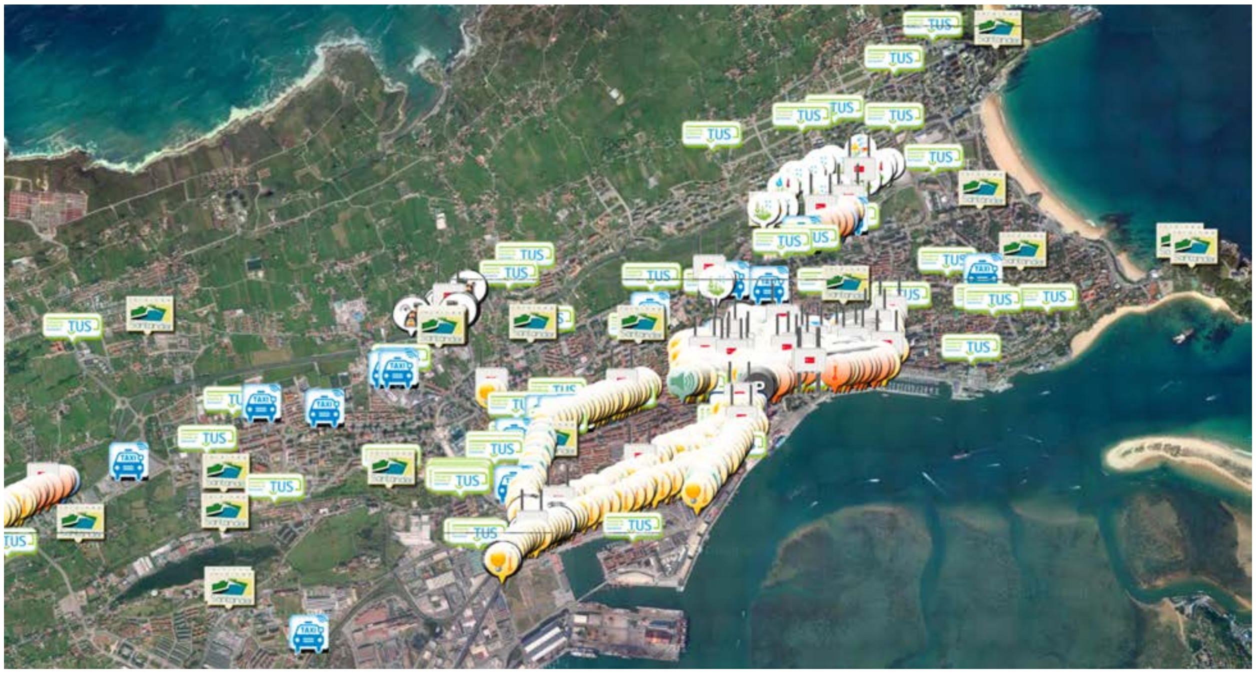


Figure 21: Devices deployment within the SmartSantander IoT facility.

5.1.1 Santander RZ IoT Technical Framework

Santander RZ IoT framework is evolving according to advancements in IoT technologies and architectures. Its first IoT network and sensors deployments were based on IEEE 802.15.4 [26], within the European initiative SmartSantander [27], setting the basis for a growing heterogeneous FIWARE aligned architecture, that currently supports many different IoT services, pilots and projects. Figure 22 provides an overall approach to the technical RZ architecture.

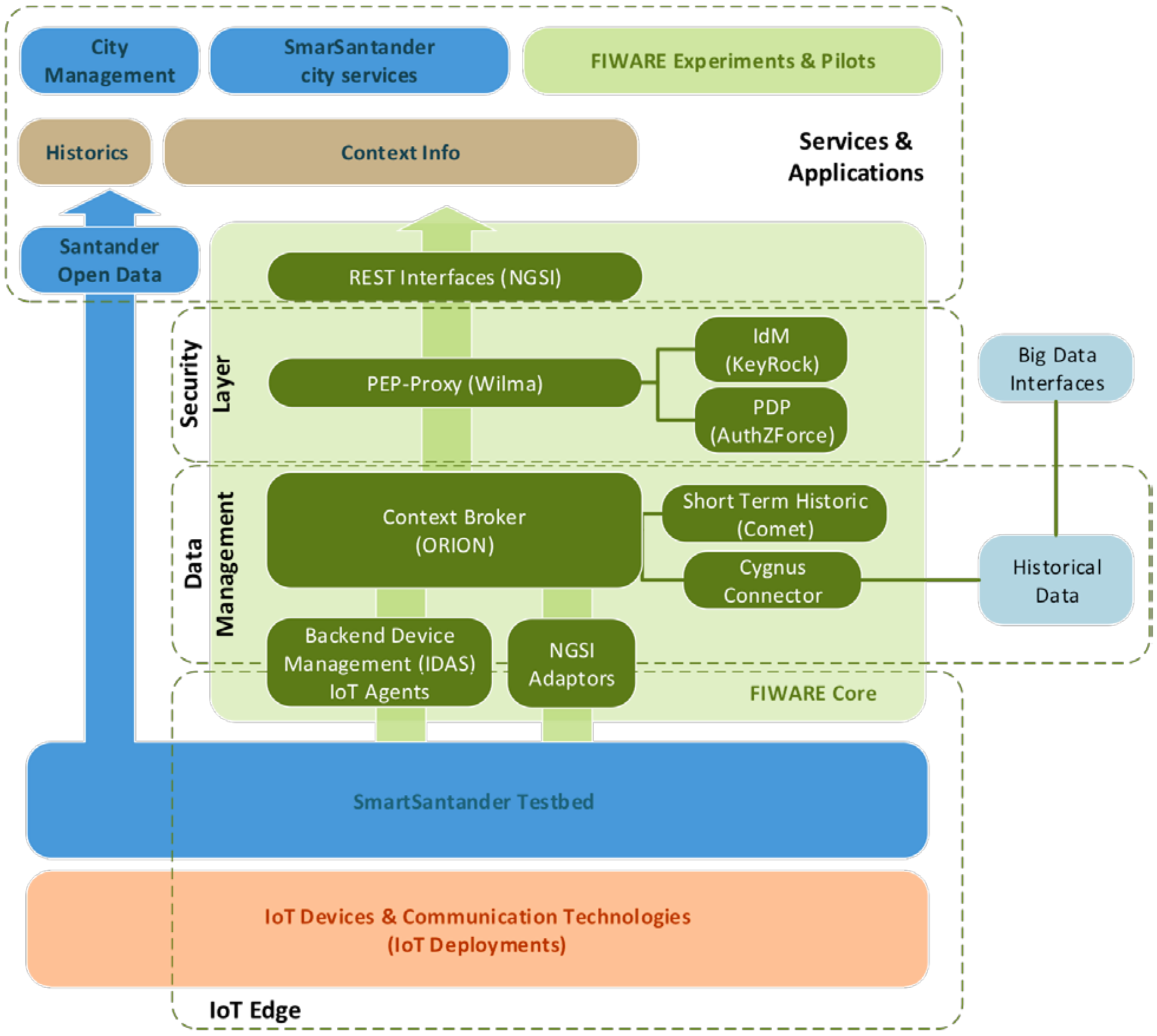


Figure 22: Santander RZ framework.

From this overall point of view, the current IoT framework is structured across four main layers, each of them built with specific components and providing specific functionalities:
The **IoT Edge** comprises all the heterogeneous sensors deployed in Santander, plus the IoT communication protocols and networks. The IoT Edge comes directly from the SmartSantander IoT project and provides physical IoT infrastructure to the city with a wide WSN based in IEEE 802.15.4 that covers the main part of the city. WiFi, Ethernet and 3G gateways, 2G/3G nodes, and complements of these infrastructures provide access to the Santander's backbone networks and support mobile sensing nodes that considerably expands its sensing capabilities. A new LoRaWAN network is currently being deployed and managed by the University of Cantabria in

order to support new IoT devices deployments. The IoT Edge directly feeds data to the Santander's Open Data Portal and the SmartSantander City Services, as well as to the Santander's FIWARE based backbone platform to be used in the SynchroniCity reference framework implementation.

Within Santander's FIWARE framework, the **Data Management** core is built by an own-managed instance of the Orion Context Broker [28]. This component will distribute (Publish/Subscribe messaging system) and manage the context information uploaded by the SmartSantander IoT Edge, using an NGSIv2 [29] interface, to both gather context (southbound) and provide context (northbound).

In the southbound layer, the FIWARE Backend Device Management IDAS [30] provides connectors (IoT Agents) for different IoT protocols (oneM2M, MQTT, etc.). This sublayer is complemented with specific NGSI adaptors, designed and developed by the University of Cantabria, to import the SmartSantander IoT Edge data plus other external info sources managed by the municipality and third parties. This way, Santander RZ exposes its datasets through a RESTFul NGSIv2 API to third parties. The Context Broker is complemented by two FIWARE components to manage historical data: the Short Term Historic Comet [31] and the Cygnus Connector [32], which are included in the SynchroniCity reference framework implementation. These two components are currently being tested within Santander IoT architecture, as part of the pilots. They will be linked to the SynchroniCity architecture implementation, according to the project's guidelines.

Santander's RZ FIWARE **security framework** instantiates the classical FIWARE approach: PEP Wilma proxy [33], its Identity Manager KeyRock [34] and the Authorization PDP AuthZForce [35]. These three components together identify the data consumer and protect the data access. This framework is managed by the University of Cantabria and will be used/integrated within the proposed SynchroniCity security schema.

5.1.2 Santander RZ datasets and data models

Santander IoT data deployment is aligning with current FIWARE Data Models [36], based on OMA NGSI meta-model and offered through an NGSIv2 RESTFul API. The available datasets will be expanded as the SynchroniCity project progresses, while the RZ adapts the existing and incoming data sources to the agreed SynchroniCity data models and implements the corresponding NGSI adaptors. Currently, the already available datasets refer to the following applications:

- environmental monitoring
- parks and garden deployment
- smart parking
- traffic measurement
- bus information
- bike hiring
- City Points of Interest (PoI)

Next, we describe the first two data sources as they are used in the use case scenario, as detailed in Section 5.2.2.

5.1.2.1 Environmental monitoring devices

The city of Santander is trying to carry out an effective policy for environmental management through the signing of agreements that aid improvements in air quality and quality of life for its citizens. Key elements in undertaking this task are:

- Monitoring of pollutants
- Noise and temperature measurement

With the aim to develop this environmental monitoring policy, temperature, CO index, noise and luminosity sensors have been installed in street lamps and facades. All these devices send their information, in a multi-hop fashion (if needed), to the corresponding gateway that gathers all the received information making it available to the Santander backbone. Figure 23 shows several examples of the installation of these devices, at different streetlamps and facades of the city:

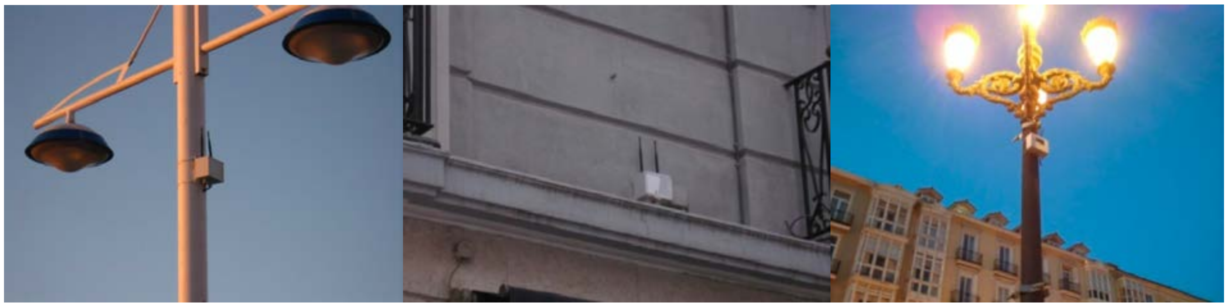


Figure 23: Example of repeaters installation.

More than 1,000 of these fixed nodes have been installed at the Santander city centre. The Environmental Monitoring service was later expanded by mobile nodes installed on municipal public buses, parks and gardens vehicles and taxis. This way the service is able to cover a much wider area on a much more efficient way. Mobile nodes send the collected information to the internet/intranet, and also, interact with the corresponding static nodes placed at streetlamps and facades.

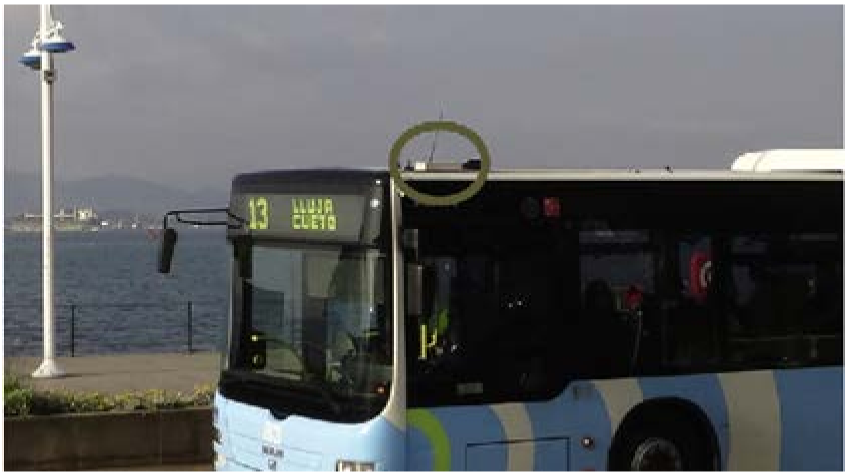


Figure 24: Mobile environmental monitoring sensors.

