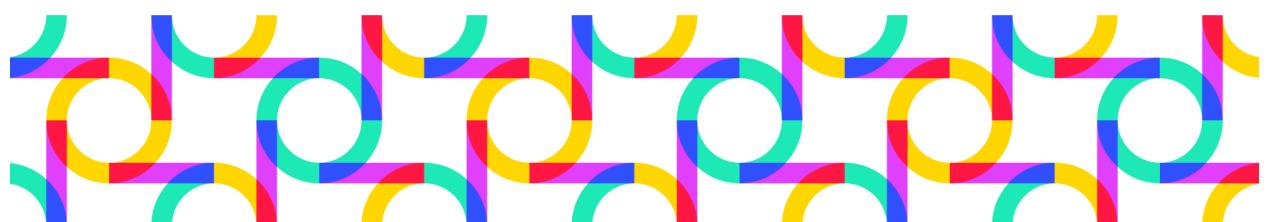




**Data Spaces for effective and trusted data sharing**

[i4Trust.org](http://i4Trust.org)

## **i4Trust Building Blocks (version 3.0)**



# Table of Contents

|  |           |
|--|-----------|
| <b>1 Introduction</b>                          | <b>5</b>  |
| 1.1 Executive Summary                          | 5         |
| 1.2 Intended audience                          | 5         |
| 1.3 Structure of the document                  | 6         |
| 1.4 Changes from previous version              | 7         |
| 1.5 Related documents and resources            | 7         |
| <b>2 Introduction to Data Spaces</b>           | <b>8</b>  |
| <b>3 About FIWARE</b>                          | <b>11</b> |
| 3.1 Introduction to FIWARE                     | 11        |
| 3.2 Vertical Smart Solutions powered by FIWARE | 15        |
| 3.3 Smart Organizations powered by FIWARE      | 17        |
| 3.4 Getting started with FIWARE                | 18        |
| <b>4 About iSHARE</b>                          | <b>20</b> |
| 4.1 Introduction to iSHARE                     | 20        |
| 4.2 A deeper look at the iSHARE standard       | 22        |
| 4.3 Logistics data space                       | 25        |
| 4.4 Getting Started with iSHARE                | 26        |
| <b>5 i4Trust Data Spaces</b>                   | <b>27</b> |
| 5.1 Overview                                   | 27        |
| 5.2 i4Trust Building Blocks                    | 29        |
| 5.2.1 Data interoperability                    | 29        |

|  |           |
|--|-----------|
| 5.2.2 Data sovereignty and Trust   | 32        |
| 5.2.3 Data value creation  | 34        |
| 5.2.4 Data Space Governance  | 36        |
| 5.2.4.1 What are Business Agreements?                                    | 36        |
| 5.2.4.2 What constitutes Operational Agreements?                         | 37        |
| 5.2.4.3 What are organisational agreements?                              | 38        |
| 5.3 i4Trust roles  | 39        |
| <b>6 Bringing the pieces together: Packet Delivery reference example</b> | <b>41</b> |
| 6.1 Overall description  | 41        |
| 6.2 Parties involved   | 43        |
| 6.2.1 Data Service Provider: Packet Delivery Company                     | 43        |
| 6.2.2 Data Service Consumer: HappyPets Inc.                              | 45        |
| 6.2.3 Data Service Consumer: NoCheaper Ltd                               | 46        |
| 6.2.4 i4Trust Marketplace  | 46        |
| 6.2.5 i4Trust Trust Provider (iSHARE Satellite)                          | 47        |
| 6.3 Deployment of components   | 48        |
| 6.4 Prerequisites  | 51        |
| 6.5 Create Offering  | 60        |
| 6.5.1 Sequence description (Packet Delivery Co.)                         | 60        |
| 6.6 Acquisition of Rights / Activation                                   | 68        |
| 6.6.1 Sequence description (Happy Pets Inc.)                             | 68        |
| 6.6.2 Sequence description (No Cheaper Ltd)                              | 73        |
| 6.7 Access to data service   | 75        |
| 6.7.1 Sequence description (Happy Pets Customer)                         | 75        |
| 6.7.2 Scenario: No Cheaper   | 101       |
| 6.8 Role-based access  | 102       |
| 6.8.1 Create Offering  | 103       |

|   |            |
|---|------------|
| 6.8.2 Acquisition of Rights / Activation                                | 104        |
| 6.8.3 Access to data service  | 107        |
| 6.9 Direct access to data service by client applications (M2M)          | 110        |
| 6.9.1 Create Offering   | 111        |
| 6.9.2 Acquisition of Rights / Activation                                | 112        |
| 6.9.3 Access to data service  | 113        |
| 6.10 Subscriptions and Notifications                                    | 117        |
| 6.10.1 Deployment and configuration of components                       | 118        |
| 6.10.2 Create Offering  | 121        |
| 6.10.3 Acquisition of rights / Activation                               | 121        |
| 6.10.4 Sending notifications  | 126        |
| <b>7 Bringing the pieces together: Smart Shepherd reference example</b> | <b>128</b> |
| 7.1 Overall description   | 128        |
| 7.2 Parties involved  | 131        |
| 7.2.1 AI Service Provider: Smart Shepherd Inc.                          | 131        |
| 7.2.2 AI Service Consumer: HappyCattle Co.                              | 132        |
| 7.2.3 Data Service Provider: Real Time Weather Ltd.                     | 133        |
| 7.3 Prerequisites   | 134        |
| 7.4 Create offering   | 134        |
| 7.5 Acquisition of Rights / Activation                                  | 135        |
| 7.6 Usage of AI service   | 150        |

# 1 Introduction

## 1.1 Executive Summary

The main goal of i4Trust is to boost the development of innovative services around new data value chains. i4Trust helps to achieve this by providing the right tools, education, coaching and initial funding for the creation of Data Spaces enabling trustworthy and effective data sharing. Ecosystems of collaborating SMEs and supporting DIHs will emerge in a sustainable way around such Data Spaces.

This document is intended to be distributed to CTOs, architects and developers of DIHs and SMEs as a description of the different building blocks to be used for creation of i4Trust data spaces and how they work together. These building blocks result from the integration of open, standard-based, CEF-compatible components coming from the iSHARE Scheme and FIWARE. The document comes at the end with a rather detailed description of a Proof of Concept for a specific use case that illustrates how the different Building Blocks work together.

## 1.2 Intended audience

This document is intended for technical experts of DIHs and SMEs, namely CTOs, architects and developers.

They should have experience on:

- REST - JSON technologies
- Security Standards: OpenID Connect, OAuth2, XACML, PKI, JWT/S
- NGSIv2/NGSI-LD API and FIWARE Context Broker technology

Deployment instructions are provided in the form of Kubernetes recipes, therefore it is recommended to have hands-on experience on:

- Container Technologies (e.g., Docker)
- Kubernetes
- Microservice architectures and technologies

## 1.3 Structure of the document

This document is divided into 6 sections:

Section 1: Introduction - Summary of this document

Section 2: Introduction to Data Spaces - This section gives a brief introduction to data spaces in general. It explains the different technology and governance building blocks of a data space and outlines its ecosystem.

Section 3: About FIWARE - In this section, an introduction to FIWARE is given. It outlines the framework FIWARE brings, and explains reference architectures for Vertical Smart Solutions and Smart Organizations. This is concluded by a collection of resources to get started with FIWARE.

Section 4: About iSHARE - This section gives an introduction to iSHARE. The iSHARE scheme and framework is explained and a reference example of a logistics use case is depicted. Finally, it provides suggestions on how to get started with iSHARE.

Section 5: i4Trust Data Spaces - This section provides details about the building blocks required for the creation of i4Trust Data Spaces. It gives an overview of the different FIWARE and iSHARE building blocks and explains how these can be combined to materialize such data spaces. The section is concluded by an overview of the different roles of actors participating in an i4Trust Data Space.

Section 6: Bringing the pieces together - This section describes an i4Trust use case built with a Proof-of-Concept (PoC) which aims to showcase a data space for trusted data sharing among different parties. The PoC incorporates the different technology building blocks of i4Trust Data Spaces on the basis of an use case, where a fictitious packet delivery company is offering digital services on the marketplace to certain

online retailers. The offered service allows to view and change specific attributes of a packet delivery order. On the marketplace, the retailers can acquire access to the packet delivery system, and delegate the access to its customers.

## 1.4 Changes from previous version

This document incorporates the following changes compared to v2.0:

- Added a more representative reference example of an AI Service provider "Smart Shepherd" (chapter 7), showcasing the full potential of the i4Trust platform and giving more insights to users on how to use it in a data space involving several service providers and consumers.

This new use case involves an AI service provider providing a service for the calculation of predicted animal activity. Furthermore, there is a data service provider for weather measurements, whose data is used to enrich the dataset used for the prediction calculations by the AI service provider. Finally, there is one example of a service consumer, representing a majority of farmers that want to acquire access to the AI service.

Considering that there is also a training of the ML models at the AI service provider, these consumers also could provide data in order to improve the prediction calculations.

- Added description of the iSHARE Satellite ([section 6.2.5](#)).

## 1.5 Related documents and resources

- GitHub repository: <https://github.com/i4Trust>
- Training material on YouTube:  
<https://www.youtube.com/channel/UCScaleWPmKfs-VZrurDLFg/playlists>
- Slides of i4Trust Train the Trainers program:  
<https://www.slideshare.net/FI-WARE/clipboards/i4trust-train-the-trainer-program>

## 2 Introduction to Data Spaces

A **data space** can be defined as a decentralized data ecosystem built around commonly agreed building blocks enabling an effective and trusted sharing of data among participants. From a technical perspective, a number of **technology building blocks** are required to materialize data spaces ensuring:

- **Data interoperability** - Data spaces should provide a solid framework for an efficient exchange of data among participants, supporting full decoupling of data providers and consumers. This requires the adoption of a “common lingua” every participant uses, materialized in the adoption of common APIs for the data exchange, and the definition of common data models. Common mechanisms for traceability of data exchange transactions and data provenance, are also required.
- **Data Sovereignty and trust** - Data spaces should bring technical means for guaranteeing that participants in a data space can trust each other and exercise sovereignty over data they share. This requires the adoption of common standards for managing the identity of participants, the verification of their truthfulness and the enforcement of policies agreed upon data access and usage control.
- **Data value creation** - Data spaces should provide support for the creation of multi-sided markets where participants can generate value out of sharing data (i.e., creating data value chains). This requires the adoption of common mechanisms enabling the definition of terms and conditions (including pricing) linked to data offerings, the publication and discovery of such offerings and the management of all the necessary steps supporting the lifecycle of contracts that are established when a given participant acquires the rights to access and use data.

Besides the adoption of a common technology foundation, data spaces also require **governance**, that is the adoption of a number of business, operational and organizational agreements among participants. Business agreements, for example, specify what kind of terms and conditions can regulate the sharing of data between participants and the legal framework supporting contracts established through the

data space. Operational agreements, on the other hand, regulate policies that have to be enforced during data space operation like, for example, compliance with GDPR (General Data Protection Regulation) or the 2nd Payment Services Directive (PSD2) in the finance sector. They may also comprise the definition of tools that operators of cloud infrastructures or global services supporting data spaces must implement, enabling auditing of certain processes or the adoption of cyber-security practices. Last but not least, organizational agreements establish the governance bodies (very much like ICANN for the Internet). They deal with the identification of concrete specifications that products implementing technology building blocks in a data space should comply with, as well as the business and operational agreements to be adopted. The complete taxonomy of building blocks required for creating data spaces is illustrated in Figure 2.1.

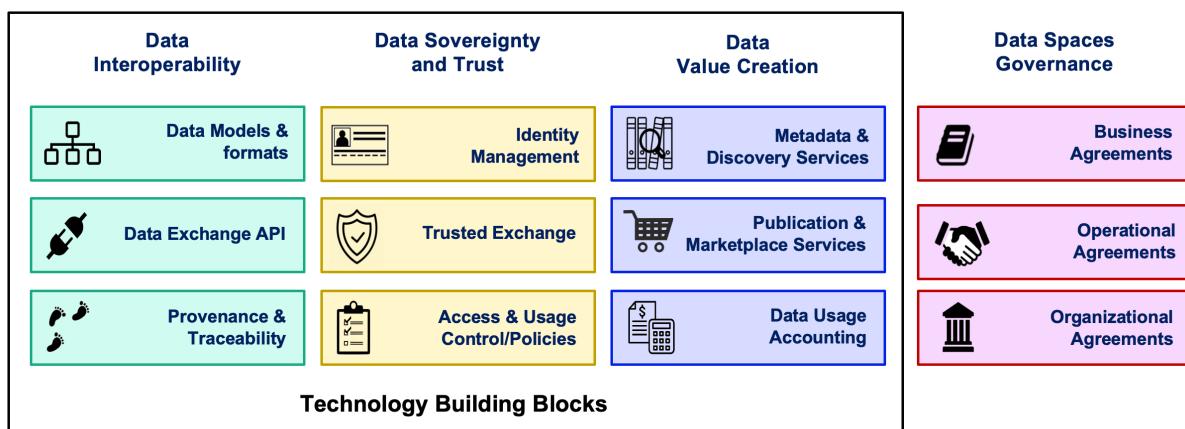


Figure 2.1 - Building blocks in a Data Space

Sharing of data within a given data space should not be limited to a single domain. This would severely limit the creation of new innovative services since individuals and organizations usually act in multiple domains at the same time and many opportunities will flourish when data generated within organizations operating in certain domain (management of traffic in cities, for example) is shared for its exploitation in processes relevant to other domains (continuing with the example, logistics). Therefore, technology building blocks for data spaces must be domain-agnostic. On the other hand, they should rely on open standards, allowing multiple infrastructure and global service providers to emerge and support data spaces, without getting locked in any particular provider. Given this, while making things work in living labs and pilots is relatively easy, the main challenge towards definition of successful data spaces is the decision of what concrete standards and design principles are adopted, since they have to be accepted by all participants.

A data space ecosystem built around the mentioned building blocks basically involves participants playing the role of data consumers and/or data providers on one hand and a number of actors playing the role of service providers at overall data space level on the other hand. Service providers include providers of trust authority services as well as providers of service and data discovery, brokering, logging and trading services enabling data-driven multi-side markets. Data space participants may implement Identity Provider and Authorization Registry roles themselves or may rely on independent service providers supporting such roles. As mentioned earlier, certain agreements need to be defined binding participants and service providers and this typically implies establishing some data space governance bodies. Figure below summarize the different roles played by actors involved in a data space ecosystem

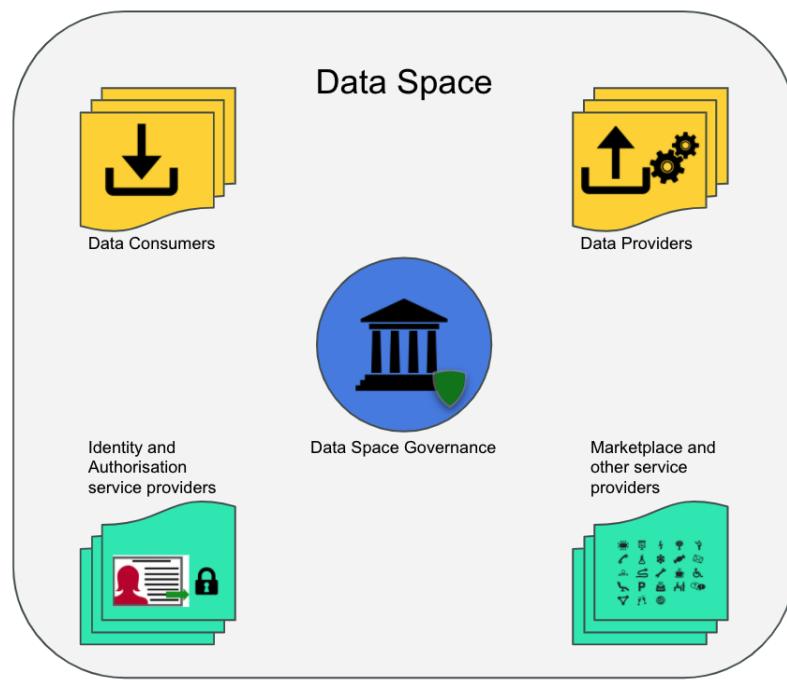


Figure 2.2 - Data space ecosystem

## 3 About FIWARE

### 3.1 Introduction to FIWARE

FIWARE<sup>1</sup> was created with the ultimate goal of creating an open sustainable ecosystem around public, royalty-free and implementation-driven software platform standards easing the development of smart solutions and supporting organizations in their transition into smart organizations. From a technical perspective, FIWARE brings a curated framework of open source software components which can be assembled together and combined with other third-party platform components to build platforms easing the development of smart solutions and smart organizations in multiple application domains: cities, manufacturing, utilities, agrifood, etc. Since creation of the FIWARE Foundation in late 2016, a vibrant FIWARE Community has been formed with a true worldwide dimension, comprising more than 90 member organizations<sup>2</sup>, including large corporations, SMEs, technology centres and universities, and hundreds of individual members. Parallel to this growth in membership, the number of organizations adopting FIWARE has also not stopped growing.

---

<sup>1</sup> <https://www.fiware.org/>

<sup>2</sup> <https://www.fiware.org/foundation/members/>

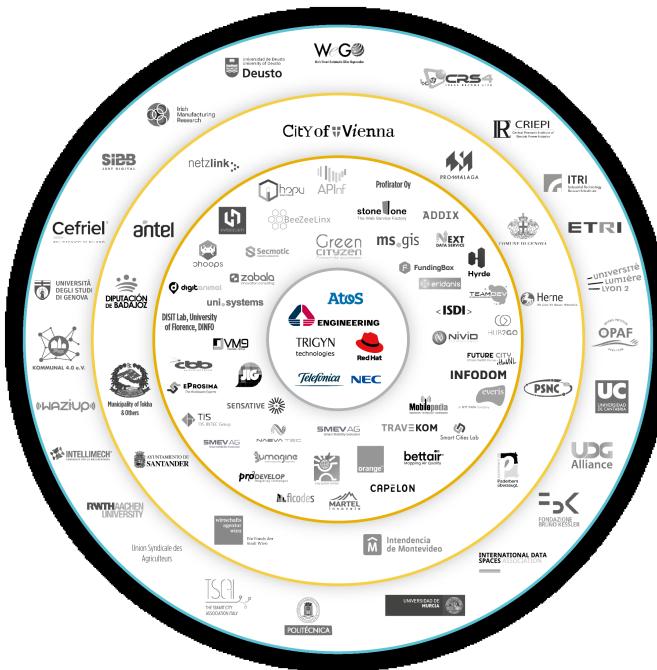


Figure 3.1 - Current member organizations in the FIWARE Community

**Any software architecture “powered by FIWARE” gravitates around management of a Digital Twin data representation of the real world.** This Digital Twin data representation is built based on information gathered from many different sources, including sensors, cameras, information systems, social networks, end users through mobile devices, etc. It is constantly maintained and accessible in near real-time (“right-time” is the term also often used, reflecting that the interval between the instants of time at which some data is gathered and made accessible is enough short to allow a proper reaction). Applications constantly process and analyze this data (not only current values but also history generated over time) in order to automate certain tasks or bring support to smart decisions by end users. The collection of all digital twins modelling the real world that is managed is also referred to as **Context** and the data associated with attributes of digital twins is also referred to as **context information**.

In FIWARE, a digital twin is an entity which digitally represents a real-world physical asset (e.g. a bus in a city, a milling machine in a factory) or a concept (e.g., a weather forecast, a product order). Each digital twin:

- is universally identified with an URI (Universal Resource Identifier),
- belongs to a well-known type (e.g., the Bus type, or the Room type) also universally identified by an URI, and
- is characterized by several attributes which in turn are classified as:
  - properties holding data (e.g., the “current speed” of a Bus, or “max temperature” in a Room) or
  - relationships, each holding an URI identifying a third digital twin entity the given entity is linked to (e.g., the concrete Building where a concrete Room is located).

Attributes of a digital twin may vary ranging from attributes that are quite static (e.g., the “license plate” of a Bus), to attributes that change very dynamically (e.g., the “speed” or “number of passengers” in a Bus) to attributes which still change but not that often (e.g., the “driver” in a Bus which may change twice a day). Very important, the attributes of a digital twin are not only limited to observable data but also inferred data. Thus, for example, the digital twin of a Street may not only have an attribute “current traffic”, which may be automatically measured through sensors or cameras, but an attribute “forecasted traffic in 30 minutes” which might be calculated based on AI algorithms that keep the value of this attribute updated based on current traffic data, other relevant data impacting traffic (e.g., current weather observed and forecasted, schedule of events nearby, etc) and historical information about traffic in the given street. Therefore, the Digital Twin data representation of the world that is managed in a “powered by FIWARE” architecture is expected to contain all the information needed by smart applications, not only measurable data but also other augmented insights and knowledge acquired over time.

A Digital Twin approach provides the basis for data integration at different levels, as illustrated in Figure 3.2:

- Within a **vertical smart solution**, solving how main building blocks within the architecture can be integrated.
- Within a **smart organization**, bringing support to the integration of the different systems within a smart organization following a system of systems approach.
- Within a **smart data space**, establishing the basic “common lingua” that systems linked to the different organizations speak and understand.

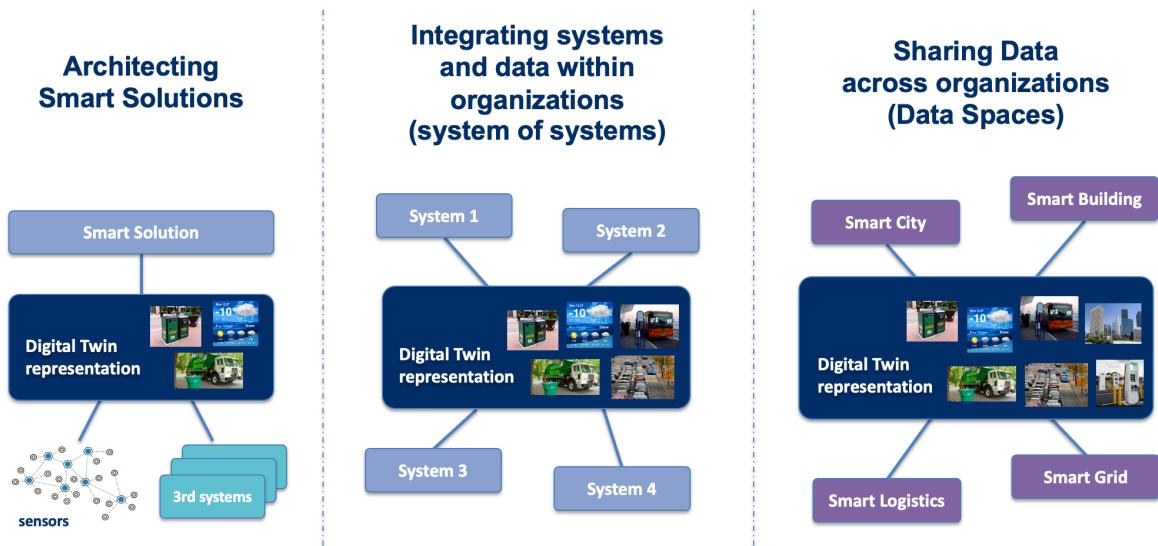


Figure 3.2 - Levels of Integration supported following a Digital Twin approach

Two critical elements need to be standardized in order to support an effective data integration at these three levels: the API to get access to digital twin data and the data models describing the attributes and semantics associated with the different types of digital twins being considered. The FIWARE Community has driven and continues to drive standardization at both fronts:

- The **NGSI API**, provides a simple yet powerful RESTful API for getting access to context / digital twin data. NGSI API specifications have evolved over time driven by feedback from developers and implementation experiences. A first mature version of the API is the **NGSIV2 API**, which was defined by members of the FIWARE Community and is currently used in many systems in production within multiple sectors. Evolution of the API has taken place within the **ETSI CIM ISG<sup>3</sup>** (Context Information Management Industry Specification Group), where members of the FIWARE Community and the FIWARE Foundation have led the definition of an evolved version of the API, known as the **NGSI-LD API**, whose specifications were first published by ETSI in 2019 and continue to evolve<sup>4</sup>. The NGSI-LD API is used as the data integration API and is implemented by the core component of any “powered by FIWARE” architectures: the so-called Context Broker component. Different alternative open source implementations of a Context Broker are available within the FIWARE Community, namely the Orion-LD, Scorpio and Stellio products.
- The **Smart Data Models initiative<sup>5</sup>**, launched by the FIWARE Foundation, provides a library of Data Models described in JSON/JSON-LD format which are compatible respectively with the NGSIV2/NGSI-LD APIs as well as any other RESTful interfaces compliant with the Open API specification. Data Models published under the initiative are compatible with schema.org and comply with other existing de-facto sectoral standards when they exist. They solve one major issue developers are facing like it is the fact that a given data model specification may be mapped into JSON/JSON-LD in many different ways, all of them valid. Thanks to the Smart Data Models initiative, developers can rely on concrete mappings into JSON/JSON-LD, compatible with the NGSIV2/NGSI-LD APIs, that are made available within this library, avoiding interoperability problems derived from alternative mappings. Since its creation, more than 500 data models have been published and the number of organizations contributing data model descriptions is constantly growing. Relevant organizations like TM Forum<sup>6</sup> or IUDX<sup>7</sup> are joining forces with the FIWARE Foundation bringing support to an open governance model for the initiative, following best open source practices.

Building around the FIWARE Context Broker, which is the core mandatory component in any “powered by FIWARE” architecture, a rich set of complementary open-source components are available listed as part of the FIWARE Catalogue<sup>8</sup>. These components

---

<sup>3</sup> <https://www.etsi.org/committee/cim>

<sup>4</sup> [https://www.etsi.org/deliver/etsi\\_gs/CIM/001\\_099/009/01.03.01\\_60/gs\\_CIM009v010301p.pdf](https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.03.01_60/gs_CIM009v010301p.pdf)

<sup>5</sup> Github: <https://github.com/smart-data-models>; website: <https://smartdatamodels.org/>

<sup>6</sup> <https://www.tmforum.org/>

<sup>7</sup> <https://www.iudx.org.in/>

<sup>8</sup> <https://github.com/FIWARE/catalogue>

can be classified in the following categories or chapters:

- components easing development of interfaces with the Internet of Things, Robotic and third-party systems
- components supporting context / digital twin data processing and monitoring or the connection with data processing engines (e.g., Apache Spark, Apache Flink), monitoring tools (e.g., Grafana) and analysis platforms (e.g., Apache Superset)
- components covering aspects related to Identity and Access Management (IAM) as well as Publication and Monetization of Data (including data accessible via APIs like NGSI-LD)

## 3.2 Vertical Smart Solutions powered by FIWARE

Figure 3.3 depicts the reference architecture of a vertical smart solution powered by FIWARE. The concrete example corresponds to a smart solution for picking and palletizing products from a warehouse using robots. This reference architecture is structured in essentially three layers:

- A Context Broker component is at the core of the architecture, keeping a digital twin representation of the real world objects and concepts relevant to the specific problem tackled: AGV robots, palletizer robots, shelf sections where products are stored in the warehouse, automatic doors AGV robots have to pass, operators in the shopfloor, items of stored products, orders generated from the CRM system, etc.
- Southbound to the Context Broker, the NGSI IoT Agents, available as part of the FIWARE IDAS framework, are used for connections to robotic systems exporting the OPC-UA IoT protocol or to specific sensors or actuators, used for example to detect items in shelf sections or to be able to open the shop floor doors. They perform the necessary conversions between IoT protocols and NGSI. In addition, System Adapters developed based on the IDAS Agent library cope with the connection to the CRM and the Warehouse Inventory Management system that the solution has to interface with. FIWARE components like FIROS/SOSS allow, on the other hand, to perform the adaptation to robotic systems based on ROS/ROS2. Last but not least, the FIWARE component Kurento is able to process the video streams of cameras deployed in the shop floor, which are helpful to detect potential obstacles or risky situations.
- Northbound to the Context Broker, a number of tools are targeted to support real-time big data processing of the streams of history data generated as context / digital twin information evolves over time. A combination of third party open source components (Apache Superset, Grafana) and FIWARE components (e.g., Wirecloud) are shown in the picture targeted to support the creation of operational dashboards and advanced data maps for monitoring processes. A number of FIWARE Data Connectors (Cygnus, Draco, Cosmos, STH Comet,

QuantumLeap) are available as part of FIWARE to facilitate transference of historic context / digital twin information to these tools.

Transversal to all these layers, a number of FIWARE components support Identity and Access Management. They control the flow of data across the different layers. With regards to the access to the Context Broker, they enforce the policies establishing what users can update, query or subscribe to changes on context / digital twin data. Note that the flow of data is not only south to north in the picture. Northbound applications can perform updates on context data, which in turn will trigger changes in the devices, robots or systems that are connected southbound.

An important point to highlight is that FIWARE is not about taking it all or nothing. You are not forced to use all the complementary FIWARE components mentioned above but you are free to use other third party platform components as well to design the hybrid platform of your choice. Thus, for example, you may opt for using a concrete IoT platform instead of IDAS IoT Agents to interface with sensors and actuators as reflected in the picture. As long as it uses the FIWARE Context Broker technology to manage context information, your platform can be labeled as “Powered by FIWARE” and solutions build on top as well.

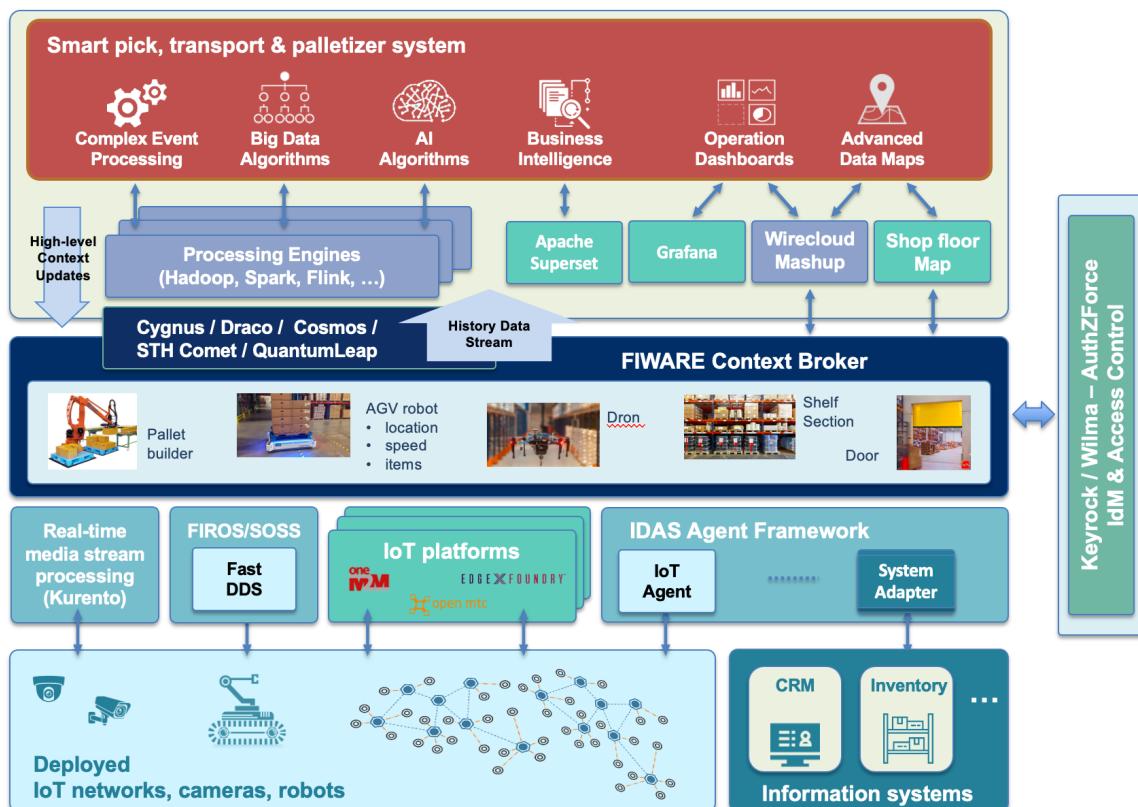


Figure 3.3 - Smart Solution for Picking and Palletizing products from a warehouse

### 3.3 Smart Organizations powered by FIWARE

Figure 3.4 depicts the reference architecture of a smart city powered by FIWARE. Again, the Context Broker component is at the core of the architecture, holding a digital twin representation of the real world objects and concepts and describing what is going on in the city: streets, waste bins and containers, waste trucks, buses, electric vehicle chargers, buildings, events, citizen claims, etc. The different vertical smart solutions deployed in the city (e.g., Air Quality Monitoring, Smart Traffic Management, Smart Parking, Smart Waste Management) are connected to the Context Broker contributing that information they manage which is relevant for creating a holistic Context / Digital Twin representation of the whole City, thereby breaking the information silos . Some of these vertical smart solutions may be powered by FIWARE (e.g., Traffic Control and Waste Management systems in the figure) in which case their interface with the global city-level Context Broker does not require any adaptation. Others may not be powered by FIWARE but this doesn't represent a major problem because creation of NGSI system adapters which translate from whatever API those systems export to NGSI-LD has proven not to be difficult. Last but not least, the City may deploy sensor/camera infrastructures through which valuable data is extracted.

Exploiting the complete Context / Digital Twin representation of the City, the Smart City Governance System (or City Operation Center) can be developed. Real-time BigData processing tools can be used relying on data coming from multiple sources, extracting more valuable insights for the support of decisions. Similarly, monitoring tools can leverage in this holistic Context / Digital Twin representation of the City.

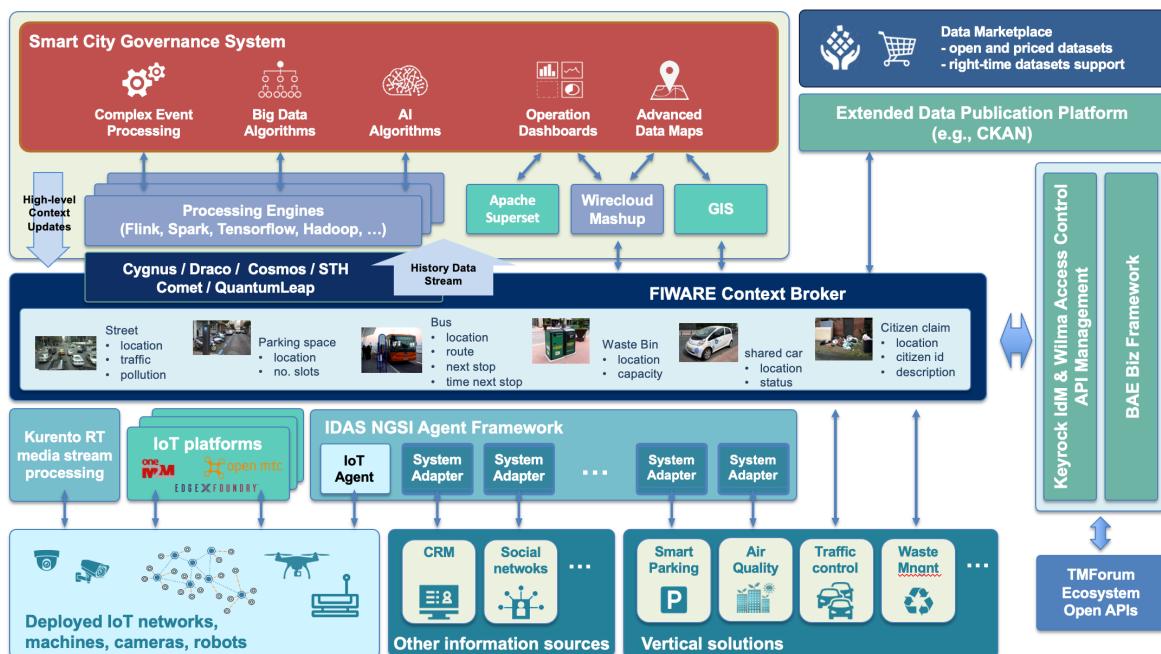


Figure 3.4 - Smart City Reference Architecture

## 3.4 Getting started with FIWARE

A number of valuable resources can help in getting started with FIWARE. As a first step, it is recommended to watch some of the webinars about FIWARE fundamentals which can be found on the FIWARE webinars webpage:

<https://www.fiware.org/community/fiware-webinars/>

Watching these webinars you will get an overview of the vision and value proposition behind FIWARE already introduced in this chapter.

The catalogue of FIWARE components is available on GitHub and it may be worth taking a first look at the contents of its home (readme) page:

<https://github.com/FIWARE/catalogue>

Developers are then recommended to go through the step by step tutorials available on GitHub:

<https://github.com/FIWARE/tutorials.Step-by-Step> (NGSI-LD tutorials)

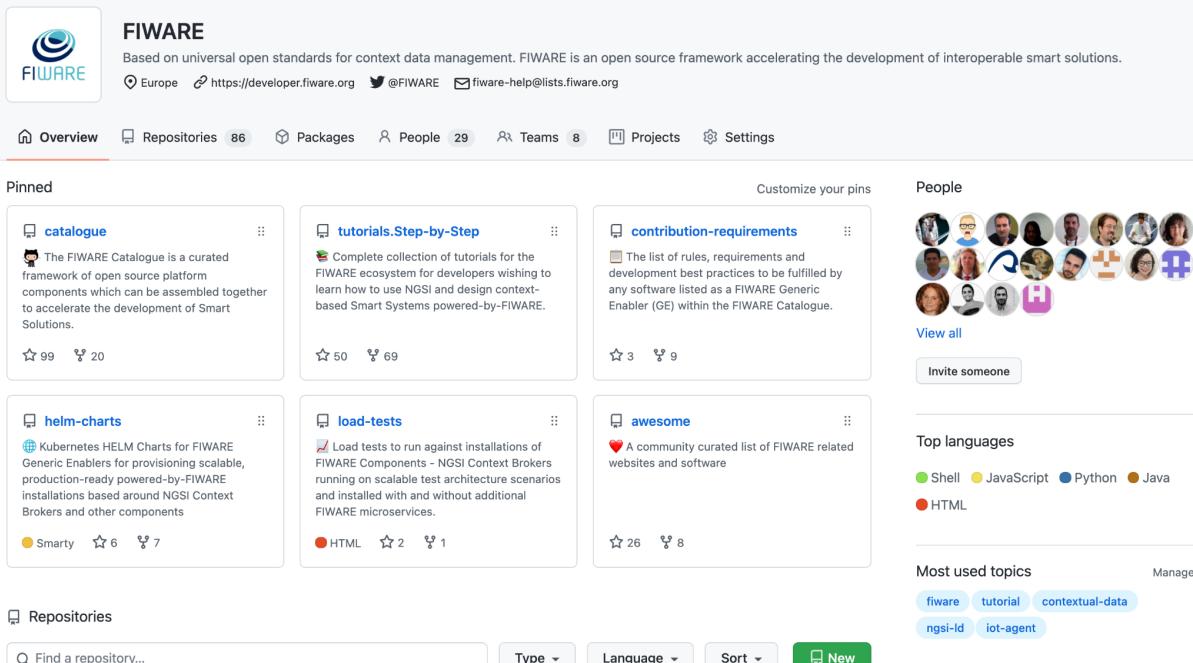
Each tutorial demonstrates the correct use of individual FIWARE components in connection to a case example. When starting with a given FIWARE component, it is useful to take a look at webinars connected to those components, available on the FIWARE webinars webpage mentioned above. Those webinars may also be helpful to CTOs and architects who may want to understand what each FIWARE component brings without the need to deploy and play with the software.

More information can be found in the space linked to the FIWARE organization in GitHub where you can find receipts how to deploy and configure FIWARE components for production environments using Kubernetes (helm-charts) or learn about load test results for some components, for example:

<https://github.com/FIWARE>

Also on the FIWARE website:

<https://www.fiware.org/>



**Pinned**

- catalogue** ⚡ The FIWARE Catalogue is a curated framework of open source platform components which can be assembled together to accelerate the development of Smart Solutions. ⭐ 99 ⚡ 20
- tutorials.Step-by-Step** 📖 Complete collection of tutorials for the FIWARE ecosystem for developers wishing to learn how to use NGSI and design context-based Smart Systems powered-by-FIWARE. ⭐ 50 ⚡ 69
- contribution-requirements** 📄 The list of rules, requirements and development best practices to be fulfilled by any software listed as a FIWARE Generic Enabler (GE) within the FIWARE Catalogue. ⭐ 3 ⚡ 9
- helm-charts** 🛠 Kubernetes HELM Charts for FIWARE General Enablers for provisioning scalable, production-ready powered-by-FIWARE installations based around NGSI Context Brokers and other components. ⚡ Smarty ⚡ 6 ⚡ 7
- load-tests** 🚗 Load tests to run against installations of FIWARE Components - NGSI Context Brokers running on scalable test architecture scenarios and installed with and without additional FIWARE microservices. ⚡ HTML ⚡ 2 ⚡ 1
- awesome** ❤️ A community curated list of FIWARE related websites and software. ⚡ 26 ⚡ 8

**Customize your pins**

**People**

**Top languages**

**Most used topics**

Information about smart data models which can be used royalty-free can be found on GitHub:

<https://github.com/smart-data-models>

As well as on the website associated with the Smart Data Models initiative:

<https://smartdatamodels.org/>

## 4 About iSHARE

### 4.1 Introduction to iSHARE

iSHARE brings a standard for secure and controlled exchange of data. All organizations that meet iSHARE's technical and legal requirements, and may display the iSHARE trademark, apply the same secure methods of identification, authentication and authorization. They can also authorize each other to share data with third parties – usually further downstream in the value chain – without losing control over their data.

The iSHARE scheme allows for the creation of data spaces, in which people and systems can share data in a trusted and controlled manner. It is already in use in the logistics sector, as well as various other industries, both nationally and internationally.

iSHARE was developed as a solution to enable businesses to connect with each other in a secure and controlled manner. It was initiated in late 2016 by the Dutch government's Top Sector Logistics, to address the key challenge of how to create a basis of trust, so that parties can share time-critical information simply and efficiently in a decentralized manner. Since its inception, a number of key players in logistics have been instrumental in defining a logistics data space using iSHARE. Today, this logistics data space has grown into a strong ecosystem of more than 5,000 companies providing access to their data via data hubs, and over 45,000 identities have been issued by iSHARE identity providers.