Environmental data models

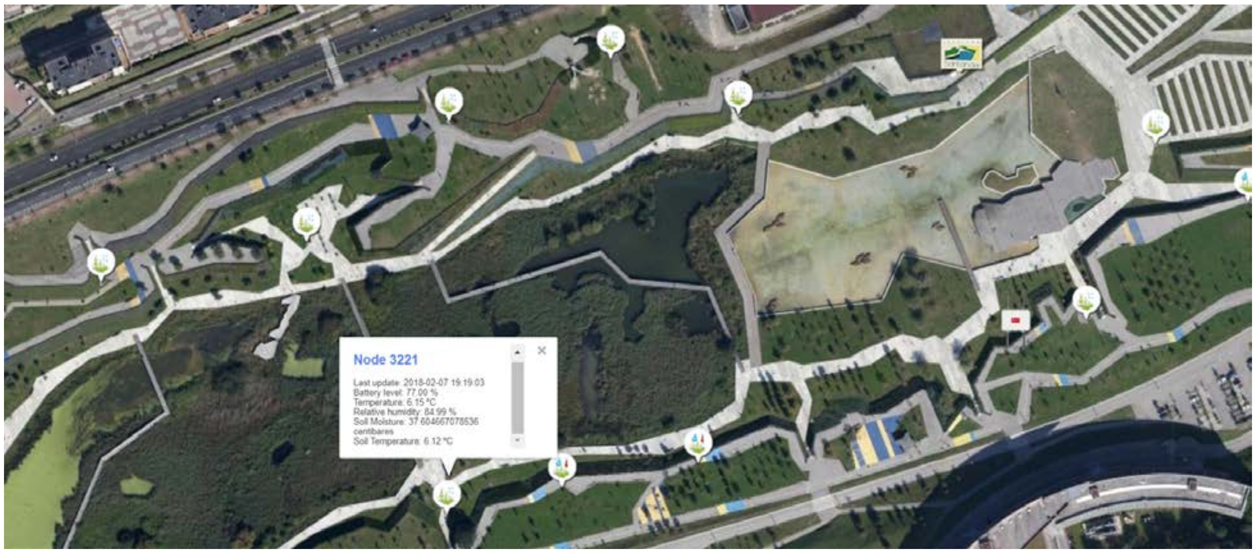
Data captured is available according to FIWARE Environment [76] and Weather [40] data models. Currently, three entities are managed within the Santander SynchroniCity infrastructure:

- **AirQualityObserved** [38]. It represents an observation of air quality conditions at a certain place and time.
- **NoiseLevelObserved** [39]. It represents an observation of those parameters that estimate noise pressure levels at a certain place and time.
- **WeatherObserved** [40]. It represents an observation of weather conditions at a certain place and time.

5.1.2.2 Parks and garden irrigation deployment

To control the irrigation in parks and gardens in the City of Santander and make it more efficient, 50 devices have been deployed in two green zones of the city. They monitor irrigation-related parameters, such as moisture temperature, humidity, etc., evaluate the current situation and opportunely act over the irrigation systems. For this purpose, the following types of sensors are installed:

- Weather station: anemometer, pluviometer;
- Atmospheric pressure, solar radiation, air humidity and temperature sensors;
- Soil temperature and humidity sensors;
- Evaluation of water consumption sensor.



Node 3221

Last update: 2018-02-07 19:19:03
Battery level: 77.00 %
Temperature: 6.15 °C
Relative humidity: 84.99 %
Soil Moisture: 37.604667078536 centibares
Soil Temperature: 6.12 °C

Figure 25: Parks & Gardens nodes deployment.

Parks and Gardens data model

Data captured is available according to FIWARE Parks and Garden [40] data model. Currently, one entry is managed within the Santander SynchroniCity infrastructure:

- **GreenSpaceRecord** [42]. It contains a harmonized description of the conditions recorded on a particular area or point inside a greenspace (flower bed, garden, etc.).

5.2 Validation

To validate our assumptions on the marketplace enablers, we developed a Proof-of-Concept (PoC) prototype, based on the reference implementation of the FIWARE/TMForum Business API Ecosystem (BAE). We integrated the prototype with the Santander IoT infrastructure and tested it in the context of a typical smart city use cased, defined along with Santander RZ, as detailed next.

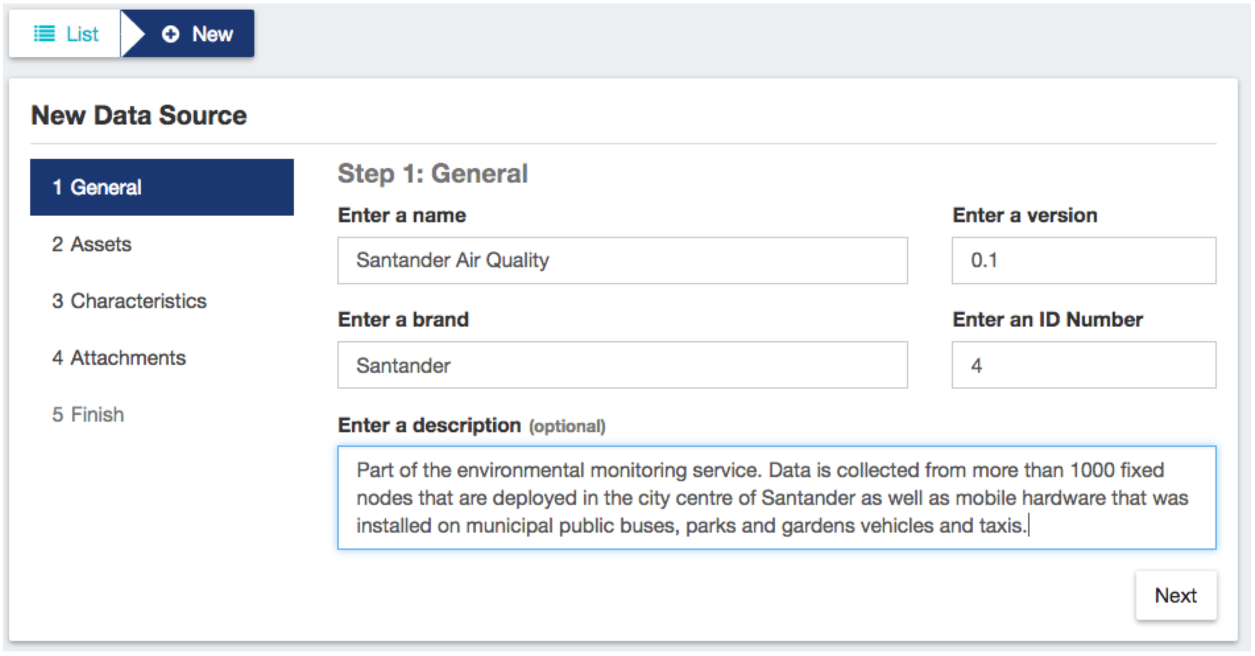
5.2.1 Use Case Description

As part of the validation, we aim to illustrate how the various stakeholders of a specific smart city use can benefit from the data marketplace. The use case we are describing is named *Weather Forecasting* and falls under two of the application themes defined in D3.1, which are *Multimodal Transportation* and *Community Policy Suite.*
Weather forecasting, and in particular, local forecasting in cities and surroundings is the domain of the application created by *company X*. Its current expertise on this area is based on the forecasting skills of its experts' staff, captured in validated models of reasoning and models of knowledge. *Company X* plans to stimulate its experienced forecasters by the introduction of new technologies, which give them the opportunity to analyze meteorological data in new and creative ways as well as improve their decision making. In this sense, *company X* requires diverse environmental data, captured by cities' IoT infrastructures, to be included within its forecasting algorithms and improve the accuracy of the offered services. This will also allow *company X* to provide specific city-centered weather forecasts, updated in real time, based on real-time captured data.
Multiple stakeholders are taking part in the use case, either as enabler, as beneficiary or as both. Santander RZ, as he *city council*, is the main manager of the environmental sensor nodes and the parks and gardens deployment. Within the SynchroniCity IoT framework, the *city council* is responsible for the context management component (Orion CB), the security framework (IdM, PEP, PDP), and the provision of the data marketplace.
Company X, as the weather forecaster, is responsible for providing the weather forecasting algorithms and methodology as well as the weather forecast app that provides the weather forecasts to the end user.
The *citizens* using the weather forecasting app are considered the end users of the *company X* app. *App developers* and *IoT service providers*, exploiting the weather forecast by the app to provide further services to users, can also be considered as stakeholders in this use case. An example for that would be a multimodal transportation planner that adapts the route recommendations according to the results of the weather forecasting app.

5.2.2 Marketplace in action

As described in detail in Section 5.2.2.1 and Section 5.2.2.1), the city of Santander owns two initial sets of IoT deployments that provide relevant information for the envisioned scenarios.

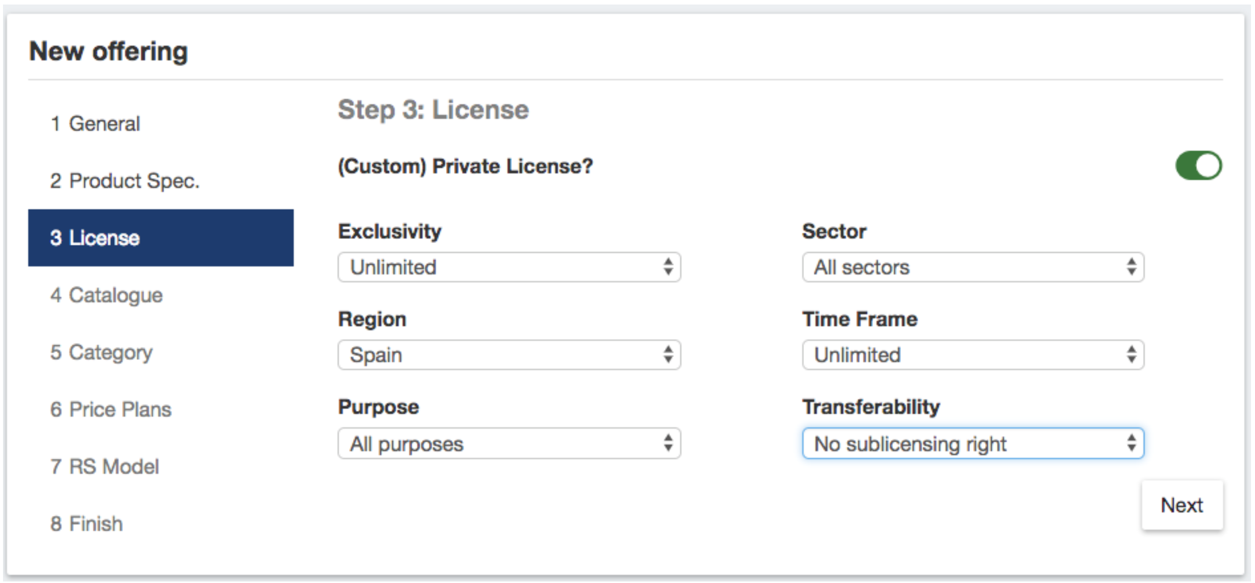
These systems report, in real time, environmental information of the city (weather, air quality, land status), using the SynchroniCity framework instance deployed in the Santander RZ: the associated datasets from Santander will be exposed to third parties through the SynchroniCity IoT data marketplace, according the specified SLA and license agreements.
Obeying these conditions, *company X* gets access to Santander environmental datasets, so it can assimilate this information into its forecasting algorithms and be exploited by its forecasters.
Company X provides its users with regional and national weather forecasts. By capturing also city's environmental data, it is also able to provide weather status, forecasts, evolutions, historical data and specific expanded info (beaches, waves, alerts, etc.) related to the city and its surroundings.
The different data sources that are involved in this use case are Environmental Sensor Nodes (**WeatherObserved** [40] entities and **AirQualityObserved** [38] entities) and Park and Gardens nodes (**GreenSpaceRecord** [42] entites). These datasets, formatted according to the FIWARE data models, are provided by Santander RZ to the SynchroniCity IoT data marketplace using the NGSI standards, which have been chosen for the reference implementation of the SynchroniCity framework.
As described in Section 4.2, in order to be able to address specific requirements of an IoT data marketplace designed for Smart City/IoT data, we adapted and streamlined the process to create data offerings from creating generic product specifications to data source specifications.
Figure 26 shows the updated form for creating the data source specification.



Enter a name	Santander Air Quality
Enter a version	0.1
Enter a brand	Santander
Enter an ID Number	4
Enter a description (optional)	Part of the environmental monitoring service. Data is collected from more than 1000 fixed nodes that are deployed in the city centre of Santander as well as mobile hardware that was installed on municipal public buses, parks and gardens vehicles and taxis.

Figure 26: Create a new data source specification.

In this example, Santander RZ specifies the characteristics of its Air Quality data source. To create data offerings linked to the created data source specification, Santander RZ needs to define the terms of the new data offering. This includes specifying a license agreement under which the offered data source can be used and the pricing model (e.g., free, one-time payment, subscription, pay-per-use). Figure 27 highlights the definition of the license agreement for the previously defined data source.



Step 3: License

(Custom) Private License? [✓]

Exclusivity: Unlimited	Sector: All sectors
Region: Spain	Time Frame: Unlimited
Purpose: All purposes	Transferability: No sublicensing right

Figure 27: Create a new offering.

We offer all the commonly available data licenses as a predefined option to be attached to the data offering. In addition to that, we also provide the option to provide custom licenses, that allow the data provider to specify the following properties:

- **Exclusivity:**
    - Exclusive. Only the licensee has a right to exploit the data that has been licensed to her. Most importantly, this also excludes the licensor itself to exploit the data after the license has been handed out.
    - Sole license. Only the licensee and the licensor have the right to exploit the licensed data.
    - Non-exclusive. This is the most common mode for use cases in the data marketplace. The license can be given out to any number of licensees and the licensor holds the right to exploit the licensed data after giving out the data license.
- **Sector:**
    - This specifies the business sector for which the license allows exploitation of the data. The most common modes will be to allow exploitation in all sectors, however, there may be use cases with a more restrictive license for the business sector.
- **Region:**
    - The data license can be restricted to be exploited only in certain geographical regions or unlimited and be applicable in all regions.
- **Time frame:**
    - This option specifies the time frame in which the data license remains valid. This can be only a select time window after which the license has to be renewed/newly acquired or can be set to be unlimited.
- **Purpose:**

    - Research. In this case, exploitation of the data is only granted for research purposes. This includes the use in education, publication in scientific journals/conferences and similar non-profit usage.
    - Commercial. This allows the licensee to use the exploitation of the data for business purposes, i.e. to generate profit from the exploitation of the data.
- **Transferability:**
    - No sublicensing right. The license only holds for the licensee; he is not granted the right to transfer his license privileges to others.
    - Sublicensing right. The licensee has the right to transfer the license to other parties without restrictions. The sublicense can include the right to exploit the whole data or only parts of the data.
    - Sublicensing right with restrictions. The licensee has the right to transfer the license to other parties, however, restrictions apply which are defined by the licensor.

Figure 28 shows a set of data offerings published by Santander RZ, in the specific example the ones involved in this use case which are: (i) *Santander Air Quality*, (ii) *Santander Weather Data*, and (iii) *Santander Greenspace Data*.

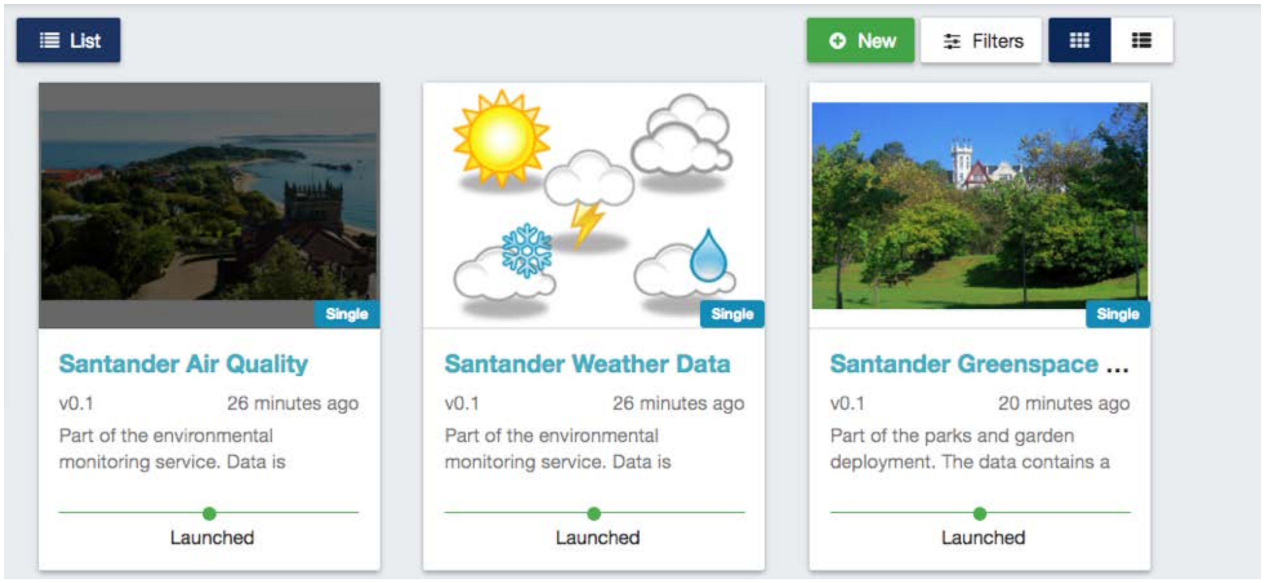
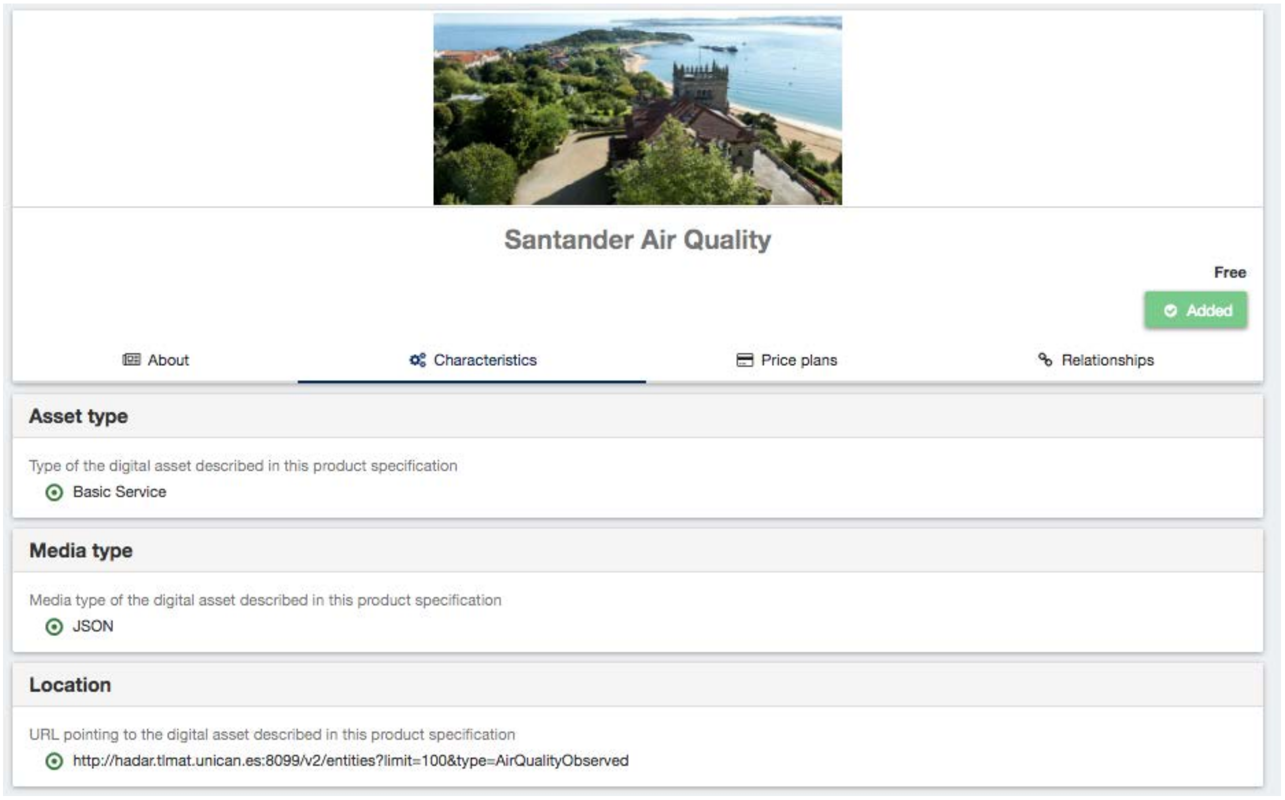


Figure 28: Santander RZ data offerings in the marketplace PoC.

As previously mentioned, *company X* is interested in purchasing all these data offerings to use as input for its weather forecasting service. After successfully purchasing a particular offering, the marketplace updates the role of *company X* so that the company will be authorized to access the data. Figure 29 shows details of the purchase of *Santander Air Quality* including the location where the data can be accessed from.


Santander Air Quality

Free

Added

About | Characteristics | Price plans | Relationships

Asset type

Type of the digital asset described in this product specification

Basic Service

Media type

Media type of the digital asset described in this product specification

JSON

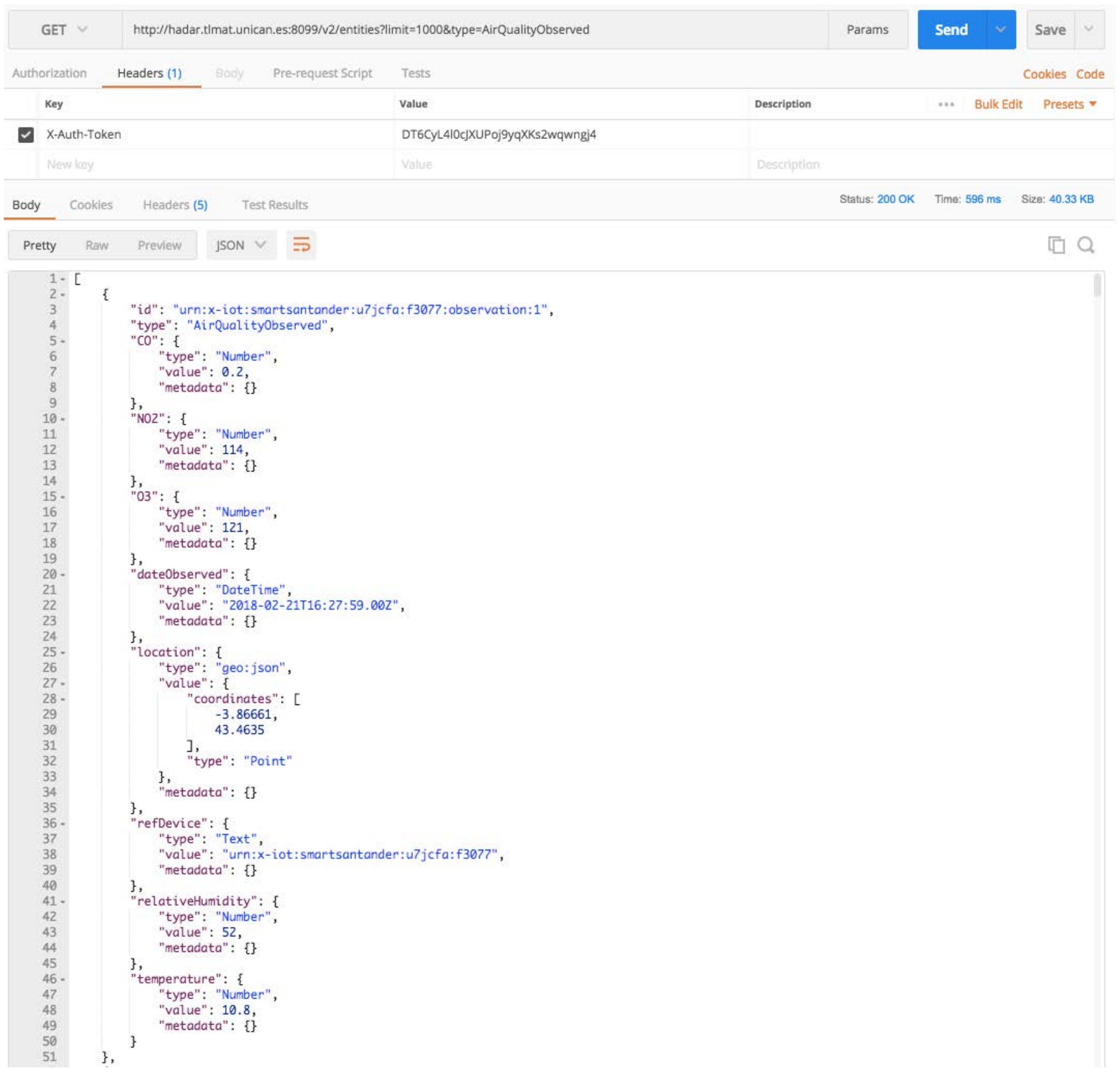
Location

URL pointing to the digital asset described in this product specification

http://hadar.tlmat.unican.es:8099/v2/entities?limit=100&type=AirQualityObserved

Figure 29: Purchasing a data offering.

Company X can now access the data by providing the OAuth2 token (as a X-Auth-Token header field) along with the *GET* request of the acquired resource.
We use Postman [45] to show how the purchased data can be accessed. More specifically, Figure 30 shows a screenshot of *company X* requesting the Air Quality data by using the resource location and the OAuth2 token obtained during the purchase phase, and the related response which includes the received data.
Figure 31 shows a new offering being published on the marketplace, named *Santander Weather Forecasting Service* provided by *company* X. This process showcases how more value can be added iteratively through the usage of the marketplace, as there may later be another company which uses the newly offered weather forecasting data stream as input to offer a new service (for example an outdoor activity planner using the weather forecast) through the marketplace.



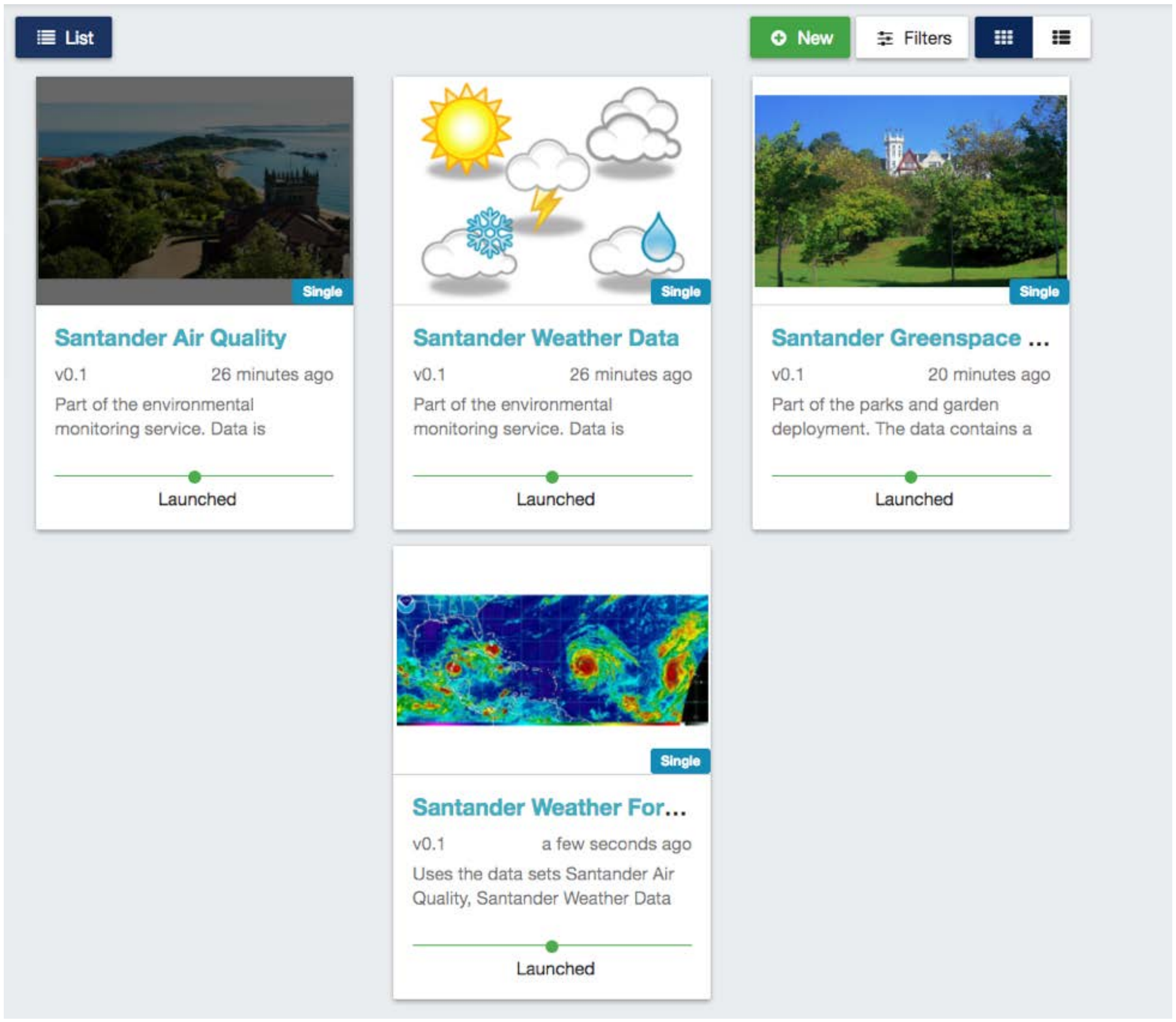
```
GET http://hadar.tlmat.unican.es:8099/v2/entities?limit=1000&type=AirQualityObserved
```