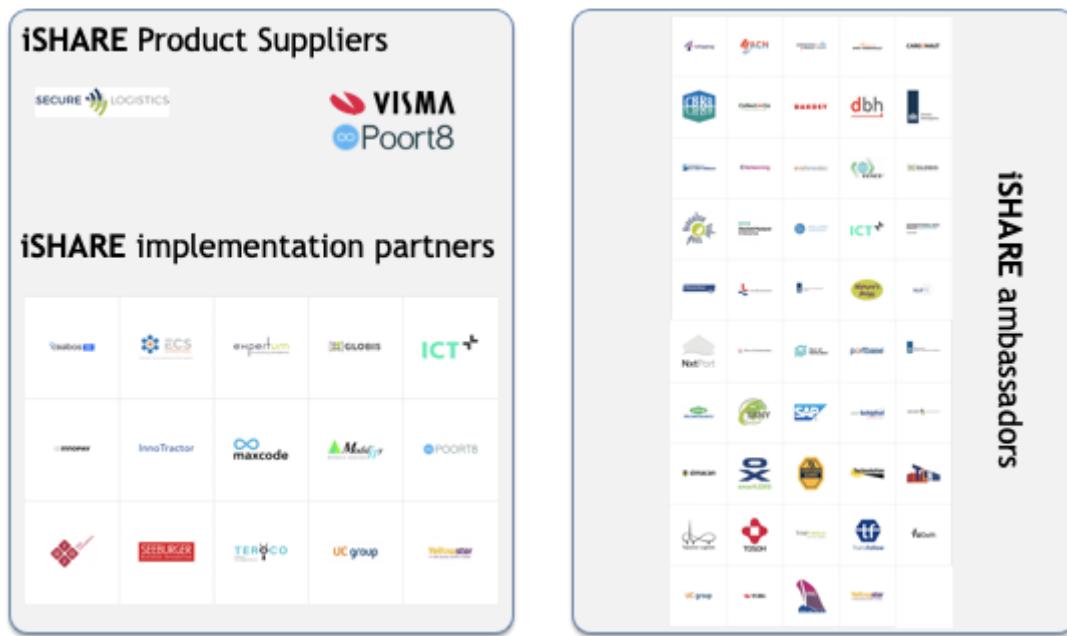


Figure 4.1 - Current ecosystem of iSHARE in Logistics Space

Thanks to being purposefully developed as a generic, open, federated and not-for-profit standard, iSHARE can be applied across sectors and across international borders. It facilitates the creation of ecosystems, as iSHARE is not just a technical specification, but also covers the legal and operational aspects which form the basis for a data space.

### What are the benefits?

1. Secure and simple login using iSHARE

An iSHARE digital identity enables you to log in securely when conducting business online. Identities that meet the requirements of the iSHARE standard are much more secure than alternative identities, such as those of Google and Microsoft. In the case of an iSHARE identity, your data is guaranteed not to be shared with third parties without your consent, for example, and your identity is verified on a regular basis.

This ensures that the service portals/websites you access know that you really are who you claim to be, and what you are authorized to do. You are also assured that the websites you log in to are reliable and secure. The higher the security level of iSHARE is, the more sensitive data you can exchange.

The great convenience of iSHARE is that you can use a single digital identity to log in securely and easily to a growing number of service portals/websites owned by different organizations.

## 2. Secures and controlled data sharing thanks to iSHARE authorization

iSHARE authorizations enable you to authorize other organizations to retrieve or modify your data on your behalf. This immediately saves you time and money, plus everything is done according to the requirements of the iSHARE standard: in a secure and controlled way.

You determine and manage who is allowed to use which data and under what conditions, in the iSHARE authorization register of an iSHARE-approved data hub. To do so, you must make use of a digital identity that meets the requirements of the iSHARE standard. Further, while defining authorizations you can attach a data license to it so the data consumer knows and understands the conditions under which data will be provided. The data consumer has to abide by the data license, as they have signed the terms of use agreement which covers the legal framework for sharing data as described in iSHARE standard.

Using iSHARE authorisation registry you can also authorize a data hub that is storing your data to share it with selected third parties who need it for their business processes.

## 4.2 A deeper look at the iSHARE standard

iSHARE enables trusted data exchange between organizations via its framework which standardizes the way these organizations can perform identification, authentication and authorizations. This is important from an interoperability perspective. Additionally, the framework provides for organizations to attach a data license while giving out authorisations, which determine the usage policy for the data that is being shared. It is quite important, given the value of data, that the owner of the data remains in control, even after it is shared. This is known as “data sovereignty”, and in iSHARE is achieved through use of data licenses. Additionally, the use of eIDAS digital certificates to sign the authorizations and the possibility to sign the data itself provides not only trust, but also proof of data source and its associated usage policy. Organisations who want to efficiently use these provisions can register themselves with an iSHARE satellite, and sign agreements to abide by the terms of use. This helps avoid signing multiple bilateral agreements for organizations with whom one wants to share data, as one can refer to the same terms of use and specifications when they have signed up with iSHARE. This is called multilateral agreement, which in the long run brings efficiency.

The following figure shows the roles that actors may play in a data space using the iSHARE scheme.

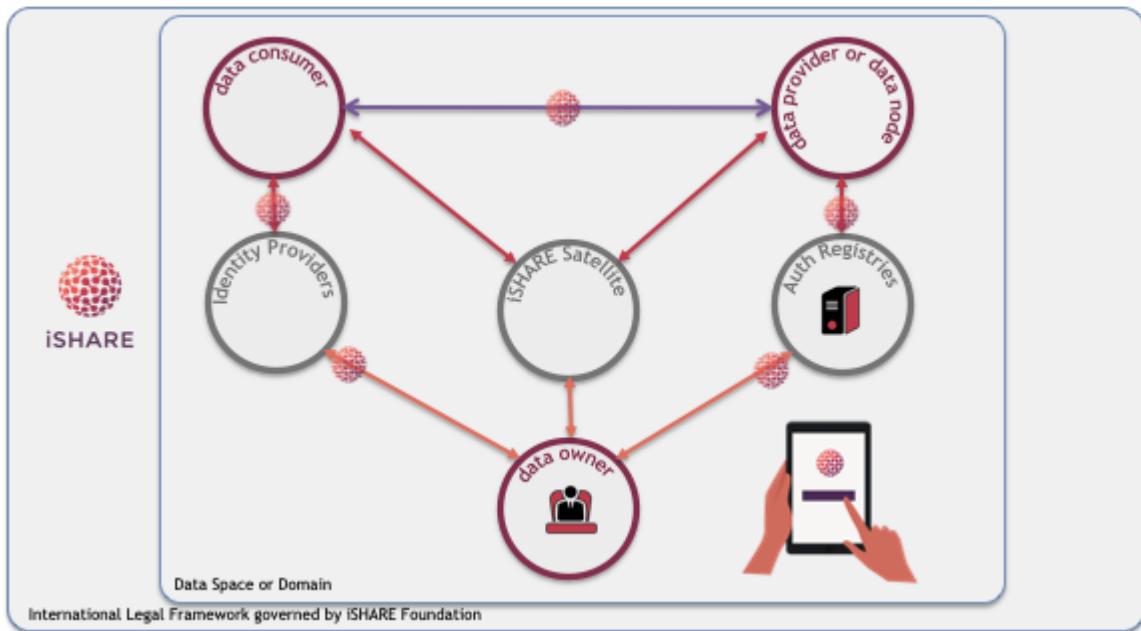


Figure 4.2 - Roles in Data Spaces using iSHARE

The following iSHARE roles are considered:

**Data Owner** - a party who owns data in the context of the use case (or business process) at a given time, and has entitled rights over the data.

**Data Consumer** - a party who wants to consume data provided by a data provider.

**Data Provider** - a party who hosts the service that provides data.

**Identity Provider** - a party who provides validated authentic identities of human actors. In many use cases the identity of the human actor is necessary, to assess the authorisations that person may have. Moreover it allows existing identities to be reused to login at multiple service providers, thereby reducing the need to create and maintain accounts with each service provider. It also reduces the overhead associated with validating identities for service providers.

**Authorisation Registry Provider** - a party who provides an Authorisation Registry as a Service. Such service provides a means for any given Data Owner to provide rights over its data to any party it desires. Usually, a Data Owner would either hire a third party to provide this service, or they themselves can host their own Authorisation Registry.

**iSHARE Satellite** - iSHARE satellite is responsible for holding registrations of participants of iSHARE. Any iSHARE party can check if any other party with whom it wants to connect and exchange data is a valid iSHARE participant or not. As explained earlier, this in essence allows participants to know if the other party would comply with legal terms of use, and would follow the usage policy as described in the data licenses.

Additional roles such as marketplace, authorisation registry broker, data vocabulary provider, etc. can also play a role in the data space when they are necessary.

The following figure depicts how a Data Owner can provide rights to Data Consumers to access data by means of registering the authorizations at a Authorisation Registry.

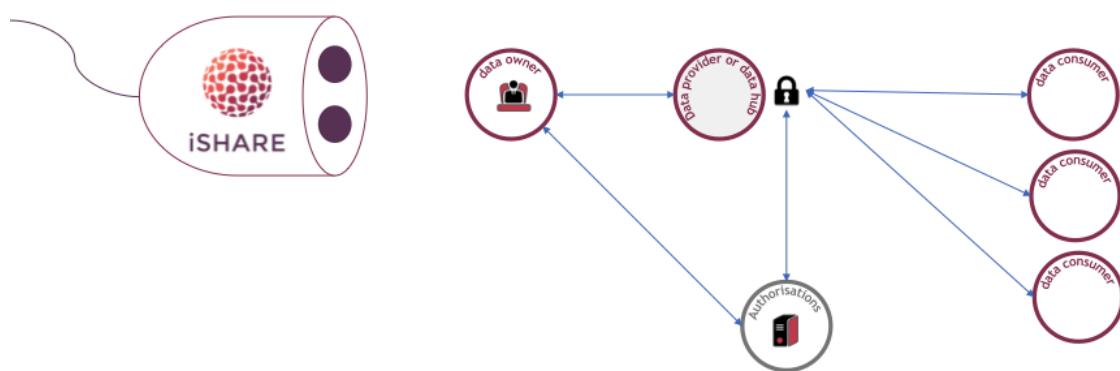


Figure 4.3 - Easy data access in decentralized manner

iSHARE enables creation of data spaces in multitudes in different sectors and different regions.

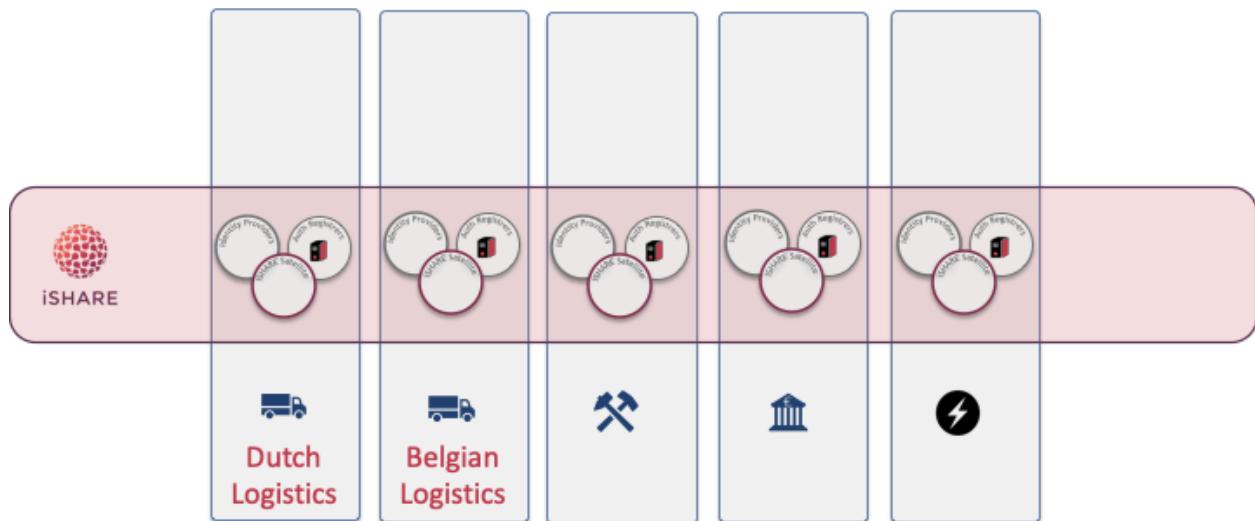


Figure 4.4 - Different dataspaces

## 4.3 Logistics data space

The figure below depicts one of the use cases in logistics which is enabled by iSHARE.

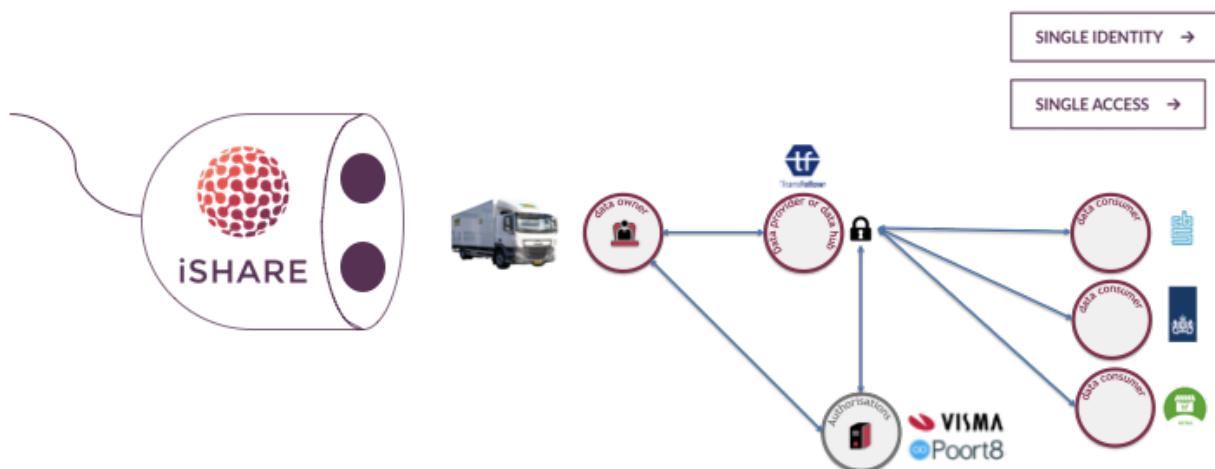


Figure 4.5 - Logistics example use case

In this example, the transporter is the data owner for the data about the shipment that it is carrying. That data is of course useful for the consignee who is awaiting the shipment to be delivered, but so it is for the department of traffic and transportation for the cities and regions within the transport route. With that information, these departments could predict the traffic and accordingly adjust road regulation, to avoid congestion and reduce pollution. Additionally, the custom border control (CMR)

process) can benefit from paperless access to information about the transportation directly from the source (transporter) about its clearance and payment of necessary taxes and duties. With access to such information in real-time, the process can be made efficient, and it reduces congestion and unnecessary waiting times at the borders. This is all made possible when the transporter registers in the authorisation registry who has access to which data and under what conditions. iSHARE standard enables the transporter to share only necessary information with relevant authorities/parties, and no need to share complete information which may be of sensitive nature.

## 4.4 Getting Started with iSHARE

Getting started with iSHARE is simple.

1. **Educate:** educate yourself with the iSHARE scheme specifications which cover all aspects of iSHARE like: Functional, Technical, Legal and Operational. It also explains the role model of iSHARE, and what is covered and what is not in the iSHARE specifications.
2. **Select and design use case:** Once you have a good understanding of iSHARE, it is time to select a use case which you want to solve using iSHARE. iSHARE experts can also help in designing use cases for participants.
3. **Implement and get it certified:** Once you have selected the use case and designed it, it is time to implement and test it. After implementation, the specifications of iSHARE can be certified using a certification test tool. The testing and certification varies according to the role of the participants.
4. **Sign the agreement:** When you sign the agreement with iSHARE Foundation or iSHARE satellite, you are registered as an iSHARE participant in the database. When you connect to other participants, they can check your status at the iSHARE satellite.

More detailed information can be found on the website:

<https://www.ishareworks.org/en>

The iSHARE scheme specification can be found here:

<https://ishareworks.atlassian.net/wiki/spaces/IS/pages/70222191/iSHARE+Scheme>

The iSHARE Technical specification can be found on the developer portal:

<https://dev.ishareworks.org/>

# 5 i4Trust Data Spaces

## 5.1 Overview

The main goal of i4Trust is to boost the development of innovative services around new data value chains. i4Trust helps to achieve this by providing the right tools, education, coaching and initial funding for the creation of Data Spaces enabling trustworthy and effective data sharing. Ecosystems of collaborating SMEs and supporting DIHs will emerge in a sustainable way around such Data Spaces.

i4Trust integrates standard-based building blocks from the FIWARE and iSHARE frameworks. Together with common data models, the FIWARE Context Broker building block supports effective data exchange among parties by using the standard NGSI-LD API. Additionally, FIWARE Data Marketplace components, based on relevant global industrial recommendations and data catalog (DCAT) standards, bring support to data publication and trading, including the monetization of data. On the other hand, the iSHARE scheme brings the foundation for trustworthy exchange among parties based on well-established security standards and robust legal frameworks.

This way, i4Trust provides the foundation for the core functionalities in every Data Space, namely data interoperability, data sovereignty and trust and data value creation. It also supports core concepts such as loose coupling (meaning that involved participants are not required to have knowledge about each other *a priori* and may discover them dynamically), as well as reusability and replaceability of data sources and applications.

Since all the tools are standard-based and supported by open source reference implementations, organizations that are sharing data through i4Trust Data Spaces — based on such references — will avoid vendor lock-in scenarios.

The proposed building blocks are in line with recommendations from the European Commission's Connecting Europe Facility (CEF) program, a pillar instrument materializing the EU Digital Single Market by means of creating cross-border Digital Service Infrastructures that enable the flourishing of innovative digital services across all the EU member states. The mentioned compatibility is achieved thanks to FIWARE Context Broker technology already being a CEF Building Block. Additionally, the iSHARE scheme is also compatible with eIDAS, the CEF Building Block that electronically identifies users from all across Europe. The integration of FIWARE components enabling the integration of the Context Broker with Distributed Ledger Technologies (DLTs) will facilitate the integration with the European Blockchain Service Infrastructure (EBSI) CEF Building Block in a later phase, enabling trusted digital audit trails, automated compliance checks and data integrity proofs.

The following picture summarizes how FIWARE and iSHARE building blocks complement each other and bring a complete and solid baseline for creation of data spaces. The following sections will elaborate further on the details of how the building blocks in each of the defined pillars (data interoperability, data sovereignty and trust, data value creation, data space governance) are materialized in i4Trust.

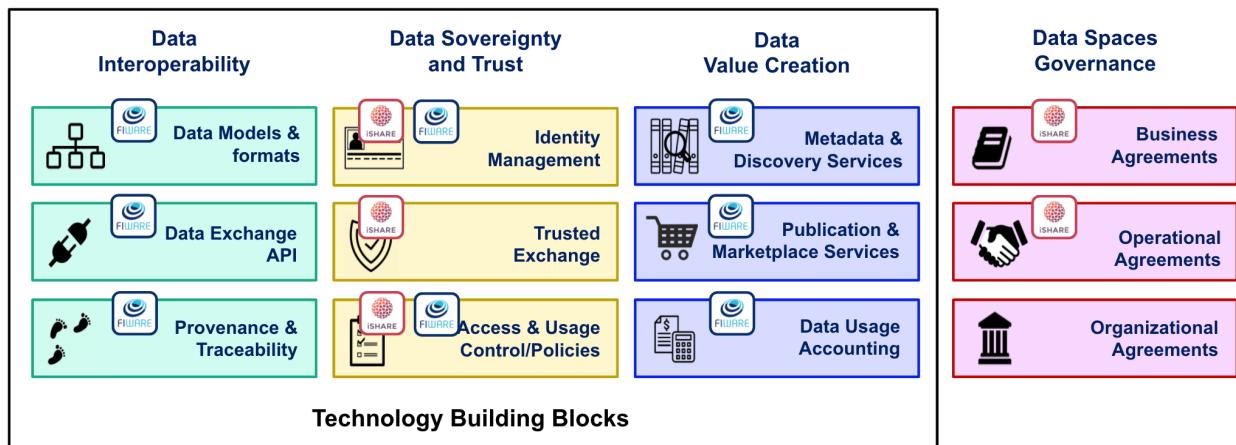


Figure 5.1 - Combining FIWARE and iSHARE building blocks to materialize data spaces

## 5.2 i4Trust Building Blocks

### 5.2.1 Data interoperability

Data providers joining data spaces must be able to publish data resources at well defined endpoints knowing that data consumers, unknown by them a priori, will know how to retrieve and consume data through those endpoints. Data consumers, on the other hand, must know how data available through endpoints they discover can be consumed. This is a key principle which was observed in the design of the world wide web: content providers publish web pages on web servers (endpoints) knowing that web browsers will be able to connect to them and retrieve web pages whose content they can render and display to end users. It means that all participants in data spaces should ‘speak the same language’, which translates into adopting domain-agnostic common APIs and security schemas for data exchange (the way of constructing sentences) together with data models represented in data formats compatible with those APIs (the vocabulary used in constructed sentences).

Participants in i4Trust data spaces will essentially exchange digital twin data using the **NGSI-LD API**. This API is domain-agnostic so many different systems have been developed using NGSI-LD in domains such as smart cities, smart manufacturing, smart energy, smart water, smart agrifood, smart ports, or smart health, to mention a few. This facilitates data sharing because each system participating in a i4Trust data space will be publishing data that simply enriches a Digital Twin data representation of the world that the rest of systems connecting to the data space will know how to access. Participant systems don’t know a priori what systems will be consuming the data they publish (although they will be able to set up concrete terms and conditions for accessing/using data as we will explain in the next section).

Figure 5.2 illustrates how different systems participating in i4Trust data spaces will exchange data. Context Broker servers are the endpoints through which systems connected to the data space publish digital twin data, very much like web servers publish html content on the world wide web. Those systems can in turn connect to Context Broker servers in order to obtain information they need. Note that i4Trust data spaces enable near real-time (right-time) exchange of digital twin data which is fundamental in the design of innovative value chains demanding a very dynamic exchange of data among participants. Just think about scenarios like a city managing traffic lights in streets close to a given train station in order to facilitate that travelers arriving and taking a taxi can leave faster to their destinations. NGSI-LD brings very simple therefore easy to use operations for creating, updating and consuming context / digital twin data but also more powerful operations like sophisticated queries, including geo-queries, or the subscription to get notified on changes of digital twin entities. On the other hand, i4Trust data spaces can also support the exchange of large files using standard file transfer protocols, since this kind of file transferences may be required for certain scenarios like training of AI algorithms.

Note that systems participating in i4Trust data spaces do not need to be themselves

"powered by FIWARE". Systems which have not been architected using FIWARE can still use the NGSI-LD API to share data they produce and consume data they need in the form of data associated with attributes of digital twin entities which represent that part of the world they deal with. This can be done directly by the systems or through NGSI-LD system adapters which have been programmed to perform a conversion between NGSI-LD and the API that the system natively supports for managing data.

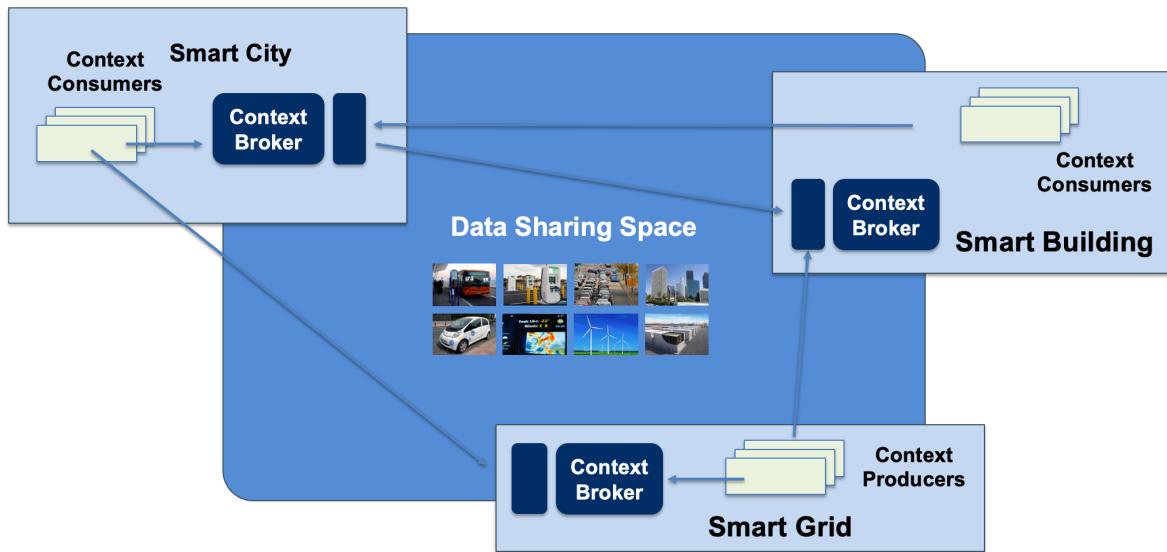


Figure 5.2 - Data Exchange in a i4Trust data space

From a theoretical perspective, systems connected to a data space should be able to share data using the API they prefer. The specification (information model) of each API could be published as some kind of manifesto that certain gateway components, integrated as part of the platform that systems should use to connect to the data space, may dynamically process in order to perform an automated adaptation from/to the APIs. However, such an approach faces important challenges. In the first place, such an approach has only been demonstrated in very simple scenarios and involving very simple APIs. Thus, the ability to exploit the kind of sophisticated features NGSI-LD would support for accessing digital twin data will be rather limited. On the other hand, creating a "common lingua" has proven to work for creating the kind of ecosystems we pursue: we shall ask ourselves if the world wide web had experienced the speed on adoption reached if HTTP and HTML hadn't been adopted as "common lingua" for web servers and browsers and each web server had the ability to choose a different protocol (as opposed to HTTP) or a different document format (as opposed to HTML). NGSI-LD has the advantages of being an open standard (defined by ETSI), has a strong open source community behind (FIWARE) and, quite relevant within Europe, it will pave the way for alignment with developments in the Connecting Europe Facility (CEF) program. Creation of system adapters that transform from specific APIs a given system may still require to use from/to NGSI-LD has proven to be not a

complex task. On the other hand, following the proposed approach, smart solutions which may already be based on FIWARE would not require any adaptation.

As mentioned before, the NGSI-LD API is domain-agnostic, therefore it is designed to work for any type (class) of digital twin. Consequently, achieving full interoperability also requires the adoption of common data models to be represented in formats compatible with the API. Here, the **Smart Data Models initiative**<sup>9</sup>, already introduced in the previous FIWARE chapter, reaches great relevance. It brings a powerful resource for developers who can rely on the way data model specifications are mapped into concrete JSON and JSON-LD structures under the initiative, the latter being compatible with NGSI-LD.

Figure 5.3 illustrates how resources are organized within the Smart Data Models initiative on GitHub. Data models are grouped into “subjects” (weather, parking, aquaculture, etc) which in turn are referred from repositories associated with the multiple application domains being considered (Smart Cities, Smart Agrifood, Smart Manufacturing, Smart Water, Smart Energy, etc). Note that there are subjects which are very specific to a given application domain (e.g., “street lighting” with regards to smart cities and communities) while others may be relevant to multiple domains (e.g., “weather” that is relevant to almost every domain or “sewage” that is relevant to the Smart Cities and Smart Water domains). Published data models are open to contributions and royalty-free.

An open governance model has been defined for the Smart Data Models initiative defining the lifecycle of data models comprising incubation of brand new data models as well as curation of data models via harmonization of different contributions. Processes and procedures for management of the different activities follow best practices from open source communities, guided by principles of transparency and meritocracy.

---

<sup>9</sup> <https://github.com/smart-data-models>

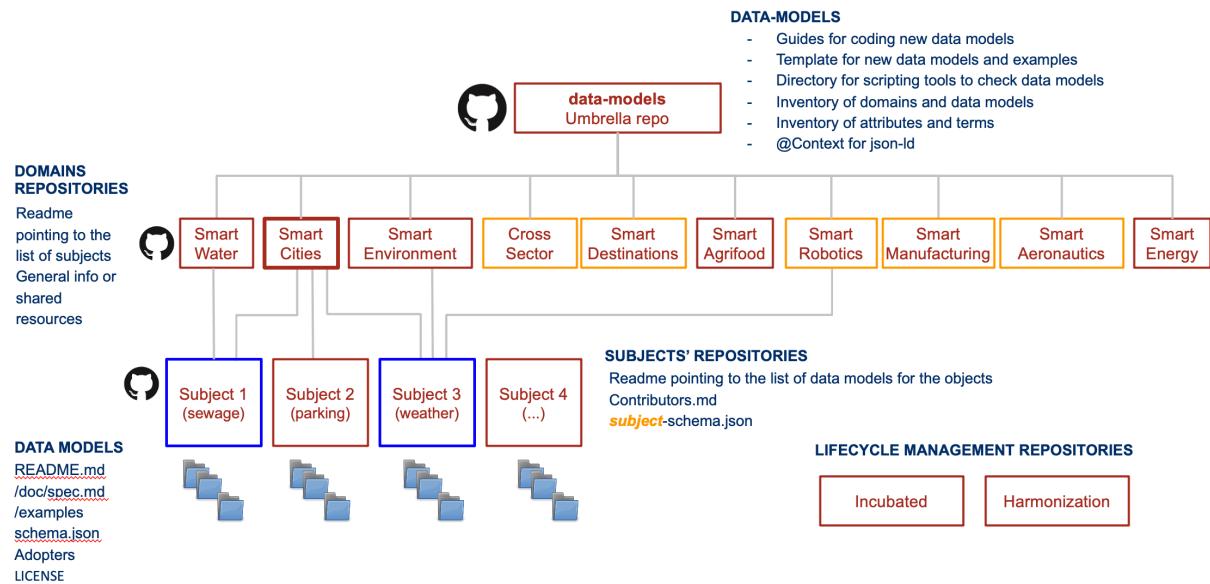


Figure 5.3 - Smart Data Models organization on GitHub

Completing the picture of building blocks for Data Interoperability, i4Trust is planning to incorporate components which provide the means for tracing and tracking in the process of data provision and data consumption/use. They will provide the basis for a number of important functions, from identification of the provenance of data to audit-proof logging of NGSI-LD transactions, very relevant for data spaces with strong requirements on transparency and certification. In this respect, i4Trust will leverage existing FIWARE components (i.e., Canis Major<sup>10</sup>) that ease recording of NGSI-LD transaction logs into different **Distributed Ledgers / Blockchains**.

## 5.2.2 Data sovereignty and Trust

Data spaces must provide means for guaranteeing organizations joining data spaces that they can trust the other participants and that they will be able to exercise sovereignty on their data. That requires the definition of common building blocks, based on mature security standards that will be used by all participants in the data space.

A first fundamental building block to support within data spaces has to do with **Identity Management (IM)**. This building block is implemented in i4Trust through multiple distributed **Identity Providers (IdP)** allowing identification and authentication of organizations, individuals, machines and other actors participating in a data space. i4Trust IdPs provide identity services by implementing APIs defined in the iSHARE specifications. For human identities the specifications are based on the **OpenID Connect** standard, which has been adapted in the iSHARE specification to overcome limitations in business and legal contexts and to improve interoperability. For

<sup>10</sup> <https://github.com/fiware/CanisMajor>

organizational identities iSHARE relies on **Public Key Infrastructure (PKI)** and **OAuth2.0** standards, which have been adapted as well to overcome their limitations. Currently, for EU wide acceptability, iSHARE uses **eIDAS**<sup>11</sup> based digital certificates for digitally signing data and assertions. The **eIDAS** framework is a building block provided by the European Commission that enables the mutual recognition of national electronic identification schemes (eID) across borders, allowing European citizens to use their national eIDs when accessing online services from other European countries. Additionally, regional certificates that match the requirements similar to eIDAS are legally recognized for the purpose of digital signatures and can be added and made interoperable.

i4Trust participants may rely on IdPs provided by iSHARE as a service or instantiate their own ones using the FIWARE **Keyrock** component which supports **OpenIdConnect**<sup>12</sup>, **SAML 2.0**<sup>13</sup> and **OAuth2**<sup>14</sup> standards and has been evolved to fully comply with iSHARE specifications.

A second fundamental building block will be the one which facilitates trusted data exchange among participants, providing certainty that participants involved in the data exchange are who they claim to be, and that they comply with defined rules/agreements.

Trust on the one hand refers to the fact that data providers and data consumers can rely on the identity of the members of the data ecosystem and beyond that, between different security domains. The iSHARE standard defines a Scheme Owner (also called iSHARE Satellite), which is playing the role of a trust authority providing a trusted framework which keeps the scheme, and its network of participants, operating properly. Every organisation participating in the data ecosystem is registered at the iSHARE satellite, enabling them to check at the iSHARE Satellite whether other organisations participate as well and are trustable. In order to also ensure the integrity of message contents, all exchanged messages among the organisations are protected by signed JSON Web Tokens (JWTs) using the PKI.

On the other hand trust refers to the fact that data consumers will abide by the terms of data usage that is defined by means of data license. The technical and functional specifications of the iSHARE standard have been developed keeping in mind the legal and business aspects. Furthermore, the iSHARE Satellite allows to ensure that consumers have agreed to the terms of use.

A third fundamental building block will be the one which facilitates authorisations at data attribute level and further complements it with usage rights. This building block is key to data sovereignty and enables data sovereign data spaces. The iSHARE standard specifies the role of an Authorisation Registry which enables organizations to specify

<sup>11</sup> <https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/eID>

<sup>12</sup> <https://openid.net/connect/>

<sup>13</sup> <http://docs.oasis-open.org/security/saml/Post2.0/sstc-saml-tech-overview-2.0.html>

<sup>14</sup> <https://oauth.net/2/>

access rights on specific data points to other organizations and users. Additionally, it allows attaching the data license to the authorisation so that consumers know what their usage rights are on the data. iSHARE refers to the XACML standard for the language inspiration of the API specifications of Authorisation Registries. The Authorization Registries and API proxy components in i4Trust implement the Policy Decision Point (PDP) and Policy Access Point (PAP) roles in the XML-based XACML standard, but the data exchanged is specified as JSON instead of XML and REST APIs are used instead. Other roles as defined in the XACML architecture are kept open and any standard could be used for that matter.

### 5.2.3 Data value creation

Loose coupling of participants is a fundamental principle in data spaces. Data providers and consumers do not necessarily know about each other. Therefore, it becomes essential to incorporate building blocks enabling the management of **data resources as true assets** with a business value. Assets which can be published, discovered and, eventually, traded. This way boosting the creation of multi-side markets where innovative services can be created.

**FIWARE Business Application Ecosystem (BAE)** components enable creation of **Marketplace services** which participants in data spaces can rely on for publishing their offerings around data assets they own. Different types of data assets can be defined via plugins that can be installed in the BAE, taking care of data validation, provider permissions and service activation. Nevertheless, three kinds of standard data asset types are supported by default, namely static data files, right-time data resources provided via NGSI-LD at well defined endpoints, and data processing services, which typically have associated well defined endpoints for providing input data and publish results, both in right-time, using NGSI-LD. Marketplace services are accessible through a portal or via APIs. Users, either end users through the portal or applications via APIs, of the Marketplace service can perform the following main actions:

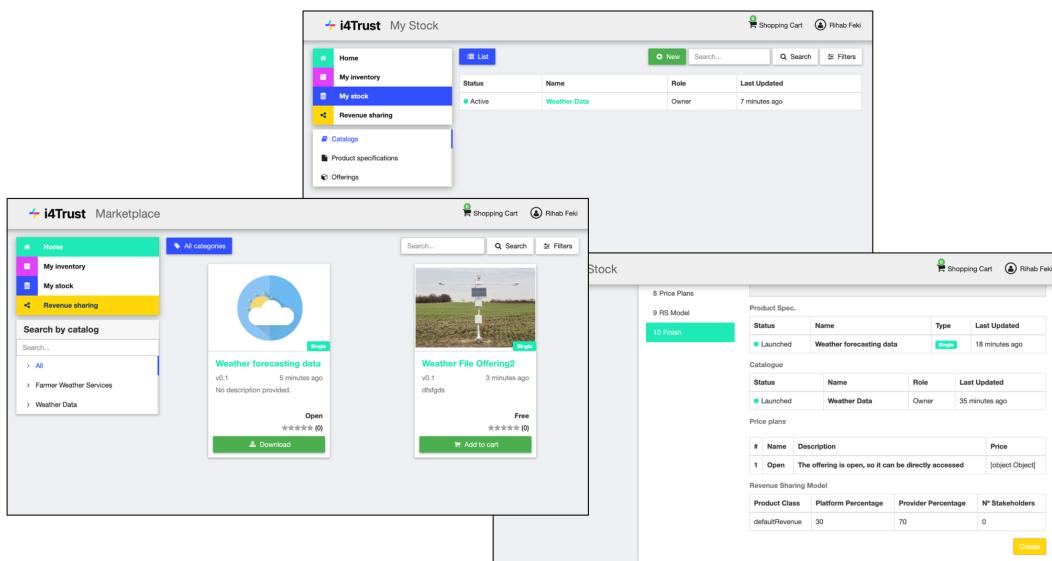
- register offerings around defined data asset types which typically means providing description of the asset, data models behind, endpoints (URLs) through with the data asset will be accessible and, very important, terms and conditions defined around the data asset, including SLAs, legal clauses, access control policies users of the asset have to comply with and associated pricing schema, which may be based on:
  - Free access (open data): user pay no money
  - One-time payments: users pay only once.
  - Recurring payments: users pay periodically (monthly, yearly, etc) for getting access to some data. In addition, users will be able to cancel the subscription, but they won't be able to access data anymore.
  - Usage payment: users pay per use. Their payments are based on the amount of information consumed.

- ability to navigate and search/discover existing offerings based on selected criteria.
- register revenue sharing models establishing how incomes generated by an offering with multiple stakeholders involved must be distributed.

Following is the list of backend components and APIs associated to the FIWARE BAE Marketplace:

- Backend implementing standard TM Forum APIs supporting configuration of the marketplace:
  - Catalog Management API
  - Product Ordering Management API
  - Product Inventory Management API
  - Party Management API
  - Customer Management API
  - Billing Management API
  - Usage Management API
- Rating, Charging, and Billing backend
- Revenue Settlement and Sharing System.
- Authentication, API Orchestrator, and Web portal

Figure 9 shows the FIWARE Data Marketplace portal deployed and configured for the i4Trust project<sup>15</sup>.



| Product Spec. |                          |      |                |
|---------------|--------------------------|------|----------------|
| Status        | Name                     | Type | Last Updated   |
| Launched      | Weather forecasting data |      | 18 minutes ago |

| Price Plans |   |      |  |
|-------------|---|------|--|
| Price       | # | Name | Description  |
| Free        | 1 | Open | The offering is open, so it can be directly accessed [object Object] |

| Revenue Sharing Model |                     |                     |                 |
|-----------------------|---------------------|---------------------|-----------------|
| Product Class         | Platform Percentage | Provider Percentage | N° Stakeholders |
| defaultRevenue        | 30                  | 70                  | 0               |

Figure 5.4 - FIWARE Data Marketplace portal

<sup>15</sup> <https://i4trust.org/>

FIWARE also comprise components for publication of data resources linked to data assets around which offerings are managed through the FIWARE Data Marketplace. For this purpose, we have the Idra publication platform and have developed as well extensions to the CKAN open data platform, which is an open data publication platform widely adopted in the market. These extensions support enhanced data management capabilities and integration with FIWARE technologies including NGSI. In particular, data publication and discovery features provided by CKAN have been enhanced with the following features:

- Right-time (near real time) time data publication - thanks to this extension, CKAN is not limited to list data resources linked to static files as part of its catalogue but also data resources linked to NGSI-LD requests served by Context Broker components deployed in a data space. This brings the ability to discover data resources relying on DCAT capabilities publication platforms support,
- Identity Management, Authentication and Access Control functions based on Keyrock components - therefore supporting OpenId Connect, OAuth2 and XACML standards adopted at overall data space level.
- Publication of priced data resources - thanks to this extension, it is possible to mark data resources listed as part of the catalogue as linked to offerings visible in the Data Marketplace. Users can therefore click on those data resources and navigate to the Marketplace to proceed with the acquisition of access rights
- Enhanced Data Visualization - thanks to this extension WireCloud (Configurable Dash-boards component) to create adaptive and incremental data visualization. This way, farmers (sellers) can decide the way they want their data to be shown.

## 5.2.4 Data Space Governance

Participants as well as potential participants of the data space need certainty not only on the above mentioned technical building blocks but also on non-technical aspects of data space like Operational, Legal and Business aspects. Data space governance is not only necessary for specifying these aspects but also in evolving them with time so that it is always relevant for the participants of the data space. Before sharing data with another organisation, first they both need to sit together and understand each other's requirements and agree on all aspects related to data sharing.

### 5.2.4.1 What are Business Agreements?

Data spaces should provide for data sovereignty to its participants. This can be achieved by using the right technical solution with a **legal framework** like the one provided by iSHARE. Apart from that the participants need to agree on other business aspects like the costing model, purpose and usage of data based on its criticality and liabilities. Such agreements are part of the **Business Agreements** building block.

**iSHARE authorisation registry** provides means for the participants to share only relevant information to other parties and also specify what they are allowed to do with that information by using **data licenses** as specified in the iSHARE standard. Another key element to sharing data in a sovereign way is to make sure that you are sharing the data with the right party which is made possible by using authentic identities that also provides a level of assurance on its authenticity. **iSHARE identity providers**, who are **eIDAS** compliant, provide the authentic identities along with desired level of assurance which enables trusted data sharing.

Additionally, the data space participants also need to identify and decide which **data exchange standards** they would like to use to interoperate with each other. Equally important is for them to decide the **data vocabularies** so that all participants can interpret the data correctly. A service provider could choose to provide vocabulary for data and its related translation and other services towards the participants.

#### 5.2.4.2 What constitutes Operational Agreements?

Another fundamental building block is about the operational reliability within the data space. Service providers such as identity providers, authorisation registry providers, marketplaces, etc. should be reliable so that parties can trust them and conduct business with confidence when using their services. Service providers are needed to be legally obliged to maintain uptime and availability of the services as it is critical for participants' businesses. Such arrangements can be made via **Service Level agreements**. iSHARE already specifies basic Service Level Agreements for service providers which they need to agree and sign off when they apply to become iSHARE certified service providers. It can be used as a basis by participants for further elaborating requirements for service providers that are critical for business continuity in the data space.

Another key aspect that is part of this Operational Agreement building block is **operational processes** which include:

- Participant lifecycle management -
- Incident management,
- release and version management of the different i4Trust building blocks
- change management,

Additionally, it may include other management processes as necessary as well as reporting as determined by the ecosystem. When starting a data space based on i4Trust, a base set of operational processes for Participant lifecycle management and Incident management will be available based on iSHARE. On the other hand, the different i4Trust building blocks come with proper release and version management handled within the corresponding FIWARE and iSHARE Communities. i4Trust Data

spaces can use these as foundation for operational agreements and if necessary evolve them further for data space specific requirements.

Most data spaces are initiated by few organizations who come together for a purpose, however, most of them want the data space to grow in the domain they operate in and often cross domain as well. In order for data space to gain the visibility it requires to excite other organizations to join them, **communication** becomes fundamental. Additionally service providers within the data space may want to publish about their services and use cases with reference to the data space. So some form of communication agreements are necessary to provide guidelines for organizations involved in the data space.