Key	Value	Description
X-Auth-Token	DT6CyL4l0cJXUPoj9yqXKs2wqwngj4	

Status: 200 OK Time: 596 ms Size: 40.33 KB

```
[
    {
        "id": "urn:x-iot:smartsantander:u7jcfa:f3077:observation:1",
        "type": "AirQualityObserved",
        "CO": {
            "type": "Number",
            "value": 0.2,
            "metadata": {}
        },
        "NO2": {
            "type": "Number",
            "value": 114,
            "metadata": {}
        },
        "O3": {
            "type": "Number",
            "value": 121,
            "metadata": {}
        },
        "dateObserved": {
            "type": "DateTime",
            "value": "2018-02-21T16:27:59.00Z",
            "metadata": {}
        },
        "location": {
            "type": "geo:json",
            "value": {
                "coordinates": [
                    -3.86661,
                    43.4635
                ],
                "type": "Point"
            },
            "metadata": {}
        },
        "refDevice": {
            "type": "Text",
            "value": "urn:x-iot:smartsantander:u7jcfa:f3077",
            "metadata": {}
        },
        "relativeHumidity": {
            "type": "Number",
            "value": 52,
            "metadata": {}
        },
        "temperature": {
            "type": "Number",
            "value": 10.8,
            "metadata": {}
        }
    },
```

Figure 30: Data access through Postman.


List

New

Filters

Single

Santander Air Quality

v0.1 26 minutes ago

Part of the environmental monitoring service. Data is

Launched

Single

Santander Weather Data

v0.1 26 minutes ago

Part of the environmental monitoring service. Data is

Launched

Single

Santander Greenspace ...

v0.1 20 minutes ago

Part of the parks and garden deployment. The data contains a

Launched

Single

Santander Weather For...

v0.1 a few seconds ago

Uses the data sets Santander Air Quality, Santander Weather Data

Launched

Figure 31: Santander RZ data offerings updated with company X *Santander Weather Forecasting Service* in the marketplace PoC.

5.2.3 Benefits

In this section, we provide an overview of the benefits and value that the stakeholders of the use case receive by using the data marketplace. It allows the *city council* to foster business models based on the exploitation of IoT data. The service portfolio offered to citizens and based on new technologies is improved. A newly available dataset (the output of the weather forecast app) is available for exploitation according to the SLA and license agreements defined by *company X*.
The data sources available on the data marketplace offer *company X* a value by improving the weather forecasts with more specific, relevant and city-oriented results. It provides new incentives to obtain more accurate results and foster innovation. Furthermore, if *company X* wants to provide their apps and services in a new city that provides the relevant data sources needed for the input on the data marketplace, the integration work required is kept to a minimum as the data format has

to adhere to the NGSI standards to be used in the reference implementation of the SynchroniCity framework.
The *citizens* benefit from more accurate weather forecasts that are updated in real time with additional and expanded information referred to its city and surroundings. Third party services that integrate the weather forecast into their service provision offer improved services to the *citizens*.
App developers and *IoT service providers* benefit from the new dataset available on the data marketplace, namely the *Santander Weather Forecasting Service*.

6 Conclusion

In this deliverable, we introduced the basic data marketplace enablers that underpin the SynchroniCity's vision to create a global market for IoT-enabled urban services with standardized interfaces and common information models. To this aim, we have first conducted an extensive literature survey exploring strengths and weaknesses of current approaches to the development of IoT data marketplaces.

Following a series of interviews with various project partners, we conveyed the insights gained into a combined value proposition and identify the most important characteristics the SynchroniCity IoT data marketplace needs to have in order to both alleviate the pains of the stakeholders and provide the maximum gain.

We presented the architecture design of the marketplace. We described each of the individual components, including a detailed API specification in the Appendix, and show how they interact with each other by examining different workflows. Moreover, we discussed security related aspects regarding the registration, authentication, authorization and access management as well as the secure communication among components.

We validated an initial PoC prototype of the marketplace by showing the way it can be leveraged in a typical smart city use case. The latter was defined along with the City of Santander, one of the SynchroniCity RZs, and involved environmental sensor deployments throughout the city and a weather forecast service. The aim of the validation was also to highlight the benefits it can provide to the key stakeholders (City of Santander, the service provider, customers of the service and third party service providers).

For the next iteration of the marketplace we plan to finalize the development of the *federation management* module which will be able to interact with other data marketplaces in order to discover, buy and access data within a federated ecosystem. Moreover, order creation will be integrated with digital signatures capabilities allowing both data consumer and provider to agree and sign terms and conditions (e.g., pricing, license, SLA) of the specific data offering. SLA definition (i.e., SLA and license management) will allow customers to agree upon specific characteristics of the way data is delivered. This can include information about the quality of data that is being provided such as *update frequency* to ensure the data is being kept up-to-date, the *amount of data* that is being delivered in a predefined time interval in the case of data streams, and the *response time* of the devices providing the data. The *feedback and reputation* module and the *transparency and accountability* service will instrument the marketplace with tools to increase the overall customers' trust by leveraging on transparent operations and technologically verifiable behaviors. We believe that emerging consensus-based distributed ledger technology, e.g., Blockchain, could be instrumental to realize this vision. These features will be presented in D2.5 (Advanced data marketplace enablers), due by M20.

References

1. Florian Stahl, Fabian Schomm, Gottfried Vossen and Lara Vomfell. A classification framework for data marketplaces | SpringerLink. Retrieved January 21, 2018, from https://link.springer.com/article/10.1007/s40595-016-0064-2
2. Florian Stahl, Fabian Schomm and Gottfried Vossen. EconPapers: The data marketplace survey revisited. Retrieved January 21, 2018, from https://econpapers.repec.org/RePEc:zbw:ercisw:18
3. Bhaskar Krishnamachari, Jerry Power, Cyrus Shahabi and Seon Ho Kim. IoT Marketplace: A data and API market for IoT devices. Retrieved January 21, 2018, from https://msbfile03.usc.edu/digitalmeasures/gerardpo/intellcont/USCIoTMarketplace_Jan152017-1.pdf
4. Florian Stahl, Fabian Schomm, Gottfried Vossen and Lara Vomfell. EconStor: Marketplaces for digital data: Quo vadis?. Retrieved January 21, 2018, from https://www.econstor.eu/handle/10419/129780
5. City Data Exchange - Smart Copenhagen. Retrieved January 21, 2018, from https://www.citydataexchange.com/
6. Thingful. Retrieved January 21, 2018, from https://www.thingful.net/
7. BIG IoT. Retrieved January 21, 2018, from http://big-iot.eu/
8. BIG IoT Marketplace. https://market.big-iot.org/. Accessed 21 Jan. 2018.
9. Thingful Blog (2017, August 7) — Interoperable Internet of Things – BIG IoT Project. Retrieved January 21, 2018, from http://blog.thingful.net/post/163911459061/interoperable-internet-of-things-big-iot-project
10. Business API Ecosystem - Biz Ecosystem RI | FIWARE Catalogue. Retrieved January 21, 2018, from https://catalogue.fiware.org/enablers/business-api-ecosystem-biz-ecosystem-ri
11. MK Data Hub. Retrieved January 21, 2018, from https://datahub.mksmart.org/
12. Dawex: Sell, buy and share data. Retrieved January 21, 2018, from https://www.dawex.com/en/
13. GitHub - dawex/signed-contract-vault: A forgery-proof contract. Retrieved January 21, 2018, from https://github.com/dawex/signed-contract-vault
14. Terbine. Retrieved January 21, 2018, from http://terbine.com/
15. Streamr. Retrieved January 21, 2018, from https://www.streamr.com/
16. IPFS is the Distributed Web. Retrieved January 21, 2018, from https://ipfs.io/
17. DataBrokerDAO · Global market for local data. Retrieved January 21, 2018, from https://databrokerdao.com/
18. Ocean Protocol. Retrieved January 21, 2018, from https://oceanprotocol.com/
19. BigchainDB. Retrieved January 21, 2018, from https://www.bigchaindb.com/
20. IOTA Data Market. Retrieved January 21, 2018, from https://data.iota.org/
21. FIWARE TMF Business API Ecosystem · Apiary. https://fiwaretmfbizecosystem.docs.apiary.io/. Accessed 30 Jan. 2018.
22. GitHub - FIWARE-TMForum/business-ecosystem-logic-proxy. https://github.com/FIWARE-TMForum/business-ecosystem-logic-proxy. Accessed 31 Jan. 2018.
23. RFC 6749 - The OAuth 2.0 Authorization. https://tools.ietf.org/html/rfc6749. Accessed 29 Jan. 2018.
24. RFC 7061 - eXtensible Access Control Markup Language - IETF Tools. https://tools.ietf.org/html/rfc7061. Accessed 29 Jan. 2018.
25. NIST Guideline for use of TLS http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r1.pdf. Accessed 30 Jan. 2018.
26. IEEE 802.15.4-2015 - IEEE Standard for Low-Rate Wireless Networks. https://standards.ieee.org/findstds/standard/802.15.4-2015.html. Accessed 20 Feb. 2018.
27. Smartsantander.eu. http://www.smartsantander.eu/. Accessed 20 Feb. 2018.

28. Publish/Subscribe Context Broker - Orion - FIWARE Catalogue. 9 Mar. 2017, https://catalogue.fiware.org/enablers/publishsubscribe-context-broker-orion-context-broker/documentation. Accessed 20 Feb. 2018.
29. NGSIv2 Specification document. http://fiware.github.io/context.Orion/api/v2/. Accessed 20 Feb. 2018.
30. Backend Device Management - IDAS | FIWARE Catalogue. 12 Feb. 2018, https://catalogue.fiware.org/enablers/backend-device-management-idas. Accessed 20 Feb. 2018.
31. STH-Comet | FIWARE Catalogue. 20 Nov. 2017, https://catalogue.fiware.org/enablers/sth-comet. Accessed 20 Feb. 2018.
32. Cygnus | FIWARE Catalogue. 20 Nov. 2017, https://catalogue.fiware.org/enablers/cygnus. Accessed 20 Feb. 2018.
33. PEP Proxy - Wilma | FIWARE Catalogue. 12 Jun. 2017, https://catalogue.fiware.org/enablers/pep-proxy-wilma. Accessed 20 Feb. 2018.
34. Identity Management - KeyRock | FIWARE Catalogue. 17 Jan. 2018, https://catalogue.fiware.org/enablers/identity-management-keyrock. Accessed 20 Feb. 2018.
35. Authorization PDP - AuthZForce | FIWARE Catalogue. 11 Dec. 2017, https://catalogue.fiware.org/enablers/authorization-pdp-authzforce. Accessed 20 Feb. 2018.
36. Fiware DataModels. https://fiware-datamodels.readthedocs.io/en/latest/index.html. Accessed 20 Feb. 2018.
37. Fiware DataModels - Environment - Read the Docs. https://fiware-datamodels.readthedocs.io/en/latest/Environment/doc/introduction/index.html. Accessed 20 Feb. 2018.
38. Fiware DataModels - AirQualityObserved - Read the Docs. https://fiware-datamodels.readthedocs.io/en/latest/Environment/AirQualityObserved/doc/spec/index.html. Accessed 20 Feb. 2018.
39. Fiware DataModels - NoiseLevelObserved - Read the Docs. https://fiware-datamodels.readthedocs.io/en/latest/Environment/NoiseLevelObserved/doc/spec/index.html. Accessed 20 Feb. 2018.
40. Fiware DataModels - WeatherObserved - Read the Docs. https://fiware-datamodels.readthedocs.io/en/latest/Weather/WeatherObserved/doc/spec/index.html. Accessed 20 Feb. 2018.
41. Fiware DataModels - Garden - Read the Docs. https://fiware-datamodels.readthedocs.io/en/latest/ParksAndGardens/Garden/doc/spec/index.html. Accessed 20 Feb. 2018.
42. Fiware DataModels - GreenspaceRecord - Read the Docs. https://fiware-datamodels.readthedocs.io/en/latest/ParksAndGardens/GreenspaceRecord/doc/spec/index.html. Accessed 20 Feb. 2018.
43. CKAN is an open-source DMS. https://github.com/ckan/ckan. Accessed 20 Feb. 2018.
44. Hypercat. http://www.hypercat.io/standard.html. Accessed 20 Feb. 2018.
45. Postman. https://www.getpostman.com/. Accessed 20 Feb. 2018.

Appendix

FORMAT: 1A
TITLE: SynchroniCity IoT data marketplace APIs Specification
DATE: 10 November 2017
VERSION: v0.1
APIARY_PROJECT: synchronicityiotdm
SPEC_URL: https://synchronicityiotdm.docs.apiary.io/#

SynchroniCity IoT data marketplace APIs

The SynchroniCity IoT data marketplace Open API is an extension of the Business API Ecosystem made up of the FIWARE Business Framework and a set of APIs (and its reference implementations) provided by the TMForum (https://catalogue.fiware.org/enablers/business-api-ecosystem-biz-ecosystem-ri).
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 catalog management, ordering management, federation management, revenue management, customer management, SLA and license management, feedback and reputation, and transparency and accountability service.

Editors

- Angelo Capossele, Digital Catapult
- Andrea Gaglione, Digital Catapult
- Daniel Puschmann, Digital Catapult

Status

This is a work in progress and is changing on a daily basis.

Copyright

Specification

Authentication

Each HTTP request against the SynchroniCity IoT data marketplace Open API requires the inclusion of specific authentication credentials.
The specific implementation of this API may support multiple authentication schemes (OAuth, Basic Auth, Token) and will be determined by the specific provider that implements the data marketplace.
Please contact the provider to determine the best way to authenticate against this API. Remember that some authentication schemes may require that the API operates using SSL over HTTP (HTTPS).
The reference implementation of the SynchroniCity IoT data marketplace Open APIs provides support for Cookie, and OAuth2 authentication.

Synchronous Faults

Error responses will be encoded using the most appropriated <code>content-type</code> in base to the <code>Accept</code> header of the request.
JSON Example

```
{
    "error": "No JSON object could be decoded",
}
```

XML Example

```
<?xml version="1.0" encoding="utf-8"?>
<error>No JSON object could be decoded</error>
```

Used HTTP Codes

HTTP Code	Type	Description
200	OK	Your request has been completed properly
201	Created	Your resource has been created.
204	No content	Your request has been processed, but a response is not available. Generally used when deleting entities
400	Bad Request	The content of your request is not correct
401	Unauthorized	You are not logged in or the Authorization token you are providing is not valid
403	Forbidden	You have no rights to perform the request
404	Not Found	The resource you are looking for does not exists
409	Conflict	The resource you are trying to create already exists
422	Unprocessable Entity	The entity included in the request cannot be processed (e.g., it includes invalid fields)

500	Internal server error	There was an internal error in the system so your request cannot be completed

Group Data Source Specification Management API

This API is a subset of the Catalog 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.
The Data Source Specification Management API uses the following fields:

- **id** - Unique identifier of the data source specification
- **href** - URL pointing to the product specification info
- **productNumber** - An id number assigned by the data provider/seller to identify its data source specification
- **version** - Version of the data source specification
- **lastUpdate** - Date and time of the last update
- **name** - Name of the data source specification
- **description** - Narrative text that explains what the data source is
- **isBundle** - Determines whether the current specification represents a single data source specification or a bundle of data source specifications
- **brand** - The provider's name or trademark of the data source
- **lifecycleStatus** - Current lifecycle status of the data source specification
- **relatedParty** - List of parties and its roles related to the current data source specification. For each party, it is included the id and the href as described in the Party Management section. Additionally, it is included a *role* field specifying the role of the user in the current data source specification
- **attachment** - List of product attachments, such as video, pictures, pdf documents, etc. Which help describing the data source specification. Each attachment contains the following fields:
    - **type** - Attachment type, video, picture, document, etc. The type *Picture* can be included only once and is used by the platform as the logo of the data source specification
    - **url** - URL pointing to the attachment itself. Note that if the attachment has been uploaded to the system using the asset management API, you can use the URL returned by the upload task in the *Location* header
- **bundledProductSpecification** - In case the current data source is a bundle (isBundle is true), this field contains the list of data source specifications that made up the bundle. Each element of the list must contain the id, the href and the name of the data source specification being bundled
- **productSpecCharacteristic** - Characteristics define attributes and properties of data sources specification and are used to describe the data source specification in compliance with the SynchroniCity data models. Characteristics can be a discrete value, or can be a range of values. Characteristics have the following fields:
    - **name** - Name of the characteristic
    - **description** - Narrative text explaining what the characteristic is
    - **valueType** - The kind of value the characteristic could have. Valid values are *String* and *Number*

- **productSpecCharacteristicValue** - List of values of the characteristic. If configurable is false, this field must contain a single value. Each of these elements contain the following fields:
  - **default** - Indicates whether the current value is the default for the characteristic
  - **unitOfMeasure** - Could be minutes, MB, etc. This field is only used when the type is *Number*
  - **value** - Value of the characteristic when it is a discrete value. If this field is included, valueFrom and valueTo must be empty
  - **valueFrom** - Starting value of the characteristic when it is a range. If this field is included, valueTo must be also included and value must be empty
  - **valueTo** - Ending value of the characteristic when it is a range. If this field is included, valueFrom must be also included and value must be empty
  - **valueType** - The kind of value the characteristic could have. Valid values are *String* and *Number*

Data Source Specification Collection

[/DSProductCatalog/api/catalogManagement/v2/productSpecification{?offset}{?size}{?isBundle}{?productNumber}{?relatedParty.id}{?lifecycleStatus}{?sort}{?body}]

List Data Source Specifications [GET]

- Parameters
  - offset: 1 (optional) - Optional parameter used to specify the first element to be returned
  - size: 10 (optional) - Optional parameter used to limit the number of elements returned
  - isBundle: true (optional) - Optional parameter used to filter the returned data source specifications by isBundle
  - productNumber: 1 (optional)- Optional parameter used to filter the returned data source specifications by product number
  - relatedParty.id: userA (optional) - Optional parameter used to filter the returned data source specifications by owner
  - lifecycleStatus: Active (optional) - Optional parameter used to filter the returned data source specifications by lifecycle status
  - sort: name (optional) - Optional parameter used to specify how to sort results
  - body: Parking (optional) - Optional parameter used to make keyword searches
- Request
  - Headers
  - Authorization: Bearer YOUR_OAUTH2_TOKEN

- Response 200 (application/json):

```
[{
      "id": "22",
      "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/22",
      "productNumber": "I42-340-DX",
      "version": "2.0",
      "lastUpdate": "2017-10-19T16:42:23.0Z",
      "name": "Parking spots",
      "description": "Parking spots available in the parking lot XYZ",
      "isBundle": true,
      "brand": "Parking Co",
      "lifecycleStatus": "Active",
      "relatedParty": [{
          "id": "userA",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
          "role": "Owner"
      }],
      "attachment": [{
          "type": "Picture",
          "url": "https://URL_API/media/picture.png"
      }],
      "bundledProductSpecification": [{
          "id": "15",
          "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/15",
          "name": "Sensor 15"
      }, {
          "id": "64",
          "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/64",
          "name": "Sensor 64"
      }],
      "productSpecCharacteristic": [{
          "name": "data model",
          "description": "Specification of the data model",
          "valueType": "string",
          "productSpecCharacteristicValue": [{
              "valueType": "string",
              "default": true,
              "value": "http://fiware-
datamodels.readthedocs.io/en/latest/Parking/ParkingSpot/doc/spec/index.html"
          }]
      }]
}]
```

Create Data Source Specification [POST]

- Request (application/json)
    - Headers
    - Authorization: Bearer YOUR_OAUTH2_TOKEN

    - Body:

```
{
      "productNumber": "I42-340-DX",
      "version": "2.0",
      "lastUpdate": "2017-10-19T16:42:23.0Z",
      "name": "Parking spots",
      "description": "Parking spots available in the parking lot XYZ",
      "isBundle": true,
      "brand": "Parking Co",
      "lifecycleStatus": "Active",
      "relatedParty": [{
          "id": "userA",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
          "role": "Owner"
```

```
        }],
        "attachment": [{
            "type": "Picture",
            "url": "https://URL_API/media/picture.png"
        }],
        "bundledProductSpecification": [{
            "id": "15",
            "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecificatio
n/15",
            "name": "Sensor 15"
        }, {
            "id": "64",
            "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecificatio
n/64",
            "name": "Sensor 64"
        }],
        "productSpecCharacteristic": [{
            "name": "data model",
            "description": "Specification of the data model",
            "valueType": "string",
            "productSpecCharacteristicValue": [{
                "valueType": "string",
                "default": true,
                "value": "http://fiware-
datamodels.readthedocs.io/en/latest/Parking/ParkingSpot/doc/spec/index.html"
            }]
        }]
}
```

- Response 201 (application/json):

```
{
      "id": "22",
      "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/22",
      "productNumber": "I42-340-DX",
      "version": "2.0",
      "lastUpdate": "2017-10-19T16:42:23.0Z",
      "name": "Parking spots",
      "description": "Parking spots available in the parking lot XYZ",
      "isBundle": true,
      "brand": "Parking Co",
      "lifecycleStatus": "Active",
      "relatedParty": [{
          "id": "userA",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
          "role": "Owner"
      }],
      "attachment": [{
          "type": "Picture",
          "url": "https://URL_API/media/picture.png"
      }],
      "bundledProductSpecification": [{
          "id": "15",
          "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/15",
          "name": "Sensor 15"
      }, {
          "id": "64",
          "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/64",
          "name": "Sensor 64"
      }],
      "productSpecCharacteristic": [{
          "name": "data model",
          "description": "Specification of the data model",
          "valueType": "string",
          "productSpecCharacteristicValue": [{
              "valueType": "string",
              "default": true,
              "value": "http://fiware-
datamodels.readthedocs.io/en/latest/Parking/ParkingSpot/doc/spec/index.html"
          }]
      }]
}
```

Data Source Specification Entry

[/DSProductCatalog/api/catalogManagement/v2/productSpecification/{id}]

Get Data Source Specification [GET]

- Parameters
  - id: 1 - Id of the product to be retrieved
- Request
  - Headers
  - Authorization: Bearer YOUR_OAUTH2_TOKEN

- Response 200 (application/json):

```
{
    "id": "22",
    "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/22",
    "productNumber": "I42-340-DX",
    "version": "2.0",
    "lastUpdate": "2017-10-19T16:42:23.0Z",
    "name": "Parking spots",
    "description": "Parking spots available in the parking lot XYZ",
    "isBundle": true,
    "brand": "Parking Co",
    "lifecycleStatus": "Active",
    "relatedParty": [{
        "id": "userA",
        "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
        "role": "Owner"
    }],
    "attachment": [{
            "type": "Picture",
            "url": "https://URL_API/media/picture.png"
        }],
        "bundledProductSpecification": [{
            "id": "15",
            "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/15",
            "name": "Sensor 15"
        }, {
            "id": "64",
            "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/64",
            "name": "Sensor 64"
        }],
         "productSpecCharacteristic": [{
            "name": "data model",
            "description": "Specification of the data model",
            "valueType": "string",
            "productSpecCharacteristicValue": [{
                "valueType": "string",
                "default": true,
                "value": "http://fiware-
datamodels.readthedocs.io/en/latest/Parking/ParkingSpot/doc/spec/index.html"
            }]
        }]
}
```

Update Data Source Specification [PATCH]

- Parameters
    - id: 1 - Id of the data source specification to be updated
- Request - Partial update of the data source specification, only the fields to be updated need to be provided - (application/json)
    - Headers
    - Authorization: Bearer YOUR_OAUTH2_TOKEN

    - Body:

```
{
      "lifecycleStatus": "Retired"
}
```

- Response 200 (application/json)

```
{
        "id": "22",
        "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/22",
        "productNumber": "I42-340-DX",
        "version": "2.0",
        "lastUpdate": "2017-10-19T16:42:23.0Z",
        "name": "Parking spots",
        "description": "Parking spots available in the parking lot XYZ",
        "isBundle": true,
        "brand": "Parking Co",
        "lifecycleStatus": "Retired",
        "relatedParty": [{
            "id": "userA",
            "href":
  "https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
            "role": "Owner"
        }],
        "attachment": [{
            "type": "Picture",
            "url": "https://URL_API/media/picture.png"
        }],
        "bundledProductSpecification": [{
            "id": "15",
            "href":
  "https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/15",
            "name": "Sensor 15"
        }, {
            "id": "64",
            "href":
  "https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/64",
            "name": "Sensor 64"
        }],
         "productSpecCharacteristic": [{
            "name": "data model",
            "description": "Specification of the data model",
            "valueType": "string",
            "productSpecCharacteristicValue": [{
                "valueType": "string",
                "default": true,
                "value": "http://fiware-
  datamodels.readthedocs.io/en/latest/Parking/ParkingSpot/doc/spec/index.html"
            }]
        }]
  }
```

Group 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 catalog. This resource includes pricing information and links (optional) Service Level Agreements and license terms.
The Product Offering Management API uses the following fields:

- **id** - Unique identifier of the data offering
- **href** - URL pointing to the data offering info
- **version** - Version of the data offering
- **lastUpdate** - Date and time of the last update
- **name** - Name of the data offering
- **description** - Narrative text that explains what the data offering is
- **isBundle** - Determines whether the current data offering represents a single offering or a bundle of offerings. If false, then a productSpecification will be returned, but the bundledProductOfferings will be absent or empty and vice-versa if isBundle is true.
- **lifecycleStatus** - Current lifecycle status of the data offering
- **category** - List of categories of the data offering. For each category the id, href, and name fields are included as described in Category Management section of the FIWARE Business API Ecosystem specification.