#### 5.2.4.3 What are organisational agreements?

For a data space to be successful and ever evolving, so that it stays relevant for the participants, it needs **governance** which can act on the previously mentioned agreements as its constitution and take necessary steps to make sure that data space operates as agreements were made. However, that does not mean that agreements are set in stone and cannot change over periods of time with new learnings and as the ecosystem evolves so does the agreements. For effective governance a right governing body is required to be set up by the data space participants with consensus. This is a key agreement in the **organisational agreements** building block. Ideally, for open data spaces a neutral and unbiased body suits better so as to create a level playing field for participants and service providers and not allow it to be skewed towards the organisations with deep pockets. On the other hand a closed data space can be created by a consortium of organizations where the purpose is different and often restricted to certain requirements for that consortium, however, here the difference is that it is generally not open for any other organization to join the data space or it has to comply by the requirements set by the initial consortium which may not be fair to them. iSHARE and FIWARE have an open, not-for-profit and unbiased governance structure in place so that i4Trust Data spaces can refer to iSHARE and FIWARE governance and can use it as the basis for their governance structure.

Since in a data space most of the transactions involve multiple parties it is apparent that a **business model** is agreed upon to at least fund the common infrastructure like governing body on one hand and if desired some common technical infrastructure on the other.

## 5.3 i4Trust roles

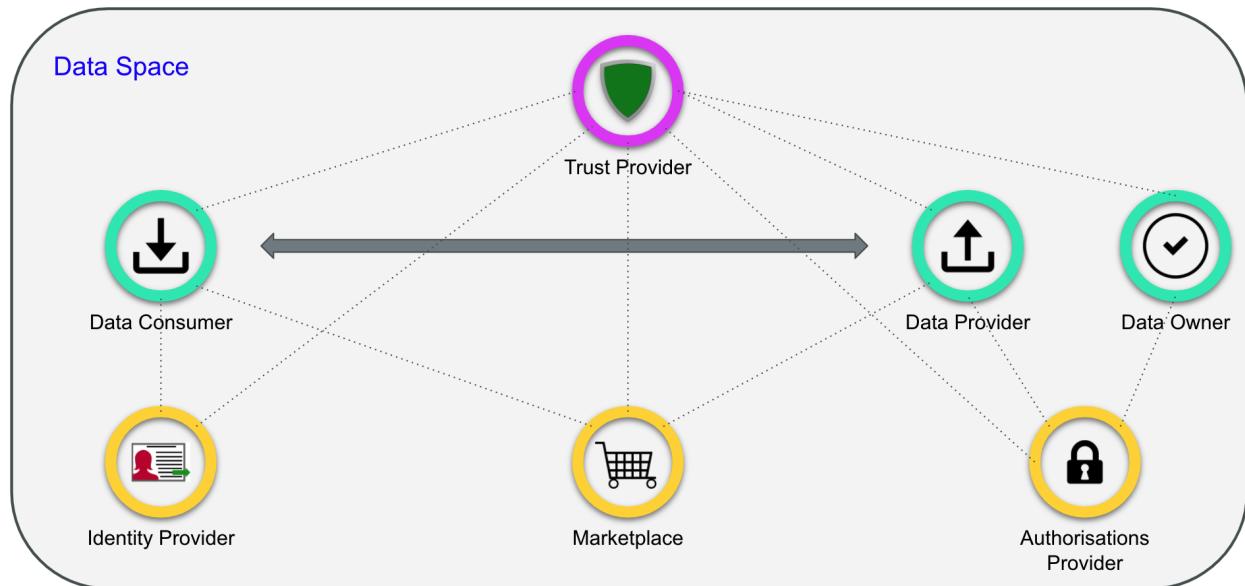


Figure 5.5 - Roles in i4Trust

Though these roles are mentioned separately, it does not restrict an organization from playing multiple roles.

- **Trust Authority Provider:** Every data space needs a trust provider who acts as a register of participants of the data space and keeps their status up to date. In a large data space there can be more than one trust authority provider. In i4Trust this role will be fulfilled by iSHARE satellite.
- **Data sharing roles:**
  - **Data Consumer:** A party or person who consumes data provided by the data provider. The data consumer has to agree to use the data as defined by the data license under which it received the authorization to access that data.
  - **Data Provider:** An organization that hosts a service to provide data on behalf of data owners. Data provider has the responsibility to verify who the data consumers are and provide only data that they are authorized to have access to.
  - **Data Owner:** A party or person who owns the data in the context of the process at a given time. Do note that a data owner does not necessarily have to be a data creator, it could be an application that generates data as a result of some processing.

- **Service Provider roles:**

- **Identity Provider:** A service provider that provides authentic human identities of data consumers or data providers so that either of them can be sure that they are dealing with the right party. Identity provider also provides the level of assurance of the identity.
- **Authorisation Registry Provider:** A service provider that provides services to data owners to register authorisations to data consumers/data providers. Alternatively, a data owner could choose to play the role themselves as long as they implement the authorisation registry provider APIs.
- **Marketplace Provider:** A service provider which allows data owners to publish data available for data consumers to be purchased or publish services data consumers can purchase. Marketplace enables discovery of data and data services for data consumers.

# 6 Bringing the pieces together: Packet Delivery reference example

## 6.1 Overall description

This chapter describes an i4Trust use case built with a Proof-of-Concept (PoC) which aims to showcase a data space for trusted data sharing among different parties.

The target is to allow a data service provider to offer its service on a public marketplace, where a service consuming party can acquire access to this offer. Furthermore this party can delegate the access to this acquired service offering to a different party.

In this use case, the provider is a packet delivery company, offering a service to view and change specific attributes of a packet delivery order. The consuming parties will be different retailers providing shop systems to their customers. These retailers will acquire access to the packet delivery system, and delegate the access to its customers.

In the PoC several parties are involved, each hosting its own infrastructure. Namely:

- Marketplace: Public marketplace for creating service offerings and acquiring access to them
- Packet Delivery Company: Provider which offers a service for retrieving and updating data of packet delivery orders

- Happy Pets: Premium pets retailer. Additionally there are two human actors involved: Happy Pets employee (actor working on behalf of Happy Pets company) and Happy Pets Customer (Customer of the pets shop system)
- No Cheaper: Retailer offering products at big discounts. Additionally there are two human actors: No Cheaper employee (actor working on behalf of No Cheaper company) and No Cheaper Customer (Customer of the No Cheaper shop system)
- iSHARE Satellite: Scheme administrator deciding about the admittance of each party within the trusted framework

Figure 10.1 depicts the overall architecture of the PoC. The packet delivery company and the shop system provider each have their own identity provider and authorization registry. In addition, the packet delivery company hosts a portal which allows users to view and modify attributes of packet delivery orders. The order entities are stored in an instance of the Context Broker. Read and write access to the packet delivery order entities is controlled by a PEP Proxy and PDP according to the described roles in section 6.2.1.

The iSHARE Satellite fulfills the role of a scheme administrator which holds information about each participating party (including a global UID called EORI) and allows to check for the admittance of each party.

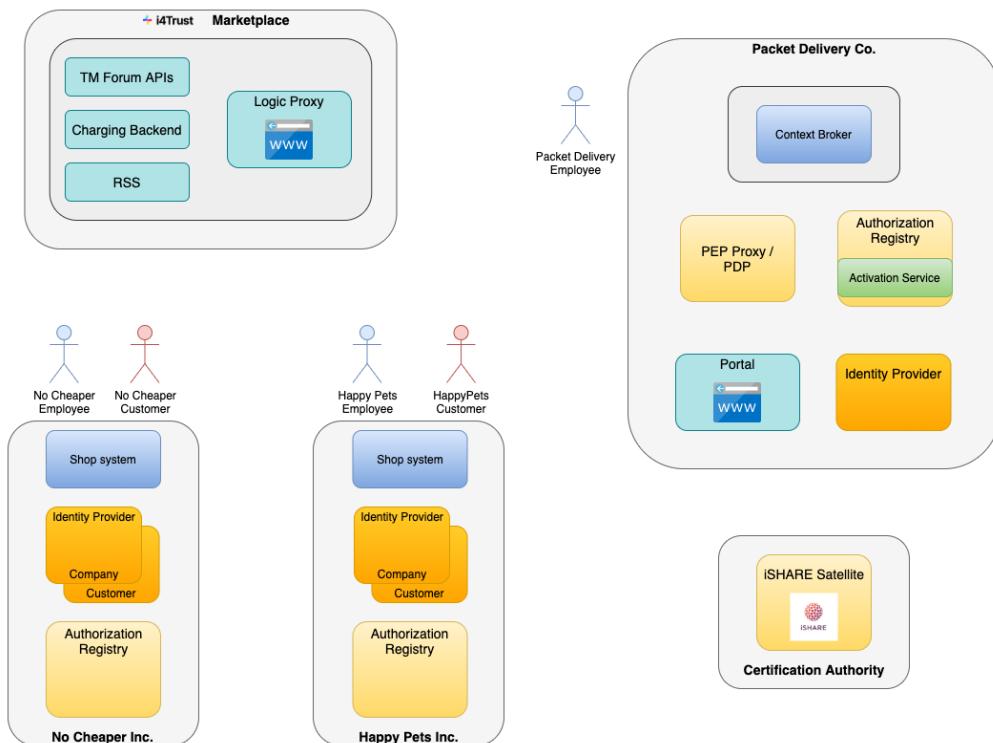


Figure 6.1: Overall architecture

The PoC implements the [iSHARE Scheme](#) in what refers to warrant trusted exchange of data between organizations, and the enforcement of access control policies. Technical details can be found [here](#).

## 6.2 Parties involved

### 6.2.1 Data Service Provider: Packet Delivery Company

Packet Delivery company is a parcel service provider delivering packets all over the world. It offers two kind of Packet Delivery services:

- A "Standard Packet Delivery" service for which the customer simply is given the opportunity to specify the issuer (sender) of the packet, the address, date and time at which the packet to be delivered is ready for collection, and the name and address of the destinee to whom the package has to be delivered. When Packet Delivery Company receives a packet delivery order from a given customer, it returns the target date at which the packet is planned to be delivered. Under defined terms and conditions (e.g., there are no problems with customs, addresses are valid, etc), it commits to deliver the packet in 48 hours within the same country and 5-6 days if it requires international shipping. However, customers are not allowed to adjust the concrete date of delivery (e.g., delaying it to a more suitable date) nor fine-tune the concrete time of delivery within the selected date of delivery.
- A "Gold Packet Delivery" service for which the customer enjoys all the benefits of the "Standard Packet Delivery" but also is allowed to adjust the concrete address of delivery, date of delivery within an offered period, as well as concrete time of delivery within the selected date of delivery, provided such adjustments are feasible (e.g., are requested enough time in advance and do not imply additional costs).

Packet Delivery Company offers its services electronically to different retailers, bringing them access to its Packet Delivery Info system (P.Info) via a REST API in order to allow them to issue packet delivery orders, trace location of orders and allow their customers to perform requests for adjustments on address, date and time of planned delivery when their clients are entitled to.

This is implemented because the P.Info system offers access to data about DELIVERYORDER entities through a Context Broker using NGSI-LD. A DELIVERYORDER is an entity with attributes like:

- issuer
- pickingAddress
- pickingDate
- pickingTime
- destinee
- deliveryAddress
- PDA (planned date of arrival)

- PTA (planned time of arrival)
- EDA (expected date of arrival)
- ETA (expected time of arrival)

Packet Delivery Company has defined two roles “P.Info.standard” and “P.Info.gold” for the P.Info system based on which the operations that can be requested on the above attributes through the Context Broker service it publishes have been defined. To simplify the description of the scenario, we will focus on attributes deliveryAddress, PDA and PTA since we could assume that the other ones will be assigned values at the time an order is created, will be always readable but will not be able to be changed by users with the defined roles. In that sense, the following policies apply for the defined roles regarding modification of these three attributes (deliveryAddress, PDA, PTA) once an order has been created:

| Path:  | Verb                 |             |
|--|----------------------|-------------|
| /ngsi-Id/v1/entities/{entityID}/attrs/{attrName} | GET                  | PATCH       |
| deliveryAddress                                  | P.Info.standard/gold | P.Info.gold |
| EDA  | P.Info.standard/gold | ---         |
| ETA  | P.Info.standard/gold | ---         |
| PDA  | P.Info.standard/gold | P.Info.gold |
| PTA  | P.Info.standard/gold | P.Info.gold |

Note that orders will be created using POST but with a different path (/ngsi-Id/v1/entities/). For issuing such requests an additional role “P.Create” is defined which will be assigned to the retailers Happy Pets and No Cheaper only.

Packet Delivery Company has decided to publish two different Packet Delivery offerings targeted to potential retailers and other kind of companies:

- Basic Delivery: which allows the company which acquires the offer to provide just a Standard Packet Delivery Services to its customers
- Premium Delivery: which allows the company which acquires the offer to provide Standard and Gold Delivery Services to its customers

Both have different pricing assigned.

Note that the Packet Delivery Company should not know about the identity of users of applications of any Retailer company. It simply should be able, when it receives a request, to a) recognize that such request comes from an user linked to an application that belongs to a Retailer company that acquired one of its offerings in the

Marketplace, b) find out what is the role within the P.Info application that such user has been assigned by the given Retailer company (i.e., either "P.Info.standard" or "P.Info.gold"), and c) check that such a role is a role that the given Retailer company could assign, considering the offering in the Marketplace it had acquired. After such steps, the Packet Delivery Company will simply check whether a user with the given role can perform the operation requested.

An application created by organization NoCheaper, no matter if it defines users whom it assigns role "P.Info.gold" to, is unable to successfully change the value of the PTA attribute of a given order.

### **6.2.2 Data Service Consumer: HappyPets Inc.**

HappyPets Inc. (HappyPets for short) is a company that sells products for pets. It will acquire the "Premium Packet Delivery" offering in the Marketplace.

Retailer company HappyPets (a company that sells products for pets) may then acquire the Premium Delivery offering in the Marketplace. This will allow it to offer, in turn, standard and gold delivery services to its customers through the store application of HappyPets (HappyPetsStore). In addition, there may be certain employees within its own organization, namely supervisors and agents in the phone help-desk service it offers, who may change the deliveryAddress, PDA and PTA of a given order using an internal application (HappyPetsBackOffice).

When a customer signs up in the HappyPetsStore, it can do as "regular" customer or "prime" customer (paying an annual fee). "Regular" customers are provided the standard packet delivery services while "prime" customers are provided the gold packet delivery service. This means they are assigned the "P.Info.standard" role and the "P.Info.gold" role within the HappyPetsStore application, respectively.

On the other hand, different employees are given different roles within the HappyPetsBackOffice application, so certain employees with supervisor roles at physical shops or agents at the central help-desk also have the "P.Info.gold" role assigned. The Happy Pets employee:

- Acquires the offering "Premium Packet Delivery" at the marketplace

The Happy Pets Customer:

- Signs up at the shop system of Happy Pets and gets assigned the "prime customer" role
  - For simplicity, we will assume that there is already a Happy Pets customer which already registered as "prime customer"
- Makes an order on the shop system, which results in the creation of a packet delivery order
  - For simplicity, we will assume that there is already a delivery order for this customer at the Packet Delivery company system

- Successfully changes the PTA of the order via the packet delivery company portal

### 6.2.3 Data Service Consumer: NoCheaper Ltd

NoCheaper Ltd is a company that sells products of any kind at rather big discounts. It will acquire the “Basic Packet Delivery” offering from the Packet Delivery Service company in the Marketplace.

The No Cheaper employee:

- Acquires the offering “Basic Packet Delivery” at the marketplace

The No Cheaper Customer:

- Signs up at the No Cheaper shop system and gets assigned the “standard customer” role
  - For simplicity, we will assume that there is already a No Cheaper customer which already registered as “standard customer”
- Makes an order on the shop system, which results in the creation of a packet delivery order
  - For simplicity, we will assume that there is already a delivery order for this customer at the Packet Delivery company system
- When trying to change the PTA of the order via the packet delivery company portal, it is denied
- It can be also shown that this request will get denied, even when the No Cheaper employee is assigning the “Prime Customer” role to the No Cheaper customer in its own Identity Provider system

### 6.2.4 i4Trust Marketplace

The i4Trust Marketplace is built based on the [FIWARE BAE \(Business Application Ecosystem\) component](#) that is made up of the combination of the FIWARE Business Framework and a set of APIs provided by the [TMForum](#). It allows the monetization of different kinds of assets during the whole service life cycle, from offering creation to its charging, accounting and revenue settlement required for billing and payment to involved participants.

Figure 6.2 shows the overall Architecture of the FIWARE Marketplace and the interactions between the different components.

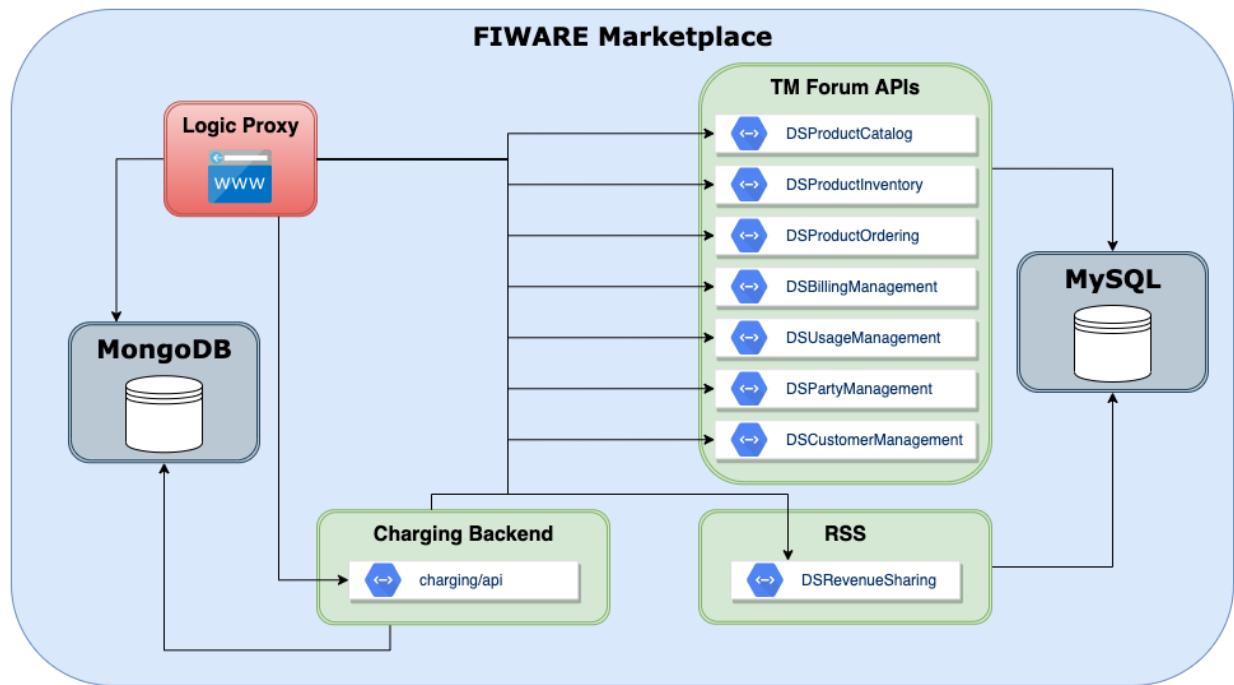


Figure 6.2: FIWARE Marketplace architecture overview

The packet delivery order asset parameters when creating the offer, and implementation of the necessary steps performed by the marketplace during the acquisition and activation phase, are provided by a dedicated [plugin](#) to be installed within the Charging Backend component.

A dedicated theme for the Marketplace UI can be found [here](#).

Deployment instructions for the marketplace can be found [here](#).

### 6.2.5 i4Trust Trust Provider (iSHARE Satellite)

A trust provider is a critical role in any dataspace since it ensures that participants can trust each other when sharing data and abide by terms and conditions of usage of data based on legal framework which all of these participants have agreed to. In a dataspace it usually also acts as a governing body which coordinates with the participants on a regular basis for keeping the dataspace definition relevant for its participants. Additionally, it may act as a neutral body which helps in mediation when there is a dispute amongst the participants. This makes it necessary then the organisation playing this role is a neutral and independent organisation.

iSHARE Framework defines this role as iSHARE Satellite. iSHARE Satellite component allows the trust provider to register the participants in the database and since it is based on distributed ledger technology, the participants' are discoverable across

different dataspaces. This enables interoperability of basic trust among all participants across dataspaces as long as dataspaces are based on iSHARE framework.

As explained earlier, each dataspace will have at least one trust provider and that role is ideally suited for a not for profit, neutral body responsible for registration of participants of that data space. The primary role of a trust provider is to onboard and register the participants of the dataspace after performing due diligence of the participant. Additionally, they facilitate the participants with agreements based on trust framework as well as covering dataspace specific agreements which they need to sign to become participants of the dataspace.

In i4Trust each experiment will assign/nominate an organisation to play the role of trust provider. This organisation will implement their own iSHARE Satellite component which is a node on the distributed ledger based on iSHARE Framework. This will enable each experiment to fully set up their own dataspace which are interoperable and they can independently register participants.

## 6.3 Deployment of components

In the following it is described, which enablers could be deployed, representing the components described in the architecture as shown in Figure 10.1. Information about the marketplace is provided in the previous section 6.2.4.

Note that additional features and bug fixes will be implemented continuously. Up-to-date deployment instructions and the latest releases of the components can be always found at the [i4Trust Tutorials](#) repository on GitHub.

### Context Broker

The packet delivery order service is provided by an instance of the [orion context broker](#) via the [NGSI-LD API](#). An instance of this context broker needs to be deployed within the Packet Delivery company environment. Deployment instructions can be found [here](#).

### Identity Provider / Authorisation Registry

The components of the identity provider and the authorisation registry are combined in a single enabler, the identity manager [Keyrock](#). This identity provider implements the required endpoints for authentication and authorisation of users, as well as for storing and retrieving delegation information of access policies. A separate instance needs to be deployed for each of the environments of Packet Delivery company, Happy Pets and No Cheaper. Furthermore, at Happy Pets and No Cheaper, in order to distinguish

between employees and shop customers, two separate IDPs need to be deployed for each of the two retailers. In order to differentiate in the log ins of employees at the marketplace and shop customers at the Packet delivery portal, for both, Happy Pets and No Cheaper, two Keyrock instances are deployed.

Deployment instructions for Keyrock in the environments of service provider and consumers can be found [here](#) and [here](#), respectively.

The implementation of the authorization registry functionality into Keyrock is still ongoing. Until this is finished, a test instance of the [iSHARE Authorization Registry](#) is used as a central instance for all parties.

In order to create policies at this Authorization Registry, an additional [Activation Service](#) needs to be deployed where requests dedicated for the Authorization Registry need to be sent to. This Activation Service is placed within the environment of Packet Delivery Company in order to be able to create policies at its Authorization Registry. It will check at the Authorization Registry, whether the requesting party is allowed to create policies and then will create the requested policies on behalf of the requesting party.

### PEP Proxy / PDP

The access management is handled by the API Gateway [Kong](#) together with the [ngsi-ishare-policies](#) plugin, which incorporates the necessary PEP/PDP functionality for i4Trust data spaces.

With every new version of the plugin, an Image of Kong which incorporates the plugin is built and pushed to [quay.io](#).

Deployment instructions can be found [here](#).

### Shop systems

For both retailers, Happy Pets and No Cheaper, a shop system needs to be set up, allowing users to register with correct roles/policies assigned, and to issue packet delivery orders. For simplicity, these systems can be skipped. In this case, customer users need to be pre-registered in the corresponding identity providers and their assigned policies need to be created manually in the authorisation registries. Furthermore, pre-existing delivery orders need to be created within the Context Broker.

### Packet delivery portal

For the Packet Delivery company, a portal system application needs to be set up, allowing to view information about packet delivery orders, and to change specific attributes of these orders. The source code of this application including deployment

instructions can be found [here](#). Further deployment instructions for this reference example can be also found [here](#).

### iSHARE Satellite

An instance of an iSHARE Satellite needs to be deployed centrally.

For Kubernetes deployment, Helm Charts are provided for most of the components. Charts for the activation service and the Packet Delivery Company portal application can be found [here](#). For the components of Orion Context Broker, Business API Ecosystem and Keyrock, the charts can be found [here](#).

When using Kubernetes, note that the ingress controller must be capable of handling large request headers. When using nginx as ingress controller, these are the proposed parameters to be set:

```
large-client-header-buffers: "8 32k"  
http2-max-field-size: "32k"  
http2-max-header-size: "32k"
```

The following Docker images have been released officially:

| Component            | Docker Image   |
|----------------------|--|
| Orion Context Broker | <a href="#">quay.io/fiware/orion-lb:1.0.1</a>                |
| Keyrock              | <a href="#">fiware/idm:8.2.0</a>                             |
| Kong                 | <a href="#">quay.io/fiware/kong:0.2.0</a>                    |
| PDC Portal           | <a href="#">i4trust/pdc-portal:v0.1.0</a>                    |
| Activation Service   | <a href="#">i4trust/activation-service:v0.1.0</a>            |
| BAE APIs             | <a href="#">fiware/biz-ecosystem-apis:v7.6.0</a>             |
| BAE RSS              | <a href="#">fiware/biz-ecosystem-rss:v8.0.0</a>              |
| BAE Charging Backend | <a href="#">fiware/biz-ecosystem-charging-backend:v8.0.0</a> |
| BAE Logic Proxy      | <a href="#">fiware/biz-ecosystem-logic-proxy:v8.0.0</a>      |

All of the components will be reachable via specific hostnames, depending on their environment and infrastructure. For the sake of clarity of the following description of the use case in this document, all shown requests will use the hostnames of the components deployed on the i4Trust demo infrastructure listed below:

| Component                           | Hostname  |
|-------------------------------------|---|
| Context Broker (Packet Delivery Co) | Not publicly accessible   |
| IDP (Packet Delivery Co)            | <a href="https://pdc-keyrock.i4trust-demo.fiware.dev">https://pdc-keyrock.i4trust-demo.fiware.dev</a>   |
| IDP (Happy Pets)                    | <a href="https://happypets-keyrock.i4trust-demo.fiware.dev">https://happypets-keyrock.i4trust-demo.fiware.dev</a>   |
| IDP (Happy Pets Shop)               | <a href="https://happypets-keyrock-shop.i4trust-demo.fiware.dev">https://happypets-keyrock-shop.i4trust-demo.fiware.dev</a>                                     |
| IDP (No Cheaper)                    | <a href="https://nocheaper-keyrock.i4trust-demo.fiware.dev">https://nocheaper-keyrock.i4trust-demo.fiware.dev</a>   |
| IDP (No Cheaper Shop)               | <a href="https://nocheaper-keyrock-shop.i4trust-demo.fiware.dev">https://nocheaper-keyrock-shop.i4trust-demo.fiware.dev</a>                                     |
| PEP Proxy / PDP                     | <a href="https://pdc-kong.i4trust-demo.fiware.dev">https://pdc-kong.i4trust-demo.fiware.dev</a>   |
| Shop system (Happy Pets)            | Not used at the moment  |
| Shop system (No Cheaper)            | Not used at the moment  |
| Packet delivery portal              | <a href="https://pdc-portal.i4trust-demo.fiware.dev">https://pdc-portal.i4trust-demo.fiware.dev</a>   |
| iSHARE Satellite                    | <a href="https://scheme.isharetest.net">https://scheme.isharetest.net</a>   |
| Marketplace (Logic Proxy)           | <a href="https://marketplace.i4trust-demo.fiware.dev">https://marketplace.i4trust-demo.fiware.dev</a>   |
| AR / Activation Service (temporary) | <a href="https://ar.isharetest.net">https://ar.isharetest.net</a> / <a href="https://pdc-as.i4trust-demo.fiware.dev">https://pdc-as.i4trust-demo.fiware.dev</a> |

## 6.4 Prerequisites

Before following the described use case, several pre-conditions have to be created at the components of the different parties involved. This involves pre-registrations at the iSHARE Satellite for each participant, as well as creating users and organizations

within the different identity providers and providing the access configuration at the PEP proxy and PDP.

In the following it is assumed that each party hosts its components, like identity provider or authorization registry, within its own environment. Therefore each party will get assigned a single unique identifier, the Economic Operators Registration and Identification number ([EORI](#)), to be registered at the iSHARE Satellite. Every party should be registered as an active participant, which will be checked at different steps of the use case. Note that in the case that some components are not part of these parties/organizations, because they are hosted externally, these components require separate EORIs.

Additionally, each party needs to acquire an X.509 certificate which is distributed by a trusted root under certain [PKIs](#). These certificates are required in the following use case for signing and validating requests between the different components.

The following table lists the EORI numbers for each participant of this use case.

| Participant        | GUID/EORI            |
|--------------------|----------------------|
| HappyPets          | EU.EORI.NLHAPPYPETS  |
| Packet Delivery Co | EU.EORI.NLPACKETDEL  |
| Market Place       | EU.EORI.NLMARKETPLA  |
| iSHARE satellite   | EU.EORI.NL0000000000 |
| NoCheaper          | EU.EORI.NLNOCHEAPER  |

Some participating parties host their own identity provider. For each of them, some pre-configuration needs to be done before following this use case:

- Packet Delivery company: Create user for employee
- Happy Pets (Company IDP): Create user for employee
- No Cheaper (Company IDP): Create user for employee

For simplicity, it is assumed that the two customers have already been registered at the Identity Providers (Customer IDP) of Happy Pets and No Cheaper, respectively. Therefore, these users must be created initially and get assigned the required user-level policies (Premium or Standard), which will be described later in this section.

This means to manually create the necessary policies using the UI or API of the Authorization Registries of Happy Pets and No Cheaper, respectively.

Access control for the packet delivery order service, provided by a context broker instance, is managed by a PEP proxy in conjunction with a PDP, namely Kong with `ngsi-ishare-policies` plugin. An [example setup](#) based on Helm charts shows how to perform a declarative configuration of the Kong plugin to create a route pointing to the local Orion Context Broker instance.

The PDP component is able to automatically analyse the incoming NGSI-LD request and to determine the necessary policy required for each request.

In the following, an attribute based policy concept will be used for the access management instead of simple roles. Policies are defined according to the iSHARE-compliant model of [delegation evidence](#). The roles defined in [section 6.2.1](#) match to the following policies, which will be required to be stored at the Authorization Registries for accessing the different resources described in [section 6.2.1](#):

| Role            | Policies (example)   |
|-----------------|--|
| P.Info.standard | {<br>"delegationEvidence": {<br>"notBefore": 1541058939,<br>"notOnOrAfter": 2147483647,<br>"policyIssuer": "EU.EORI.NLPACKETDEL",<br>"target": {<br>"accessSubject": "EU.EORI.NLNOCHEAPER"<br>},<br>"policySets": [<br>{<br>"maxDelegationDepth": 1,<br>"target": {<br>"environment": {<br>"licenses": ["ISHARE.0001"]<br>}<br>},<br>"policies": [<br>{<br>"target": {<br>"resource": {<br>"type": "DELIVERYORDER",<br>"identifiers": ["*"],<br>"attributes": ["*"]<br>},<br>"actions": ["GET"]<br>},<br>"rules": [<br>{<br>"effect": "Permit"<br>}<br>]<br>]<br>]<br>]<br>]<br>]<br>} |

```

        ]
    }
}
}

{
  "delegationEvidence": {
    "notBefore": 1541058939,
    "notOnOrAfter": 2147483647,
    "policyIssuer": "EU.EORI.NLPACKETDEL",
    "target": {
      "accessSubject": "EU.EORI.NLHAPPYPETS"
    },
    "policySets": [
      {
        "maxDelegationDepth": 1,
        "target": {
          "environment": {
            "licenses": ["ISHARE.0001"]
          }
        },
        "policies": [
          {
            "target": {
              "resource": {
                "type": "DELIVERYORDER",
                "identifiers": ["*"],
                "attributes": [
                  "pta", "pda", "deliveryAddress"
                ]
              },
              "actions": ["PATCH"]
            },
            "rules": [
              {
                "effect": "Permit"
              }
            ]
          },
          {
            "target": {
              "resource": {
                "type": "DELIVERYORDER",
                "identifiers": ["*"],
                "attributes": ["*"]
              },
              "actions": ["GET"]
            },
            "rules": [
              {
                "effect": "Permit"
              }
            ]
          }
        ]
      }
    ]
  }
}

```

P.Info.gold

```
        ]
    }
}
```

Above policies represent the access rights at organizational level stored at the Authorization Registry of the Packet Delivery Company. This means that these policies constitute the access rights, which Happy Pets and No Cheaper are allowed to delegate to their customers. In addition, similar policies at user level need to be stored at the Authorization Registry of Happy Pets and No Cheaper for each of their customers. Such policies represent the actual access rights of each individual customer and might be restricted to certain delivery orders by adding specific entity IDs in the identifiers attribute.

The organizational access policies will be created during the phase [acquisition of rights](#), where employees of Happy Pets and No Cheaper acquire access to the Packet Delivery Company service offerings on behalf of their organizations. In order that the Marketplace is able to create these policies via the Activation Service at the Authorization Registry of Packet Delivery Company, an initial policy must be created at its Authorization registry:

```
{
  "delegationEvidence": {
    "notBefore": 1614354348,
    "notOnOrAfter": 1614354348,
    "policyIssuer": "EU.EORI.NLPACKETDEL",
    "target": {
      "accessSubject": "EU.EORI.NLMARKETPLA"
    },
    "policySets": [
      {
        "policies": [
          {
            "target": {
              "resource": {
                "type": "delegationEvidence",
                "identifiers": ["*"],
                "attributes": ["*"]
              },
              "actions": ["POST"]
            },
            "rules": [
              {
                "effect": "Permit"
              }
            ]
          }
        ]
      }
    ]
  }
}
```

```

        }
    ]
}
}
```

The user-level policies stored at the Authorization Registries would be created when customers sign up at the corresponding shop systems of Happy Pets and No Cheaper, respectively. As written earlier in this section, it is assumed that the two customers have already been registered at the Identity Providers of Happy Pets and No Cheaper. Therefore their user-level access policies need to be initially stored at the Authorisation Registries (can be done via the UI or API), as given in the examples below, where access is restricted only to specific delivery orders:

| User   | Policy (example)   |
|--|--|
| No Cheaper customer with GET access to delivery order with ID urn:ngsi-ld:DELIVERYORDER:NOCHEAPER001 | <pre>{   "delegationEvidence": {     "notBefore": 1541058939,     "notOnOrAfter": 2147483647,     "policyIssuer": "EU.EORI.NLNOCHEAPER",     "target": {       "accessSubject": "0a6fd729-b52d-467c-9b10-0ba42dbaff4a"     },     "policySets": [       {         "maxDelegationDepth": 1,         "target": {           "environment": {             "licenses": [               "ISHARE.0001"             ]           }         },         "policies": [           {             "target": {               "resource": {                 "type": "DELIVERYORDER",                 "identifiers": [                   "urn:ngsi-ld:DELIVERYORDER:NOCHEAPER001"                 ],                 "attributes": [                   "*"                 ]               },               "actions": [                 "GET"               ]             }           }         ]       }     ] }</pre> |

```

        },
        "rules": [
            {
                "effect": "Permit"
            }
        ]
    }
}
}

{
    "delegationEvidence": {
        "notBefore": 1541058939,
        "notOnOrAfter": 2147483647,
        "policyIssuer": "EU.EORI.NLHAPPYPETS",
        "target": {
            "accessSubject": "cccc7d10-29e7-4a9f-944a-e6c3d39774c0"
        },
        "policySets": [
            {
                "maxDelegationDepth": 1,
                "target": {
                    "environment": {
                        "licenses": [
                            "ISHARE.0001"
                        ]
                    }
                }
            },
            "policies": [
                {
                    "target": {
                        "resource": {
                            "type": "DELIVERYORDER",
                            "identifiers": [
                                "urn:nsgsi-ld:DELIVERYORDER:HAPPYPETS001"
                            ],
                            "attributes": [
                                "pta", "pda", "deliveryAddress"
                            ]
                        },
                        "actions": [
                            "PATCH"
                        ]
                    },
                    "rules": [
                        {
                            "effect": "Permit"
                        }
                    ]
                },
                {
                    "target": {
                        "resource": {
                            "type": "DELIVERYORDER",
                            "identifiers": [
                                "urn:nsgsi-ld:DELIVERYORDER:HAPPYPETS001"
                            ]
                        }
                    }
                }
            ]
        }
    }
}

```

Happy Pets  
customer with  
GET and  
PATCH access  
to delivery  
order with ID  
urn:nsgsi-ld:DEL  
IVERYORDER:H  
APPYPETS001

```
        ],
        "attributes": [
            "*"
        ]
    },
    "actions": [
        "GET"
    ]
},
"rules": [
    {
        "effect": "Permit"
    }
]
}
]
```

The organisations of Happy Pets and No Cheaper need to be able to create packet delivery orders. This is done by a POST request to the packet delivery service and requires the role “P.Create”, as described in [section 6.2.1](#). For simplicity, the corresponding policies allowing POST requests for creating delivery order entities need to be created initially for both of them at the Packet Delivery Authorisation Registry.

The following presents how to create a delivery order through a request that is sent to the PEP proxy of Packet Delivery company (assuming there is a route /orion pointing to the Context Broker) which will handle the access control.

```
> Authorization: Bearer IIeD...NIQ // Bearer JWT for PEP proxy access control
> Content-Type: application/json

POST https://pdc-kong.i4trust-demo.fiware.dev/orion/ngsi-ld/v1/entities

> Payload
{
    "id": "urn:ngsi-ld:DELIVERYORDER:HAPPYPETS001",
    "type": "DELIVERYORDER",
    "issuer": {
        "type": "Property",
        "value": "Happy Pets"
    },
    "destinee": {
        "type": "Property",
        "value": "Happy Pets customer"
    },
    "deliveryAddress": {
        "type": "Property",
        "value": {
            "addressCountry": "DE",
            "addressLocality": "Berlin"
        }
    }
}
```

```
        "addressRegion": "Berlin",
        "addressLocality": "Berlin",
        "postalCode": "12345",
        "streetAddress": "Customer Strasse 23"
    },
},
"originAddress": {
    "type": "Property",
    "value": {
        "addressCountry": "DE",
        "addressRegion": "Berlin",
        "addressLocality": "Berlin",
        "postalCode": "12345",
        "streetAddress": "HappyPets Strasse 15"
    }
},
"pda": {
    "type": "Property",
    "value": "2021-10-02"
},
"pta": {
    "type": "Property",
    "value": "14:00:00"
},
"eda": {
    "type": "Property",
    "value": "2021-10-02"
},
"eta": {
    "type": "Property",
    "value": "14:00:00"
},
"@context": [
    "https://schema.lab.fiware.org/ld/context",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
]
}
```

For simplicity in the rest of this chapter, it is assumed that there are already packet delivery orders existing for the customers of Happy Pets and No Cheaper. Therefore these entities had been created.

In a real scenario, the packet delivery system would calculate and enrich the entity with the information of estimated arrivals picking dates.

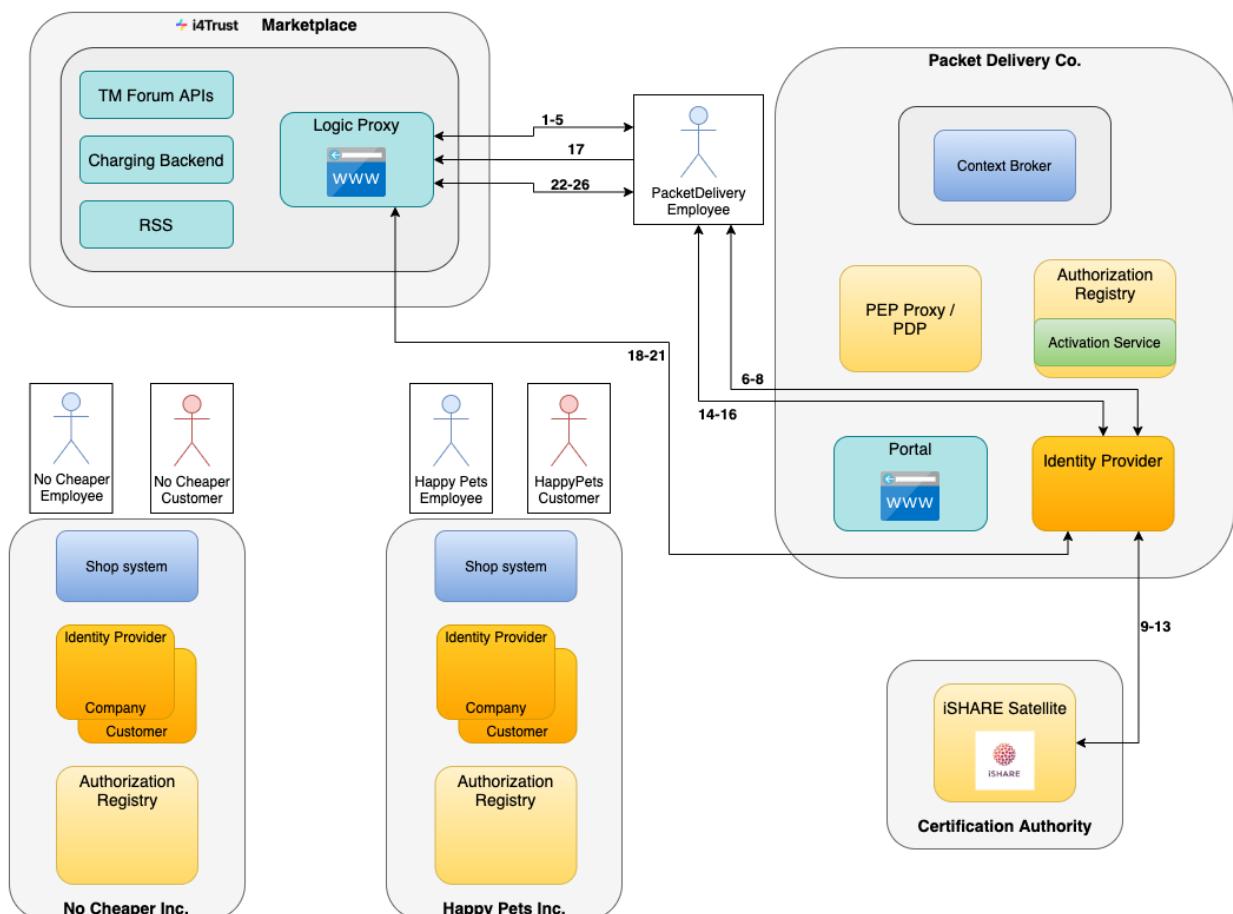
## 6.5 Create Offering

The process of creating an offer is explained. It needs to be performed twice for creating the offerings for “Basic Delivery” and “Premium Delivery”, providing a different set of offering information by an employee of the Packet Delivery company.