- **place** - List of places where the data offering is available. Each object includes the name of the place
- **bundledProductOffering** - List of data sources included when the data offering is a bundle. For each offering is included the id, the name, and the href
- **productSpecification** - Data source specification offered when the offering is not a bundle, it includes the id, the name, and the href
- **slaSpecification** - Service Level Agreement to link with the current data offering, it includes the id, the name, and the href
- **licenseSpecification** - License to link with the current data offering, it includes the id, the name, and the href
- **serviceCandidate** - Object used to specify the class of the current data offering as described in the Revenue Sharing Management sections of the FIWARE Business API Ecosystem specification.
- **productOfferingPrice** - List of pricing models of the data offering. Each of the pricing models included defines a price that can be selected by the customers during the ordering process. Each pricing model includes the following fields:
    - **name** - Name of the pricing model
    - **description** - Narrative text explaining what the pricing models is
    - **priceType** - Type of the pricing model. It could be *one time* for payments made once at acquisition time, *recurring* for payments made periodically, and *usage* for payments calculated based on the usage made by the customer of the data source
    - **unitOfMeasure** - Unit that is monitored when the priceType is usage, otherwise this field is empty
    - **recurringChargePeriod** - Specifies the period between charges when the priceType is recurring, otherwise the field is empty
    - **price** - Object describing the price of the pricing model. It contains the following fields:
        - **taxIncludedAmount** - Price of the model
        - **dutyFreeAmount** - Price of the model without including taxes
        - **taxRate** - Percentage of taxes that apply to the price
        - **currencyCode** - Currency of the price

Data Offering Collection

[/DSProductCatalog/api/catalogManagement/v2/catalog/{catId}/productOffering{?offset}{?size}{?isBundle}{?name}{?relatedParty.id}{?category.id}{?category.name}{?productSpecification.id}{bundledProductOffering.id}{?lifecycleStatus}{?sort}{?body}]

List Data Offerings [GET]

- Parameters
    - catId: 1 - Id of the catalog whose offerings are going to be retrieved
    - offset: 1 (optional) - Optional parameter used to specify the first element to be returned
    - size: 10 (optional) - Optional parameter used to limit the number of elements returned
    - isBundle: true (optional) - Optional parameter used to filter the returned data offerings by isBundle
    - name: 1 (optional)- Optional parameter used to filter the returned data offerings by name
    - relatedParty.id: userA (optional) - Optional parameter used to filter the returned data offerings by owner
    - category.id: 1 (optional) - Optional parameter used to filter the returned data offerings by category id

    - category.name: Cloud (optional) - Optional parameter used to filter the returned data offerings by category name
    - productSpecification.id: 1 (optional) - Optional parameter used to filter the returned data offerings by product specification id
    - bundledProductOffering.id: 1 (optional) - Optional parameter used to filter the returned data offerings by bundled offerings id
    - lifecycleStatus: Active (optional) - Optional parameter used to filter the returned data offering by lifecycle status
    - sort: name (optional) - Optional parameter used to specify how to sort results
    - body: Cloud (optional) - Optional parameter used to make keyword searches
- Request
    - Headers
    - Authorization: Bearer YOUR_OAUTH2_TOKEN

- Response 200 (application/json):

```
[{
      "id": "42",
      "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/catalog/1/productOffering/
42",
      "version": "1.0",
      "lastUpdate": "2017-10-19T16:42:23.0Z",
      "name": "Air quality",
      "description": "Observations of the air quality in Manchester",
      "isBundle": false,
      "lifecycleStatus": "Active",
      "category": [{
           "id": "12",
           "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/category/12",
           "name": "Environment"
      }],
          "place": [{
              "name": "Manchester"
      }],
      "bundledProductOffering": [],
      "productSpecification": {
           "id": "13",
           "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/13",
           "name": "air quality sensor 1"
      },
      "slaSpecification": {
           "id": "61",
           "href":
"https://URL_API/DSLAlicenseManagement/api/slaManagement/v1/slaSpecification/61",
           "name": "SLA V1"
      },
      "licenseSpecification": {
           "id": "31",
           "href":
"https://URL_API/DSLAlicenseManagement/api/licenseManagement/v1/licenseSpecification/
31",
           "name": "Non exclusive license"
```

```
        },
        "serviceCandidate": {
             "id": "defaultRevenue",
             "name": "Revenue Sharing Model"
        },
        "productOfferingPrice": [{
             "name": "Monthly Price",
             "description": "monthlyprice",
             "priceType": "recurring",
             "unitOfMeasure": "",
             "price": {
                "taxIncludedAmount": 12,
                "dutyFreeAmount": 10,
                "taxRate": 20,
                "currencyCode": "EUR"
            },
         "recurringChargePeriod": "monthly"
        }, {
         "name": "Usage Price",
         "description": "usageprice",
         "priceType": "usage",
         "unitOfMeasure": "second",
         "price": {
            "taxIncludedAmount": 12,
            "dutyFreeAmount": 10,
            "taxRate": 20,
            "currencyCode": "EUR"
         },
         "recurringChargePeriod": ""
        }]
}]
```

Create Data Offering [POST]

- Parameters
  - catId: 1 - Id of the catalog whose data offerings are going to be retrieved
- Request (application/json)
  - Headers
  - Authorization: Bearer YOUR_OAUTH2_TOKEN

  - Body:

```
{
      "version": "1.0",
      "name": "Air quality",
      "description": "Observations of the air quality in Manchester",
      "isBundle": true,
      "lifecycleStatus": "Active",
      "category": [{
          "id": "12",
          "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/category/12",
          "name": "Environment"
      }],
      "place": [{
          "name": "Manchester"
```

```
        }],
        "bundledProductOffering": [],
        "productSpecification": {
            "id": "13",
            "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecificatio
n/13",
            "name": "air quality sensor 1"
        },
        "slaSpecification": {
            "id": "61",
            "href":
"https://URL_API/DSLAlicenseManagement/api/slaManagement/v1/slaSpecification/6
1",
            "name": "SLA V1"
        },
        "licenseSpecification": {
            "id": "31",
            "href":
"https://URL_API/DSLAlicenseManagement/api/licenseManagement/v1/licenseSpecifi
cation/31",
            "name": "Non exclusive license"
        },
        "serviceCandidate": {
            "id": "defaultRevenue",
            "name": "Revenue Sharing Model"
        },
        "productOfferingPrice": [{
            "name": "Monthly Price",
            "description": "monthlyprice",
            "priceType": "recurring",
            "unitOfMeasure": "",
            "price": {
                "taxIncludedAmount": 12,
                "dutyFreeAmount": 10,
                "taxRate": 20,
                "currencyCode": "EUR"
            },
            "recurringChargePeriod": "monthly"
        }, {
            "name": "Usage Price",
            "description": "usageprice",
            "priceType": "usage",
            "unitOfMeasure": "second",
            "price": {
                "taxIncludedAmount": 12,
                "dutyFreeAmount": 10,
                "taxRate": 20,
                "currencyCode": "EUR"
            },
            "recurringChargePeriod": ""
        }]
}
```

- Response 201 (application/json):

```
{
      "id": "42",
      "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/catalog/1/productOffering/
42",
      "version": "1.0",
      "lastUpdate": "2017-10-19T16:42:23.0Z",
      "name": "Air quality",
      "description": "Observations of the air quality in Manchester",
      "isBundle": false,
      "lifecycleStatus": "Active",
      "category": [{
          "id": "12",
          "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/category/12",
          "name": "Environment"
      }],
      "place": [{
          "name": "Manchester"
      }],
      "bundledProductOffering": [],
      "productSpecification": {
          "id": "13",
          "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/13",
          "name": "air quality sensor 1"
      },
      "slaSpecification": {
          "id": "61",
          "href":
"https://URL_API/DSLAlicenseManagement/api/slaManagement/v1/slaSpecification/61",
          "name": "SLA V1"
      },
      "licenseSpecification": {
          "id": "31",
          "href":
"https://URL_API/DSLAlicenseManagement/api/licenseManagement/v1/licenseSpecification/
31",
          "name": "Non exclusive license"
      },
      "serviceCandidate": {
          "id": "defaultRevenue",
          "name": "Revenue Sharing Model"
      },
      "productOfferingPrice": [{
          "name": "Monthly Price",
          "description": "monthlyprice",
          "priceType": "recurring",
          "unitOfMeasure": "",
          "price": {
              "taxIncludedAmount": 12,
              "dutyFreeAmount": 10,
              "taxRate": 20,
              "currencyCode": "EUR"
          },
        "recurringChargePeriod": "monthly"
    }, {
        "name": "Usage Price",
        "description": "usageprice",
        "priceType": "usage",
        "unitOfMeasure": "second",
        "price": {
            "taxIncludedAmount": 12,
            "dutyFreeAmount": 10,
            "taxRate": 20,
            "currencyCode": "EUR"
        },
        "recurringChargePeriod": ""
    }]
}
```

Data Offering Entry

[/DSProductCatalog/api/catalogManagement/v2/catalog/{catId}/productOffering/{id}]

Get Data Offering [GET]

- Parameters
  - catId: 1 - Id of the catalog whose data offerings are going to be retrieved
  - id: 1 - Id of the data offering to be retrieved
- Request
  - Headers
  - Authorization: Bearer YOUR_OAUTH2_TOKEN

- Response 200 (application/json):

```
{
    "id": "42",
    "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/catalog/1/productOffering/
42",
    "version": "1.0",
    "lastUpdate": "2017-10-19T16:42:23.0Z",
    "name": "Air quality",
    "description": "Observations of the air quality in Manchester",
    "isBundle": false,
    "lifecycleStatus": "Active",
    "category": [{
        "id": "12",
        "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/category/12",
        "name": "Environment"
    }],
    "place": [{
        "name": "Manchester"
    }],
    "bundledProductOffering": [],
    "productSpecification": {
        "id": "13",
        "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/13",
            "name": "air quality sensor 1"
        },
        "slaSpecification": {
            "id": "61",
            "href":
"https://URL_API/DSLAlicenseManagement/api/slaManagement/v1/slaSpecification/61",
            "name": "SLA V1"
        },
        "licenseSpecification": {
            "id": "31",
            "href":
"https://URL_API/DSLAlicenseManagement/api/licenseManagement/v1/licenseSpecification/
31",
            "name": "Non exclusive license"
        },
        "serviceCandidate": {
            "id": "defaultRevenue",
            "name": "Revenue Sharing Model"
        },
        "productOfferingPrice": [{
            "name": "Monthly Price",
            "description": "monthlyprice",
            "priceType": "recurring",
            "unitOfMeasure": "",
            "price": {
                "taxIncludedAmount": 12,
                "dutyFreeAmount": 10,
                "taxRate": 20,
                "currencyCode": "EUR"
            },
            "recurringChargePeriod": "monthly"
        }, {
            "name": "Usage Price",
            "description": "usageprice",
            "priceType": "usage",
            "unitOfMeasure": "second",
            "price": {
                "taxIncludedAmount": 12,
                "dutyFreeAmount": 10,
                "taxRate": 20,
                "currencyCode": "EUR"
            },
            "recurringChargePeriod": ""
        }]
}
```

Update Data Offering [PATCH]

- Parameters
    - catId: 1 - Id of the catalog whose data offerings are going to be retrieved
    - id: 1 - Id of the data offering to be retrieved
- Request - Partial update of the data offering, only the fields to be updated need to be provided - (application/json)
    - Headers

- Authorization: Bearer YOUR_OAUTH2_TOKEN
- Body:

```
{
    "lifecycleStatus": "Retired"
}
```

- Response 200 (application/json):

```
{
      "id": "42",
      "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/catalog/1/productOffering/
42",
      "version": "1.0",
      "lastUpdate": "2017-10-19T16:42:23.0Z",
      "name": "Air quality",
      "description": "Observations of the air quality in Manchester",
      "isBundle": false,
      "lifecycleStatus": "Retired",
      "category": [{
          "id": "12",
          "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/category/12",
          "name": "Environment"
      }],
      "place": [{
          "name": "Manchester"
      }],
      "bundledProductOffering": [],
      "productSpecification": {
          "id": "13",
          "href":
"https://URL_API/DSProductCatalog/api/catalogManagement/v2/productSpecification/13",
          "name": "air quality sensor 1"
      },
      "slaSpecification": {
          "id": "61",
          "href":
"https://URL_API/DSLAlicenseManagement/api/slaManagement/v1/slaSpecification/61",
          "name": "SLA V1"
      },
      "licenseSpecification": {
          "id": "31",
          "href":
"https://URL_API/DSLAlicenseManagement/api/licenseManagement/v1/licenseSpecification/
31",
          "name": "Non exclusive license"
      },
      "serviceCandidate": {
          "id": "defaultRevenue",
          "name": "Revenue Sharing Model"
      },
      "productOfferingPrice": [{
          "name": "Monthly Price",
          "description": "monthlyprice",
          "priceType": "recurring",
          "unitOfMeasure": "",
          "price": {
              "taxIncludedAmount": 12,
              "dutyFreeAmount": 10,
              "taxRate": 20,
              "currencyCode": "EUR"
          },
          "recurringChargePeriod": "monthly"
      }, {
          "name": "Usage Price",
          "description": "usageprice",
          "priceType": "usage",
          "unitOfMeasure": "second",
          "price": {
              "taxIncludedAmount": 12,
              "dutyFreeAmount": 10,
              "taxRate": 20,
              "currencyCode": "EUR"
          },
          "recurringChargePeriod": ""
      }]
}
```

Group 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.
This API manages the following fields:

- **id** - Unique identifier of the order
- **href** - URL pointing to the product order info
- **externalId** - Id of the order given by customer, which can be used by them to identify the order in their own systems
- **description** - Description of the order
- **state** - Status of the order, relative to the status of the different order items
- **orderDate** - Date when the order was created
- **completionDate** - Date when the order was completed
- **requestedStartDate** - Order start date wished by the requestor
- **notificationContact** - Contact attached to the order to send back information regarding the current order
- **relatedParty** - Defines parties which are involved in the order and the role they are playing. For each party, it is included the id and the href as described in the Party Management section of the FIWARE Business API Ecosystem specification. Additionally, it is inlcluded a *role* field specifing the role of the user in the current order
- **orderItem** - List of order items that have to be treated. For each order item the following information is managed:
    - **id** - Id of the order item relative to the order (Only need to be unique within the order)
    - **action** - Type of the order item. Currently only *add* is supported (acquisition)
    - **state** - Status of the order item

    - **billingAccount** - Billing account selected by the customer to acquire the offering according to the Billing Management API section as specified in the FIWARE Business API Ecosystem specification
    - **productOffering** - Data offering being acquired. It includes the id and the href of the product offering
    - **product** - Information provided to create the inventory digital asset. It also contains the selected pricing. The different fields managed by this object are the same as the described in the Inventory Management API Section of the FIWARE Business API Ecosystem specification
    - **signature** - Digital signature of the customer agreeing on SLA and license as specified by the related data offering. It contains the id, href, public parameters, algorithm, public key (MUST match the one of the customer ordering the related data offer) and the value of the signature.

Order Collection

[/DSProductOrdering/api/productOrdering/v2/productOrder{?offset}{?size}{?priority}{?state}{?relatedParty.id}{?relatedParty.role}]

List Orders [GET]

- Parameters
    - offset: 1 (optional) - Optional parameter used to specify the first element to be returned
    - size: 10 (optional) - Optional parameter used to limit the number of elements returned
    - priority: 1 (optional) - Optional parameter used to filter the returned orders by priority
    - state: InProgress (optional) - Optional parameter used to filter the returned orders by state
    - relatedParty.id: userA (optional) - Optional parameter used to filter the returned orders by user id
    - relatedParty.role: seller (optional) - Optional parameter used to filter the returned orders by user role (customer or seller)
- Request
    - Headers
    - Authorization: Bearer YOUR_OAUTH2_TOKEN

- Response 200 (application/json):

```
[{
       "id": "42",
       "href":
"https://URL_API/DSProductOrdering/api/productOrdering/v2/productOrder/42",
       "externalId": "CustomerId",
       "description": "Observations of the air quality in Manchester",
       "state": "InProgress",
       "orderDate": "2017-10-12T16:42:23-04:00",
       "completionDate": "2017-10-12T16:43:23-04:00",
       "requestedStartDate": "2017-10-12T16:42:23-04:00",
       "notificationContact": "[email protected]",
       "relatedParty": [{
           "role": "customer",
           "id": "userA",
           "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA"
```

```
       }, {
           "role": "seller",
           "id": "userB",
           "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userB"
       }],
       "orderItem": [{
           "id": "1",
           "action": "add",
           "state": "Acknowledged",
           "billingAccount": [{
               "id": "5",
               "href":
"https://URL_API/DSBillingManagement/api/billingManagement/v2/billingAccount/5"
           }],
           "productOffering": {
               "id": "42",
               "href": "http: //serverlocation:
port/catalogManagement/productOffering/42"
           },
           "product": {
               "productPrice": {
                   "name": "Monthly Price",
                   "description": "monthlyprice",
                   "priceType": "recurring",
                   "recurringChargePeriod": "monthly",
                   "unitOfMeasure": "",
                   "price": {
                       "amount": "12",
                       "currency": "EUR"
                   }
               }
           },
           "signature": {
               "id": "132",
               "href":
"https://URL_API/DSProductORdering/api/agreementManagement/v2/132",
               "parameter": "secp256r1",
               "algorithm": "SHA256withECDSA",
               "publicKey":
"04a01532a3c0900053de60fbefefcca58793301598d308b41e6f4e364e388c2711bef432c599148c9414
3d4ff46c2cb73e3e6a41d7eef23c047ea11e60667de425",
               "value":
"30450221009c7ccbcab30752a40407f4192e954f4e4401c02b22c69b0ac01a239e3a113b3002207beb24
67a931ca356061d6ca576aec530c009754fafcdea71acbe08348dedaa3"
           }
       }]
}]
```

Create Order [POST]

- Request (application/json)
    - Headers
    - Authorization: Bearer YOUR_OAUTH2_TOKEN

- Body:

```
{
      "externalId": "CustomerId",
      "description": "Observations of the air quality in Manchester",
      "state": "InProgress",
      "requestedStartDate": "2017-10-12T16:42:23-04:00",
      "notificationContact": "[email protected]",
      "relatedParty": [{
          "role": "customer",
          "id": "userA",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA"
      }, {
          "role": "seller",
          "id": "userB",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userB"
      }],
      "orderItem": {
          "id": "1",
          "action": "add",
          "state": "Acknowledged",
          "billingAccount": [{
              "id": "5",
              "href":
"https://URL_API/DSBillingManagement/api/billingManagement/v2/billingAccount/5
"
          }],
          "productOffering": {
              "id": "42",
              "href": "http: //serverlocation:
port/catalogManagement/productOffering/42"
          },
          "product": {
              "productPrice": {
                  "name": "Monthly payment",
                  "description": "A monthly payment price model",
                  "priceType": "recurring",
                  "recurringChargePeriod": "monthly",
                  "unitOfMeasure": "",
                  "price": {
                      "amount": "12",
                      "currency": "EUR"
                  }
              }
          },
          "signature": {
              "id": "132",
              "href":
"https://URL_API/DSProductORdering/api/agreementManagement/v2/132",
              "parameter": "secp256r1",
              "algorithm": "SHA256withECDSA",
              "publicKey":
"04a01532a3c0900053de60fbefefcca58793301598d308b41e6f4e364e388c2711bef432c5991
48c94143d4ff46c2cb73e3e6a41d7eef23c047ea11e60667de425",
              "value":
"30450221009c7ccbcab30752a40407f4192e954f4e4401c02b22c69b0ac01a239e3a113b30022
07beb2467a931ca356061d6ca576aec530c009754fafcdea71acbe08348dedaa3"
          }
      }
}
```

- Response 201 (application/json):

```
{
      "id": "42",
      "href":
"https://URL_API/DSProductOrdering/api/productOrdering/v2/productOrder/42",
      "externalId": "CustomerId",
      "description": "Observations of the air quality in Manchester",
      "state": "InProgress",
      "orderDate": "2017-10-12T16:42:23-04:00",
      "completionDate": "2017-10-12T16:43:23-04:00",
      "requestedStartDate": "2017-10-12T16:42:23-04:00",
      "notificationContact": "[email protected]",
      "relatedParty": [{
          "role": "customer",
          "id": "userA",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA"
      }, {
          "role": "seller",
          "id": "userB",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userB"
      }],
      "orderItem": [{
          "id": "1",
          "action": "add",
          "state": "Acknowledged",
          "billingAccount": [{
              "id": "5",
              "href":
"https://URL_API/DSBillingManagement/api/billingManagement/v2/billingAccount/5"
          }],
          "productOffering": {
              "id": "42",
              "href": "http: //serverlocation:
port/catalogManagement/productOffering/42"
          },
          "product": {
              "productPrice": {
                  "name": "Monthly Price",
                  "description": "monthlyprice",
                  "priceType": "recurring",
                  "recurringChargePeriod": "monthly",
                  "unitOfMeasure": "",
                  "price": {
                      "amount": "12",
                      "currency": "EUR"
                  }
```

```
                }
            },
            "signature": {
                "id": "132",
                "href":
"https://URL_API/DSProductORdering/api/agreementManagement/v2/132",
                "parameter": "secp256r1",
                "algorithm": "SHA256withECDSA",
                "publicKey":
"04a01532a3c0900053de60fbefefcca58793301598d308b41e6f4e364e388c2711bef432c599148c9414
3d4ff46c2cb73e3e6a41d7eef23c047ea11e60667de425",
                "value":
"30450221009c7ccbcab30752a40407f4192e954f4e4401c02b22c69b0ac01a239e3a113b3002207beb24
67a931ca356061d6ca576aec530c009754fafcdea71acbe08348dedaa3"
            }
        }]
}
```

Order Entry [/DSProductOrdering/api/productOrdering/v2/productOrder/{id}]

Get Order [GET]

- Parameters
  - id: 1 - Id of the order to be retrieved
- Request
  - Headers
  - Authorization: Bearer YOUR_OAUTH2_TOKEN

- Response 200 (application/json):

```
{
      "id": "42",
      "href":
"https://URL_API/DSProductOrdering/api/productOrdering/v2/productOrder/42",
      "externalId": "CustomerId",
      "description": "Observations of the air quality in Manchester",
      "state": "InProgress",
      "orderDate": "2017-10-12T16:42:23-04:00",
      "completionDate": "2017-10-12T16:43:23-04:00",
      "requestedStartDate": "2017-10-12T16:42:23-04:00",
      "notificationContact": "[email protected]",
      "relatedParty": [{
          "role": "customer",
          "id": "userA",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA"
      }, {
          "role": "seller",
          "id": "userB",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userB"
      }],
      "orderItem": [{
          "id": "1",
          "action": "add",
          "state": "Acknowledged",
          "billingAccount": [{
              "id": "5",
              "href":
"https://URL_API/DSBillingManagement/api/billingManagement/v2/billingAccount/5"
          }],
          "productOffering": {
              "id": "42",
              "href": "http: //serverlocation:
port/catalogManagement/productOffering/42"
          },
          "product": {
              "productPrice": {
                  "name": "Monthly Price",
                  "description": "monthlyprice",
                  "priceType": "recurring",
                  "recurringChargePeriod": "monthly",
                  "unitOfMeasure": "",
                  "price": {
                      "amount": "12",
                      "currency": "EUR"
                  }
              }
          },
          "signature": {
              "id": "132",
              "href":
"https://URL_API/DSProductORdering/api/agreementManagement/v2/132",
              "parameter": "secp256r1",
              "algorithm": "SHA256withECDSA",
              "publicKey":
"04a01532a3c0900053de60fbefefcca58793301598d308b41e6f4e364e388c2711bef432c599148c9414
3d4ff46c2cb73e3e6a41d7eef23c047ea11e60667de425",
              "value":
"30450221009c7ccbcab30752a40407f4192e954f4e4401c02b22c69b0ac01a239e3a113b3002207beb24
67a931ca356061d6ca576aec530c009754fafcdea71acbe08348dedaa3"
          }
      }]
}
```

Update Order [PATCH]

- Parameters
  - id: 1 - Id of the order to be updated
- Request - Partial update of the order, only the fields to be updated need to be provided - (application/json)
  - Headers
  - Authorization: Bearer YOUR_OAUTH2_TOKEN

  - Body:

```
{
      "state": "Completed"
}
```

- Response 200 (application/json)

-
  ```
      {
        "id": "42",
        "href":
  "https://URL_API/DSProductOrdering/api/productOrdering/v2/productOrder/42",
        "externalId": "CustomerId",
        "description": "Observations of the air quality in Manchester",
        "state": "Completed",
        "orderDate": "2017-10-12T16:42:23-04:00",
        "completionDate": "2017-10-12T16:43:23-04:00",
        "requestedStartDate": "2017-10-12T16:42:23-04:00",
        "notificationContact": "[email protected]",
        "relatedParty": [{
            "role": "customer",
            "id": "userA",
            "href":
  "https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA"
        }, {
            "role": "seller",
            "id": "userB",
            "href":
  "https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userB"
        }],
        "orderItem": [{
            "id": "1",
            "action": "add",
            "state": "Acknowledged",
            "billingAccount": [{
                "id": "5",
                "href":
  "https://URL_API/DSBillingManagement/api/billingManagement/v2/billingAccount/5
  "
            }],
            "productOffering": {
                "id": "42",
                "href": "http: //serverlocation:
  port/catalogManagement/productOffering/42"
            },
            "product": {
                "productPrice": {
                    "name": "Monthly Price",
                    "description": "monthlyprice",
                    "priceType": "recurring",
                    "recurringChargePeriod": "monthly",
                    "unitOfMeasure": "",
                    "price": {
                        "amount": "12",
                        "currency": "EUR"
                    }
                }
            },
            "signature": {
  ```

```
              "id": "132",
              "href":
"https://URL_API/DSProductORdering/api/agreementManagement/v2/132",
              "parameter": "secp256r1",
              "algorithm": "SHA256withECDSA",
              "publicKey":
"04a01532a3c0900053de60fbefefcca58793301598d308b41e6f4e364e388c2711bef432c5991
48c94143d4ff46c2cb73e3e6a41d7eef23c047ea11e60667de425",
              "value":
"30450221009c7ccbcab30752a40407f4192e954f4e4401c02b22c69b0ac01a239e3a113b30022
07beb2467a931ca356061d6ca576aec530c009754fafcdea71acbe08348dedaa3"
          }
      }]
}
```

Group 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.
The License Specification Management API uses the following fields:

- **id** - Unique identifier of the license specification
- **href** - URL pointing to the license specification info
- **lastUpdate** - Date and time of the last update
- **name** - Name of the license specification
- **description** - Narrative text that explains the body of the license specification
- **relatedParty** - List of parties and its roles related to the current license specification. For each party, it is included the id and the href as described in the Party Management section. Additionally, it is included a *role* field specifying the role of the user in the current license specification
- **attachment** - List of request attachments, such as pdf, documents, which help describing the license specification. Each attachment contains the following fields:
    - **type** - Attachment type, such as a document, etc.
    - **url** - URL pointing to the attachment itself. Note that if the attachment has been uploaded to the system using the asset management API, you can use the URL returned by the upload task in the *Location*header
- **exclusivity** - Whether the license specification grants the exclusive right to use the data linked to the current license specification
- **sector** - List of sectors where the license specification applies. Each item of the list contains a value and an exception field
- **region** - List of territories where the license specification applies. Each item of the list contains a value and an exception field
- **timeFrame** - Time frame in which the license specification remains valid
- **purpose** - List of allowed purposes (e.g., research, commercial, etc.) when using the related data linked to the currenti license specification. Each item of the list contains a value and an exception field
- **transferability** - Expression of the right to sublicense

License Specification Collection

[/DSLAlicenseManagement/api/licenseManagement/v1/licenseSpecification{?offset}{?size}{?id}{?relatedParty.id}{?exclusivity}{?sector}{?region}{?purpose}{?transferability}]

List License Specifications [GET]

- Parameters
  - offset: 1 (optional) - Optional parameter used to specify the first element to be returned
  - size: 10 (optional) - Optional parameter used to limit the number of elements returned
  - id: 22 (optional)- Optional parameter used to filter the returned license specifications by its ID
  - relatedParty.id: userA (optional) - Optional parameter used to filter the returned license specifications by owner
  - exclusivity: nonExclusive (optional) - Optional parameter used to filter the returned license specifications by exclusivity
  - sector: all (optional) - Optional parameter used to filter the returned license specifications by sector
  - region: all (optional) - Optional parameter used to filter the returned license specifications by region
  - purpose: all (optional) - Optional parameter used to filter the returned license specifications by purpose
  - transferability: sublicensingRight (optional) - Optional parameter used to filter the returned license specifications by transferability
- Request
  - Headers
  - Authorization: Bearer YOUR_OAUTH2_TOKEN