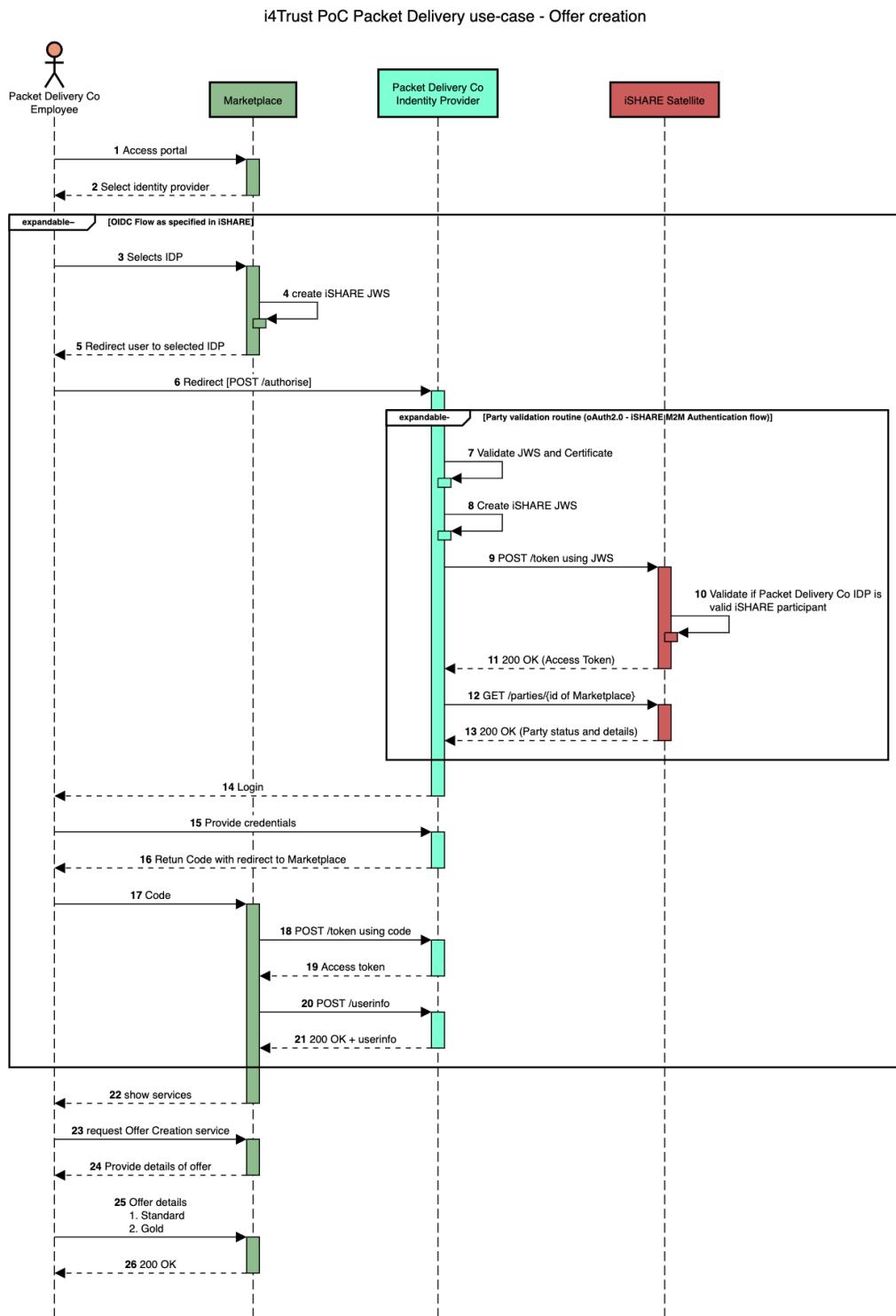
### 6.5.1 Sequence description (Packet Delivery Co.)

The following gives a detailed description of the offer creation process. Figure 6.3 presents the different interactions in an architectural overview, whereas Figure 6.4 shows a detailed sequence diagram of the whole process. The numberings in the architectural overview map to the different steps of the sequence diagram.

In the following, a description is given for each of the sequence steps.



*Figure 6.3: Architectural interactions for step “Create offering”*



*Figure 6.4: Sequence diagram for step “[Create offering](#)”*

1. Packet Delivery company employee accesses the Marketplace page (provided by the BAE Logic Proxy), in order to login.
2. Packet Delivery company employee is displayed a list of Identity Providers for selecting the desired Identity Provider for login.
3. Packet Delivery company employee selects the Identity Provider (of Packet Delivery company).
4. Marketplace Logic Proxy generates an [iSHARE JWT](#) signed using the JSON Web Signature standard (JWS). This also applies to all JWT generated in further steps.

```
> Headers
{
  "alg": "RS256",
  "typ": "JWT",
  "x5c": [ // Complete certificate chain for validating JWT signature (see step 7)
    "MIIEhjCC....Zy9w=="
  ]
}

> Payload
{
  "iss": "EU.EORI.NLMARKETPLA", // Client-ID of Marketplace (EORI)
  "sub": "EU.EORI.NLMARKETPLA", // Client-ID of Marketplace (EORI)
  "aud": "EU.EORI.NLPACKETDEL", // ID (EORI) of Packet Delivery Co IDP
  "jti": "378a47c4-2822-4ca5-a49a-7e5a1cc7ea59", // Unique JWT ID
  "iat": 1504683445,
  "exp": 1504683475,
  "nbf": 1504683445,
  "response_type": "code",
  "client_id": "EU.EORI.NLPACKETDEL",
  "scope": "openid iSHARE sub name contact_details",
  "redirect_uri":
  "https://marketplace.i4trust-demo.fiware.dev/openid_connect1.0/return",
  "state": "af0ifjsldkj",
  "nonce": "c428224ca5a",
  "acr_values": "urn:http://eidas.europa.eu/LoA/NotNotified/high",
  "language": "en"
}
```

5. Packet Delivery company employee is redirected to the Identity Provider of Packet Delivery company via it's browser. The request sent to the [/authorize](#) endpoint contains the signed JWT created in step 4 in the request URL parameter.

```
> Content-Type: application/x-www-form-urlencoded
```

```
POST https://pdc-keyrock.i4trust-demo.fiware.dev/authorise
```

```
response_type=code&
client_id=EU.EORI.NLMARKETPLA&
scope=iSHARE openid&
request=eyJ0eXA...YkNKOQ

Decoded Request parameter
{
    "iss": "EU.EORI.NLMARKETPLA", // Client-ID of Marketplace (EORI)
    "sub": "EU.EORI.NLMARKETPLA", // Client-ID of Marketplace (EORI)
    "aud": "EU.EORI.NLPACKETDEL", // ID (EORI) of Packet Delivery Co IDP
    "jti": "378a47c4-2822-4ca5-a49a-7e5a1cc7ea59", // Unique JWT ID
    "iat": 1504683445,
    "exp": 1504683475,
    "nbf": 1504683445,
    "response_type": "code",
    "client_id": "EU.EORI.NLPACKETDEL",
    "scope": "openid iSHARE sub name contact_details",
    "redirect_uri":
    "https://marketplace.i4trust-demo.fiware.dev/openid_connect1.0/return",
    "state": "af0ifjsldkj",
    "nonce": "c428224ca5a",
    "acr_values": "urn:http://idas.europa.eu/LoA/NotNotified/high",
    "language": "en"
}
```

6. The request of step 5 is forwarded to the Packet Delivery company Identity Provider.
7. Packet Delivery company Identity Provider validates the JWT and certificate of the Marketplace (an example for implementation can be found [here](#)).
8. Packet Delivery company Identity Provider generates an iSHARE JWT (also see description in step 4).

```
> Headers
{
    "alg": "RS256",
    "typ": "JWT",
    "x5c": [
        "MIIEhjCC....Zy9w=="
    ]
}

> Payload
{
    "iss": "EU.EORI.NLPACKETDEL", // Client-ID (EORI) of Packet Delivery Co
    "sub": "EU.EORI.NLPACKETDEL", // Client-ID (EORI) of Packet Delivery Co
    "aud": [
        "EU.EORI.NL0000000000", // ID (EORI) and token-URL of iSHARE Satellite
        "https://scheme.isharetest.net/connect/token"
    ]
}
```

```
        ],
        "jti": "99ab5bca41bb45b78d242a46f0157b7d", // Unique JWT ID
        "iat": "1540827435",
        "exp": 1540827465,
        "nbf": 1540827435
    }
```

9. Packet Delivery company Identity Provider sends a request to the iSHARE Satellite [/token](#) endpoint. The signed JWT created in step 8 is provided with the `client_assertion` parameter of the request.

```
> Content-Type: application/x-www-form-urlencoded  
  
POST https://scheme.isharetest.net/connect/token  
  
grant_type=client_credentials&  
scope=iSHARE&  
client_id=EU.EORI.NLPACKETDEL&  
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&  
client_assertion=eyJhbGc...9hvW
```

10. iSHARE Satellite validates the JWT and certificate, and checks the status of the Packet Delivery company Identity Provider (also see step 7). If it is no valid iSHARE participant, an error is returned.
11. The [access token](#) is returned to the Packet Delivery company Identity Provider.

```
< Content-Type: application/json  
< Cache-Control: no-store  
< Pragma: no-cache  
  
{  
    "access_token": "aW2ys9NGE8RjHPZ4mytQivkWJ05HGQCYJ7VyMBGGDLIow",  
    "expires_in": 3600,  
    "token_type": "Bearer"  
}
```

12. Packet Delivery company Identity Provider sends a request to the [/parties](#) endpoint of the iSHARE Satellite, in order to retrieve information about the Marketplace for verification of its status as iSHARE participant. The access token from the previous step 11 is provided as Bearer authorization token. The request contains the Marketplace EORI as query parameter, as well as the subject name as encoded in the certificate of the Marketplace.

```
> Authorization: Bearer IIe...VNIQ // access_token from step 11  
  
GET https://scheme.isharetest.net/parties?  
eori=EU.EORI.NLMARKETPLA&  
certificate_subject_name=C=NL, SERIALNUMBER=EU.EORI.NLMARKETPLA, CN=Packet  
Delivery Co&
```

```
active_only=true
```

13. iSHARE Satellite returns the information about the Marketplace. The `party_info.adherence.status` field contains the status of the Marketplace. When there is an error response, or the status is not active, then an error is presented to the Packet Delivery company employee.

```
> Content-Type: application/json
{
  "parties_token": "eyJ4N...xY_aaQ"
}

Decoded parties_token parameter
{
  "iss": "EU.EORI.NL000000000",
  "sub": "EU.EORI.NLMARKETPLA",
  "jti": "77e8179fbfe6469eb64c054da26a77c3",
  "iat": 1589282112,
  "exp": 1589282142,
  "aud": "EU.EORI.NLPACKETDEL",
  "party_info": {
    "party_id": "EU.EORI.NLMARKETPLA",
    "party_name": "Marketplace",
    "adherence": {
      "status": "Active",
      "start_date": "2020-04-26T00:00:00",
      "end_date": "2021-07-25T00:00:00"
    },
    "certifications": [
    ],
    "capability_url": "https://bae-keyrock.i4trust-demo.fiware.dev/capabilities"
  }
}
```

14. The login page of the Packet Delivery company Identity Provider is presented to the Packet Delivery company employee.
15. Packet Delivery company employee provides its credentials on the login page of Packet Delivery company Identity Provider.
16. Packet Delivery company Identity Provider returns the code according to OpenID Connect to the [redirect URI](#) of the Marketplace Logic Proxy, as provided in the request of step 5.

```
< Location: https://marketplace.i4trust-demo.fiware.dev/openid_connect1.0/return?
code=Dmn-TbSj70cK15ym1j5xZsgkabzVP8dMugC81nzmeW4&
state=ZqVQm4zHaEDyBhzpm1ZRH7fsxy7031q2
```

17. The code from step 16 is redirected to the Marketplace Logic Proxy.

18. Marketplace Logic Proxy sends a request to the [/token](#) endpoint of the Packet Delivery company Identity Provider. The signed JWT created in step 4 is provided with the `client_assertion` parameter of the request. The code from steps 16/17 is provided with the `code` parameter.

```
> Content-Type: application/x-www-form-urlencoded  
  
POST https://pdc-keyrock.i4trust-demo.fiware.dev/token  
  
grant_type=authorization_code&  
client_id=EU.EORI.NLMARKETPLA&  
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&  
client_assertion=eyJhbGci...9hvW&  
redirect_uri=https://marketplace.i4trust-demo.fiware.dev/openid_connect1.0/return&  
code=Dmn-TbSj70cK15ym1j5xZsgkabzVP8dMugC81nzmeW4
```

19. The [access token](#) is returned to the Marketplace Logic Proxy. In addition, an iSHARE JWT compliant [id\\_token](#) is returned which is associated with the authenticated session.

```
< Content-Type: application/json  
< Cache-Control: no-store  
< Pragma: no-cache  
  
{  
  "id_token": "eyJhb...V2jA",  
  "access_token": "aW2ys9NGE8RjHPZ4mytQivkJ05HGQCYJ7VyMBGGDLIOw",  
  "expires_in": 3600,  
  "token_type": "Bearer"  
}  
  
Decoded id_token parameter  
{  
  "iss": "EU.EORI.NLPACKETDEL",  
  "sub": "419404e1-07ce-4d80-9e8a-eca94vde0003de",  
  "aud": "EU.EORI.NLMARKETPLA",  
  "jti": "378a47c4-2822-4ca5-a49a-7e5a1cc7ea59",  
  "iat": 1504683445,  
  "exp": 1504683475,  
  "auth_time": 1504683435,  
  "nonce": "c428224ca5a",  
  "acr": "urn:oid:1.3.6.2.1.5.4.1.1.1.1",  
  "azp": "EU.EORI.NLMARKETPLA",  
}
```

20. Marketplace Logic Proxy sends a request to the [/userinfo](#) endpoint of Packet Delivery company Identity Provider, in order to retrieve profile information about the Packet Delivery company employee user. The access token from the previous step 19 is provided as Bearer authorization token.

```
> Authorization: Bearer IIeDIRdnYo2ngwDQYJKoZIhvcNAQELBQAwSDEZMBcGA1UEAwwQaVNIQ
< Content-Type: application/json; charset=UTF-8
```

```
GET https://pdc-keyrock.i4trust-demo.fiware.dev/userinfo
```

21. Packet Delivery company Identity Provider [returns](#) the information about the Packet Delivery company employee user, based on the scopes in the request of step 5.

```
Content-Type: application/json
```

```
{
  "iss": "EU.EORI.NLPACKETDEL",
  "sub": "419404e1-07ce-4d80-9e8a-eca94vde0003de", // ID of employee user
  "organizations": [], // Keyrock attribute
  "roles": [ "customer" ], // Keyrock attribute
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "preferred_username": "j.doe",
  "email": "janedoe@example.com",
}
```

22. The Packet Delivery company employee is logged in to the Marketplace. The user is now able to create catalogs, products and offerings.
23. The Packet Delivery company employee chooses to create a new product and offering, within an existing catalog, for the Packet Delivery order service.
24. The user is asked to provide details about the product and offering.
25. The Packet Delivery company employee provides the details about the product and offering (either for Standard Delivery or Premium Delivery), which includes the target endpoint of the service provided  
(<https://pdc-kong.i4trust-demo.fiware.dev/packetdelivery/ngsi-ld/v1>), URL of the Authorisation Registry where to store the policies (for the moment it is the activation service in front of the AR <https://pdc-as.i4trust-demo.fiware.dev>) and its EORI (EU.EORI.NL000000004), the type of entities for which access will be granted (DELIVERYORDER), number of minutes until created policies should expire (can be set to a very large value in the case that the policies should never expire), and the allowed attributes for GET ("\*", since one should be able to read all information) and PATCH ("pta,pda" when allowing to change the PTA and PDA values for the premium offering) requests, respectively. Additionally, further information like price plans, license agreement and SLAs can be provided.
26. Product and Offering are created on the Marketplace and the user gets a successful response.

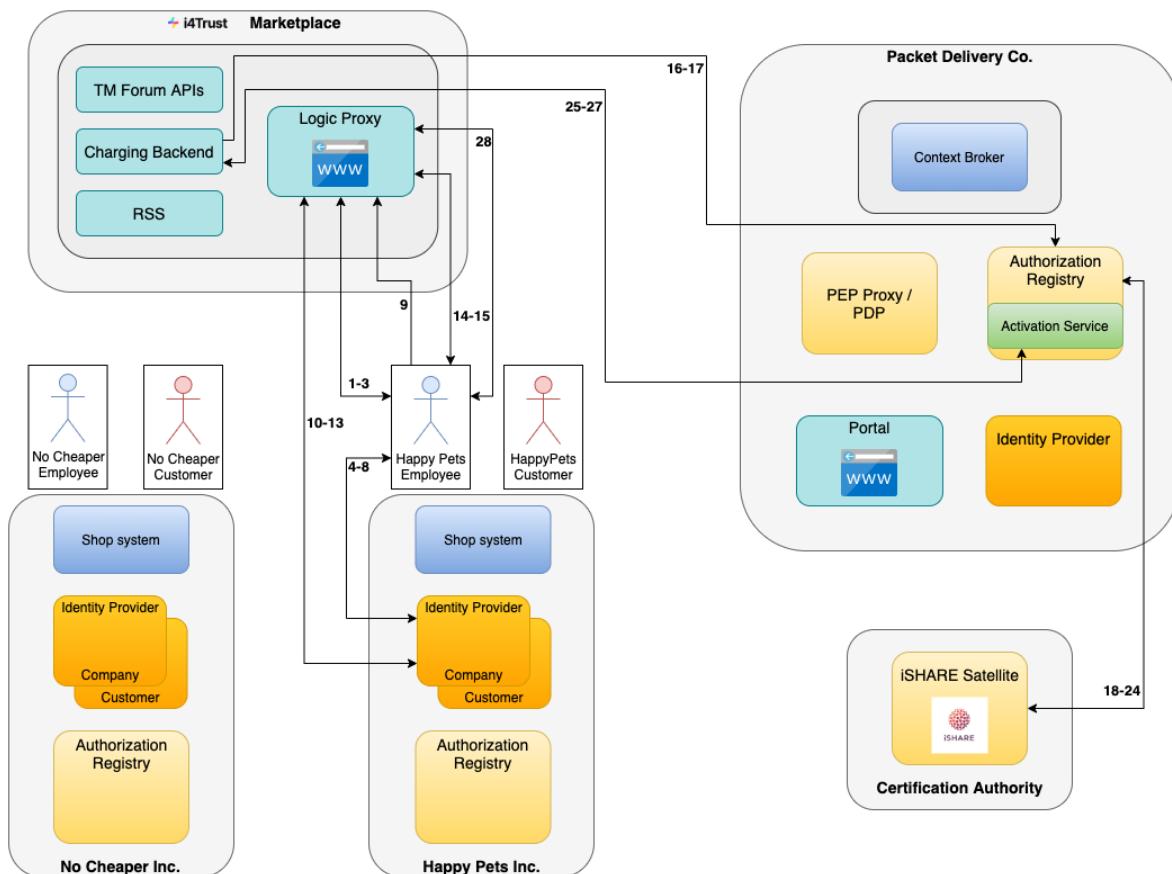
## 6.6 Acquisition of Rights / Activation

The process of acquiring access to the packet delivery service is displayed. It is performed by employees of both parties separately, Happy Pets and No Cheaper, where the former one acquires access to the "Premium Delivery" offering and the latter acquires the "Basic Delivery" offering.

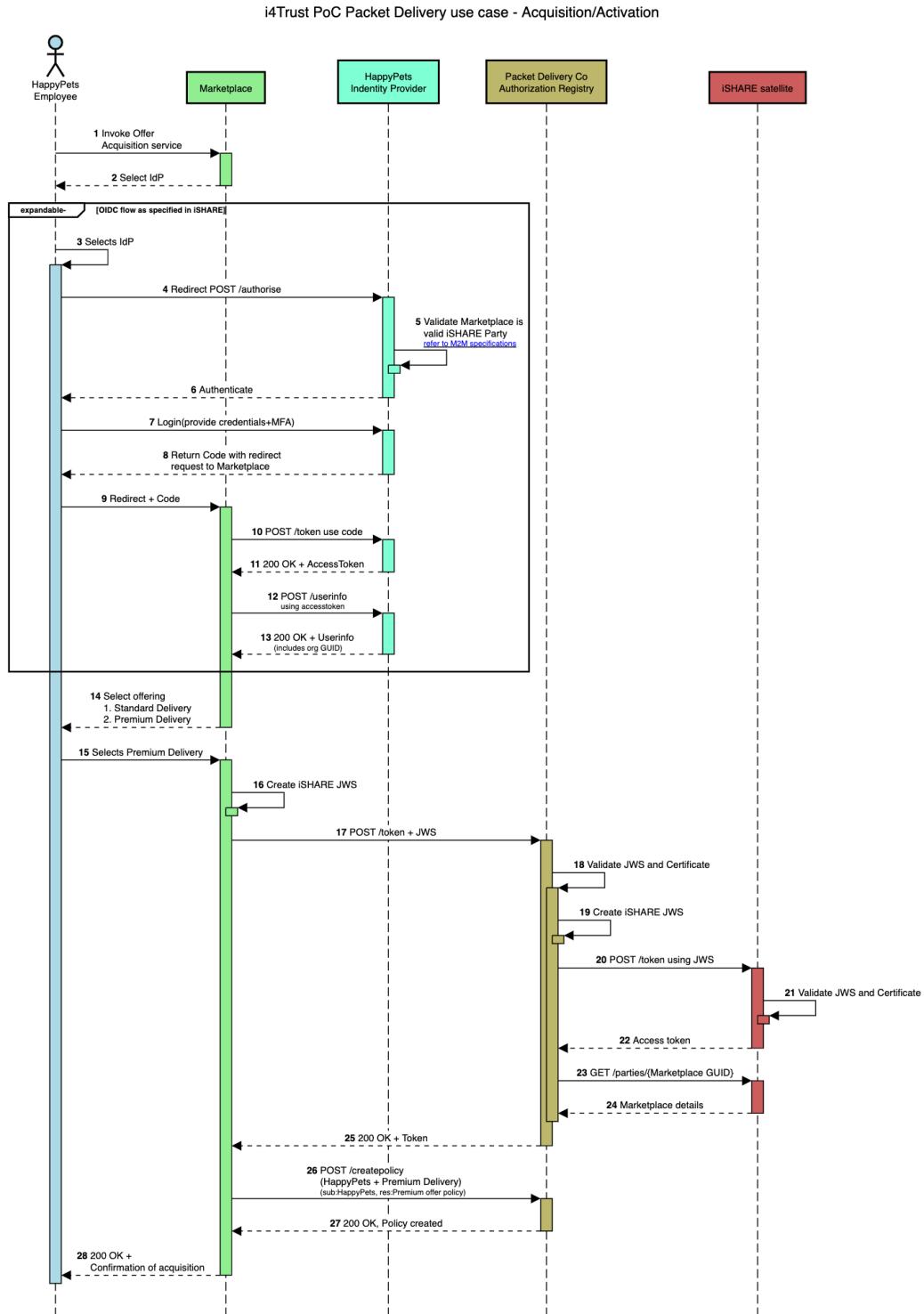
### 6.6.1 Sequence description (Happy Pets Inc.)

The following gives a detailed description of the acquisition process. Figure 6.5 presents the different interactions in an architectural overview, whereas Figure 6.6 shows a detailed sequence diagram of the whole process. The numberings in the architectural overview map to the different steps of the sequence diagram.

In the following, a description is given for each of the sequence steps.



*Figure 6.5: Architectural interactions for step "Acquisition by Happy Pets"*



*Figure 6.6: Sequence diagram for step "Acquisition by Happy Pets"*

1. Happy Pets employee accesses the Marketplace page (provided by the BAE Logic Proxy), in order to login.
2. Happy Pets employee gets forwarded to a page for selecting the desired Identity Provider for login.
3. (until 13.) These steps represent the standard OIDC flow, as described in steps 3-21 of the “Create offering” description in section 3.2.1. Here, the interactions take place between the Happy Pets employee, Marketplace and Happy Pets Identity Provider (Company IDP).
14. The Happy Pets employee is logged in to the Marketplace. The user is now able to browse available offerings. In order to acquire access to such an offering, the user needs to purchase it.
15. The Happy Pets employee chooses to acquire the “Premium Delivery” service offering of the Packet Delivery company. After finishing the checkout process on the Marketplace, the service has been acquired. The following steps of activation at the Packet Delivery company for Happy Pets will be performed afterwards by the Marketplace Charging Backend.
16. Marketplace Charging Backend generates an [iSHARE JWT](#) signed using the JSON Web Signature standard (JWS).

```
> Headers
{
  "alg": "RS256",
  "typ": "JWT",
  "x5c": [ // Complete certificate chain for validating JWT signature
    "MIIEhjCC....Zy9w=="
  ]
}

> Payload
{
  "iss": "EU.EORI.NLMARKETPLA", // Client-ID (EORI) of Marketplace
  "sub": "EU.EORI.NLMARKETPLA", // Client-ID (EORI) of Marketplace
  "aud": [
    "EU.EORI.NL000000000", // ID (EORI) and token-URL of Packet Delivery Co
    "authorisation registry
      "https://pdc-keyrock.i4trust-demo.fiware.dev/connect/token"
  ],
  "jti": "99ab5bca41bb45b78d242a46f0157b7d", // Unique JWT ID
  "iat": "1540827435",
  "exp": 1540827465,
  "nbf": 1540827435
}
```

17. Marketplace Charging Backend sends a request to the Packet Delivery company Authorisation Registry [/token](#) endpoint (note that in the current setup this is sent to the activation service component in front, as described in [6.3](#))

[Deployment of components](#)). The signed JWT created in step 16 is provided with the `client_assertion` parameter of the request.

```
> Content-Type: application/x-www-form-urlencoded  
  
POST https://pdc-as.i4trust-demo.fiware.dev/connect/token  
  
grant_type=client_credentials&  
scope=iSHARE&  
client_id=EU.EORI.NLMARKETPLA&  
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&  
client_assertion=eyJhbGc...9hvW
```

18. (until 24.) These steps represent the standard Party Validation routine, as described in steps 7-13 of the “Create offering” description in section 3.2.1. Here, the interactions take place between the Packet Delivery company Authorisation Registry and the iSHARE Satellite so that the Packet Delivery company Authorisation Registry can verify that the Marketplace (Charging Backend) is a trusted party.
25. The [access token](#) is returned to the Marketplace Charging Backend, in response to the request of step 17.

```
< Content-Type: application/json  
< Cache-Control: no-store  
< Pragma: no-cache  
  
{  
  "access_token": "aw2ys9NGE8RjHPZ4mytQivkWJ05HGQCYJ7VyMBGGDLIOW",  
  "expires_in": 3600,  
  "token_type": "Bearer"  
}
```

26. Marketplace Charging Backend sends a request to the `/createpolicy` endpoint of the Packet Delivery company Authorisation Registry (note that in the current setup this is sent to the activation service component in front, as described in [6.3 Deployment of components](#)), in order to create an access policy for Happy Pets allowing to read and update the attributes as described for the P.Info.gold role.

```
> Authorization: Bearer IIe...VNIQ // access_token from step 25  
  
POST https://pdc-as.i4trust-demo.fiware.dev/createpolicy  
  
> Payload  
{  
  "delegationEvidence": {  
    "notBefore": 1617796688,  
    "notOnOrAfter": 1617798488,  
    "policyIssuer": "EU.EORI.NLPACKETDEL",  
    "target": {  
      "accessSubject": "EU.EORI.NLHAPPYPETS"
```

```
        },
        "policySets": [
            {
                "maxDelegationDepth": 0,
                "target": {
                    "environment": {
                        "licenses": [
                            "ISHARE.0001"
                        ]
                    }
                },
                "policies": [
                    {
                        "target": {
                            "resource": {
                                "type": "DELIVERYORDER",
                                "identifiers": ["*"],
                                "attributes": ["pta", "pda"]
                            },
                            "actions": ["PATCH"]
                        },
                        "rules": [
                            {
                                "effect": "Permit"
                            }
                        ]
                    },
                    {
                        "target": {
                            "resource": {
                                "type": "DELIVERYORDER",
                                "identifiers": ["*"],
                                "attributes": ["*"]
                            },
                            "actions": ["GET"]
                        },
                        "rules": [
                            {
                                "effect": "Permit"
                            }
                        ]
                    }
                ]
            }
        ]
    }
```

27. A successful response with HTTP status 200 is returned to the Marketplace Charging Backend.
28. The acquisition and activation process has finished and the user gets a successful response.

## 6.6.2 Sequence description (No Cheaper Ltd)

The process is basically the same as for the acquisition described in the previous section 3.3.1. In this case the steps are processed by the No Cheaper employee user, acquiring access to the Basic Delivery service offering of Packet Delivery company. In the process, the Identity Provider of No Cheaper is used. An access policy is created for No Cheaper according to the role P.Info.standard.

Figure 6.7 shows the detailed sequence diagram for the process in the case that it is performed by No Cheaper.

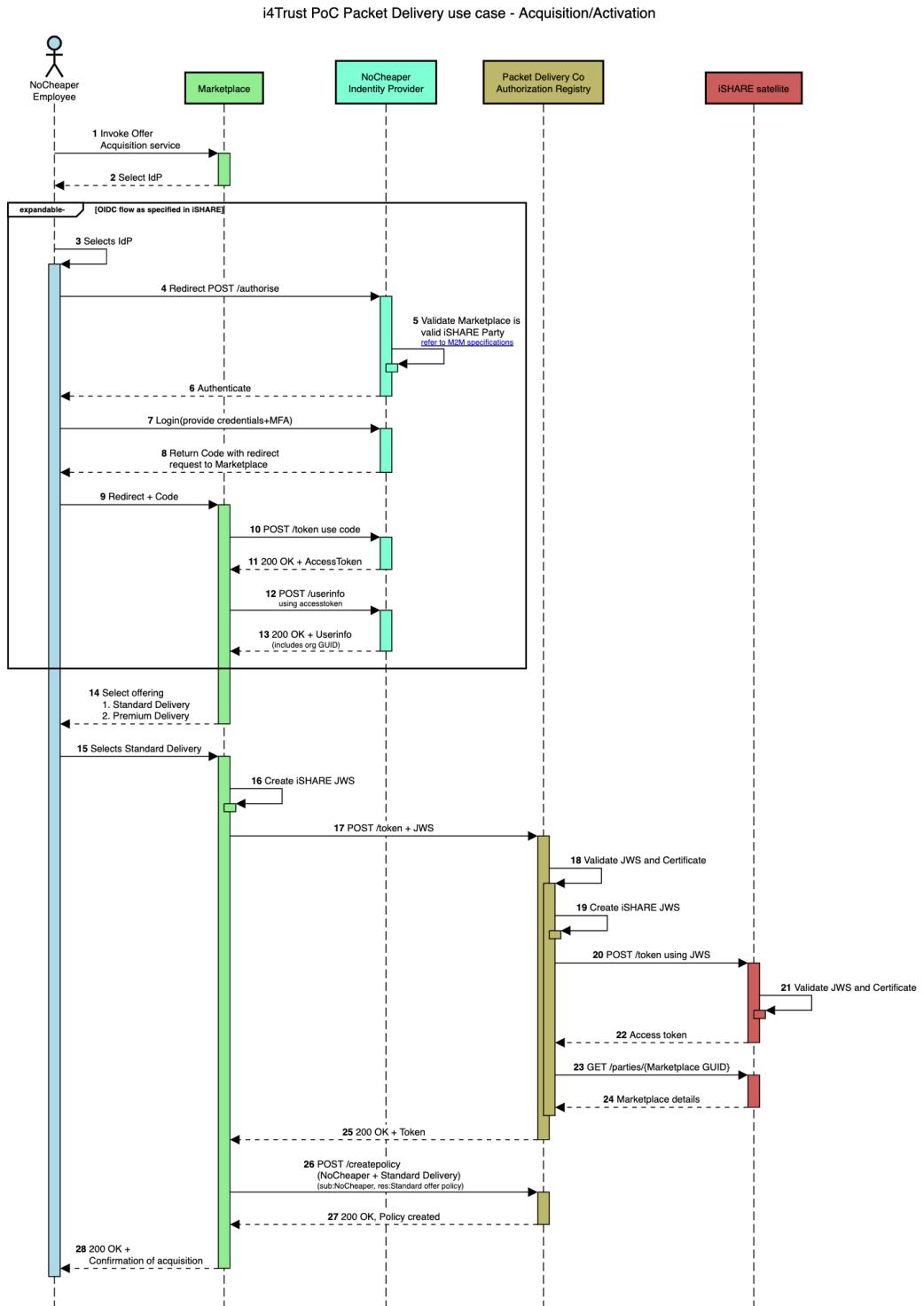


Figure 6.7: Sequence diagram for step "[Acquisition by No Cheaper](#)"

## 6.7 Access to data service

The process of changing the PTA attribute of a packet delivery order via the packet delivery portal is explained. The process would be similar, when trying to change the PDA or delivery address.

In the following the sequences are shown for the scenario of the Happy Pets customer changing the PTA of the delivery order. In the case of the No Cheaper customer, the sequences would be the same with the only difference being that the request for changing the PTA would be denied.

### 6.7.1 Sequence description (Happy Pets Customer)

The following gives a detailed description of the process of changing the PTA attribute by the Happy Pets customer. Figure 6.8 presents the different interactions in an architectural overview, whereas Figure 6.9 shows a detailed sequence diagram of the whole process. The numberings in the architectural overview map to the different steps of the sequence diagram.

In the following, a description is given for each of the sequence steps.

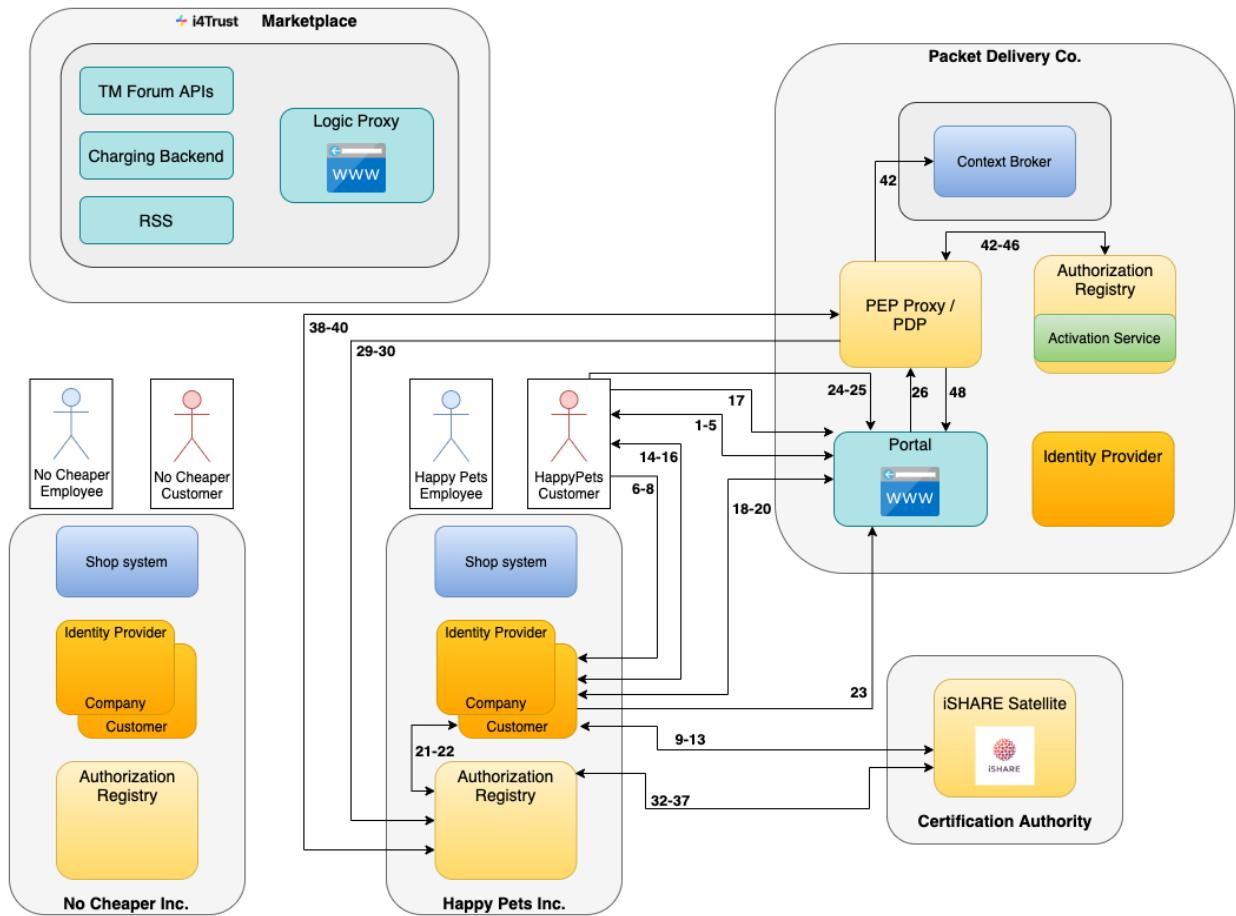


Figure 6.8: Architectural interactions for step "Change PTA by Happy Pets customer"

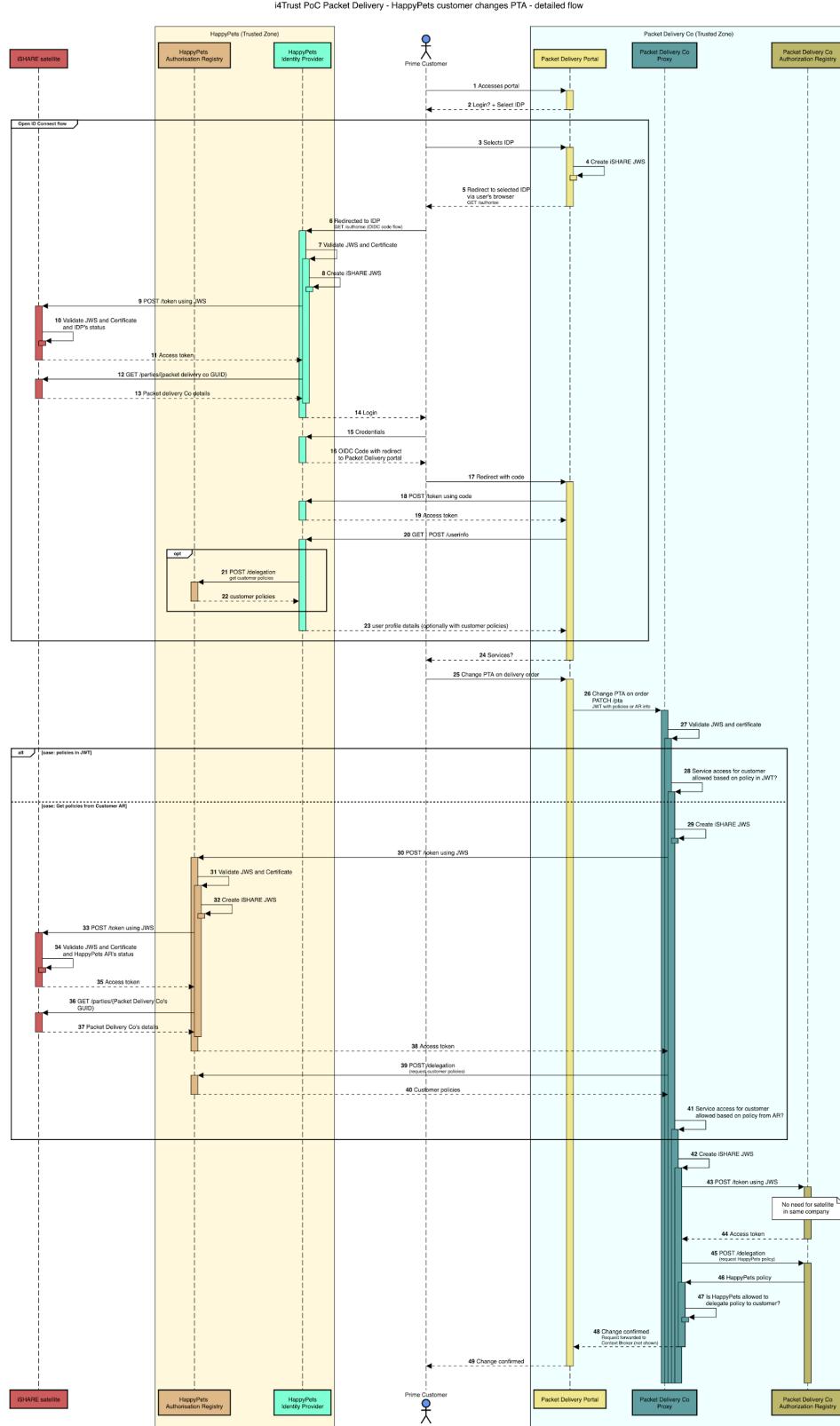


Figure 6.9: Sequence diagram for step “[Change PTA by Happy Pets customer](#)”

1. Happy Pets customer accesses the Packet delivery company portal or starts the Packet Delivery company app in its smartphone, in order to login.
2. Happy Pets customer gets forwarded to a page for selecting the desired Identity Provider for login.
3. Happy Pets customer selects the Identity Provider (Happy Pets Customer IDP).
4. Packet delivery company portal generates an [iSHARE JWT](#) signed using the JSON Web Signature standard (JWS). This also applies to all JWT generated in further steps.

```
> Headers
{
  "alg": "RS256",
  "typ": "JWT",
  "x5c": [ // Complete certificate chain for validating JWT signature (see step 7)
    "MIIEhjCC....Zy9w=="
  ]
}

> Payload
{
  "iss": "EU.EORI.NLPACKETDEL", // Client-ID of Packet Delivery company (EORI)
  "sub": "EU.EORI.NLPACKETDEL", // Client-ID of Packet Delivery company (EORI)
  "aud": [
    "EU.EORI.NLHAPPYPETS", // ID (EORI) and token-URL of Happy pets IDP
    "https://happypets-keyrock-shop.i4trust-demo.fiware.dev/connect/token"
  ],
  "jti": "99ab5bca41bb45b78d242a46f0157b7d", // Unique JWT ID
  "iat": "1540827435",
  "exp": 1540827465,
  "nbf": 1540827435
}
```