- Response 200 (application/json):

```
[
      {
          "id": "22",
          "href":
"https://URL_API/DSLAlicenseManagement/api/licenseManagement/v1/licenseSpecification/
22",
          "lastUpdate": "2017-10-19T16:42:23.0Z",
          "name": "Customised license A",
          "description": "This license lets consumers distribute, exploit, and build
upon, even commercially, as long as they credit the owner of the original data",
          "relatedParty": [{
              "id": "userA",
              "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
              "role": "Owner"
          }],
          "attachment": [{
              "type": "pdf",
              "url": "https://URL_API/media/document.pdf"
          }],
          "exclusivity": "nonExclusive",
          "sector": "all",
          "region": "all",
          "timeFrame": "2020-12-31T00:00:00.0Z",
          "purpose": "all",
          "transferability": "sublicenseRight"
      }
]
```

Create License Specification [POST]

- Request (application/json)
    - Headers
    - Authorization: Bearer YOUR_OAUTH2_TOKEN

    - Body:

```
{
      "name": "Customised license A",
      "description": "This license lets consumers distribute, exploit, and
build upon, even commercially, as long as they credit the owner of the
original data",
      "relatedParty": [{
          "id": "userA",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
          "role": "Owner"
      }],
      "attachment": [{
          "type": "pdf",
          "url": "https://URL_API/media/document.pdf"
      }],
      "exclusivity": "nonExclusive",
      "sector": "all",
      "region": "all",
      "timeFrame": "2020-12-31T00:00:00.0Z",
      "purpose": "all",
      "transferability": "sublicenseRight"
}
```

- Response 201 (application/json):

```
{
      "id": "22",
      "href":
"https://URL_API/DSLAlicenseManagement/api/licenseManagement/v1/licenseSpecification/
22",
      "lastUpdate": "2017-10-19T16:42:23.0Z",
      "name": "Customised license A",
      "description": "This license lets consumers distribute, exploit, and build
upon, even commercially, as long as they credit the owner of the original data",
      "relatedParty": [{
          "id": "userA",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
          "role": "Owner"
      }],
      "attachment": [{
          "type": "pdf",
          "url": "https://URL_API/media/document.pdf"
      }],
      "exclusivity": "nonExclusive",
      "sector": "all",
      "region": "all",
      "timeFrame": "2020-12-31T00:00:00.0Z",
      "purpose": "all",
      "transferability": "sublicenseRight"
}
```

License Specification Entry

[/DSLAlicenseManagement/api/licenseManagement/v1/licenseSpecification/{id}]

Get License Specification [GET]

- Parameters
  - id: 22 - Id of the license specification to be retrieved
- Request
  - Headers
  - Authorization: Bearer YOUR_OAUTH2_TOKEN

- Response 200 (application/json):

```
{
      "id": "22",
      "href":
"https://URL_API/DSLAlicenseManagement/api/licenseManagement/v1/licenseSpecification/
22",
      "lastUpdate": "2017-10-19T16:42:23.0Z",
      "name": "Customised license A",
      "description": "This license lets consumers distribute, exploit, and build
upon, even commercially, as long as they credit the owner of the original data",
      "relatedParty": [{
              "id": "userA",
              "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
              "role": "Owner"
      }],
      "attachment": [{
              "type": "pdf",
              "url": "https://URL_API/media/document.pdf"
      }],
      "exclusivity": "nonExclusive",
      "sector": "all",
      "region": "all",
      "timeFrame": "2020-12-31T00:00:00.0Z",
      "purpose": "all",
      "transferability": "sublicenseRight"
}
```

Update License Specification [PATCH]

- Parameters
  - id: 1 - Id of the license specification to be updated

- Request - Partial update of the license specification, only the fields to be updated need to be provided - (application/json)
    - Headers
    - Authorization: Bearer YOUR_OAUTH2_TOKEN

    - Body:

```
{
      "timeFrame": "2022-12-31T00:00:00.0Z",
      "region": "UK"
}
```

- Response 200 (application/json):

```
{
          "id": "22",
          "href":
"https://URL_API/DSLAlicenseManagement/api/licenseManagement/v1/licenseSpecification/
22",
          "lastUpdate": "2017-10-19T16:42:23.0Z",
          "name": "Customised license A",
          "description": "This license lets consumers distribute, exploit, and build
upon, even commercially, as long as they credit the owner of the original data",
          "relatedParty": [
              {
                  "id": "userA",
                  "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
                  "role": "Owner"
              }
          ],
          "attachment": [
              {
                  "type": "pdf",
                  "url": "https://URL_API/media/document.pdf"
              }
          ],
          "exclusivity": "nonExclusive",
          "sector": "all",
          "region": "UK",
          "timeFrame": "2022-12-31T00:00:00.0Z",
          "purpose": "all",
          "transferability": "sublicenseRight"
}
```

Group 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.
The SLA Specification Management API uses the following fields:

- **id** - Unique identifier of the SLA specification
- **href** - URL pointing to the SLA specification info
- **lastUpdate** - Date and time of the last update

- **name** - Name of the SLA specification
- **description** - Narrative text that explains the body of the SLA specification
- **relatedParty** - List of parties and its roles related to the current SLA specification. For each party, it is included the id and the href as described in the Party Management section. Additionally, it is included a *role*field specifying the role of the user in the current SLA specification
- **attribute** - List of object describing the attributes of the SLA specification. It contains the following fields:
    - **name** - Name of the attribute
    - **description** - Detailed description of the attribute
    - **valueType** - The kind of value the attribute could have. Valid values are *String* and *Number*
    - **unitOfMeasure** - Could be minutes, MB, etc. This field is only used when the type is *Number*
    - **value** - Value of the characteristic when it is a discrete value. If this field is included, valueFrom and valueTo must be empty
    - **valueFrom** - Starting value of the attribute when it is a range. If this field is included, valueTo must be also included and value must be empty
    - **valueTo** - Ending value of the attribute when it is a range. If this field is included, valueFrom must be also included and value must be empty

SLA Specification Collection

[/DSLAlicenseManagement/api/SLAManagement/v1/SLASpecification{?offset}{?size}{?id}{?relatedParty.id}]

List SLA Specifications [GET]

- Parameters
    - offset: 1 (optional) - Optional parameter used to specify the first element to be returned
    - size: 10 (optional) - Optional parameter used to limit the number of elements returned
    - id: 52 (optional)- Optional parameter used to filter the returned license specifications by its ID
    - relatedParty.id: userA (optional) - Optional parameter used to filter the returned license specifications by owner
- Request
    - Headers
    - Authorization: Bearer YOUR_OAUTH2_TOKEN

- Response 200 (application/json):

```
[
      {
          "id": "52",
          "href":
"https://URL_API/DSLAlicenseManagement/api/SLAManagement/v1/SLASpecification/52",
          "lastUpdate": "2017-10-19T16:42:23.0Z",
          "name": "SLA response time",
          "description": "This Service Level Agreement defines response time and
update frequency",
          "relatedParty": [{
              "id": "userA",
              "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
              "role": "Owner"
          }],
          "attribute": [{
              "name": "response time",
              "description": "Specification of the response time provided",
              "valueType": "number",
              "unitOfMeasure": "ms",
              "valueFrom": 200,
              "valueTo": 700
          },{
              "name": "frequency",
              "description": "Specification of the update frequency provided",
              "valueType": "number",
              "unitOfMeasure": "pkt/day",
              "value": 10
          }]
      }
]
```

Create SLA Specification [POST]

- Request (application/json)
    - Headers
    - Authorization: Bearer YOUR_OAUTH2_TOKEN

    - Body:

```
{
      "name": "SLA response time",
      "description": "This Service Level Agreement defines response time and
update frequency",
      "relatedParty": [{
          "id": "userA",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
          "role": "Owner"
      }],
      "attribute": [{
          "name": "response time",
          "description": "Specification of the response time provided",
          "valueType": "number",
          "unitOfMeasure": "ms",
          "valueFrom": 200,
          "valueTo": 700
      },{
          "name": "frequency",
          "description": "Specification of the update frequency provided",
          "valueType": "number",
          "unitOfMeasure": "pkt/day",
          "value": 10
      }]
}
```

- Response 201 (application/json):

```
{
      "id": "52",
      "href":
"https://URL_API/DSLAlicenseManagement/api/SLAManagement/v1/SLASpecification/52",
      "lastUpdate": "2017-10-19T16:42:23.0Z",
      "name": "SLA response time",
      "description": "This Service Level Agreement defines response time and update
frequency",
      "relatedParty": [{
          "id": "userA",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
          "role": "Owner"
      }],
      "attribute": [{
          "name": "response time",
          "description": "Specification of the response time provided",
          "valueType": "number",
          "unitOfMeasure": "ms",
          "valueFrom": 200,
          "valueTo": 700
      },{
          "name": "frequency",
          "description": "Specification of the update frequency provided",
          "valueType": "number",
          "unitOfMeasure": "pkt/day",
          "value": 10
      }]
}
```

SLA Specification Entry

[/DSLAlicenseManagement/api/SLAManagement/v1/SLASpecification/{id}]

Get SLA Specification [GET]

- Parameters
  - id: 52 - Id of the SLA specification to be retrieved
- Request
  - Headers
  - Authorization: Bearer YOUR_OAUTH2_TOKEN

- Response 200 (application/json):

```
{
      "id": "52",
      "href":
"https://URL_API/DSLAlicenseManagement/api/SLAManagement/v1/SLASpecification/52",
      "lastUpdate": "2017-10-19T16:42:23.0Z",
      "name": "SLA response time",
      "description": "This Service Level Agreement defines response time and update
frequency",
      "relatedParty": [{
          "id": "userA",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
          "role": "Owner"
      }],
      "attribute": [{
          "name": "response time",
          "description": "Specification of the response time provided",
          "valueType": "number",
          "unitOfMeasure": "ms",
          "valueFrom": 200,
          "valueTo": 700
      },{
          "name": "frequency",
          "description": "Specification of the update frequency provided",
          "valueType": "number",
          "unitOfMeasure": "pkt/day",
          "value": 10
      }]
}
```

Update SLA Specification [PATCH]

- Parameters
  - id: 52 - Id of the SLA specification to be updated
- Request - Partial update of the license specification, only the fields to be updated need to be provided - (application/json)
  - Headers
  - Authorization: Bearer YOUR_OAUTH2_TOKEN

  - Body:

```
{
      "attribute": [{
          "name": "response time",
          "description": "Specification of the response time provided",
          "valueType": "number",
          "unitOfMeasure": "ms",
          "valueFrom": 400,
          "valueTo": 900
      },{
          "name": "frequency",
          "description": "Specification of the update frequency provided",
          "valueType": "number",
          "unitOfMeasure": "pkt/day",
          "value": 20
      }]
}
```

- Response 200 (application/json):

```
{
      "id": "52",
      "href":
"https://URL_API/DSLAlicenseManagement/api/SLAManagement/v1/SLASpecification/52",
      "lastUpdate": "2017-10-19T16:42:23.0Z",
      "name": "SLA response time",
      "description": "This Service Level Agreement defines response time and update
frequency",
      "relatedParty": [{
          "id": "userA",
          "href":
"https://URL_API/DSPartyManagement/api/partyManagement/v2/individual/userA",
          "role": "Owner"
      }],
      "attribute": [{
          "name": "response time",
          "description": "Specification of the response time provided",
          "valueType": "number",
          "unitOfMeasure": "ms",
          "valueFrom": 400,
          "valueTo": 900
      },{
          "name": "frequency",
          "description": "Specification of the update frequency provided",
          "valueType": "number",
          "unitOfMeasure": "pkt/day",
          "value": 20
      }]
}
```

Group Federation Management API

API for the management of federation functionalities.
The Federation Management API uses the following fields:

- **url** - Url of the external data catalog from which import metadata
- **type** - Type of the external data catalog from which import metadata (e.g., CKAN, Hypercat)
- **externalId** - List of IDs of dataset retrieved from the external data catalog
- **title** - List of title of dataset retrieved from the external data catalog
- **description** - List of descriptions of dataset retrieved from the external data catalog
- **metadata** - List of metadata of the dataset retrieved from the external data catalog
    - **name** - Name of the metadata field
    - **value** - Value of the metadata field

Import Data Catalog Collection

[/DSFederationManagement/api/federationManagement/v1/importDataCatalog{?url}{?type}]

List Dataset [GET]

- Parameters
    - url: http://data.london.gov.uk/api/3/action/package_list - Parameter used to specify the url of the external dataset from which import metadata
    - type: CKAN - Parameter used to specify the type of the external dataset from which import metadata
- Request
    - Headers
    - Authorization: Bearer YOUR_OAUTH2_TOKEN

- Response 200 (application/json):

```
[{
    "externalId": "cd15b07e-7e95-4e29-99c7-1123aa412a91",
    "title": "Schools and educational institutions air quality exposure data broken
down by parliamentary constituency",
      "description": "This is a dataset of schools and educational institutions air
quality exposure data broken down by parliamentary constituency based on the current
2013 data from the London Atmospheric Emissions Inventory",
      "metadata": [{
          "name": "maintainer",
          "value": ""
      },{
          "name": "license_title",
          "value": "UK Open Government Licence (OGL v3)"
      }]
  },{
      "exernalId": "21b7e97b-310f-4cde-be7d-bd17b38874b1",
      "title": "Incapacity Benefit Claimants, Borough",
      "description": "Incapacity Benefit",
      "metadata": [{
          "name": "maintainer",
          "value": "Opinion Research and General Statistics (GLA)"
      },{
          "name": "license_title",
          "value": "UK Open Government Licence (OGL v2)"
      }]
}]
```