5. Happy Pets customer is redirected to the Identity Provider of Happy Pets (Customer IDP) via it's browser. The request sent to the [/authorise](#) endpoint contains the signed JWT created in step 4 in the request URL parameter. The values of the [scope](#) parameter specify the access privileges being requested, and give control over what will be returned to the request sent to the /userinfo endpoint in step 23. Due to the delegation\_evidence scope, the customers policies will be retrieved as delegation evidence along with the user info. In the case that the policies need to be retrieved from a separate authorisation registry, the information for this authorisation registry would be provided along with the user info due to the [authorisation\\_registry](#) scope

```
> Content-Type: application/x-www-form-urlencoded
POST https://happypets-keyrock-shop.i4trust-demo.fiware.dev/authorise
response_type=code&
```

```

client_id=EU.EORI.NLPACKETDEL&
scope=iSHARE openid&
request=eyJ0eXA...YkNK0Q

Decoded Request parameter
{
  "iss": "EU.EORI.NLPACKETDEL",
  "sub": "EU.EORI.NLPACKETDEL",
  "aud": "EU.EORI.NLHAPPYPETS",
  "jti": "378a47c4-2822-4ca5-a49a-7e5a1cc7ea59",
  "iat": 1504683445,
  "exp": 1504683475,
  "response_type": "code",
  "client_id": "EU.EORI.NLPACKETDEL",
  "scope": "openid iSHARE sub name contact_details delegation_evidence
authorisation_registry",
  "redirect_uri":
  "https://pdc-portal.i4trust-demo.fiware.dev/openid_connect1.0/return",
  "state": "af0ifjsldkj",
  "nonce": "c428224ca5a",
  "acr_values": "urn:http://eidas.europa.eu/LoA/NotNotified/high",
  "language": "en"
}

```

6. The request of step 5 is forwarded to the Happy Pets Identity Provider (Customer IDP).
7. Happy Pets Identity Provider (Customer IDP) validates the JWT and certificate of the Packet Delivery company portal.
8. Happy Pets Identity Provider (Customer IDP) generates an iSHARE JWT (also see description in step 4).

```

> Headers
{
  "alg": "RS256",
  "typ": "JWT",
  "x5c": [
    "MII EhjCC....Zy9w=="
  ]
}

> Payload
{
  "iss": "EU.EORI.NLHAPPYPETS", // Client-ID (EORI) of Happy Pets
  "sub": "EU.EORI.NLHAPPYPETS", // Client-ID (EORI) of Happy Pets
  "aud": [
    "EU.EORI.NL000000000", // ID (EORI) and token-URL of iSHARE Satellite
    "https://scheme.isharetest.net/connect/token"
  ],
  "jti": "99ab5bca41bb45b78d242a46f0157b7d", // Unique JWT ID
  "iat": "1540827435",
  "exp": 1540827465,
}

```

```
        "nbf": 1540827435
    }
```

9. Happy Pets Identity Provider (Customer IDP) sends a request to the iSHARE Satellite [/token](#) endpoint. The signed JWT created in step 8 is provided with the `client_assertion` parameter of the request.

```
> Content-Type: application/x-www-form-urlencoded
POST https://scheme.isharetest.net/connect/token
grant_type=client_credentials&
scope=iSHARE&
client_id=EU.EORI.NLHAPPYPETS&
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&
client_assertion=eyJhbGc...9hvW
```

10. iSHARE Satellite validates the JWT and certificate, and checks the status of the Happy Pets Identity Provider (also see step 7). If it is no valid iSHARE participant, an error is returned.
11. The [access token](#) is returned to the Happy Pets Identity Provider (Customer IDP).

```
< Content-Type: application/json
< Cache-Control: no-store
< Pragma: no-cache

{
  "access_token": "aW2ys9NGE8RjHPZ4mytQivkWJ05HGQCYJ7VyMBGGDLIOw",
  "expires_in": 3600,
  "token_type": "Bearer"
}
```

12. Happy Pets Identity Provider (Customer IDP) sends a request to the [/parties](#) endpoint of the iSHARE Satellite, in order to retrieve information about the Packet Delivery Company for verification of its status as iSHARE participant. The access token from the previous step 11 is provided as Bearer authorization token. The request contains the Packet Delivery company EORI as query parameter, as well as the subject name as encoded in the certificate of the Packet Delivery company.

```
> Authorization: Bearer IIe...VNIQ // access_token from step 11
GET https://scheme.isharetest.net/parties?
  eori=EU.EORI.NLPACKETDEL&
  certificate_subject_name=C=NL, SERIALNUMBER=EU.EORI.NLPACKETDEL, CN=Packet
  Delivery Co&
  active_only=true
```

13. iSHARE Satellite returns the information about the Packet Delivery company. The `party_info.adherence.status` field contains the status of the Packet Delivery company. When there is an error response, or the status is not active, then an error is presented to the Happy Pets customer.

```
> Content-Type: application/json
{
  "parties_token": "eyJ4N...xY_aaQ"
}

Decoded parties_token parameter
{
  "iss": "EU.EORI.NL00000000",
  "sub": "EU.EORI.NLPACKETDEL",
  "jti": "77e8179fbfe6469eb64c054da26a77c3",
  "iat": 1589282112,
  "exp": 1589282142,
  "aud": "EU.EORI.NLHAPPYPETS",
  "party_info": {
    "party_id": "EU.EORI.NLPACKETDEL",
    "party_name": "Packet Delivery Co",
    "adherence": {
      "status": "Active",
      "start_date": "2020-04-26T00:00:00",
      "end_date": "2021-07-25T00:00:00"
    },
    "certifications": [
    ],
    "capability_url": "https://pdc-keyrock.i4trust-demo.fiware.dev/capabilities"
  }
}
```

14. The login page of the Happy Pets Identity Provider (Customer IDP) is presented to the Happy Pets customer.
15. Happy Pets customer provides its credentials on the login page of Happy Pets Identity Provider (Customer IDP).
16. Happy Pets Identity Provider (Customer IDP) returns the code according to OpenID Connect to the [redirect URI](#) of the Packet Delivery company portal/app, as provided in the request of step 5.

```
< Location: https://pdc-portal.i4trust-demo.fiware.dev/openid_connect1.0/return?
  code=Dmn-TbSj70cK15ym1j5xZsgkabzVP8dMugC81nzmeW4&
  state=ZqVQm4zHaEDyBhzpm1ZRH7fsxy7031q2
```

17. The code from step 16 is redirected to the Packet Delivery company portal/app.
18. Packet Delivery company portal/app sends a request to the [/token](#) endpoint of the Happy Pets Identity Provider (Customer IDP). The signed JWT created in

step 4 is provided with the `client_assertion` parameter of the request. The code from steps 16/17 is provided with the `code` parameter.

```
> Content-Type: application/x-www-form-urlencoded  
  
POST https://happypets-keyrock-shop.i4trust-demo.fiware.dev/token  
  
grant_type=authorization_code&  
client_id=EU.EORI.NLPACKETDEL&  
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&  
client_assertion=eyJhbGc...9hvW&  
redirect_uri=https://happypets-keyrock-shop.i4trust-demo.fiware.dev/openid_connect1.0/  
return&  
code=Dmn-TbSj70cK15ym1j5xZsgkabzVP8dMugC81nzmew4
```

19. The [access token](#) is returned to the Packet Delivery company portal. In addition, an iSHARE JWT compliant [id\\_token](#) is returned which is associated with the authenticated session.

```
< Content-Type: application/json  
< Cache-Control: no-store  
< Pragma: no-cache  
  
{  
  "id_token": "eyJhb...V2jA",  
  "access_token": "alWys9NGE8RjHPZ4mytQivkJ05HGQCYJ7VyMBGGDLIOw",  
  "expires_in": 3600,  
  "token_type": "Bearer"  
}  
  
Decoded id_token parameter  
{  
  "iss": "EU.EORI.NLHAPPYPETS",  
  "sub": "419404e1-07ce-4d80-9e8a-eca94vde0003de",  
  "aud": "EU.EORI.NLPACKETDEL",  
  "jti": "378a47c4-2822-4ca5-a49a-7e5a1cc7ea59",  
  "iat": 1504683445,  
  "exp": 1504683475,  
  "auth_time": 1504683435,  
  "nonce": "c428224ca5a",  
  "acr": "urn:http://eidas.europa.eu/LoA/NotNotified/low",  
  "azp": "EU.EORI.NLPACKETDEL",  
}
```

20. Packet Delivery company portal/app sends a request to the [userinfo](#) endpoint of Happy Pets Identity Provider (Customer IDP), in order to retrieve information about the Happy Pets customer user including its delegation evidence/policies. The access token from the previous step 19 is provided as Bearer authorization token.

In the case that the policy delegation information is stored in a separate authorisation registry for Happy Pets, the additional call described in steps 21/22 is required in order to retrieve this information which will enrich the response to the request to this /userinfo endpoint.

Otherwise, when the identity provider holds this information itself, a business logic is required to gather this information and provide it with the response.

```
> Authorization: Bearer IIeDIRdnYo2ngwDQYJKoZIhvcNAQELBQAwSDEZMBcGA1UEAwQaVNIQ  
< Content-Type: application/json; charset=UTF-8
```

```
GET https://happypets-keyrock-shop.i4trust-demo.fiware.dev/userinfo
```

21. (Optional step to be performed in the case that profile info/policy delegation information is stored in separate Happy Pets Authorization Registry) The profile information of the Happy Pets customer is requested from the Happy Pets Authorization Registry.

```
> Authorization: Bearer IIe...NIQ // OAuth2 Bearer token (obtained by M2M OAuth2)  
> Content-Type: application/json

POST 'https://ar.isharetest.net/delegation'

> Payload
{
  "delegationRequest": {
    "policyIssuer": "EU.EORI.HAPPYPETS",      //Creator/Owner of the policy, since this
                                                //is in request, the EORI of the company whose policy is in question - happy pets
    "target": {
      "accessSubject": "419404e1-07ce-4d80-9e8a-eca94vde0003de"    //Subject for whom
                                                                //this resource request is for - user pseudonym
    },
    "policySets": [
      {
        "policies": [
          {
            "target": {
              "resource": {
                //Free text attributes!!
                "type": "DELIVERYORDER",    //Resource category, Orders in this case
                "identifiers": [
                  "*"    //Identifier of the resource, Order id in this case
                ],
                "attributes": [    //Attributes about the resource
                  "*"
                ]
              },
              "actions": [
                "PATCH"
              ],
              "rules": [
                {
                  "effect": "Permit"
                }
              ]
            }
          }
        ]
      }
    ]
  }
}
```

```

        }
    ]
}
}
}
```

22. (Optional step in the case that profile info/policy delegation information is stored in separate Happy Pets Authorization Registry) The profile information of the Happy Pets customer is returned from the Happy Pets Authorization Registry

```

> decoded JWT
{
  "iss": "EU.EORI.NLHAPPYPETS",
  "sub": "EU.EORI.NLHAPPYPETS",
  "jti": "d8a7fd7465754a4a9117ee28f5b7fb60",
  "iat": 1591966224,
  "exp": 1591966254,
  "aud": "EU.EORI.NLHAPPYPETS",
  "delegationEvidence": {
    "notBefore": 1541058939,
    "notOnOrAfter": 2147483647,
    "policyIssuer": "EU.EORI.NLHAPPYPETS",
    "target": {
      "accessSubject": "419404e1-07ce-4d80-9e8a-eca94vde0003de" // ID of customer
    },
    "policySets": [
      {
        "maxDelegationDepth": 0,
        "target": {
          "environment": {
            "licenses": [
              "ISHARE.0001"
            ]
          }
        },
        "policies": [
          {
            "target": {
              "resource": {
                "type": "DELIVERYORDER",
                "identifiers": [
                  "urn:ngsi-ld:DELIVERYORDER:HAPPYPETS001" //Currently only one
order belongs to this customer so only one order id
                ],
                "attributes": [
                  "pta", "pda", "deliveryAddress" //This is based on
Prime customer
                ]
              }
            },
            "actions": [
              "GET", "PATCH"
            ]
          },
          "rules": [
            {

```

```
        "effect": "Permit"
    }
]
}
]
}
}
}
```

23. Happy Pets Identity Provider (Customer IDP) [returns](#) the information about the Happy Pets customer (optionally enriched with the information from step 22), according to the scopes of the request in step 5.

Content-Type: application/json

```
{
  "iss": "EU.EORI.NLHAPPYPETS",
  "sub": "419404e1-07ce-4d80-9e8a-eca94vde0003de", // ID of customer user
  "organizations": [], // Keyrock attribute
  "roles": [], // Keyrock attribute
  "name": "Jane Doe",
  "given_name": "Jane",
  "family_name": "Doe",
  "preferred_username": "j.doe",
  "email": "janedoe@example.com",
  "authorisationRegistry": {}
  "delegationEvidence": {
    "notBefore": 1541058939,
    "notOnOrAfter": 2147483647,
    "policyIssuer": "EU.EORI.NLHAPPYPETS",
    "target": {
      "accessSubject": "419404e1-07ce-4d80-9e8a-eca94vde0003de" // ID of customer
user
    },
    "policySets": [
      {
        "maxDelegationDepth": 0,
        "target": {
          "environment": {
            "licenses": [
              "ISHARE.0001"
            ]
          }
        },
        "policies": [
          {
            "target": {
              "resource": {
                "id": "419404e1-07ce-4d80-9e8a-eca94vde0003de"
              }
            }
          }
        ]
      }
    ]
  }
}
```

```
        "type": "DELIVERYORDER",
        "identifiers": [
            "urn:ngsi-ld:DELIVERYORDER:HAPPYPETS001"      //Currently only one
order belongs to this customer so only one order id
        ],
        "attributes": [
            "pta", "pda", "deliveryAddress"      //This is based on Prime
customer
        ]
    },
    "actions": [
        "PATCH", "GET"
    ],
    "rules": [
        {
            "effect": "Permit"
        }
    ]
}
]
```

In the case that the policies of the customer are stored in a separate authorisation registry, and are not provided by the Identity Provider's user info response, information about this authorisation registry would be provided in place of the policies, according to the `authorisation_registry` scope requested in step 5.

```
Content-Type: application/json

{
    "iss": "EU.EORI.NLHAPPYPETS",
    "sub": "419404e1-07ce-4d80-9e8a-eca94vde0003de",
    "organizations": [],
    "roles": [],
    "name": "Jane Doe",
    "given_name": "Jane",
    "family_name": "Doe",
    "preferred_username": "j.doe",
    "email": "janedoe@example.com",
    "authorisationRegistry": {
        "url": "https://ar.isharetest.net",
        "identifier": "EU.EORI.NLHAPPYPETS",
        "token_endpoint": "https://ar.isharetest.net/connect/token",
    }
}
```

```
        "delegation_endpoint": "https://ar.isharetest.net/delegation",
    }
    "delegation_evidence": { }
}

}
```

24. Happy Pets customer is logged in on the Packet Delivery company portal/app and is presented the possible services provided, including the option to change the PTA of its delivery orders.
25. Happy Pets customer searches for his packet delivery order and is presented its details. He now requests a change of the PTA of this order on the Packet Delivery company portal/app.
26. Packet Delivery company portal/app sends a request to Packet Delivery company proxy, in order to change the PTA of the delivery order. The request contains a [signed iSHARE compliant JWT](#) as Bearer token which includes the Happy Pets customer user information from step 23 with the policies in the delegation evidence (or information about the authorisation registry to retrieve these policies from, see below).

```
> Authorization: Bearer IIeD...NIQ // Bearer JWT
> Content-Type: application/json

PATCH
https://pdc-kong.i4trust-demo.fiware.dev/ngsi-ld/v1/entities/urn:ngsi-ld:DELIVERYORDER:001/attrs/pta

> Payload
{
  "value": "<new PTA>",
  "type": "Property"
}

Decoded Bearer JWT:
{
  "iss": "EU.EORI.NLHAPPYPETS",
  "sub": "419404e1-07ce-4d80-9e8a-eca94vde0003de",
  "jti": "d8a7fd7465754a4a9117ee28f5b7fb60",
  "iat": 1591966224,
  "exp": 1591966254,
  "aud": "EU.EORI.NLHAPPYPETS",
  "delegationEvidence": {
    "notBefore": 1541058939,
    "notOnOrAfter": 2147483647,
    "policyIssuer": "EU.EORI.NLHAPPYPETS",
    "target": {
      "accessSubject": "419404e1-07ce-4d80-9e8a-eca94vde0003de" // ID of customer
    },
    "policySets": [
      {
        "maxDelegationDepth": 0,
        "target": {
          "environment": {
            "id": "urn:ngsi-ld:DeliveryOrder:001"
          }
        }
      }
    ]
  }
}
```

```
        "licenses": [
            "ISHARE.0001"
        ]
    },
    "policies": [
        {
            "target": {
                "resource": {
                    "type": "DELIVERYORDER",
                    "identifiers": [
                        "urn:ngsi-ld:DELIVERYORDER:HAPPYPETS001"
                    ],
                    "attributes": [
                        "pta", "pda", "deliveryAddress"
                    ]
                },
                "environment": {
                    "serviceProviders": [
                        "EU.EORI.NLPACKETDEL"
                    ]
                }
            },
            "actions": [
                "PATCH"
            ],
            "rules": [
                {
                    "effect": "Permit"
                }
            ]
        },
        {
            "target": {
                "resource": {
                    "type": "DELIVERYORDER",
                    "identifiers": [
                        "urn:ngsi-ld:DELIVERYORDER:HAPPYPETS001"
                    ],
                    "attributes": [
                        "*"
                    ]
                },
                "environment": {
                    "serviceProviders": [
                        "EU.EORI.NLPACKETDEL"
                    ]
                }
            },
            "actions": [
                "GET"
            ],
            "rules": [
                {
                    "effect": "Permit"
                }
            ]
        }
    ]
}
```

```
        }
    ]
}
]
}
```

In the case that the policies need to be obtained from a separate authorisation registry, the JWT would contain the necessary information about this authorisation registry as described in step 23.

```
> Authorization: Bearer IIeD...NIQ // Bearer JWT
> Content-Type: application/json

PATCH
https://pdc-kong.i4trust-demo.fiware.dev/ngsi-ld/v1/entities/urn:ngsi-ld:DELIVERYORDER:001/attrs/pta

> Payload
{
  "value": "<new PTA>",
  "type": "Property"
}

Decoded Bearer JWT payload:
{
  "iss": "EU.EORI.NLHAPPYPETS", // Issuer: Happy Pets
  "sub": "419404e1-07ce-4d80-9e8a-eca94vde0003de", // Customer pseudonym
  "jti": "d8a7fd7465754a4a9117ee28f5b7fb60",
  "iat": 1591966224,
  "exp": 1591966254,
  "aud": "EU.EORI.NLHAPPYPETS",
  "authorisationRegistry": { // AR to retrieve policies from
    "url": "https://ar.isharetest.net",
    "identifier": "EU.EORI.NLHAPPYPETS",
    "token_endpoint": "https://ar.isharetest.net/connect/token",
    "delegation_endpoint": "https://ar.isharetest.net/delegation",
  }
}
```

27. Packet Delivery company proxy received the request of step 26 for changing the PTA of a delivery order. It must make sure that the issuer of the PATCH request, Happy Pets customer, was assigned the delegation evidence with a policy for updating the PTA attribute of this specific delivery order. Furthermore, since in this scenario the required customer policy was issued by a 3rd party (Happy Pets), the proxy has to check whether Happy Pets itself is allowed to delegate this policy. In general, the rule would be that the proxy needs to check the existence of valid policies through the chain of issuers, until itself (in this case the Packet Delivery company) is the issuer. In this scenario, the proxy will check policies at two different levels: issued at **organizational level** (from Packet Delivery company to Happy Pets) and issued at **user level**

(from Happy Pets to customer). Policies at both levels must match the policy access requirements in order to allow access to the service.

At first, the Packet Delivery company proxy validates the JWT and certificate which is part of the authorization header of the PATCH request (also see step 7).

28. In the case that the Happy Pets customer delegation evidence was part of the JWT in the request of step 26, the proxy (or more precisely, the PDP) can evaluate whether the contained policy allows for updating the PTA attribute of the specific delivery order. If the Happy Pets customer was assigned the P.Info.gold role in the Happy Pets shop system, the necessary policy is part of the delegation evidence in the JWT and access for changing the PTA is granted on **user level**.

In the case that it was not assigned the P.Info.gold role, the necessary policy would be missing and changing the PTA would be denied by the Packet Delivery company proxy. As a result an error would be returned to the Packet Delivery company portal, also presented to the Happy Pets customer. The following steps would be omitted.

In the case that the JWT from step 26 does not contain the Happy Pets customer delegation evidence, the optional steps 29-41 get executed, where this information is retrieved from the Happy Pets Authorization Registry.

Information about this Authorization Registry (including the endpoint) would be contained in the response of step 23 and be part of the authorisation JWT of step 26.

29. (Optional) The following steps 29-41 get executed in case that the customer policies were not part of the authorization header JWT and must be retrieved from the Happy Pets Authorisation Registry.  
Packet Delivery company proxy generates an iSHARE JWT (also see description in step 4).

```
> Headers
{
  "alg": "RS256",
  "typ": "JWT",
  "x5c": [
    "MIIIEhjCC....Zy9w=="
  ]
}

> Payload
{
  "iss": "EU.EORI.NLPACKETDEL", // Client-ID (EORI) of Packet Delivery company
  "sub": "EU.EORI.NLPACKETDEL", // Client-ID (EORI) of Packet Delivery company
  "aud": [
    "EU.EORI.NLHAPPYPETS", // ID (EORI) and token-URL of Happy Pets Authz
    "https://ar.isharetest.net/connect/token"
}
```

```
        ],
        "jti": "99ab5bca41bb45b78d242a46f0157b7d", // Unique JWT ID
        "iat": "1540827435",
        "exp": 1540827465,
        "nbf": 1540827435
    }
```

30. (Optional, see step 29) Packet Delivery company proxy sends a request to the Happy Pets Authorization Registry [/token](#) endpoint, taken from the user info obtained in step 23. The signed JWT created in step 30 is provided with the `client_assertion` parameter of the request. Using the token, the Packet Delivery company proxy will retrieve the delegation evidence/policy information of the Happy Pets customer from the Happy Pets authorisation registry in step 40.

```
> Content-Type: application/x-www-form-urlencoded

POST https://ar.isharetest.net/token

grant_type=client_credentials&
scope=iSHARE&
client_id=EU.EORI.NLPACKETDEL&
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&
client_assertion=eyJhbGc...9hvW
```

31. (Optional, see step 29) Happy Pets Authorization Registry validates the JWT and certificate of Packet Delivery company proxy (also see step 7).
32. (Optional, see step 29) Happy Pets Authorization Registry generates an iSHARE JWT (also see description in step 4).

```
> Headers
{
  "alg": "RS256",
  "typ": "JWT",
  "x5c": [
    "MIIIEhjCC....Zy9w=="
  ]
}

> Payload
{
  "iss": "EU.EORI.NLHAPPYPETS", // Client-ID (EORI) of Happy Pets
  "sub": "EU.EORI.NLHAPPYPETS", // Client-ID (EORI) of Happy Pets
  "aud": [
    "EU.EORI.NL000000000", // ID (EORI) and token-URL of iSHARE Satellite
    "https://scheme.isharetest.net/connect/token"
  ],
  "jti": "99ab5bca41bb45b78d242a46f0157b7d", // Unique JWT ID
}
```

```
        "iat": "1540827435",
        "exp": 1540827465,
        "nbf": 1540827435
    }
```

33. (Optional, see step 29) Happy Pets Authorization Registry sends a request to the iSHARE Satellite [/token](#) endpoint. The signed JWT created in step 33 is provided with the `client_assertion` parameter of the request.

```
POST https://scheme.isharetest.net/connect/token

grant_type=client_credentials&
scope=iSHARE&
client_id=EU.EORI.NLHAPPYPETS&
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&
client_assertion=eyJhbGc...9hwv
```

34. (Optional, see step 29) iSHARE Satellite validates the JWT and certificate, and checks the status of the Happy Pets Authorization Registry (also see step 7). If it is no valid iSHARE participant, an error is returned.
35. (Optional, see step 29) The [access token](#) is returned to the Happy Pets Authorization Registry.

```
< Content-Type: application/json
< Cache-Control: no-store
< Pragma: no-cache

{
    "access_token": "aw2ys9NGE8RjHPZ4mytQivkWJ05HGQCYJ7VyMBGGDLIOw",
    "expires_in": 3600,
    "token_type": "Bearer"
}
```

36. (Optional, see step 29) Happy Pets Authorization Registry sends a request to the [/parties](#) endpoint of the iSHARE Satellite, in order to retrieve information about the Packet Delivery Company for verification of its status as iSHARE participant. The access token from the previous step 36 is provided as Bearer authorization token. The request contains the Packet Delivery company EORI as query parameter, as well as the subject name as encoded in the certificate of the Packet Delivery company.

```
> Authorization: Bearer IIe...VNIQ // access_token from step 11

GET https://scheme.isharetest.net/parties?
    eori=EU.EORI.NLPACKETDEL&
    certificate_subject_name=C=NL, SERIALNUMBER=EU.EORI.NLPACKETDEL, CN=iSHARE Test
    Authorization Registry&
    active_only=true
```

37. (Optional, see step 29) iSHARE Satellite returns the information about the Packet Delivery company. The `party_info.adherence.status` field contains the status of the Packet Delivery company. When there is an error response, or the status is not active, then an error is returned to the Packet Delivery company proxy and finally presented to the Happy Pets customer.

```
> Content-Type: application/json
{
  "parties_token": "eyJ4N...xY_aaQ"
}

Decoded parties_token parameter
{
  "iss": "EU.EORI.NL000000000",
  "sub": "EU.EORI.NL000000000",
  "jti": "77e8179fbfe6469eb64c054da26a77c3",
  "iat": 1589282112,
  "exp": 1589282142,
  "aud": "EU.EORI.NLHAPPYPETS",
  "party_info": {
    "party_id": "EU.EORI.NLPACKETDEL",
    "party_name": "Packet Delivery Co",
    "adherence": {
      "status": "Active",
      "start_date": "2020-04-26T00:00:00",
      "end_date": "2021-07-25T00:00:00"
    },
    "certifications": [
    ],
    "capability_url": "https://pdc-keyrock.i4trust-demo.fiware.dev/capabilities"
  }
}
```

38. (Optional, see step 29) An [access token](#) is returned to the Packet Delivery company proxy in response to the request of step 30.

```
< Content-Type: application/json
< Cache-Control: no-store
< Pragma: no-cache

{
  "access_token": "alw2ys9NGE8RjHPZ4mytQivkWJ05HGQCYJ7VyMBGGDLIOw",
  "expires_in": 3600,
  "token_type": "Bearer"
}
```

39. (Optional, see step 29) According to the description in step 29, the Packet Delivery company proxy sends a request to the [/delegation](#) endpoint of the

Happy Pets Authorization Registry, in order to retrieve the delegation evidence of the Happy Pets customer and check for the necessary policy. The access\_token from step 38 is added as bearer token. The signed JWT with the user info contained in the authorisation header of the PATCH request in step 26 is added in the parameter previous\_steps, in order to prove to the authorisation registry that there is a valid reason to ask for the policy.

```
> Authorization: Bearer IIe...NIQ // OAuth2 Bearer token (obtained by M2M OAuth2)
> Content-Type: application/json

POST 'https://ar.isharetest.net/delegation'

> Payload
{
  "delegationRequest": {
    "policyIssuer": "EU.EORI.NLHAPPYPETS",      //Creator/Owner of the policy, since
    //this is in request, the EORI of the company whose policy is in question - Packet
    //Delivery Co. or HappyPets?
    "target": {
      "accessSubject": "419404e1-07ce-4d80-9e8a-eca94vde0003de"    //Subject for whom
      //this resource request is for - HappyPets customer ID
    },
    "policySets": [
      {
        "policies": [
          {
            "target": {
              "resource": {
                "type": "DELIVERYORDER",    //Resource category, Orders in this case
                "identifiers": [
                  "urn:ngsi-ld:DELIVERYORDER:HAPPYPETS001"    //Identifier of the
                  //resource, Order id in this case
                ],
                "attributes": [    //Attributes about the resource
                  "pta"
                ]
              },
              "actions": [
                "PATCH"
              ],
              "rules": [
                {
                  "effect": "Permit"
                }
              ]
            }
          }
        ]
      },
      {
        "previous_steps": [ <signed JWT of step 26> ]
      }
    ]
  }
}
```

40. (Optional, see step 29) Happy Pets Authorization Registry [returns](#) the delegation information as JWT delegation\_token.

```
< Content-Type: application/json
> Payload
{
  "delegation_token": "eyJ4...OaaQ"
}

Decoded delegation_token JWT payload
{
  "iss": "EU.EORI.NLHAPPYPETS",
  "sub": "EU.EORI.NLHAPPYPETS",
  "jti": "d8a7fd7465754a4a9117ee28f5b7fb60",
  "iat": 1591966224,
  "exp": 1591966254,
  "aud": "EU.EORI.NLPACKETDEL",
  "delegationEvidence": {
    "notBefore": 1541058939,
    "notOnOrAfter": 2147483647,
    "policyIssuer": "EU.EORI.NLHAPPYPETS",
    "target": {
      "accessSubject": "419404e1-07ce-4d80-9e8a-eca94vde0003de"
    },
    "policySets": [
      {
        "maxDelegationDepth": 0,
        "target": {
          "environment": {
            "licenses": [
              "ISHARE.0001"
            ]
          }
        }
      ],
      "policies": [
        {
          "target": {
            "resource": {
              "type": "DELIVERYORDER",
              "identifiers": [
                "urn:ngsi-ld:DELIVERYORDER:HAPPYPETS001"
              ],
              "attributes": [
                "pta"
              ]
            },
            "actions": [
              "PATCH"
            ]
          },
          "rules": [
            {
              "rule": "allow"
            }
          ],
          "conditions": [
            {
              "condition": "true"
            }
          ],
          "operations": [
            {
              "operation": "patch"
            }
          ]
        }
      ]
    ]
  }
}
```

```
        "effect": "Permit"
    }
]
}
]
}
}
}
```

41. Having received the delegation information, as described in step 40, the proxy (or more precisely, the PDP) can now evaluate whether the contained user policy allows for updating the PTA attribute of the specific delivery order. If the Happy Pets customer was assigned the P.Info.gold role in the Happy Pets shop system, the necessary policy is part of the delegation evidence and access is granted on **user level**. Since the policy was not issued by the Packet Delivery Company to the customer, but rather by Happy Pets, the proxy needs to check whether Happy Pets is allowed to delegate this policy to its customers. This **organizational level** access control is performed in the following steps.  
In the case that the customer was not assigned the P.Info.gold role, the necessary policy would be missing and changing the PTA would be denied by the Packet Delivery company proxy. As a result an error would be returned to the Packet Delivery company portal, also presented to the Happy Pets customer. The following steps would be omitted.
42. In order to check whether Happy Pets is allowed to delegate the policy to its customers, the proxy will check at the Packet Delivery company Authorisation Registry whether this policy exists.  
At first, Packet Delivery company proxy generates an iSHARE JWT (also see description in step 4).

```
> Headers
{
  "alg": "RS256",
  "typ": "JWT",
  "x5c": [
    "MIIEhjCC...Zy9w=="
  ]
}

> Payload
{
  "iss": "EU.EORI.NLPACKETDEL", // Client-ID (EORI) of Packet Delivery company
  "sub": "EU.EORI.NLPACKETDEL", // Client-ID (EORI) of Packet Delivery company
  "aud": [
    "EU.EORI.NLPACKETDEL", // ID (EORI) and token-URL of Packet Delivery Co. Authz
    "https://ar.isharetest.net/connect/token"
  ]
}
```

```
],
  "jti": "99ab5bca41bb45b78d242a46f0157b7d", // Unique JWT ID
  "iat": "1540827435",
  "exp": 1540827465,
  "nbf": 1540827435
}
```

43. Packet Delivery company proxy sends a request to the Packet Delivery company Authorization Registry [/token](#) endpoint. Using the token, the Packet Delivery company proxy will retrieve the delegation evidence/policy information of Happy Pets from the PPacket Delivery company authorisation registry in step 45.

```
> Content-Type: application/x-www-form-urlencoded  
  
POST https://ar.isharetest.net/token  
  
grant_type=client_credentials&  
scope=iSHARE&  
client_id=EU.EORI.NLPACKETDEL&  
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&  
client_assertion=evJhbGc...9hvW
```

44. An [access token](#) is returned to the Packet Delivery company proxy in response to the previous request.

```
< Content-Type: application/json
< Cache-Control: no-store
< Pragma: no-cache

{
  "access_token": "aW2ys9NGE8RjHPZ4mytQivkWJ05HGQCYJ7VyMBGGDLIOw",
  "expires_in": 3600,
  "token_type": "Bearer"
}
```

45. The proxy is now sending a request to the [/delegation](#) endpoint of the Packet Delivery company Authorization Registry, in order to retrieve the delegation evidence policy issued from Packet Delivery company to Happy Pets.

```
> Authorization: Bearer IIe...NIQ // OAuth2 Bearer token (obtained by M2M OAuth2)
> Content-Type: application/json

POST 'https://ar.isharetest.net/delegation'

{
  "delegationRequest": {
    "policyIssuer": "EU.EORI.NLPACKETDEL",      //Creator/Owner of the policy, since
this is in request, the EORI of the company whose policy is in question - Packet
Delivery Co.
```

```

    "target": {
      "accessSubject": "EU.EORI.HAPPYPETS" //Subject for whom this resource request
      is for - HappyPets (company in this step)
    },
    "policySets": [
      {
        "policies": [
          {
            "target": {
              "resource": {
                //Free text attributes!!
                "type": "DELIVERYORDER", //Resource category, Orders in this case
                "identifiers": [
                  "urn:ngsi-ld:DELIVERYORDER:001" //Identifier of the resource,
                  Order id in this case
                ],
                "attributes": [ //Attributes about the resource
                  "pta"
                ]
              },
              "actions": [
                "PATCH"
              ],
              "rules": [
                {
                  "effect": "Permit"
                }
              ]
            }
          }
        ]
      }
    }
  }
}

```

46. Packet Delivery company Authorization Registry [returns](#) the delegation information as JWT `delegation_token`.

```

< Content-Type: application/json
> Payload
{
  "delegation_token": "eyJ4...0aaQ"
}

Decoded delegation_token JWT payload
{
  "iss": "EU.EORI.NLPACKETDEL",
  "sub": "EU.EORI.NLPACKETDEL",
  "jti": "d8a7fd7465754a4a9117ee28f5b7fb60",
  "iat": 1591966224,
  "exp": 1591966254,
  "aud": "EU.EORI.NLPACKETDEL",
  "delegationEvidence": {
    "notBefore": 1541058939,
    "notOnOrAfter": 2147483647,
    "policyIssuer": "EU.EORI.NLPACKETDEL",
  }
}

```

```
"target": {
    "accessSubject": "EU.EORI.NLHAPPYPETS"
},
"policySets": [
    {
        "maxDelegationDepth": 1,
        "target": {
            "environment": {
                "licenses": [
                    "ISHARE.0001"
                ]
            }
        },
        "policies": [
            {
                "target": {
                    "resource": {
                        "type": "DELIVERYORDER",
                        "identifiers": [
                            "*"
                        ],
                        "attributes": [
                            "pta"
                        ]
                    },
                    "actions": [
                        "PATCH"
                    ]
                },
                "rules": [
                    {
                        "effect": "Permit"
                    }
                ]
            }
        ]
    }
}
```

In the case that Happy Pets didn't acquire access to the premium service offering, there won't be such a policy created for Happy Pets, and the response contains a Deny value in place of a Permit in the rules parameter.

```
< Content-Type: application/json
> Payload
{
    "delegation_token": "eyJ4...0aaQ"
}

Decoded delegation_token JWT payload
{
    "iss": "EU.EORI.NLPACKETDEL",
    "sub": "EU.EORI.NLPACKETDEL",
    "jti": "d8a7fd7465754a4a9117ee28f5b7fb60",
    "iat": 1591966224,
```

```

    "exp": 1591966254,
    "aud": "EU.EORI.NLPACKETDEL",
    "delegationEvidence": {
        "notBefore": 1541058939,
        "notOnOrAfter": 2147483647,
        "policyIssuer": "EU.EORI.NLPACKETDEL",
        "target": {
            "accessSubject": "EU.EORI.NLHAPPYPETS"
        },
        "policySets": [
            {
                "maxDelegationDepth": 1,
                "target": {
                    "environment": {
                        "licenses": [
                            "ISHARE.0001"
                        ]
                    }
                },
                "policies": [
                    {
                        "target": {
                            "resource": {
                                "type": "DELIVERYORDER",
                                "identifiers": [
                                    "urn:ngsi-ld:DELIVERYORDER:HappyPets001"
                                ],
                                "attributes": [
                                    "PTA"
                                ]
                            },
                            "actions": [
                                "PATCH"
                            ],
                            "rules": [
                                {
                                    "effect": "Deny"
                                }
                            ]
                        }
                    }
                ]
            }
        ]
    }
}

```

47. Having received the delegation information from the Packet Delivery company Authorization Registry, the proxy (or more precisely, the PDP) can now evaluate whether the contained **organizational policy** allows for updating the PTA attribute, and therefore whether it Happy Pets is allowed to delegate the access to its customers. If the proxy received a valid policy, access would be granted on **organizational level**.

If the requested delegation evidence can not be found or the returned policy contains the Deny rule, the change of the PTA would be denied by the Packet Delivery company proxy and an error would be returned to the Packet Delivery company portal/app, also presented to the Happy Pets customer. The following steps would be omitted.

48. As described in the previous steps, the PDP evaluated that a change of the PTA of the specific delivery order is granted, both on **organizational level** and **user level**. As a result, the request for changing the PTA from step 26 is forwarded by the Packet Delivery company proxy to the Packet Delivery company Context Broker which holds the information of the packet delivery order. The PTA of the packet delivery order is changed and the Context Broker returns a successful response with HTTP code 204 (see [section 6.7.3.1](#)). The Context Broker response is returned to the Packet Delivery company portal, in response to the request of step 26.
49. The successful change of the PTA is presented to the Happy Pets customer.

## 6.7.2 Scenario: No Cheaper

This section describes the variations of above steps in the scenario of the No Cheaper customer.

Basically the sequence of steps is the same as for Happy Pets. In contrast to Happy Pets, during the acquisition of rights described in section 6.6, No Cheaper is just acquiring the standard service and therefore its customers will only be able to read attributes of delivery orders (also see section 6.6.2). This means that at the Packet Delivery company authorisation registry, there is only a policy created allowing No Cheaper to only delegate GET access to delivery orders.

This scenario can be splitted into two cases to demonstrate the denial of access based on the different policies on organizational level and user level.

1. At No Cheaper Authorisation Registry, a policy is assigned to the No Cheaper customer allowing only GET requests to the Packet Delivery service (representing the P.Info.Standard role). When performing the steps for changing the PTA value of a delivery order, as described in the previous section, the process would stop at step 41, where access would be rejected because the No Cheaper customer was not assigned the necessary policy at user level.
2. At No Cheaper Authorisation Registry, a policy is assigned to the No Cheaper customer allowing both GET and PATCH requests to the Packet Delivery service (representing the P.Info.Gold role). When performing the steps for changing the PTA value of a delivery order, as described in the previous section, the process

would stop at step 47, where access would be rejected because No Cheaper was not assigned the necessary policy at the Packet Delivery company Authorisation Registry to delegate the premium access to its customers. Therefore access would be rejected at organizational level. This is to show that access would be still rejected, even when the No Cheaper organization issues access to the premium service to its customers within its own Authorization Registry.

In general, for both cases the request for changing the PTA should be denied. However, it can be shown that the No Cheaper customer is able to view attributes of its delivery orders.

## 6.8 Role-based access

The following describes how to use the recipes presented in sections 6.4-6.7 in order to configure access based on roles instead of attribute-based policies. The steps of creating an offering, acquisition of access rights and accessing the service will be modified. Only the differences to the previously introduced policy-based access concept will be presented in the following.

**Note that the functionalities for role-based access are not yet implemented in the corresponding components of PEP/PDP and Keyrock (AR functionality).**

In the scenario of the policy based access, the provider defines sets of policies which will correspond to certain offerings on the marketplace. By acquiring access to an offering, these policy sets are issued to the acquiring organisation within the Authorisation registry of the provider. Moreover the acquiring organisation can delegate the access to their users, meaning to issue these policy sets to their users within the Authorisation Registry of the acquiring organisation.

In the scenario of the role-based access, this would be slightly modified. The provider defines a role having a certain name, and this role is mapped to a certain policy set representing the attribute-based policies as before. An offering on the marketplace then just represents a certain role (or several roles). When acquiring access to an offering on the marketplace, these roles then get issued to the acquiring organisation within the Authorisation Registry of the provider. Furthermore, the acquiring organisation then can just assign these roles to their users within their own Authorisation Registry. When accessing the service, it is up to the PEP proxy/PDP

component Kong+plugin to obtain the set of attribute-based policies that belong to the assigned roles and to perform the evaluation of granting access based on the NGSI-LD request as before.

Per convention, the name of a role is an URL that represents an endpoint, where information can be retrieved to which set of policies this role is mapped. This endpoint can be implemented within the provider's Authorisation Registry, the Identity Provider or even a completely separate component.

### 6.8.1 Create Offering

The process of creating an offering only differs by the information provided by the provider when creating the offering on the marketplace.

Instead of providing details like HTTP methods, attributes or entity types, representing the access policies that should be issued during acquisition, the provider will just enter the name of the role that should be assigned to an acquiring organisation during acquisition of the offering. During the creation of the offering, the provider will select a specific asset type dedicated to the role-based access within i4Trust and which allows to specify the role name.

As mentioned before, a role name is an URL representing an endpoint where information can be retrieved about the policies mapped to this role. It is up to the provider to implement this endpoint and to store the necessary information about the mapping. This functionality is implemented in Keyrock. The following gives an example of how the returned object from this endpoint should look like, when there is an endpoint

'<https://pdc-keyrock.i4trust-demo.fiware.dev/policiesrolemapping?role=pdc.premium>' for a role named "pdc.premium" representing the premium service of the Packet Delivery Company. The attribute 'policies' contains an array of policy objects similar to those stored at the Authorisation Registries for the attribute-based access use case.

```
GET https://pdc-keyrock.i4trust-demo.fiware.dev/policiesrolemapping?role=pdc.premium
< Content-Type: application/json
> Payload
{
  "policiesRoleMapping": {
    "role": "pdc.premium",
    "policies": [
```

```
{  
    "target": {  
        "resource": {  
            "type": "DELIVERYORDER",  
            "identifiers": ["*"],  
            "attributes": ["pta","pda"]  
        },  
        "actions": ["PATCH"]  
    },  
    "rules": [{ "effect": "Permit" }]  
},  
{  
    "target": {  
        "resource": {  
            "type": "DELIVERYORDER",  

```

## 6.8.2 Acquisition of Rights / Activation

The process for the acquisition of access rights differs by the ‘delegationEvidence’ object to be created by the marketplace plugin in the provider’s Authorisation Registry (Packet Delivery Company), as well es the assignment of the role at user level within the Authorisation Registry of the acquiring organisation (e.g., Happy Pets).

When an employee of Happy Pets acquires access to the premium service of packet Delivery Company, the marketplace plugin will create a ‘delegationEvidence’ object at the Authorisation Registry of the Packet Delivery Company. Compared to step 26 of section 6.6.1, this object will not assign the actual policies to the organisation of Happy Pets, but rather just the role name as defined in 6.8.1. The following displays an example of the request sent by the marketplace plugin to the activation service in front of the Authorisation Registry of Packet Delivery Company:

```
> Authorization: Bearer IIe...VNIQ // access_token from step 25
POST https://pdc-as.i4trust-demo.fiware.dev/createpolicy

> Payload
{
  "delegationEvidence": {
    "notBefore": 1617796688,
    "notOnOrAfter": 1617798488,
    "policyIssuer": "EU.EORI.NLPACKETDEL",
    "target": {
      "accessSubject": "EU.EORI.NLHAPPYPETS"
    },
    "policySets": [
      {
        "maxDelegationDepth": 0,
        "target": {
          "environment": {
            "licenses": [
              "ISHARE.0001"
            ]
          }
        },
        "policies": [
          {
            "target": {
              "resource": {
                "type": "role",
                "identifiers": [
                  "https://pdc-keyrock.i4trust-demo.fiware.dev/policiesrolemapping?role=pdc.premium"
                ],
                "attributes": ["*"]
              },
              "actions": ["Assign"]
            },
            "rules": [
              {
                "effect": "Permit"
              }
            ]
          }
        ]
      }
    ]
  }
}
```

Note, that the ‘resource’ attribute contains the information about the role: type equals to “role”, and ‘identifiers’ contains the URL representing the endpoint returning the information about the policies mapped to this role. The ‘actions’ attribute contains the value “Assign”.

As described for the prerequisites in section 6.4, the user-level policies stored at the Authorization Registries of the acquiring organisations would be created when customers sign up at the corresponding shop systems of Happy Pets and No Cheaper, respectively. As written earlier, it is assumed that the two customers have already

been registered at the Identity Providers of Happy Pets and No Cheaper. Therefore their user-level access policies need to be initially stored at the Authorisation Registries (can be done via the UI or API). Instead of attribute-based access policies, the ‘delegationEvidence’ objects to be created will contain assignments of roles to users, as given in the example below for the role representing the premium service of Packet Delivery Company issued to some shop customer by Happy Pets:

```
{
  "delegationEvidence": {
    "notBefore": 1617796688,
    "notOnOrAfter": 1617798488,
    "policyIssuer": "EU.EORI.NLHAPPYPETS",
    "target": {
      "accessSubject": "cccc7d10-29e7-4a9f-944a-e6c3d39774c0"
    },
    "policySets": [
      {
        "maxDelegationDepth": 0,
        "target": {
          "environment": {
            "licenses": [
              "ISHARE.0001"
            ]
          }
        },
        "policies": [
          {
            "target": {
              "resource": {
                "type": "role",
                "identifiers": [
                  "https://pdc-keyrock.i4trust-demo.fiware.dev/policiesrolemapping?role=pdc.premium"
                ],
                "attributes": ["*"]
              },
              "actions": ["Assign"],
              "environment": {"serviceProvider": "EU.EORI.NLPACKETDEL"}
            },
            "rules": [
              {
                "effect": "Permit"
              }
            ]
          }
        ]
      }
    ]
  }
}
```

### 6.8.3 Access to data service

This process differs slightly by the JWT sent along with the NGSI-LD request to the PEP proxy, as well as the configuration and business logic within the Kong plugin.

As in step 26 of section 6.7.1, the Packet Delivery company portal/app sends a request to the Packet Delivery company proxy, in order to change the PTA of the delivery order. The request contains a [signed iSHARE compliant JWT](#) as Bearer token which includes the Happy Pets customer user information from step 23, but here with the roles assigned to the user instead of the attribute-based policies in the delegation evidence. An example is given below for the use case of the Happy Pets shop customer changing the PTA attribute of it's delivery order.

```
> Authorization: Bearer IIeD...NIQ    // Bearer JWT
> Content-Type: application/json

PATCH
https://pdc-kong.i4trust-demo.fiware.dev/ngsi-ld/v1/entities/urn:ngsi-ld:DELIVERYORDER
:001/attrs/pta

> Payload
{
  "value": "<new PTA>",
  "type": "Property"
}

Decoded Bearer JWT:
{
  "iss": "EU.EORI.NLHAPPYPETS",
  "sub": "419404e1-07ce-4d80-9e8a-eca94vde0003de",
  "jti": "d8a7fd7465754a4a9117ee28f5b7fb60",
  "iat": 1591966224,
  "exp": 1591966254,
  "aud": "EU.EORI.NLHAPPYPETS",
  "delegationEvidence": {
    "notBefore": 1541058939,
    "notOnOrAfter": 2147483647,
    "policyIssuer": "EU.EORI.NLHAPPYPETS",
    "target": {
      "accessSubject": "419404e1-07ce-4d80-9e8a-eca94vde0003de" // ID of customer
    },
    "policySets": [
      {
        "maxDelegationDepth": 0,
        "target": {
          "environment": {
            "licenses": [
              "ISHARE.0001"
            ]
          }
        },
        "policies": [
          {
            "target": {
              "resource": {
                "id": "urn:ngsi-ld:DeliveryOrder:001"
              }
            }
          }
        ]
      }
    ]
  }
}
```

```
"type": "role",
"identifiers": [
  "https://pdc-keyrock.i4trust-demo.fiware.dev/policiesrolemapping?role=pdc.premium"
],
"attributes": [
  "*"
],
"environment": {
  "serviceProviders": [
    "EU.EORI.NLPACKETDEL"
  ]
},
"actions": [
  "Assign"
],
"rules": [
  {
    "effect": "Permit"
  }
]
}
}
}
```

The alternative of providing information about the authorisation registry where to retrieve this information from, is not possible for the role-based access.

Within Kong, a different plugin `ngsi-ishare-roles` (still to be implemented) needs to be used for the route, which implements the slightly different business logic when evaluating the access to the API backend.

When receiving the NGSI-LD request, this Kong plugin will first decode the JWT and extract the included role identifiers, namely the URLs where to retrieve information about the mapped policies. For each of these assigned roles, the plugin will send a GET request to the corresponding URL for retrieval of the mapped policy sets. Based on the parameters and payload of the NGSI-LD request, the plugin will evaluate whether one of these policy sets would grant access.

If there is such a policy set, the plugin needs to check at the Authorisation Registry of Packet Delivery Company, whether the same role was also assigned to the organisation that assigned the role to the user. In this case, the plugin would query whether there is a role assignment for the desired role from Packet Delivery Company to Happy Pets. This is similar to the request sent in step 45 of section 6.7.1, but with the role assignment in the array of policies.

```
> Authorization: Bearer IIe...NIQ // OAuth2 Bearer token (obtained by M2M OAuth2)
> Content-Type: application/json

POST 'https://ar.isharetest.net/delegation'

{
  "delegationRequest": {
    "policyIssuer": "EU.EORI.NLPACKETDEL",      //Creator/Owner of the policy, since
    this is in request, the EORI of the company whose policy is in question - Packet
    Delivery Co.
    "target": {
      "accessSubject": "EU.EORI.HAPPYPETS"     //Subject for whom this resource request
    is for - HappyPets (company in this step)
    },
    "policySets": [
      {
        "policies": [
          {
            "target": {
              "resource": {
                "type": "role",
                "identifiers": [
                  "https://pdc-keyrock.i4trust-demo.fiware.dev/policiesrolemapping?role=pdc.premium"
                ],
                "attributes": [
                  "*"
                ]
              },
              "actions": [
                "Assign"
              ],
              "rules": [
                {
                  "effect": "Permit"
                }
              ]
            }
          }
        ]
      }
    ]
  }
}
```

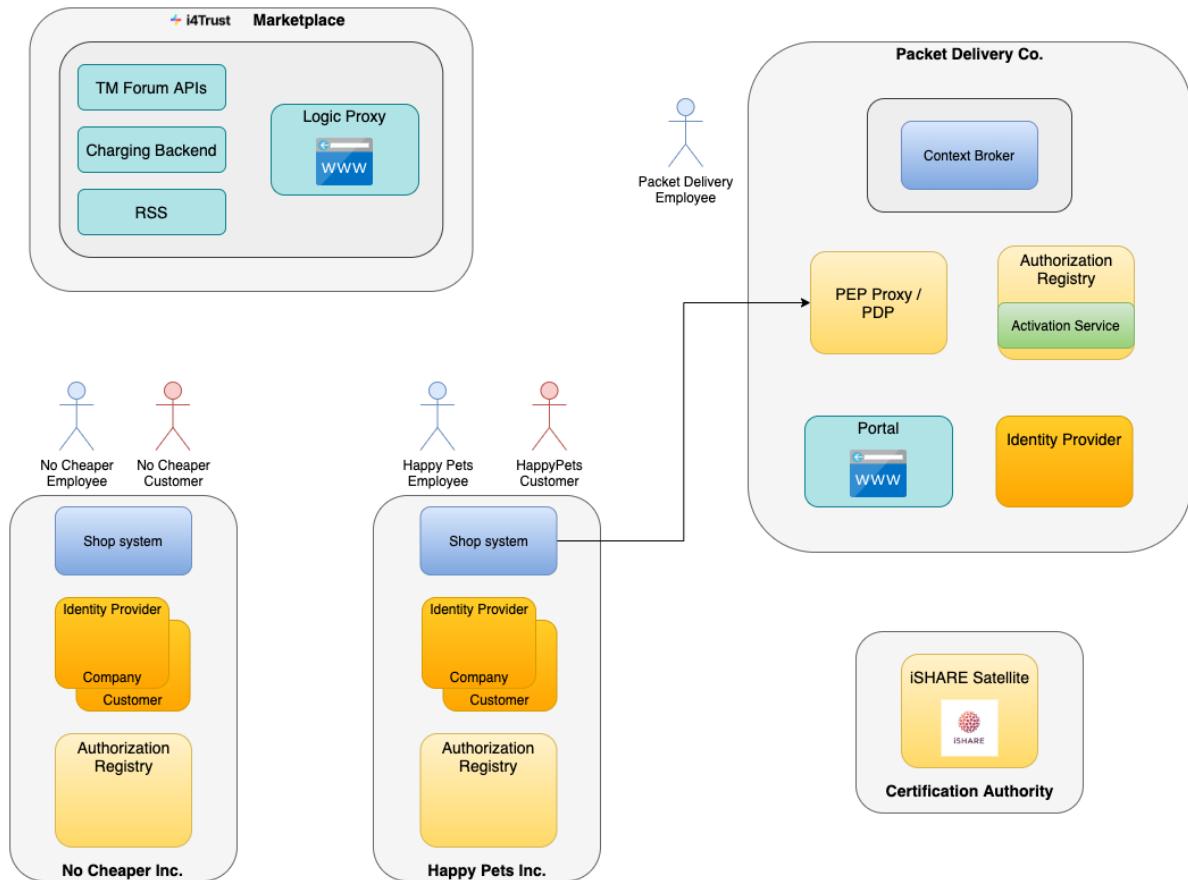
When there is a successful response, access would be granted and the NGSI-LD request would be forwarded to the Context Broker configured in the route of the Kong plugin.

## 6.9 Direct access to data service by client applications (M2M)

The previous instructions showcased how to delegate access rights from an organisational level to an user level, meaning that an organisation acquired the access rights for a particular service and delegates these rights to a certain user which will finally access the service. This is commonly called Human-To-Machine (H2M) interaction.

In the following it is presented how to use the same mechanisms in order that the organisation accesses the services, for instance by some client application within their environment. This represents a Machine-To-Machine (M2M) interaction.

Given the example of the Packet Delivery Company, this client application could be the shop system of Happy Pets. As soon as some customer orders some pet articles, a delivery order needs to be created by the shop system at the Packet Delivery Order system. This would be done by the shop system application directly accessing the Packet Delivery Order service in order to create packet delivery orders, meaning to send a POST NGSI-LD request which creates an entity of type DELIVERYORDER, as depicted in the following diagram:



In the following, the changes to the steps of creating an offering, acquisition of access rights and accessing the service will be presented, compared to the previously shown description of the attribute-based access with H2M interactions.

The M2M interaction will follow the OAuth protocol with signed iSHARE JWT, as also presented in the [iSHARE documentation](#).

## 6.9.1 Create Offering

There is no change in the process of creating an offering. The only difference compared to the H2M scenario is that the Packet Delivery Company will create an additional service offering “Create Delivery Order Service” for the creation of delivery orders, being for free or at some cost. This would be configured in a way that policies get created allowing POST NGSI-LD requests for entities of type DELIVERYORDER and all attributes (“\*”).

## 6.9.2 Acquisition of Rights / Activation

There is also no change in the process of the acquisition of access rights. Some employee of Happy Pets will access the marketplace and acquire access to the "Create Delivery Order Service". During acquisition and activation of the service access, the marketplace plugin will create an access policy at the Authorisation Registry of Packet Delivery Company, as also described in step 26 of section 6.6.1, which will allow the organisation of Happy Pets to create delivery orders. Shown below is an example of the object created at the Authorisation Registry.

```
> Authorization: Bearer IIe...VNIQ // access_token from step 25

POST https://pdc-as.i4trust-demo.fiware.dev/createpolicy

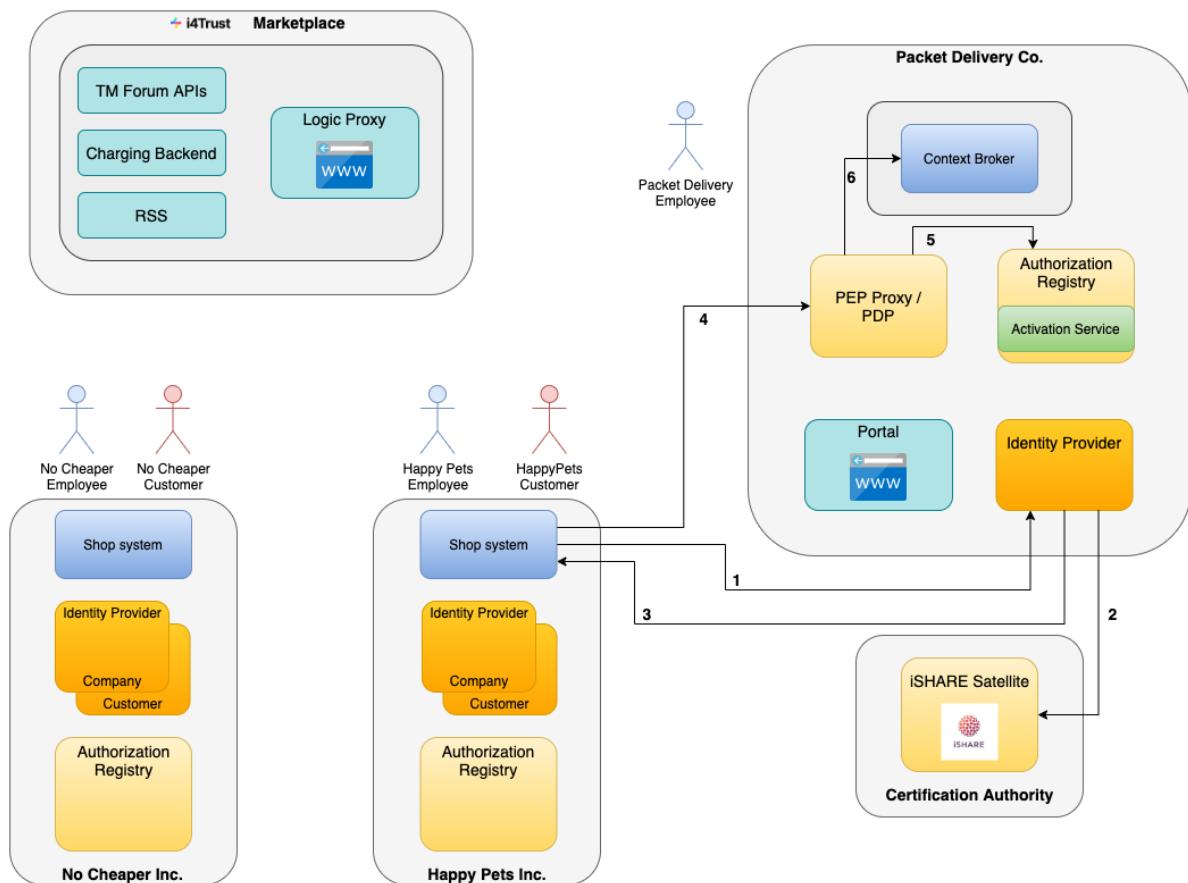
> Payload
{
  "delegationEvidence": {
    "notBefore": 1617796688,
    "notOnOrAfter": 1617798488,
    "policyIssuer": "EU.EORI.NLPACKETDEL",
    "target": {
      "accessSubject": "EU.EORI.NLHAPPYPETS"
    },
    "policySets": [
      {
        "maxDelegationDepth": 0,
        "target": {
          "environment": {
            "licenses": [
              "ISHARE.0001"
            ]
          }
        },
        "policies": [
          {
            "target": {
              "resource": {
                "type": "DELIVERYORDER",
                "identifiers": ["*"],
                "attributes": ["*"]
              },
              "actions": ["POST"]
            },
            "rules": [
              {
                "effect": "Permit"
              }
            ]
          }
        ]
      }
    ]
  }
}
```

Since Happy Pets won't delegate these access rights to their customers, there are no further steps required at Happy Pets.

### 6.9.3 Access to data service

This process differs compared to the description of section 6.7.1. Instead of a human user logging in at some IDP, obtaining an access token with its policies and invoking requests to the packet Delivery service via some portal application, the shop system of Happy Pets will obtain an access token at the Packet Delivery Company, and invoke the NGSI-LD API of the Packet Delivery Company directly.

The following architecture diagram shows the different flows involved:



Therefore the steps 1-24 of section 6.7.1 can be skipped and be replaced by the following.

First, the Happy Pets shop system needs to obtain an access token at the Packet Delivery Company. This process is similar to the steps 8-11 of section 6.7.1. The shop

system creates a signed iSHARE JWT, as shown below and also described in step 4 of section 6.7.1

```
> Headers
{
  "alg": "RS256",
  "typ": "JWT",
  "x5c": [
    "MII EhjCC....Zy9w=="
  ]
}

> Payload
{
  "iss": "EU.EORI.NLHAPPYPETS", // Client-ID (EORI) of Happy Pets
  "sub": "EU.EORI.NLHAPPYPETS", // Client-ID (EORI) of Happy Pets
  "aud": [
    "EU.EORI.NLPACKETDEL", // ID (EORI) and token-URL of PDC IDP
    "https://pdc-keyrock.i4trust-demo.fiware.dev/token"
  ],
  "jti": "99ab5bca41bb45b78d242a46f0157b7d", // Unique JWT ID
  "iat": "1540827435",
  "exp": 1540827465,
  "nbf": 1540827435
}
```

Then the shop system sends a request to the [/token](#) endpoint of the Identity Provider of Packet Delivery Company (step 1 of previous architecture diagram). Note that any component within the environment of Packet Delivery Company could be capable of issuing access tokens and it does not have to be an Identity Provider, e.g., this [/token](#) endpoint could be also implemented at the PEP Proxy. Compared to step 9 of section 6.7.1, the request will contain a different grant\_type parameter in order to obtain a signed iSHARE JWT as bearer access token. The previously created iSHARE JWT will be added as client\_assertion parameter.

```
> Content-Type: application/x-www-form-urlencoded
POST https://pdc-keyrock.i4trust-demo.fiware.dev/token
grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&
scope=iSHARE&
client_id=EU.EORI.NLHAPPYPETS&
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&
client_assertion=eyJhbGc...9hvW
```

The Identity Provider of Packet Delivery Company verifies the iSHARE JWT, checks at the trust authority whether Happy Pets is a trusted party (step 2 of previous

architecture diagram) and then returns an encoded signed iSHARE JWT as access token (step 3 of previous architecture diagram), as shown below.

```
< Content-Type: application/json
< Cache-Control: no-store
< Pragma: no-cache

{
    "access_token": "aW2y...LIOw",
    "expires_in": 3600,
    "token_type": "Bearer"
}

> Decoded payload of access token
{
    "iss": "EU.EORI.NLPACKETDEL", // Client-ID (EORI) of Packet Delivery Co
    "sub": "EU.EORI.NLHAPPYPETS", // Client-ID (EORI) of Happy Pets
    "jti": "99ab5bca41bb45b78d242a46f0157b7d", // Unique JWT ID
    "iat": "1540827435",
    "exp": 1540827465,
    "nbf": 1540827435,
    "aud": "EU.EORI.NLPACKETDEL", // Client-ID (EORI) of Packet Delivery Co
}
```

Next, the shop system will send a request to the proxy of Packet Delivery Company in order to create a delivery order (step 4 of previous architecture diagram). This request contains the previously obtained iSHARE JWT access token as authorization bearer token, as shown below, and similarly described in step 26 of section 6.7.1:

```
> Authorization: Bearer IIeD...NIQ    // Bearer JWT
> Content-Type: application/json

POST https://pdc-kong.i4trust-demo.fiware.dev/ngsi-ld/v1/entities

> Payload
{
    "id": "urn:ngsi-ld:DELIVERYORDER:HAPPYPETS001",
    "type": "DELIVERYORDER",
    "issuer": {
        "type": "Property",
        "value": "Happy Pets"
    },
    "destinee": {
        "type": "Property",
        "value": "Happy Pets customer"
    },
    "deliveryAddress": {
        "type": "Property",
        "value": {
            "type": "Address"
        }
    }
}
```

```
        "addressCountry": "DE",
        "addressRegion": "Berlin",
        "addressLocality": "Berlin",
        "postalCode": "12345",
        "streetAddress": "Customer Strasse 23"
    }
},
"originAddress": {
    "type": "Property",
    "value": {
        "addressCountry": "DE",
        "addressRegion": "Berlin",
        "addressLocality": "Berlin",
        "postalCode": "12345",
        "streetAddress": "HappyPets Strasse 15"
    }
},
"pda": {
    "type": "Property",
    "value": "2021-10-02"
},
"pta": {
    "type": "Property",
    "value": "14:00:00"
},
"eda": {
    "type": "Property",
    "value": "2021-10-02"
},
"eta": {
    "type": "Property",
    "value": "14:00:00"
},
"@context": [
    "https://schema.lab.fiware.org/ld/context",
    "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
]
}

Decoded Bearer JWT payload:
{
    "iss": "EU.EORI.NLPACKETDEL", // Client-ID (EORI) of Happy Pets
    "sub": "EU.EORI.NLHAPPYPETS", // Client-ID (EORI) of Happy Pets
    "jti": "99ab5bca41bb45b78d242a46f0157b7d", // Unique JWT ID
    "iat": "1540827435",
    "exp": 1540827465,
    "nbf": 1540827435
}
```

Now, the proxy will verify and validate the JWT access token, as described in step 27 of section 6.7.1. Since the access token contains neither the access policies (`delegationEvidence`) nor any Authorisation Registry information (`authorisationRegistry`), it will skip steps 28-41 of section 6.7.1. Based on the incoming NGSI-LD request, the proxy will evaluate the necessary policy for the operation, and request at the Authorisation Registry of Packet Delivery Company (step

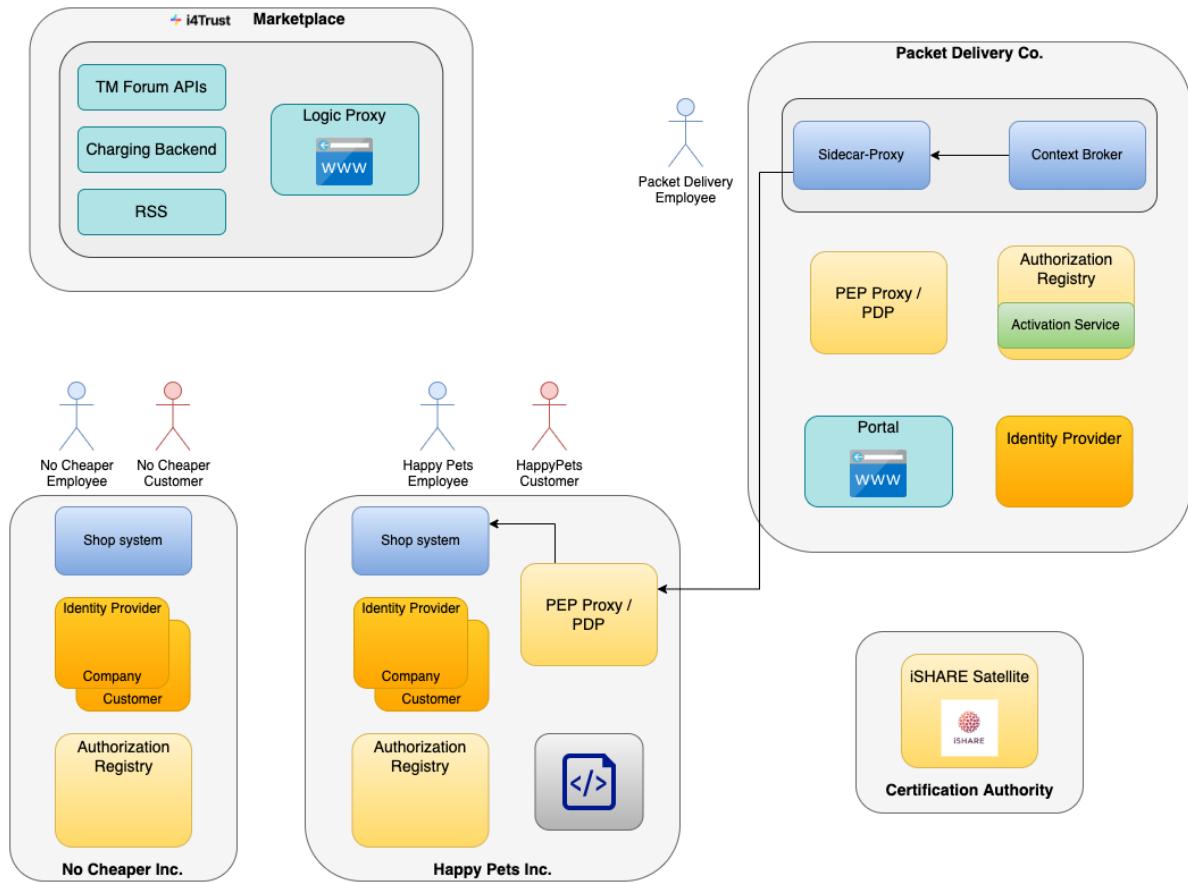
5 of previous architecture diagram) whether there is such policy issued by Packet Delivery Company (iss attribute of the JWT access token) to Happy Pets (sub attribute of the JWT access token), as described in steps 42-47. If there is a successful response from the Authorisation Registry, the NGSI-LD request for creating a delivery order is forwarded to the Context Broker of Packet Delivery Company (step 6 of previous architecture diagram).

## 6.10 Subscriptions and Notifications

Besides the access to specific entities at the Context Broker, there are also the mechanisms of subscriptions and notifications. This is an extended mechanism of the M2M interactions as described before.

When creating a subscription at the context broker, one can specify an endpoint, where the Context Broker should send notifications to, whenever there are changes to a certain entity. In the subscription, such entities can be specified either by an ID or certain query parameters.

An example use case could be an additional service offered by the Packet Delivery Company, which will allow service consumers (like Happy Pets) to subscribe to changes on certain delivery orders. Such consumers would provide an endpoint to Packet Delivery, where notifications should be sent to, whenever there is a change to delivery orders. Such process is depicted in the following diagram:



This use case requires an extended architecture compared to the one described in the previous chapters:

- Packet Delivery Company needs to deploy an additional Endpoint-Auth-Service in order to perform the necessary authorization steps before sending notifications
- Happy Pets needs to deploy an own instance of the PEP/PDP component Kong in order to protect the exposed endpoint for receiving the notifications

### 6.10.1 Deployment and configuration of components

Two additional components need to be deployed compared to the previously described architecture.

#### PEP/PDP at Happy Pets

Happy Pets requires an own instance of the PEP Proxy and PDP component Kong with the `ngsi-ishare-policies` plugin. The deployment does not differ as described in section 6.3 for the instance of the Packet Delivery Company. It needs to be configured similarly as described in section 6.4 with the “NGSI-LD attribute based” authorization

mode. The configured endpoint should forward requests to the actual notification endpoint, which could be an endpoint provided by the shop system or any other component within the environment of Happy Pets.

In order that consumers can access the Endpoint-Configuration-Service API to configure an endpoint for issuing (M2M) access tokens to the Sidecar-Proxy, Packet Delivery Company needs to also integrate the [auth-endpoint-config-ishare](#) Kong plugin and configure there an additional route which forwards requests to the Endpoint-Configuration-Service.

## Context Broker + Endpoint-Auth-Service at Packet Delivery Company

In order to extend the already deployed Context Broker with the iShare-authentication functionality , the [Endpoint-Auth-Service](#) needs to be deployed for PacketDelivery. The service essentially consists of 3 components:

### *Endpoint-Configuration-Service*

The Endpoint-Configuration-Service offers an [API](#) to configure the required authentication methods for each endpoint. If an endpoint is not configured there, it will be ignored and passed through unhandled. The service requires an SQL-Database to store the configuration-information.

### *iShare-Auth-Provider*

The Auth-Provider implements the [iShare-Authentication flow](#) with the configured credentials. Credentials can be configured through the Auth-Providers [Credentials-Management API](#). The actual JWT can be retrieved through the [Auth-Provider API](#) to be applied through the Sidecar-Proxy.

### *Sidecar-Proxy*

The Sidecar-Proxy has to be deployed co-located to each instance of the Context Broker and intercepts all outgoing traffic. Depending on the configuration provided by the Endpoint-Configuration-Service, it decides if a request needs to be authenticated or can be passed through. For configured endpoints, it requests(and caches) a JWT at the Auth-Provider and adds it to the request before forwarding it.

The proxy is implemented as a wasm-extension to the [Envoy-Proxy](#).

The Endpoint-Auth-Service can be deployed in any containerized environment. A plain docker-based setup is described here: [docker-compose-setup](#). Since the

sidecar-approach requires ipTable-manipulation, the networking part can become tricky. Therefore the recommended deployment is based on Kubernetes. Two deployment models are supported:

“Bring-your-own-envoy”: The sidecar-proxy is deployed as part of the Endpoint-Authentication-Service and injected into the Context Broker’s Pod. See the [Kubernetes-Deployment](#) description for that option.

“Service-Mesh integration”: Due to the emergence of the Service Mesh, an envoy-proxy might already exist and only needs to be configured. The Endpoint-Auth-Service currently supports integration into the OpenShift Service Mesh, as described [here](#).

To enable the Authentication for a given endpoint, two REST calls have to be made:

1. Configure the endpoint (in this use case this would be performed by the service consumer, e.g., Happy Pets, as described in the following section 6.10.3):

```
curl -X 'POST' \
  'http://endpoint-config-service/endpoint' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "domain": "happypets.com",
    "port": 80,
    "path": "/notification",
    "useHttps": false,
    "authType": "iShare",
    "authCredentials": {
      "iShareClientId": "string",
      "iShareIdpId": "string",
      "iShareIdpAddress": "https://happypets-keyrock.i4trust-demo.fiware.dev/oauth2/token",
      "requestGrantType": "urn:ietf:params:oauth:grant-type:jwt-bearer"
    }
}'
```

2. Configure the iSHARE credentials (this must be performed by the service provider, e.g., Packet Delivery Company in this use case):

```
curl -X 'POST' \
  'http://ishare-auth-provider/credentials/<iShareClientId>' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "certificateChain": "string",
    "signingKey": "string"
  }'
```

The clientId (EORI), certificateChain and signingKey have to be the valid information as registered in the iShare-Satellite for the service provider, e.g., Packet Delivery Company in this use case.

## 6.10.2 Create Offering

There is no change in the process of creating an offering for the notification service. The only difference compared to the previous scenario is that the Packet Delivery Company will create an additional service offering “Notification Service” for the notification of changes to delivery orders, being for free or at some cost. This requires a different asset type on the marketplace (plugin) which would create the necessary policies during acquisition, as described below. Note that this plugin is not implemented yet.

## 6.10.3 Acquisition of rights / Activation

This process requires additional steps compared to the descriptions before. In general it is splitted into two phases:

- **Acquisition:** Happy Pets acquires access to the notification service on the Marketplace. The Marketplace will create the necessary policies for Happy Pets at the Authorisation Registry of Packet Delivery.
- **Activation:** Happy Pets creates the necessary subscription at the Context Broker of Packet Delivery and configures the token endpoint at the Sidecar-Proxy. In addition, Happy Pets creates a policy at its own Authorization Registry, which will allow Packet Delivery to send notifications.

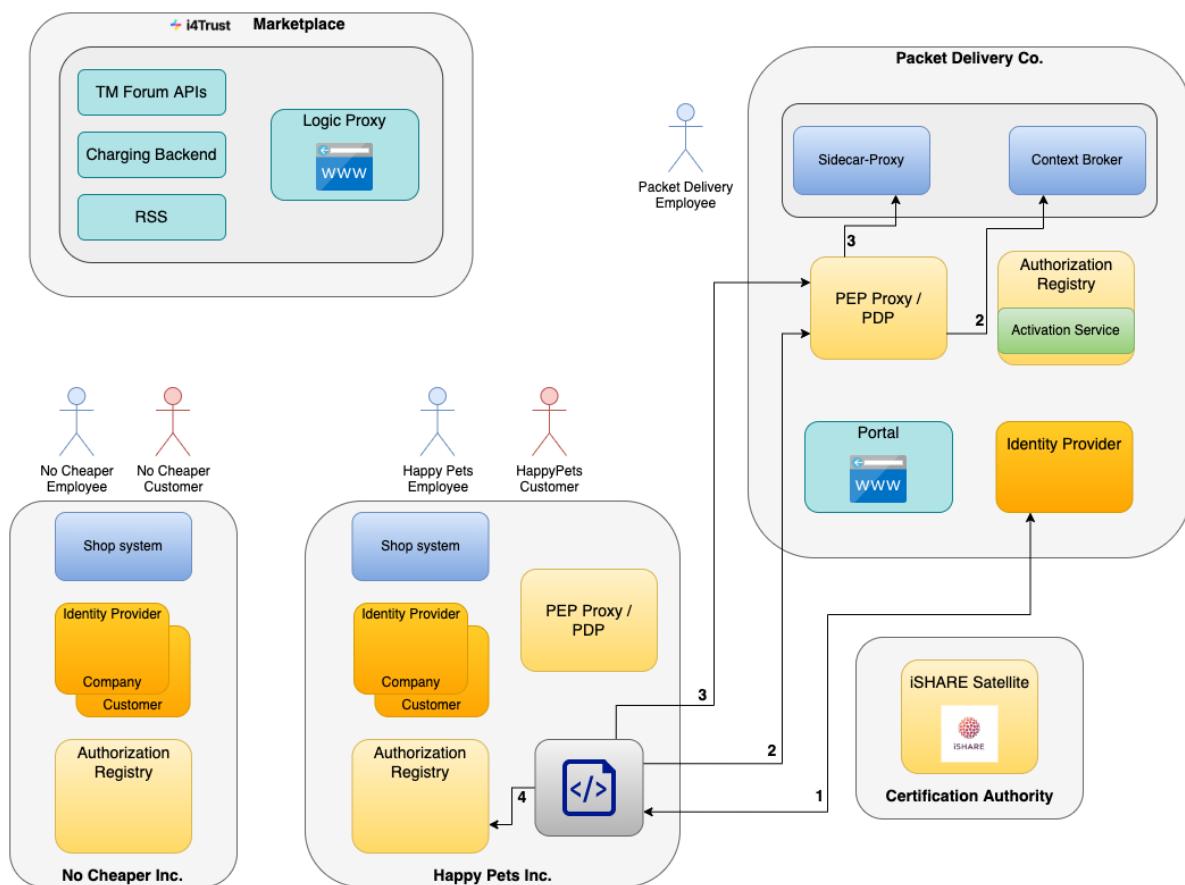
The steps during the “**Acquisition**” phase are the same as described in section 6.6. The following shows the policies that would need to be created at the Authorisation Registry of Packet Delivery Company:

```
{  
  "delegationEvidence": {  
    "policyIssuer": "EU.EORI.NLPACKETDEL",  
    "target": {  
      "accessSubject": "EU.EORI.NLHAPPYPETS"  
    },  
    "policySets": [  
      {  
        "policies": [  
          {  
            "target": {  
              "resource": {  
                "type": "DELIVERYORDER",  
                "identifiers": [  
                  "*"  
                ],  
                "attributes": [  
                  "*"  
                ]  
              },  
              "actions": [  
                "POST:Subscription"  
              ],  
              "rules": [  
                {  
                  "effect": "Permit"  
                }  
              ]  
            },  
            {  
              "target": {  
                "resource": {  
                  "type": "EndpointConfig",  
                  "identifiers": [  
                    "*"  
                  ],  
                  "attributes": [  
                    "*"  
                  ]  
                },  
                "actions": [  
                  "POST"  
                ],  
                "rules": [  
                  {  
                    "effect": "Permit"  
                  }  
                ]  
              }  
            ]  
          }  
        ]  
      }  
    ]  
  }  
}
```

In general, compared to the policies introduced before acting on entities, the action parameter of the policies gets extended compared to the regular POST, PATCH, GET and DELETE operations on entities. For instance, to allow access to get or create a certain subscription, the action parameter would be “GET:Subscription” and “POST:Subscription”, respectively. The same applies to notifications, which are only sent as POST requests and therefore require the action “POST:Notification”.

For Subscriptions, the policy parameters of type and attributes specify type and attributes of the entities that one can create subscriptions for. The same applies for notifications, where one can specify type and attributes of the entities, for which notifications can be sent.

In this use case, the created policies will allow to create subscriptions for delivery orders and to configure an endpoint for the Sidecar-Proxy authorization service. This will allow Happy Pets to perform the following steps of the **“Activation”** phase which are also depicted in the following diagram:



The following steps could be performed by some script of Happy Pets.

1. Happy Pets needs to obtain an access token at the Packet Delivery Company, which is also described in the steps 1-3 for the M2M interaction in section 6.9.3.
2. Next, Happy Pets needs to create the subscription at the Context Broker of Packet Delivery Company in order to receive notifications for changes on the PTA/PDA delivery orders. For this, Happy Pets sends a request to the PEP/PDP of Packet Delivery Company like the following, which contains the previously obtained access token.

```
> Content-Type: application/ld+json
> Authorization: Bearer IIe...NIQ

POST https://pdc-kong.i4trust-demo.fiware.dev/orion/ngsi-ld/v1/subscriptions

< Payload

{
  "description": "Notify about delivery order changes",
  "type": "Subscription",
  "entities": [{"type": "DELIVERYORDER"}],
  "watchedAttributes": ["pta", "pda"],
  "notification": {
    "attributes": ["pta", "pda", "eta", "eda"],
    "format": "normalized",
    "endpoint": {
      "uri": "https://happypets-kong.happypets.com/notify/delivery",
      "accept": "application/json"
    }
  },
  "@context": "http://context/ngsi-context.jsonld"
}
```

The subscription could also contain further query parameters in order to subscribe only to certain entities, e.g. those having specific IDs or those issued by Happy Pets only.

The Kong ngsi-ishare-policies plugin of Packet Delivery Company will verify the access token and check at the Authorization Registry of Packet Delivery Company whether Happy Pets is allowed to create such subscriptions. The request is then forwarded to the context broker of Packet Delivery Company and the subscription is created.

3. Whenever there is a change at a delivery order, the context broker of Packet Delivery Company will send a notification to the PEP/PDP endpoint of Happy Pets, as previously configured in the subscription. In order that this PEP/PDP can verify the access rights, the header of the notification requests needs to be enriched with an access token obtained from Happy Pets (like this is done in the M2M description, steps 1.3 of section 6.9.3). The sidecar-proxy will obtain the access token, add it to the header and then send the actual notification. For this, the endpoint for obtaining the access token must be configured at the

Sidecar-Proxy authorization service of Packet Delivery Company. Keyrock is capable of issuing such M2M access tokens, therefore HappyCattle sends a request to the PEP/PDP of AI4All, in order to configure the Identity Provider of HappyCattle as an endpoint at the Sidecar-Proxy of AI4All.

```
> Content-Type: application/ld+json

POST https://pdc-kong.i4trust-demo.fiware.dev/endpoint

< Payload

{
  "domain": "happypets-kong.happypets.com",
  "port": 80,
  "path": "/notify/delivery",
  "useHttps": false,
  "authType": "iShare",
  "authCredentials": {
    "iShareClientId": "EU.EORI.NLPACKETDEL",
    "iShareIdpId": "EU.EORI.NLHAPPYPETS",
    "iShareIdpAddress":
    "https://happypets-keyrock.i4trust-demo.fiware.dev/oauth2/token",
    "requestGrantType": "urn:ietf:params:oauth:grant-type:jwt-bearer"
  }
}
```

Here, IDP denotes the Keyrock instance of Happy Pets, but could be any other component that is capable to issue the required access tokens according to iSHARE specifications.

Again, the Kong plugin of Packet Delivery Company will verify the access token and check at the Authorization Registry of Packet Delivery Company whether Happy Pets is allowed to create an endpoint configuration. The request is then forwarded to the Sidecar-Proxy authorization service of Packet Delivery Company.

4. When receiving the notifications, PEP/PDP of Happy Pets will check at the Authorization Registry of Happy Pets, whether Packet Delivery Company is allowed to do so. Therefore, Happy Pets needs to create a policy issued to Packet Delivery Company at its own Authorization Registry like the one shown in the following:

```
{
  "delegationEvidence": {
    "policyIssuer": "EU.EORI.NLHAPPYPETS",
    "target": {
      "accessSubject": "EU.EORI.NLPACKETDEL"
    },
    "policySets": [
      {
        "policies": [
          {
            "id": "EU.EORI.NLPACKETDEL"
          }
        ]
      }
    ]
  }
}
```

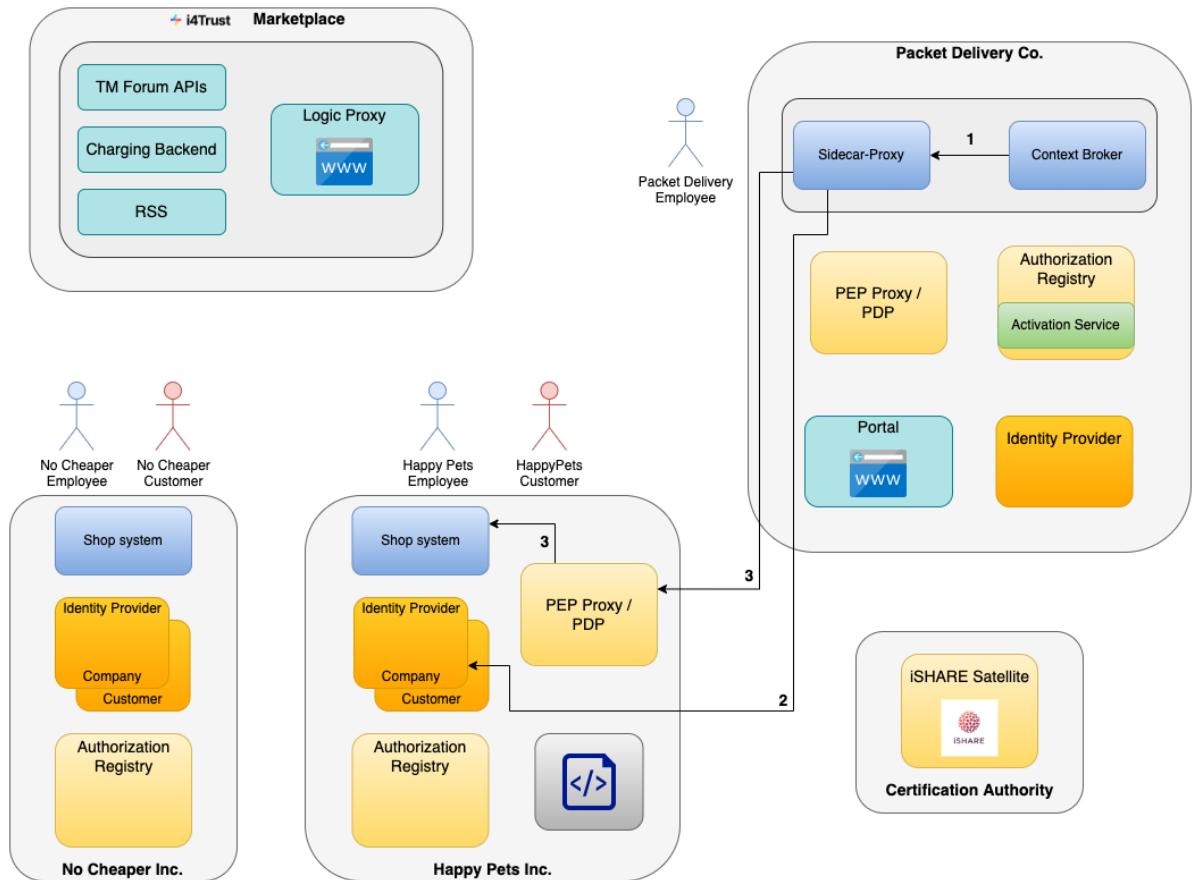
```
"target": {  
    "resource": {  
        "type": "DELIVERYORDER",  
        "identifiers": [  
            "*"  
        ],  
        "attributes": [  
            "*"  
        ]  
    },  
    "actions": [  
        "POST:Notification"  
    ],  
    "rules": [  
        {  
            "effect": "Permit"  
        }  
    ]  
}  
}  
}
```

All components are now configured, so that Happy Pets can receive notifications from Packet Delivery about changes to delivery orders. The previous steps therefore only need to be performed once after the acquisition.

#### 6.10.4 Sending notifications

In the previous steps, a subscription was created at the context broker of Packet Delivery company, so that it sends notifications to Happy Pets about changes to delivery orders.

Whenever there is a change at a delivery order, the following steps are processed which are also depicted in the following diagram:



1. The context broker sends the notification, addressed to the notification endpoint of Happy Pets specified in the subscription. Depending on the subscription, it will contain certain attributes of the delivery order. An example is given below:

```
> Content-Type: application/ld+json
POST https://happypets-kong.happypets.com/notify/delivery
< Payload
{
  "id": "urn:ngsi-ld:Notification:5fd0fa684eb81930c97005f3",
  "type": "Notification",
  "subscriptionId": "urn:ngsi-ld:Subscription:5fd0f69b4eb81930c97005db",
  "notifiedAt": "2021-12-09T16:25:12.193Z",
  "data": [
    {
      "id": "urn:ngsi-ld:DELIVERYORDER:HAPPYPETS001",
      "type": "DELIVERYORDER",
      "pda": {
        "type": "Property",
        "value": "2021-12-15"
      },
      "pta": {
        "type": "Property",
        "value": "2021-12-15"
      }
    }
  ]
}
```

```
        "type": "DELIVERYORDER",
        "value": "14:00:00"
    },
    "eda": {
        "type": "Property",
        "value": "2021-12-15"
    },
    "eta": {
        "type": "DELIVERYORDER",
        "value": "15:00:00"
    }
}
]
```

2. The notification request gets forwarded to the Sidecar-Proxy via ip-tables. The proxy now needs to obtain the access token from the Identity Provider (Keyrock) of Happy Pets, as it was configured at the endpoint authorization service. The retrieval of the access token is performed as described for the M2M interactions in steps 1-3 of section 6.9.3. The proxy then adds the access token JWT as authorization header and forwards the request to the notification endpoint of Happy Pets.
3. The PEP/PDP Kong+plugin instance of Happy Pets receives the notification request. It will verify the access token and check at the Authorization Registry of Happy Pets whether Packet Delivery Company is allowed to send such notifications. The request is then forwarded to the actual notification endpoint of Happy Pets, e.g., some endpoint of the shop system application.

## 7 Bringing the pieces together: Smart Shepherd reference example

### 7.1 Overall description

This chapter describes an i4Trust use case which aims to showcase the provisioning of an AI service and incorporation of external data within a data space. It also shows the data flow

between the data source(s) and the AI service and vice versa in a trusted and secure manner.

In general, it shows a data space where right-time data measured by the service consumer is sent to an AI service of a service provider, external data of a data service provider is incorporated into the prediction calculation, and prediction results are sent back to the service consumer.

The target is to allow an AI service provider to offer its service on a public marketplace, where a service consuming party can acquire access to this offer, inject input data into it and receive predictions resulting from the inference with the trained AI model.

The AI service is providing real-time prediction and by means of having a Context Broker deployed in the offered service, the data provisioning will be in real-time from the data source(s) to the AI model and respectively publishing the results of the predictions to the consumer application via a Context Broker on the consumer side.

In this use case, the provider is a Software company specialised in Artificial Intelligence, offering a service to predict Cattle activity based on its coordinates. The consuming parties will be a farm owner who wants to extend its farm management system to make use of the digital twin data of their cattle to get predictive animal activity.

In the PoC several parties are involved, each hosting its own infrastructure. Namely:

- **Marketplace:** Public marketplace for creating service offerings and acquiring access to them
- **Smart Shepherd Inc.:** An AI company which builds custom AI services to various use cases (Agrifood use case in this PoC)
- **HappyCattle Co.:** A Farm management system
- **Real Time Weather Ltd.:** A company which offers real-time weather data and weather forecast as services.

The figure 7.1 below presents the overall architecture of the PoC. The AI service provider and the AI consumer each have their own identity provider and authorization registry. In addition, the AI service provider subscribes to get notification based on the AI model input attributes update which are forwarded from the Context Broker hosted in the Consumer side. The AI service includes a script to pre-process the incoming data(e.g animal coordinates) to be in the fitted format as input to the AI model and to trigger the prediction. The training attributes and the prediction attribute(s) are stored in an instance of the Context Broker. Read and write access to the training and prediction attributes is controlled by a PEP Proxy and PDP according to the described roles in section 6.2.1.

The iSHARE Satellite fulfils the role of a scheme administrator which holds information about each participating party (including a global UID called EORI) and allows it to check for the admittance of each party.

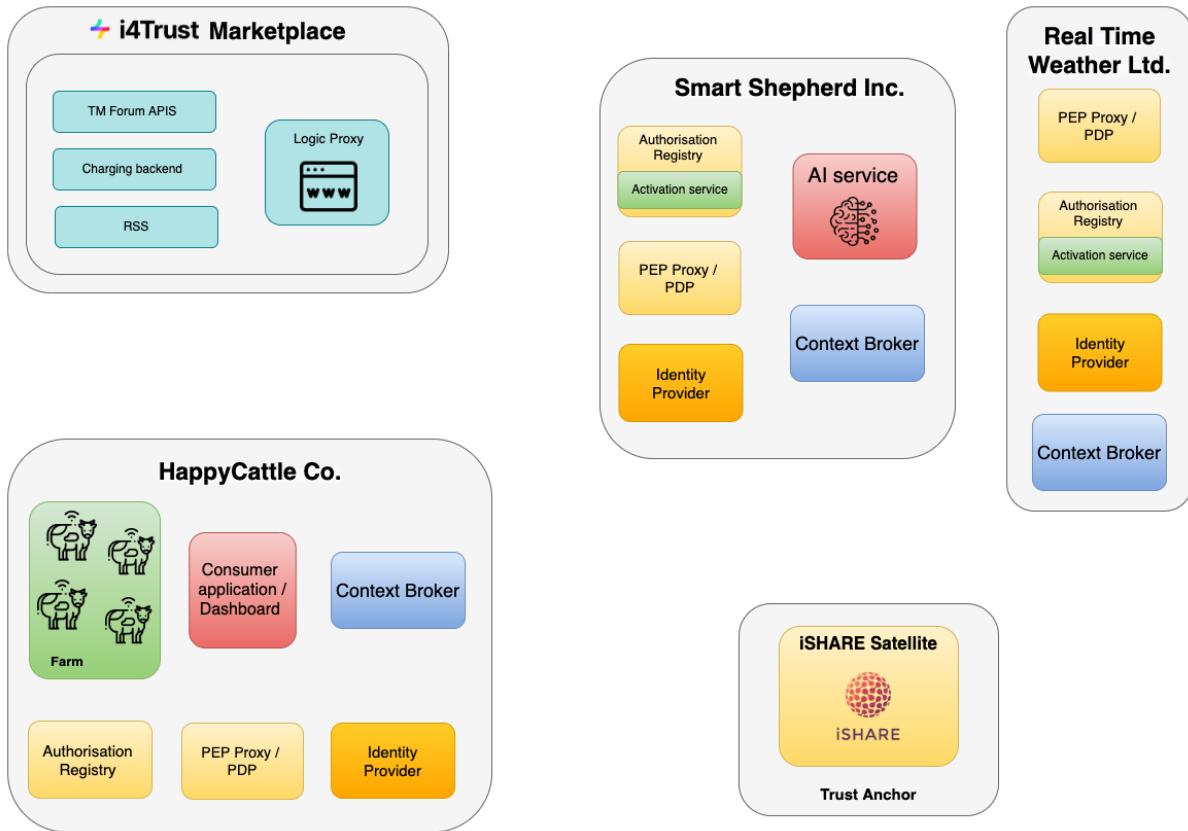


Figure 7.1: Reference Architecture

This architecture represents three different environments which will interact between each other, share data and consume an AI service in a context of a trusted data space.

The PoC implements the [iSHARE Scheme](#) in what refers to warrant trusted exchange of data between organisations, and the enforcement of access control policies. Technical details can be found [here](#).

Recipes for deploying a full running instance of the PoC can be found at [GitHub](#) (still work in progress).

## 7.2 Parties involved

In this part, the different involved parties are described with their respective roles.

### 7.2.1 AI Service Provider: Smart Shepherd Inc.

Smart Shepherd Inc. is a software development company specialized in Artificial Intelligence, they build and implement AI services and AI applications on demand depending on the proposed use case by their clients. Their solutions rely on standardized building blocks and security measures to ensure an easy integration of their services into their clients IT infrastructures/systems as well as data sovereignty.

In this use case, Smart Shepherd Inc. is offering as presented in the figure 7.2 an AI service which consists of a set of scripts that achieve data preprocessing functionality and inference with the trained AI model. In addition to that, they are deploying as part of their offering an instance of a Context Broker with the MongoDB database associated with it to manage real time data coming from the consumer. The database stores the configuration of the Context Broker, the entities and attributes of the data used to train the AI model. The data model used to create the Smart Shepherd AI services follows the [Smart Data Models](#) specification (e.g in Agrifood).

The AI service provider is responsible for training a Machine Learning model using historical data as well as providing an interface for the data provisioning needed for the inference to get future prediction based on the right-time data in NGSI-LD format received via the Context Broker. In addition to that, the output of the AI model is also being processed. Then, the prediction attribute(s) are updated in the Context Broker.

The pre-processing script is enabling the transformation of the data in NGSI-LD format to the appropriate AI model input format. Whereas, the processing script is intended to convert the data from the output of the AI model format to NGSI-LD format.

The AI service is trained and tailored to receive data about the animal through a Context Broker using NGSI-LD. An ANIMAL is an entity with attributes like:

- species: enumeration of cow, sheep, horse, pig, etc...
- location: ( x, y, z ) coordinates
- animalActivity: enumeration of Grazing, Stand up, Sternal lying, Walking, Licking calf

An example of this data model could be found under this [link](#) (still work in progress).

Before calculating the predictions, Smart Shepherd additionally retrieves weather measurement data for the Happy Cattle farm's coordinates from Real Time Weather Ltd. This process follows the mechanism of a M2M interaction with a standard data service, as described in section 6.9. This requires that the organisation Smart Shepherd acquires access to the weather data service via the i4Trust Marketplace, as described in section 6.9.2.

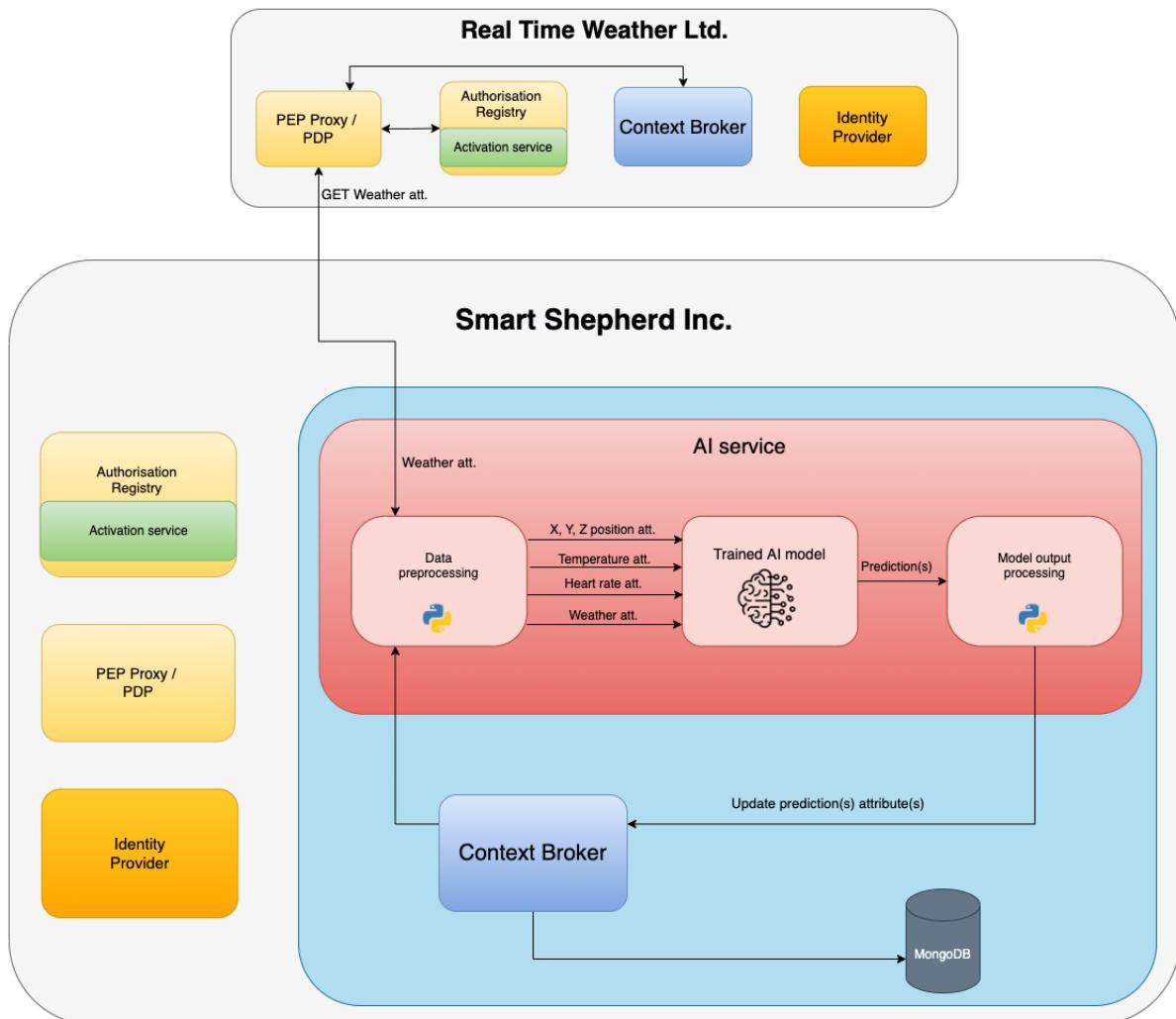


Figure 7.2: AI service infrastructure architecture

### 7.2.2 AI Service Consumer: HappyCattle Co.

HappyCattle Co. is a big farm raising animals of different kinds. They noticed that relying on the traditional human assistance they are not capable of scaling their management

process. For that reason, they have set up an IOT system and installed a farm management system that creates a digital twin of their animals in the farm. Now, they want to extend its functionality to recognize the animal activity they are raising. Therefore, they decided to rely on the Artificial Intelligence technology to gain insights from their data collected from their IOT farming system.

They will acquire the offering of the AI service for animal activity recognition on the i4Trust Marketplace.

By providing data about the animal coordinates in the x,y, z axis and labels consisting of the different animal activities (e.g Grazing, Stand up, Sternal lying, Walking, Licking calf,...), the AI service provider company could train a Machine Learning model to be able to recognize the future prediction by just providing the coordinates of the animal independently of their species.

This would make HappyCattle gain insights about their behaviour of their animals and their health condition. In return, they optimise resources and generate more profit.

The usage of the AI Service describes a purely M2M process. This use case could be also extended by additional H2M interactions.

As an example, Smart Shepherd could also provide a portal application and offer additional data services on the i4Trust Marketplace. This would incorporate the same mechanisms described in chapter 6 for the packet Delivery use case.

A human user of Happy Cattle could then access the portal using the IDP of Happy Cattle and, e.g.,

- Monitor the animal coordinates that were send by Happy Cattle
- Mark false predictions in order to improve the prediction calculations

Happy Cattle is just an example of one farmer in this use case. In a typical scenario, the data space would comprise several farmers, all using the AI service and receiving predictions. Furthermore all of the farmers could participate in further processes that involve the training of the Machine Learning model of Smart Shepherd.

### 7.2.3 Data Service Provider: Real Time Weather Ltd.

Real Time Weather Ltd. is a weather data provider, operating weather measurement stations all over the world. It allows to query current and historic weather data for specific GPS coordinates.

The weather data service is offered on the i4Trust Marketplace, so that Smart Shepherd can acquire access and incorporate the weather measurements data when calculating the predictions.

## 7.3 Prerequisites

This reference example incorporates the same type of components. For deployment and configuration, refer to the instructions in sections 6.3 and 6.4.

Here, the Kong API Gateway must be configured with the [ngsi-ishare-policies](#) and [auth-endpoint-config-ishare](#) plugins in order to act as PEP/PDP for the various requests in this reference example.

## 7.4 Create offering

This section explains the process where an employee of Smart Shepherd creates an offering for the Animal Activity prediction AI service on the Marketplace. The following architecture diagram in the figure 7.3 displays the general architecture and involved components of this process.

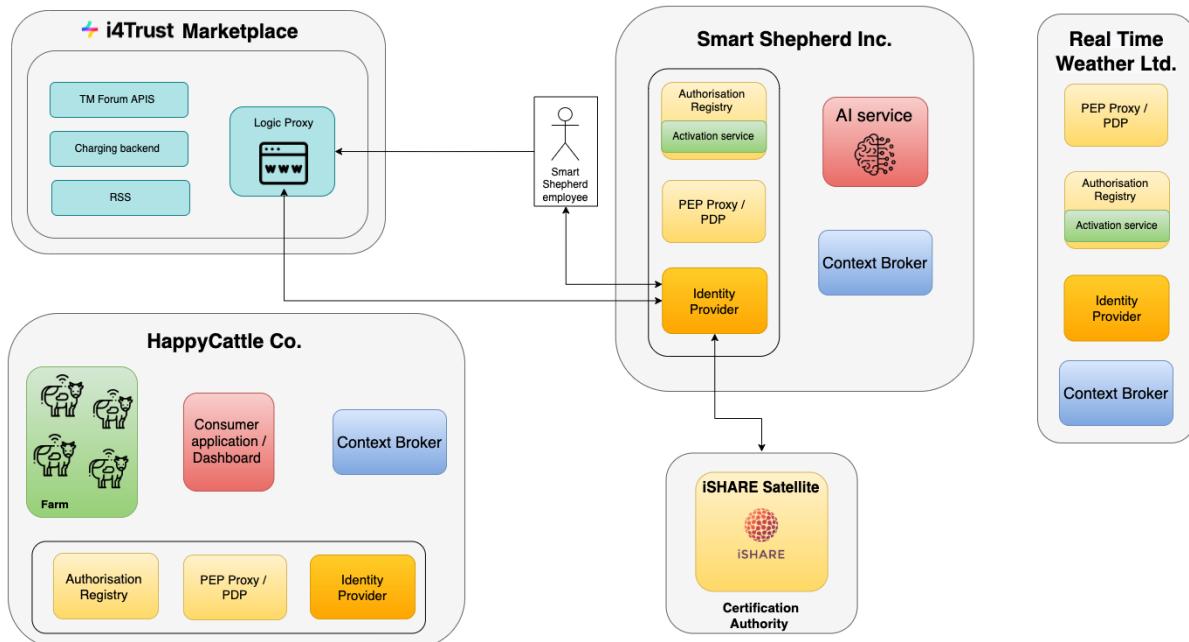


Figure 7.3: Create offering architecture

In general, this process does not differ from the description in section 6.5.1. There will be just a different offering being created on the Marketplace, providing different access rights when being acquired by another organisation.

A consumer must be able to send updates on animal coordinate measurements to the AI service, in order to trigger the calculation of predictions for the animal activity. Furthermore, the consumer must be able to receive these new predictions. Therefore, this offering should provide the following access rights:

- POST requests for entities of type “Animal” with the attribute “coordinates”
- POST requests for notifications, containing entities of type “Animal” with the attribute “coordinates”
- GET requests for entities of type “AnimalActivity” with the attributes “attr1”, “attr2”
- POST requests for creation of subscriptions for entities of type “Animal” with the attribute “coordinates”
- POST requests to configure the consumer’s Identity Provider as authorization endpoint at the Sidecar-Proxy of Smart Shepherd

At the same time, Real Time Weather Ltd. will create an offering for the weather data service. Refer to section 6.5 for the process of creating an offering for a data service. In this case the offering would be configured in a way to allow GET requests for weather measurement entities, not restricted to any attributes.

## 7.5 Acquisition of Rights / Activation

This section explains the process where HappyCattle acquires access to the AI Service of Smart Shepherd on the Marketplace. In general it is splitted into two phases:

- HappyCattle acquires access to the AI service on the Marketplace. The Marketplace will create the necessary policies for HappyCattle at the Authorisation Registry of Smart Shepherd.
- HappyCattle creates the necessary subscriptions at the Context Brokers of HappyCattle and Smart Shepherd. In addition, HappyCattle creates a policy at its own Authorization Registry, which will allow Smart Shepherd to send Notifications.

In the following, a detailed description of each single step is given. This is splitted into the two phases mentioned above, the acquisition of the AI Service offering and the creation of subscriptions.

The following architecture and sequence diagrams display the different steps involved when acquiring access to the AI Service at the Marketplace. These steps are performed by an employee of HappyCattle.

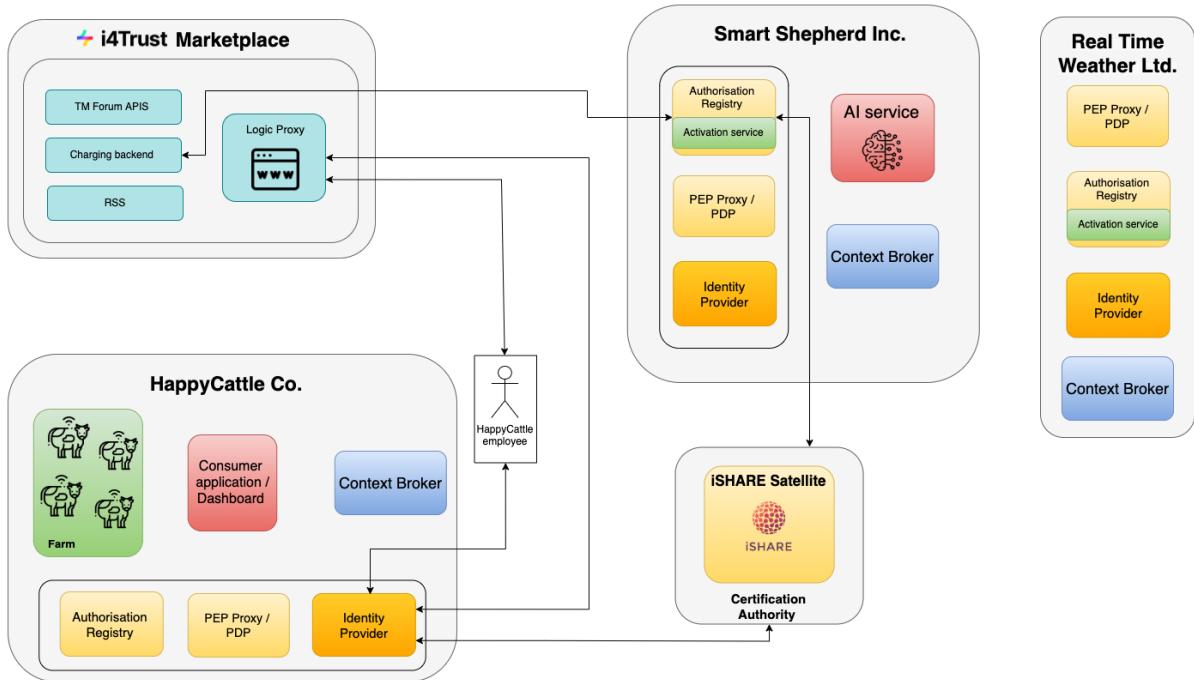


Figure 7.x: Architecture Diagram for "Acquisition on Marketplace"

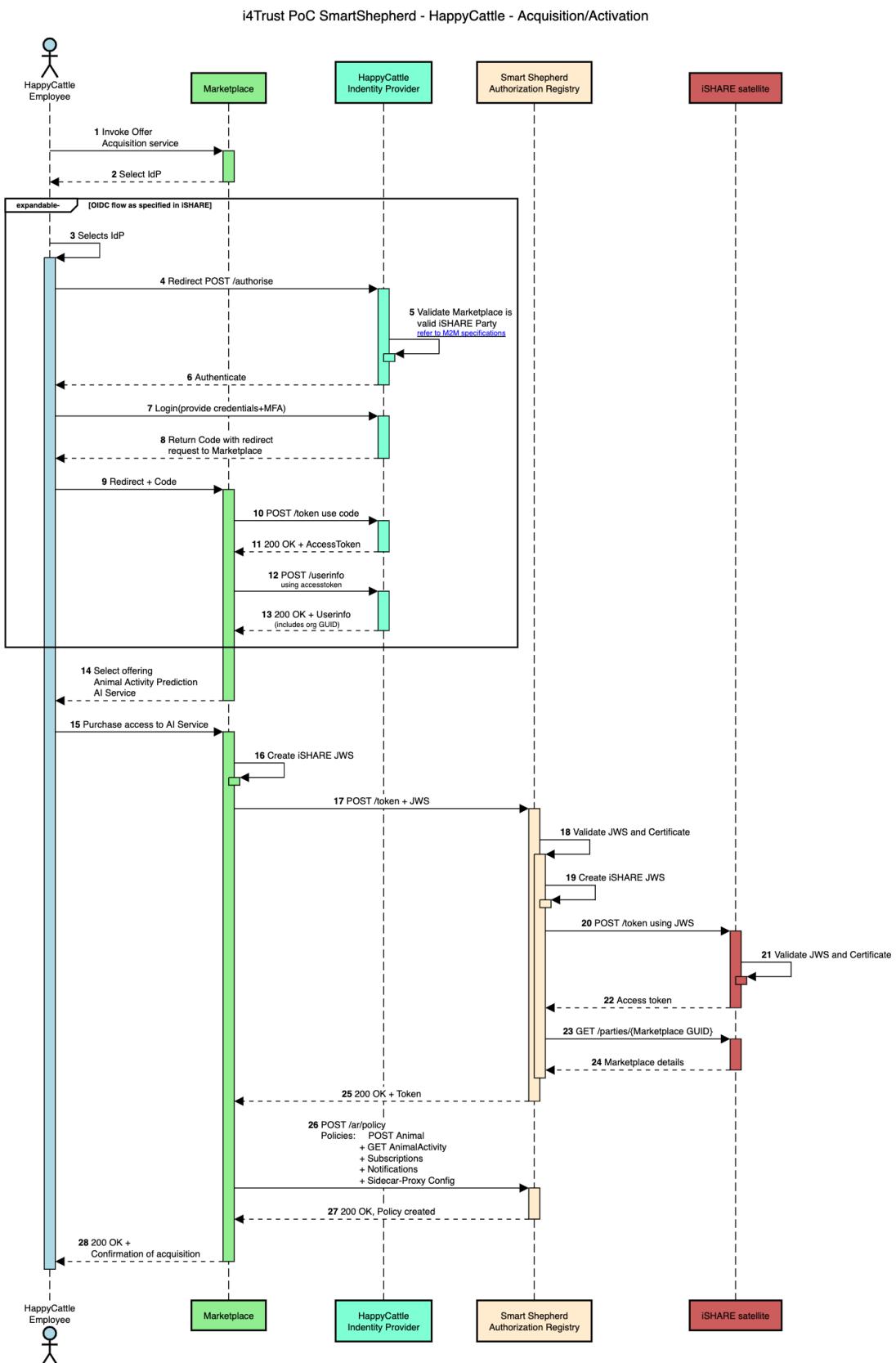


Figure 7.x: Sequence Diagram for “[Acquisition on Marketplace](#)”

In general, this process does not differ from the description from section 6.6.1. It differs by the fact that the employee of Happy Cattle acquires access to the AI Service of Smart Shepherd on the Marketplace, and there will be a different set of policies created by the Marketplace plugin at the Authorization Registry of Smart Shepherd, as displayed below:

```
{  
  "delegationEvidence": {  
    "policyIssuer": "EU.EORI.NLSMARTSHEPHERD",  
    "target": {  
      "accessSubject": "EU.EORI.NLHAPPYCATTLE"  
    },  
    "policySets": [  
      {  
        "policies": [  
          {  
            "target": {  
              "resource": {  
                "type": "Animal",  
                "identifiers": [  
                  "*"  
                ],  
                "attributes": [  
                  "coordinates"  
                ]  
              },  
              "actions": [  
                "POST:Notification", "POST"  
              ],  
              "rules": [  
                {  
                  "effect": "Permit"  
                }  
              ]  
            },  
            {  
              "target": {  
                "resource": {  
                  "type": "AnimalActivity",  
                  "identifiers": [  
                    "*"  
                  ],  
                  "attributes": [  
                    "attr1", "attr2"  
                  ]  
                },  
                "actions": [  
                  "POST:Subscription", "GET"  
                ],  
                "rules": [  
                  {  
                    "effect": "Permit"  
                  }  
                ]  
              }  
            }  
          ]  
        ]  
      }  
    ]  
  }  
}
```

```
        },
        {
          "target": {
            "resource": {
              "type": "EndpointConfig",
              "identifiers": [
                "*"
              ],
              "attributes": [
                "*"
              ]
            },
            "actions": [
              "POST"
            ],
            "rules": [
              {
                "effect": "Permit"
              }
            ]
          }
        }
      ]
    }
}
```

In the 2nd phase, HappyCattle needs to perform different steps in order to create the necessary subscriptions. It is best to automatise these steps by some script, but these steps could be also performed manually by the employee of HappyCattle. The following sequence diagram displays the different steps, which can be summarized to 5 different actions:

- Creation of a policy at the Authorization Registry of HappyCattle, which allows Smart Shepherd to send notifications to the Context Broker of HappyCattle about new animal activity predictions.
- Creation of a subscription at the Context Broker of HappyCattle, so that the Context Broker of Smart Shepherd gets notified about new animal coordinates measurements.
- Configuration of the endpoint of the Identity Provider of Smart Shepherd at the Sidecar-Proxy of HappyCattle, so that HappyCattle can retrieve the necessary access token required for sending notifications to Smart Shepherd.
- Creation of a subscription at the Context Broker of Smart Shepherd, so that the Context Broker of HappyCattle gets notified about new animal activity predictions.
- Configuration of the endpoint of the Identity Provider of HappyCattle at the Sidecar-Proxy of Smart Shepherd, so that Smart Shepherd can retrieve the necessary access token required for sending notifications to HappyCattle.

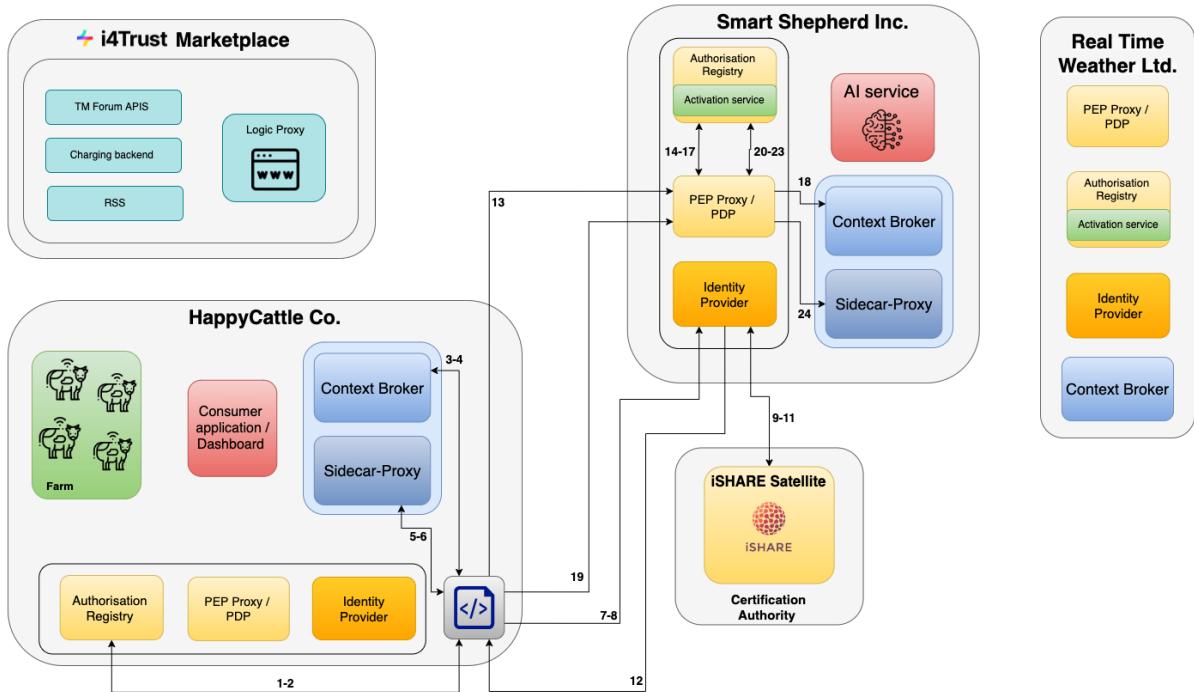
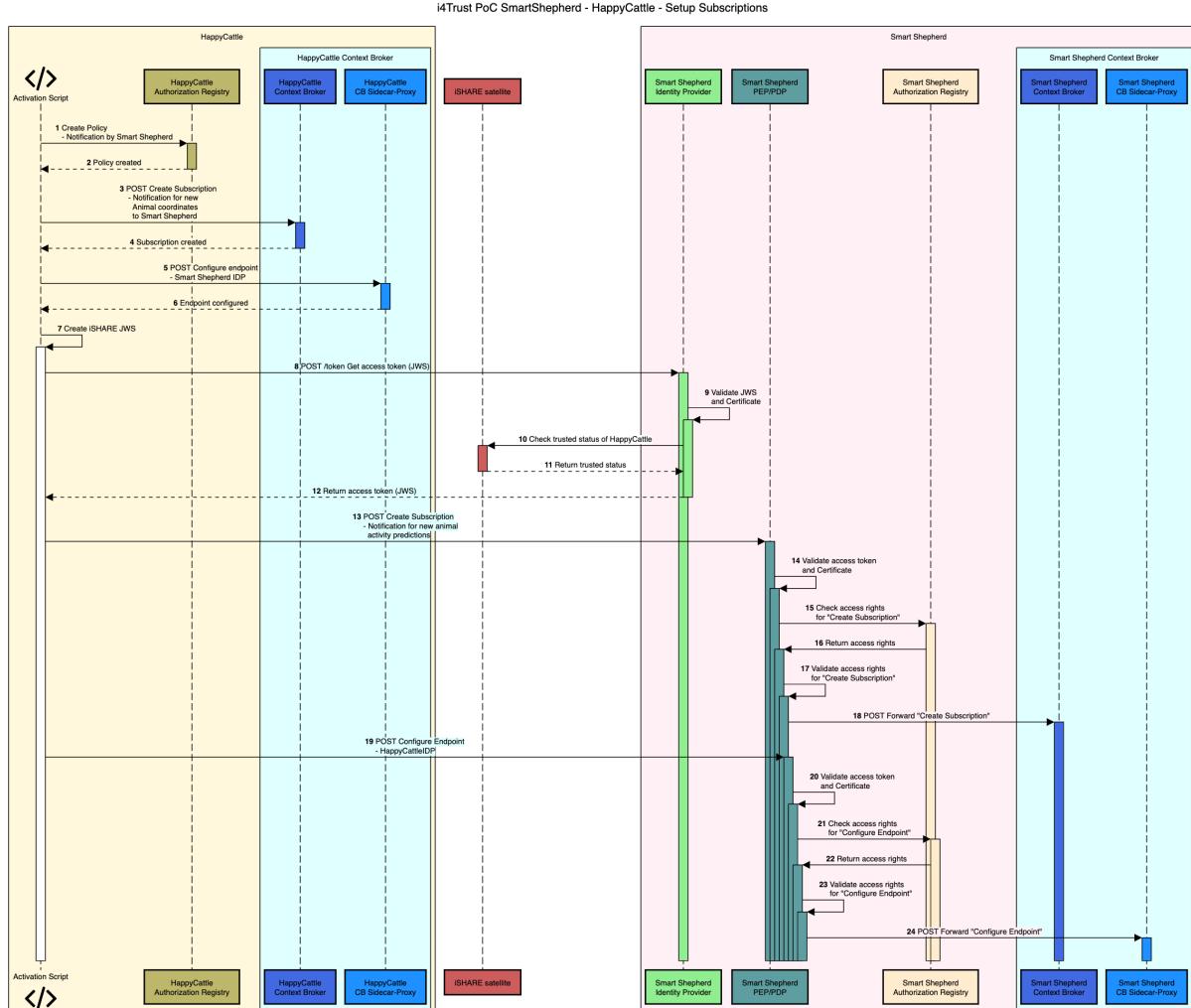


Figure 7.x: Architecture diagram for "Setup of Subscriptions"



*Figure 7.x: Sequence Diagram for "[Setup of Subscriptions](#)"*

First HappyCattle needs to create a policy at it's own Authorization Registry, that will allow Smart Shepherd to send notifications about new animal activity predictions.

1. (until 2.) In order to create a policy at the Authorization Registry, one first needs to obtain an access token, as described in steps 42-44 of section 6.7.1. Using this access token as Authorization Bearer, the policy can be created by sending the following request

```

> Authorization: Bearer <access_token>
> Content-Type: application/json

< Payload
{
  "delegationEvidence": {
    "policyIssuer": "EU.EORI.NLHAPPYCATTLE",
    "target": {
      "accessSubject": "EU.EORI.NLSMARTSHEPHERD"
    }
  }
}
  
```

```
"policySets": [
  {
    "policies": [
      {
        "target": {
          "resource": {
            "type": "Animal",
            "identifiers": [
              "*"
            ],
            "attributes": [
              "animalActivity"
            ]
          },
          "actions": [
            "POST:Notification"
          ],
          "rules": [
            {
              "effect": "Permit"
            }
          ]
        }
      }
    ]
  }
]
```

Next, HappyCattle needs to configure a subscription at its own Context Broker, so that it sends notifications to Smart Shepherd about new animal coordinate measurements.

3. (until 4.) HappyCattle creates the subscription at its own Context Broker:

```
> Content-Type: application/ld+json

POST https://orion-happycattle.i4trust.fiware.io/ngsi-ld/v1/subscriptions

< Payload

{
  "description": "Notify about new animal coordinate measurements",
  "type": "Subscription",
  "entities": [{"type": "Animal"}],
  "watchedAttributes": ["coordinates"],
  "notification": {
    "attributes": ["coordinates"],
    "format": "normalized",
    "endpoint": {
      "uri": "https://umbrella-smartshepherd.i4trust.fiware.io/notify/animal",
      "accept": "application/json"
    }
  },
  "@context": "http://context/ngsi-context.jsonld"
}
```

5. (until 6.) Additionally, HappyCattle configures the endpoint of the Identity Provider of Smart Shepherd at the Sidecar-Proxy of HappyCattle

```
> Content-Type: application/ld+json

POST https://orion-proxy-happycattle.i4trust.fiware.io/endpoint

< Payload

{
    "domain": "happycattle.com",
    "port": 80,
    "path": "/notify/animal",
    "useHttps": false,
    "authType": "iShare",
    "authCredentials": {
        "iShareClientId": "EU.EORI.NLHAPPYCATTLE",
        "iShareIdpId": "EU.EORI.NLSMARTSHEPHERD",
        "iShareIdpAddress":
        "https://smartshepherd-keyrock.i4trust-demo.fiware.dev/oauth2/token",
        "requestGrantType": "urn:ietf:params:oauth:grant-type:jwt-bearer"
    }
}
```

Next, two similar steps need to be performed at Smart Shepherd, but require first to obtain an access token in order to create a subscription and configure an endpoint at Smart Shepherd.

7. (until 8.) First, HappyCattle (e.g., the activation script) generates an [iSHARE JWT](#) signed using the JSON Web Signature standard (JWS), as also described in step 4 of section 6.7.1.

```
> Headers
{
    "alg": "RS256",
    "typ": "JWT",
    "x5c": [
        "MIIEhjCC....Zy9w=="
    ]
}

> Payload
{
    "iss": "EU.EORI.NLHAPPYCATTLE",
    "sub": "EU.EORI.NLHAPPYCATTLE",
    "aud": [
        "EU.EORI.NLSMARTSHEPHERD",
        "https://smartshepherd-keyrock.i4trust.fiware.io/token"
    ],
    "jti": "99ab5bca41bb45b78d242a46f0157b7d",
    "iat": "1540827435",
    "exp": 1540827465,
    "nbf": 1540827435
}
```

Then, HappyCattle sends a request to the Smart Shepherd Identity Provider [/token](#) endpoint. The signed JWT created before is provided with the `client_assertion` parameter of the request. Note the different `grant_type` parameter, since this is a M2M communication, as also described in section 6.9.3.

```
> Content-Type: application/x-www-form-urlencoded  
  
POST https://smartshepherd-keyrock.i4trust.fiware.io/oauth2/token  
  
grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&  
scope=iSHARE&  
client_id=EU.EORI.NLHAPPYCATTLE&  
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&  
client_assertion=eyJhbGc...9hvW
```

9. Smart Shepherd Identity Provider validates the JWT and certificate of HappyCattle.
10. Next, the Identity Provider of Smart Shepherd needs to verify the status of HappyCattle as a trusted iSHARE participant. For this, it requests this information at the iSHARE Satellite, as described in steps 9-12 of section 6.7.1.
11. iSHARE Satellite returns the information about HappyCattle and the Identity Provider of Smart Shepherd verifies its status.
12. Since HappyCattle is a trusted participant, the Identity Provider of Smart Shepherd returns the access token as an encoded and signed JWT. This JWT access token can now be used to send requests to the AI service of Smart Shepherd.

```
< Content-Type: application/json  
< Cache-Control: no-store  
< Pragma: no-cache  
  
{  
  "access_token": "aw2y...LI0w",  
  "expires_in": 3600,  
  "token_type": "Bearer"  
}  
  
> Decoded payload of access token  
{  
  "iss": "EU.EORI.NLSMARTSHEPHERD",  
  "sub": "EU.EORI.NLHAPPYCATTLE",  
  "jti": "99ab5bca41bb45b78d242a46f0157b7d",  
  "iat": "1540827435",  
  "exp": 1540827465,  
  "nbf": 1540827435,  
  "aud": "EU.EORI.NLSMARTSHEPHERD"  
}
```

Using this access token, HappyCattle can now create a subscription and configure an endpoint at Smart Shepherd

13. HappyCattle sends a request to the PEP/PDP of Smart Shepherd, in order to create a subscription at the Context Broker of Smart Shepherd that will notify HappyCattle about new animal activity predictions

```
> Content-Type: application/ld+json

POST https://smartshepherd-kong.i4trust.fiware.io/orion/ngsi-ld/v1/subscriptions

< Payload

{
  "description": "Notify about new animal activity predictions",
  "type": "Subscription",
  "entities": [{"type": "Animal"}],
  "watchedAttributes": ["animalActivity"],
  "notification": {
    "attributes": ["animalActivity"],
    "format": "normalized",
    "endpoint": {
      "uri": "https://happy cattle-kong.i4trust.fiware.io/notify/animal",
      "accept": "application/json"
    }
  },
  "@context": "http://context/ngsi-context.jsonld"
}
```

14. The PEP/PDP of Smart Shepherd validates the JWT and certificate which is part of the authorization header of the “Create Subscription” request.
15. The PEP/PDP of Smart Shepherd now needs to check whether HappyCattle is allowed to create such Subscriptions. Therefore it needs to check at its Authorization Registry, if the necessary policy for HappyCattle exists. These steps are also described in detail in steps 42-46 of section 6.7.1.  
The following presents the payload sent to the [/delegation](#) endpoint of the Authorization Registry of Smart Shepherd, in order to retrieve the delegation evidence policy issued from Smart Shepherd to HappyCattle.

```
POST 'https://idp-Smart Shepherd.i4trust.fiware.io/ar/delegation'

{
  "delegationRequest": {
    "policyIssuer": "EU.EORI.NLSMARTSHEPHERD",
    "target": {
      "accessSubject": "EU.EORI.NLHAPPYCATTLE"
    },
    "policySets": [
      {
        "policies": [
          {
            "target": {
              "resource": {
                "type": "Animal",
                "id": "urn:ngsi-ld:Animal:1234567890"
              }
            }
          }
        ]
      }
    ]
  }
}
```

```
        "identifiers": [
            "*"
        ],
        "attributes": [
            "animalActivity"
        ]
    },
    "actions": [
        "POST:Subscription"
    ],
    "rules": [
        {
            "effect": "Permit"
        }
    ]
}
]
```

16. If there is such a policy, it is returned as JWT delegation\_token.

```
< Content-Type: application/json
> Payload
{
    "delegation_token": "eyJ4...0aaQ"
}

Decoded delegation_token JWT payload
{
    "iss": "EU.EORI.NLSMARTSHEPHERD",
    "sub": "EU.EORI.NLSMARTSHEPHERD",
    "jti": "d8a7fd7465754a4a9117ee28f5b7fb60",
    "iat": 1591966224,
    "exp": 1591966254,
    "aud": "EU.EORI.NLSMARTSHEPHERD",
    "delegationEvidence": {
        "notBefore": 1541058939,
        "notOnOrAfter": 2147483647,
        "policyIssuer": "EU.EORI.NLSMARTSHEPHERD",
        "target": {
            "accessSubject": "EU.EORI.NLHAPPYCATTLE"
        },
        "policySets": [
            {
                "maxDelegationDepth": 1,
                "target": {
                    "environment": {
                        "licenses": [
                            "ISHARE.0001"
                        ]
                    }
                },
                "policies": [
                    {
                        "target": {
                            "accessSubject": "EU.EORI.NLHAPPYCATTLE"
                        }
                    }
                ]
            }
        ]
    }
}
```

```
        "resource": {
            "type": "Animal",
            "identifiers": [
                "*"
            ],
            "attributes": [
                "animalActivity"
            ]
        },
        "actions": [
            "POST:Subscription"
        ],
        "rules": [
            {
                "effect": "Permit"
            }
        ]
    }
}
```

17. The PEP/PDP checks if the policy is valid and whether it contains the necessary Permit rule.
18. If access is granted, the PEP/PDP forwards the request for the creation of a subscription to the Context Broker of Smart Shepherd.
19. HappyCattle sends a request to the PEP/PDP of Smart Shepherd, in order to configure the Identity Provider of HappyCattle as an endpoint at the Sidecar-Proxy of Smart Shepherd.

```
> Content-Type: application/ld+json

POST https://smartshepherd-kong.i4trust.fiware.io/endpoint

< Payload

{
    "domain": "smartshepherd.com",
    "port": 80,
    "path": "/notify/animal",
    "useHttps": false,
    "authType": "iShare",
    "authCredentials": {
        "iShareClientId": "EU.EORI.NLSMARTSHEPHERD",
        "iShareIdpId": "EU.EORI.NLHAPPYCATTLE",
        "iShareIdpAddress": "https://happycattle-keyrock.i4trust-demo.fiware.dev/oauth2/token",
        "requestGrantType": "urn:ietf:params:oauth:grant-type:jwt-bearer"
    }
}
```

When Smart Shepherd sends a notification to HappyCattle, the Sidecar-Proxy will first obtain an access token at this endpoint and will send it along with the notification as Authorization Bearer token.

20. The PEP/PDP of Smart Shepherd validates the JWT and certificate which is part of the authorization header of the "Create Subscription" request.

21. The PEP/PDP of Smart Shepherd now needs to check whether HappyCattle is allowed to configure such an endpoint. Therefore it needs to check at its Authorization Registry, if the necessary policy for HappyCattle exists. These steps are also described in detail in steps 42-46 of section 6.7.1.

The following presents the payload sent to the [/delegation](#) endpoint of the Authorization Registry of Smart Shepherd, in order to retrieve the delegation evidence policy issued from Smart Shepherd to HappyCattle.

```
POST 'https://smartshepherd-keyrock.i4trust.fiware.io/ar/delegation'
```

```
{
  "delegationRequest": {
    "policyIssuer": "EU.EORI.NLSMARTSHEPHERD",
    "target": {
      "accessSubject": "EU.EORI.NLHAPPYCATTLE"
    },
    "policySets": [
      {
        "policies": [
          {
            "target": {
              "resource": {
                "type": "EndpointConfig",
                "identifiers": [
                  "*"
                ],
                "attributes": [
                  "*"
                ]
              },
              "actions": [
                "POST"
              ],
              "rules": [
                {
                  "effect": "Permit"
                }
              ]
            }
          }
        ]
      }
    ]
  }
}
```

22. If there is such a policy, it is returned as JWT `delegation_token`.

```
< Content-Type: application/json
> Payload
{
  "delegation_token": "eyJ4...0aaQ"
}

Decoded delegation_token JWT payload
{
  "iss": "EU.EORI.NLSMARTSHEPHERD",
  "sub": "EU.EORI.NLSMARTSHEPHERD",
  "jti": "d8a7fd7465754a4a9117ee28f5b7fb60",
  "iat": 1591966224,
  "exp": 1591966254,
  "aud": "EU.EORI.NLSMARTSHEPHERD",
  "delegationEvidence": {
    "notBefore": 1541058939,
    "notOnOrAfter": 2147483647,
    "policyIssuer": "EU.EORI.NLSMARTSHEPHERD",
    "target": {
      "accessSubject": "EU.EORI.NLHAPPYCATTLE"
    },
    "policySets": [
      {
        "maxDelegationDepth": 1,
        "target": {
          "environment": {
            "licenses": [
              "ISHARE.0001"
            ]
          }
        },
        "policies": [
          {
            "target": {
              "resource": {
                "type": "EndpointConfig",
                "identifiers": [
                  "*"
                ],
                "attributes": [
                  "*"
                ]
              },
              "actions": [
                "POST"
              ]
            },
            "rules": [
              {
                "effect": "Permit"
              }
            ]
          }
        ]
      }
    ]
  }
}
```

23. The PEP/PDP checks if the policy is valid and whether it contains the necessary Permit rule.
24. If access is granted, the PEP/PDP forwards the request for the endpoint configuration to the Sidecar-Proxy of Smart Shepherd.

Now all necessary subscriptions have been created. As soon as the Context Broker of HappyCattle receives a new animal coordinate measurement from one of its IoT devices, the whole process of consuming the AI Service gets triggered, as described in the next section.

At the same time, Smart Shepherd needs to acquire access to the weather data service of Real Time Weather Ltd., offered on the marketplace. Refer to section 6.6 for the process of acquisition of a data service offering.

## 7.6 Usage of AI service

This section explains the process where the AI service is used. In general it is splitted into two phases:

- HappyCattle notifies the AI service about a new measurement of animal coordinates, which triggers the calculation of a new prediction for animal activity within the AI service of Smart Shepherd.
- Smart Shepherd notifies HappyCattle about the new prediction. On the dashboard of HappyCattle one can finally display the new values.

The following diagram gives a general overview of the architecture of these two phases.

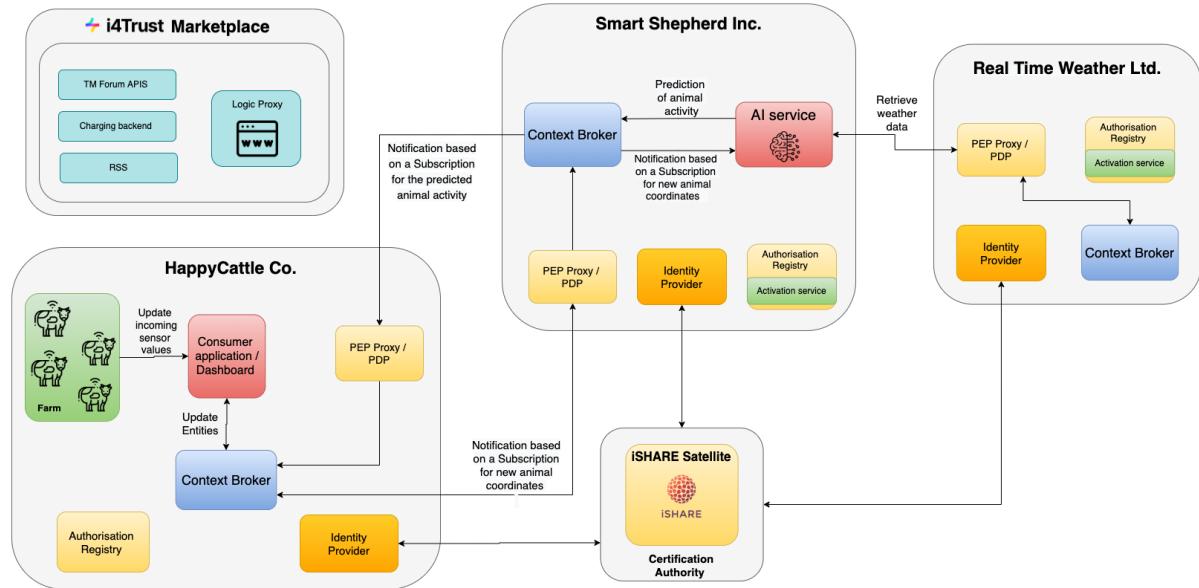


Figure 7.x: Usage of AI service architecture

In the following, a detailed description of each single step is given. This is splitted into the two phases mentioned above, the notification about a new measurement and the notification about a new prediction. Several steps do not differ compared to the descriptions in section 6.7.1, therefore this section is referenced in such cases.

The following architecture and sequence diagrams display the different steps involved, when at the farm of HappyCattle there is a new measurement of animal coordinates, and HappyCattle informs the AI service of Smart Shepherd about it. In the following, a description is given for each of the sequence steps.

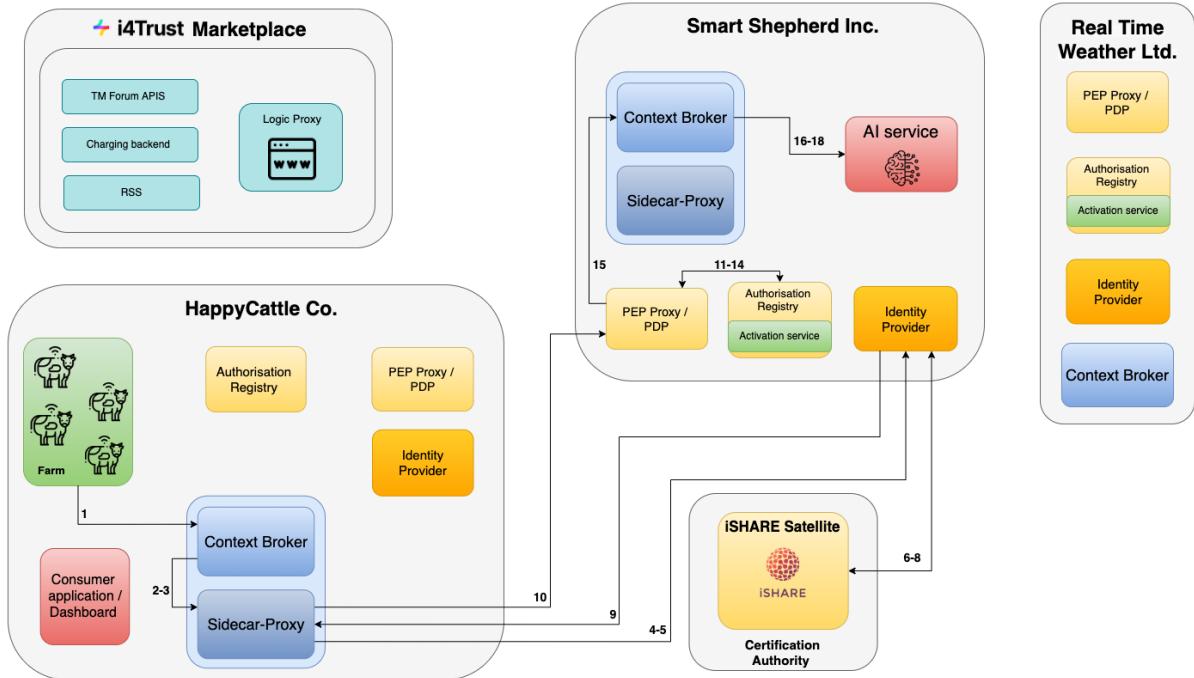


Figure 7.x: Architecture Diagram for “Update new animal coordinates”

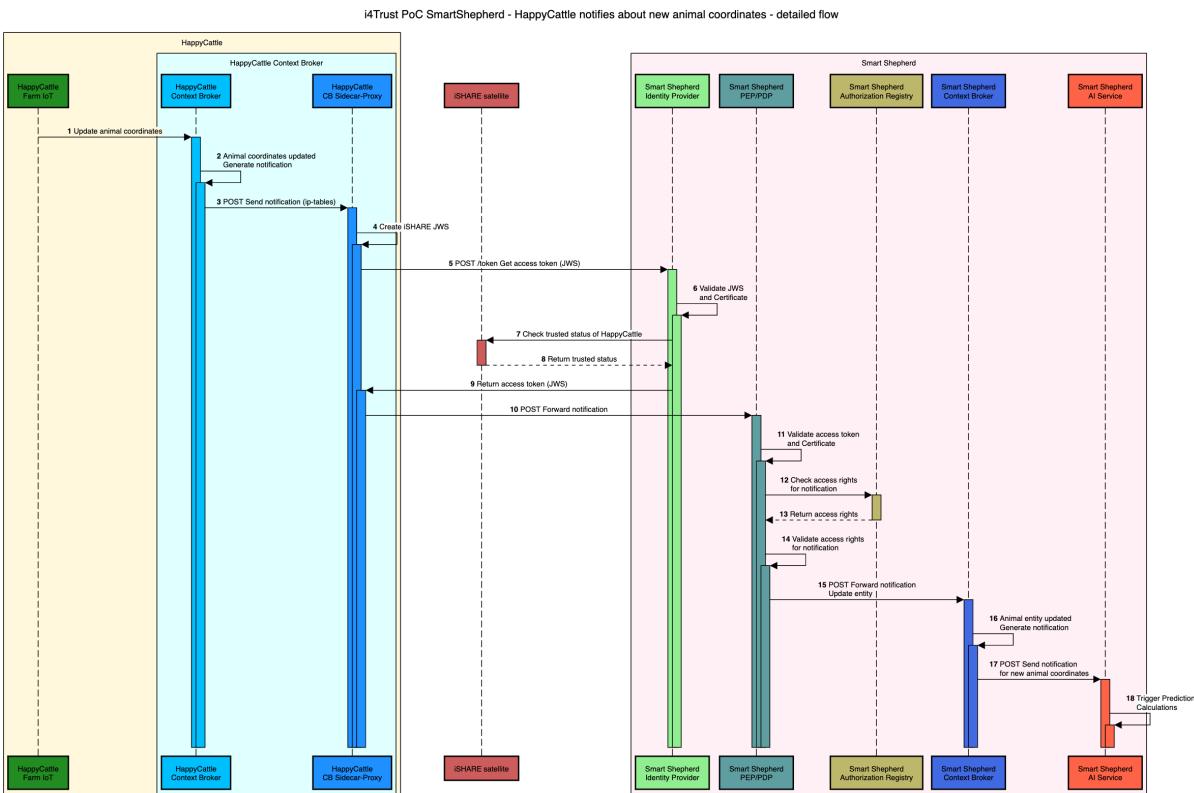


Figure 7.x: Sequence Diagram for “Update new animal coordinates”

1. One of the HappyCattle farmer's IoT devices updates the coordinates of an animal entity at the Context Broker of HappyCattle
2. (until 3.) Since there is a subscription about new animal coordinates at the Context Broker of Smart Shepherd, the Context Broker of HappyCattle creates a Notification about the updated animal entity and sends it to the PEP/PDP of Smart Shepherd.

```
> Content-Type: application/ld+json

POST https://smartshepherd-kong.i4trust.fiware.io/notify/animal

< Payload
{
  "id": "urn:ngsi-ld:Notification:5fd0fa684eb81930c97005f3",
  "type": "Notification",
  "subscriptionId": "urn:ngsi-ld:Subscription:5fd0f69b4eb81930c97005db",
  "notifiedAt": "2021-12-09T16:25:12.193Z",
  "data": [
    {
      "id": "urn:ngsi-ld:Animal:TODO-ID",
      "type": "Animal",
      "coordinates": {
        "type": "Property",
        "value": "XXX"
      }
    }
  ]
}
```

The Notification is first forwarded to the Sidecar-Proxy of HappyCattle via ip-tables, in order to retrieve the necessary access token before sending the request to the PEP/PDP of Smart Shepherd.

4. (until 5.) The Sidecar-Proxy of HappyCattle needs to retrieve an access token, before it can forward the Notification to the PEP/PDP of Smart Shepherd.

First, the proxy generates an [iSHARE JWT](#) signed using the JSON Web Signature standard (JWS), as also described in step 4 of section 6.7.1.

```
> Headers
{
  "alg": "RS256",
  "typ": "JWT",
  "x5c": [
    "MIIIEhjCC....Zy9w=="
  ]
}

> Payload
{
  "iss": "EU.EORI.NLHAPPYCATTLE",
  "sub": "EU.EORI.NLHAPPYCATTLE",
  "aud": [
```

```
    "EU.EORI.NLSMARTSHEPHERD",
    "https://smartshepherd-keyrock.i4trust.fiware.io/token"
],
"jti": "99ab5bca41bb45b78d242a46f0157b7d",
"iat": "1540827435",
"exp": 1540827465,
"nbf": 1540827435
}
```

Then, the proxy sends a request to the Smart Shepherd Identity Provider [/token](#) endpoint. The signed JWT created before is provided with the `client_assertion` parameter of the request. Note the different `grant_type` parameter, since this is a M2M communication, as also described in section 6.9.3.

```
> Content-Type: application/x-www-form-urlencoded
POST https://smartshepherd-keyrock.i4trust.fiware.io/oauth2/token
grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&
scope=iSHARE&
client_id=EU.EORI.NLHAPPYCATTLE&
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&
client_assertion=eyJhbGc...9hvW
```

5. Smart Shepherd Identity Provider validates the JWT and certificate of HappyCattle.
6. Next, the Identity Provider of Smart Shepherd needs to verify the status of HappyCattle as a trusted iSHARE participant. For this, it requests this information at the iSHARE Satellite, as described in steps 9-12 of section 6.7.1.
7. iSHARE Satellite returns the information about HappyCattle and the Identity Provider of Smart Shepherd verifies its status.
8. Since HappyCattle is a trusted participant, the Identity Provider of Smart Shepherd returns the access token as an encoded and signed JWT. This JWT access token can now be used to send requests to the AI service of Smart Shepherd.

```
< Content-Type: application/json
< Cache-Control: no-store
< Pragma: no-cache

{
  "access_token": "aW2y...LI0w",
  "expires_in": 3600,
  "token_type": "Bearer"
}
```

```
> Decoded payload of access token
{
  "iss": "EU.EORI.NLSMARTSHEPHERD",
  "sub": "EU.EORI.NLHAPPYCATTLE",
  "jti": "99ab5bca41bb45b78d242a46f0157b7d",
  "iat": "1540827435",
  "exp": 1540827465,
  "nbf": 1540827435,
  "aud": "EU.EORI.NLSMARTSHEPHERD"
}
```

10. The Sidecar-Proxy now forwards the initial Notification to the PEP/PDP of Smart Shepherd, where the previously obtained access token is put into the Authorization header of the POST request.

```
> Content-Type: application/ld+json
> Authorization: Bearer aw2y...LI0w

POST https://umbrella-Smart Shepherd.i4trust.fiware.io/notify/animal

< Payload
{
  "id": "urn:ngsi-ld:Notification:5fd0fa684eb81930c97005f3",
  "type": "Notification",
  "subscriptionId": "urn:ngsi-ld:Subscription:5fd0f69b4eb81930c97005db",
  "notifiedAt": "2021-12-09T16:25:12.193Z",
  "data": [
    {
      "id": "urn:ngsi-ld:Animal:TODO-ID",
      "type": "Animal",
      "coordinates": {
        "type": "Property",
        "value": "XXX"
      }
    }
  ]
}

Decoded Bearer JWT payload
{
  "iss": "EU.EORI.NLSMARTSHEPHERD",
  "sub": "EU.EORI.NLHAPPYCATTLE",
  "jti": "99ab5bca41bb45b78d242a46f0157b7d",
  "iat": "1540827435",
  "exp": 1540827465,
  "nbf": 1540827435,
  "aud": "EU.EORI.NLSMARTSHEPHERD"
}
```

11. The PEP/PDP of Smart Shepherd validates the JWT and certificate which is part of the authorization header of the Notification request.

12. The PEP/PDP of Smart Shepherd now needs to check whether HappyCattle is allowed to send such Notification. Therefore it needs to check at its Authorization Registry, if the necessary policy for HappyCattle exists. These steps are also described in detail in steps 42-46 of section 6.7.1. The following presents the payload sent to the [/delegation](#) endpoint of the Authorization Registry of Smart Shepherd, in order to retrieve the delegation evidence policy issued from Smart Shepherd to HappyCattle.

```
POST 'https://idp-Smart Shepherd.i4trust.fiware.io/ar/delegation'

{
  "delegationRequest": {
    "policyIssuer": "EU.EORI.NLSMARTSHEPHERD",
    "target": {
      "accessSubject": "EU.EORI.NLHAPPYCATTLE"
    },
    "policySets": [
      {
        "policies": [
          {
            "target": {
              "resource": {
                "type": "Animal",
                "identifiers": [
                  "urn:ngsi-ld:Subscription:5fd0f69b4eb81930c97005db"
                ],
                "attributes": [
                  "coordinates"
                ]
              },
              "actions": [
                "POST:Notification"
              ],
              "rules": [
                {
                  "effect": "Permit"
                }
              ]
            }
          }
        ]
      }
    ]
  }
}
```

13. If there is such a policy, it is returned as JWT `delegation_token`.

```
< Content-Type: application/json
> Payload
{
  "delegation_token": "eyJ4...0aaQ"
}

Decoded delegation_token JWT payload
{
  "iss": "EU.EORI.NLSMARTSHEPHERD",
```

```

"sub": "EU.EORI.NLSMARTSHEPHERD",
"jti": "d8a7fd7465754a4a9117ee28f5b7fb60",
"iat": 1591966224,
"exp": 1591966254,
"aud": "EU.EORI.NLSMARTSHEPHERD",
"delegationEvidence": {
    "notBefore": 1541058939,
    "notOnOrAfter": 2147483647,
    "policyIssuer": "EU.EORI.NLSMARTSHEPHERD",
    "target": {
        "accessSubject": "EU.EORI.NLHAPPYCATTLE"
    },
    "policySets": [
        {
            "maxDelegationDepth": 1,
            "target": {
                "environment": {
                    "licenses": [
                        "ISHARE.0001"
                    ]
                }
            },
            "policies": [
                {
                    "target": {
                        "resource": {
                            "type": "Animal",
                            "identifiers": [
                                "*"
                            ],
                            "attributes": [
                                "coordinates"
                            ]
                        },
                        "actions": [
                            "POST:Notification"
                        ]
                    },
                    "rules": [
                        {
                            "effect": "Permit"
                        }
                    ]
                }
            ]
        }
    ]
}

```

14. The PEP/PDP checks if the policy is valid and whether it contains the necessary Permit rule.
15. If access is granted, the PEP/PDP forwards the notification to the Context Broker of Smart Shepherd.
16. The Animal entity gets updated at the Context Broker of Smart Shepherd. Since there is a subscription of the AI service for new animal coordinate

measurements, the Context Broker will generate a Notification for the new measurement.

17. The Context Broker sends a Notification about the new measurement to the AI Service.
18. The received Notification triggers the calculation of a new animal activity prediction within the AI Service

During the phase of prediction calculation, Smart Shepherd will retrieve the current weather data from the weather data service of Real Time Weather Ltd., as depicted in the following diagram.

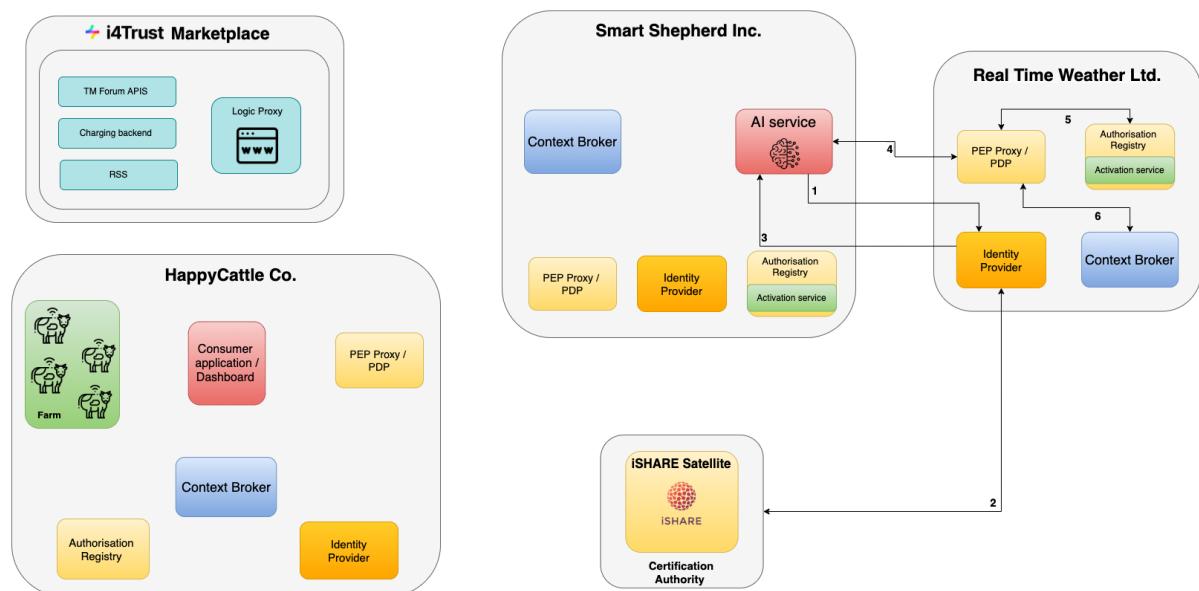


Figure 7.x: Architecture Diagram for "Retrieve weather data"

Refer to section 6.9.3 for a description of the process of usage of a data service via the M2M mechanism.

The 2nd phase begins with the calculation of a new prediction for animal activity by the AI Service of Smart Shepherd. The following architecture and sequence diagrams display the different steps involved, when at Smart Shepherd there is a new prediction of animal activity, and Smart Shepherd informs HappyCattle about it. In the following, a description is given for each of the sequence steps.

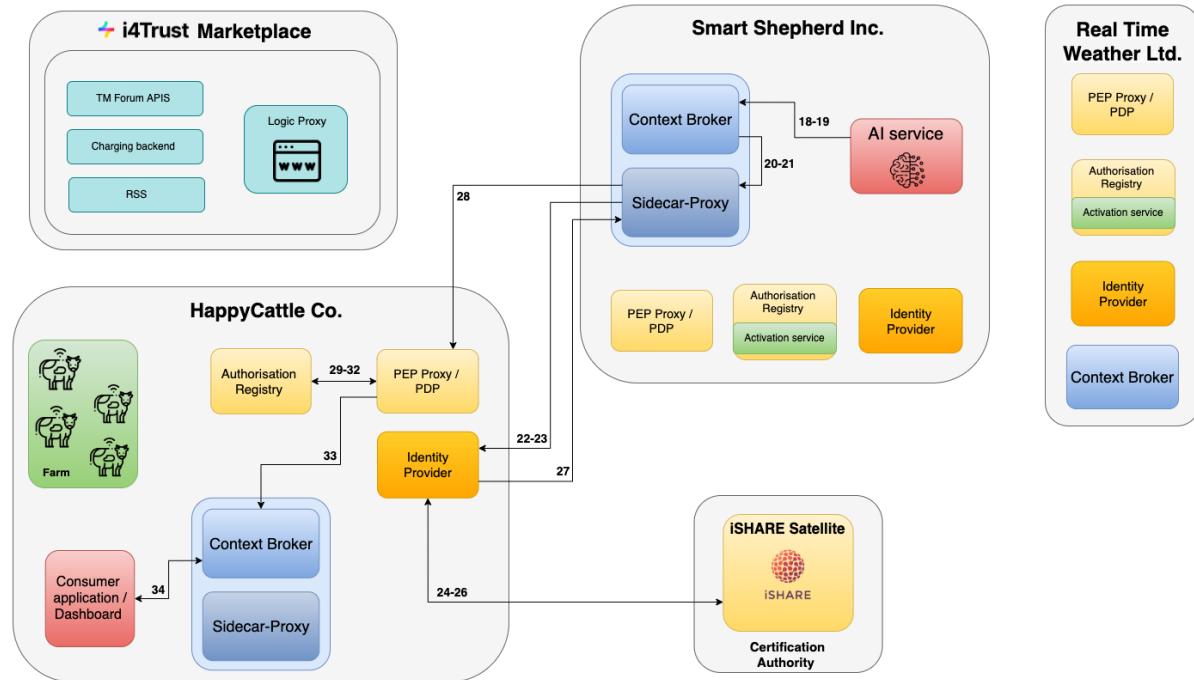


Figure 7.x: Architecture Diagram for "Send new prediction"

i4Trust PoC SmartShepherd - Smart Shepherd notifies about new animal activity prediction - detailed flow

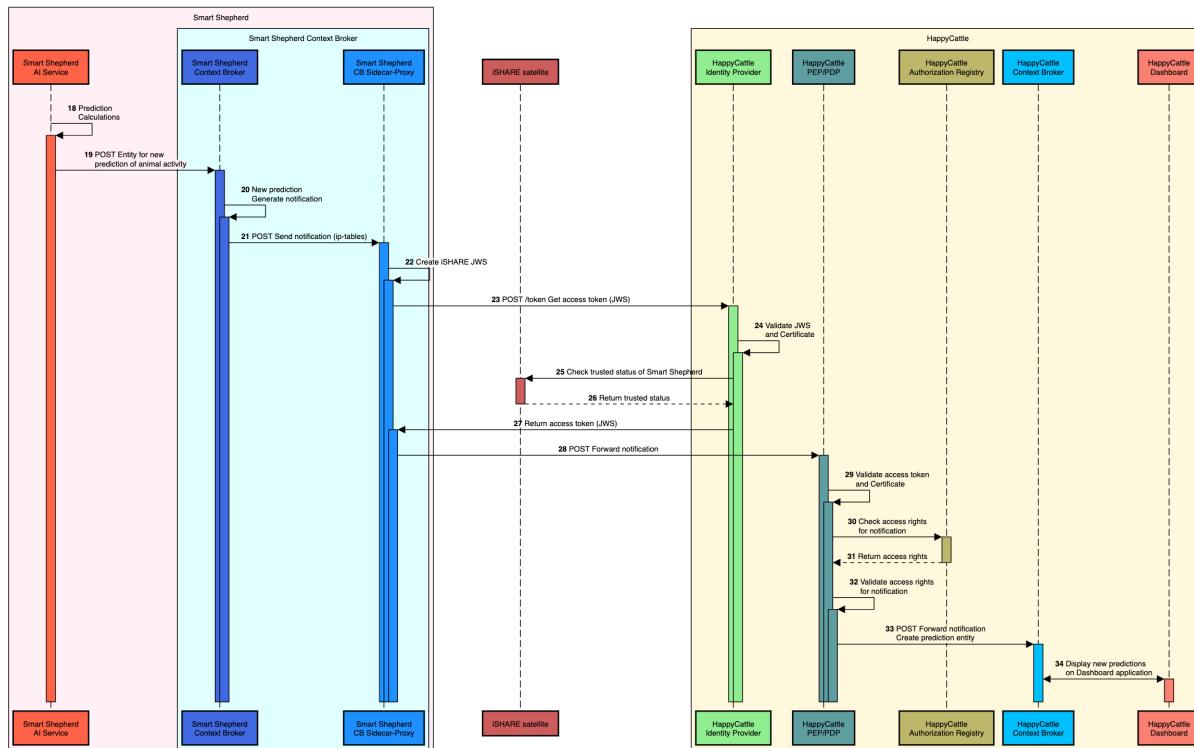


Figure 7.x: Sequence Diagram for "Send new prediction"

18. The AI Service, which previously received a new measurement of animal coordinates, finishes the calculation of a new prediction of animal activity.

19. The AI Service creates a new entity for the animal activity prediction at the Context Broker of Smart Shepherd

```
< Payload
{
    "id": "urn:ngsi-ld:AnimalActivity:001",
    "type": "Animal",
    "animalActivity": {
        "type": "Property",
        "value": "val1"
    }
}
```

20. (until 21.) The Context Broker of Smart Shepherd received a new entity for the prediction of animal activity. Since there is a subscription by HappyCattle for such new entities, the Context Broker will create a notification and send it to the PEP/PDP of HappyCattle.

```
> Content-Type: application/ld+json
POST https://happy cattle-kong.i4trust.fiware.io/notify/animal
< Payload
{
    "id": "urn:ngsi-ld:Notification:5fd0fa684eb81930c97005f3",
    "type": "Notification",
    "subscriptionId": "urn:ngsi-ld:Subscription:5fd0f69b4eb81930c97005db",
    "notifiedAt": "2021-12-09T16:25:12.193Z",
    "data": [
        {
            "id": "urn:ngsi-ld:AnimalActivity:001",
            "type": "Animal",
            "animalActivity": {
                "type": "Property",
                "value": "val1"
            }
        }
    ]
}
```

The Notification is first forwarded to the Sidecar-Proxy of Smart Shepherd via ip-tables, in order to retrieve the necessary access token before sending the request to the PEP/PDP of HappyCattle.

22. (until 23.) The Sidecar-Proxy of Smart Shepherd needs to retrieve an access token, before it can forward the Notification to the PEP/PDP of HappyCattle. First, the proxy generates an [iSHARE JWT](#) signed using the JSON Web Signature standard (JWS), as also described in step 4 of section 6.7.1.

```
> Headers
{
    "alg": "RS256",
    "typ": "JWT",
    "x5c": [
        "MIIEhjCC....Zy9w=="
```

```

        ]
    }

    > Payload
{
    "iss": "EU.EORI.NLSMARTSHEPHERD",
    "sub": "EU.EORI.NLSMARTSHEPHERD",
    "aud": [
        "EU.EORI.NLHAPPYCATTLE",
        "https://idp-happycattle.i4trust.fiware.io/token"
    ],
    "jti": "99ab5bca41bb45b78d242a46f0157b7d",
    "iat": "1540827435",
    "exp": 1540827465,
    "nbf": 1540827435
}

```

Then, the proxy sends a request to the HappyCattle Identity Provider [/token](#) endpoint. The signed JWT created before is provided with the `client_assertion` parameter of the request. Note the different `grant_type` parameter, since this is a M2M communication, as also described in section 6.9.3.

```

> Content-Type: application/x-www-form-urlencoded
POST https://happycattle-keyrock.i4trust.fiware.io/oauth2/token
grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&
scope=iSHARE&
client_id=EU.EORI.NLSMARTSHEPHERD&
client_assertion_type=urn:ietf:params:oauth:client-assertion-type:jwt-bearer&
client_assertion=eyJhbGc...9hwv

```

24. HappyCattle Identity Provider validates the JWT and certificate of Smart Shepherd.
25. Next, the Identity Provider of HappyCattle needs to verify the status of Smart Shepherd as a trusted iSHARE participant. For this, it requests this information at the iSHARE Satellite, as described in steps 9-12 of section 6.7.1.
26. iSHARE Satellite returns the information about Smart Shepherd and the Identity Provider of HappyCattle verifies its status.
27. Since Smart Shepherd is a trusted participant, the Identity Provider of HappyCattle returns the access token as an encoded and signed JWT. This JWT access token can now be used to send requests to the PEP/PDP endpoint of HappyCattle.

```

< Content-Type: application/json
< Cache-Control: no-store
< Pragma: no-cache

{
    "access_token": "aW2y...LI0w",
    "expires_in": 3600,

```

```
"token_type": "Bearer"  
}  
  
> Decoded payload of access token  
{  
    "iss": "EU.EORI.NLHAPPYCATTLE",  
    "sub": "EU.EORI.NLSMARTSHEPHERD",  
    "jti": "99ab5bca41bb45b78d242a46f0157b7d",  
    "iat": "1540827435",  
    "exp": 1540827465,  
    "nbf": 1540827435,  
    "aud": "EU.EORI.NLHAPPYCATTLE"  
}
```

28. The Sidecar-Proxy now forwards the initial Notification to the PEP/PDP of HappyCattle, where the previously obtained access token is put into the Authorization header of the POST request.

```
> Content-Type: application/ld+json  
> Authorization: Bearer aw2y...LI0w  
  
POST https://happycattle-kong.i4trust.fiware.io/notify/animal  
  
< Payload  
{  
    "id": "urn:ngsi-ld:Notification:5fd0fa684eb81930c97005f3",  
    "type": "Notification",  
    "subscriptionId": "urn:ngsi-ld:Subscription:5fd0f69b4eb81930c97005db",  
    "notifiedAt": "2021-12-09T16:25:12.193Z",  
    "data": [  
        {  
            "id": "urn:ngsi-ld:AnimalActivity:001",  
            "type": "Animal",  
            "animalActivity": {  
                "type": "Property",  
                "value": "val1"  
            }  
        }  
    ]  
}  
  
Decoded Bearer JWT payload  
{  
    "iss": "EU.EORI.NLHAPPYCATTLE",  
    "sub": "EU.EORI.NLSMARTSHEPHERD",  
    "jti": "99ab5bca41bb45b78d242a46f0157b7d",  
    "iat": "1540827435",  
    "exp": 1540827465,  
    "nbf": 1540827435,
```

```
        "aud": "EU.EORI.NLHAPPYCATTLE"  
    }
```

29. The PEP/PDP of HappyCattle validates the JWT and certificate which is part of the authorization header of the Notification request.
30. The PEP/PDP of HappyCattle now needs to check whether Smart Shepherd is allowed to send such Notification. Therefore it needs to check at its Authorization Registry, if the necessary policy for Smart Shepherd exists. These steps are also described in detail in steps 42-46 of section 6.7.1.  
The following presents the payload sent to the [/delegation](#) endpoint of the Authorization Registry of HappyCattle, in order to retrieve the delegation evidence policy issued from HappyCattle to Smart Shepherd.

```
POST 'https://happycattle-keyrock.i4trust.fiware.io/ar/delegation'  
  
{  
  "delegationRequest": {  
    "policyIssuer": "EU.EORI.NLHAPPYCATTLE",  
    "target": {  
      "accessSubject": "EU.EORI.NLSMARTSHEPHERD"  
    },  
    "policySets": [  
      {  
        "policies": [  
          {  
            "target": {  
              "resource": {  
                "type": "Animal",  
                "identifiers": [  
                  "urn:ngsi-ld:Subscription:5fd0f69b4eb81930c97005db"  
                ],  
                "attributes": [  
                  "animalActivity"  
                ]  
              },  
              "actions": [  
                "POST:Notification"  
              ],  
              "rules": [  
                {  
                  "effect": "Permit"  
                }  
              ]  
            }  
          ]  
        ]  
      }  
    ]  
  }  
}
```

```
        }
    ]
}
```

31. If there is such a policy, it is returned as **JWT delegation\_token**.

```
< Content-Type: application/json
> Payload
{
  "delegation_token": "eyJ4...0aaQ"
}

Decoded delegation_token JWT payload
{
  "iss": "EU.EORI.NLHAPPYCATTLE",
  "sub": "EU.EORI.NLHAPPYCATTLE",
  "jti": "d8a7fd7465754a4a9117ee28f5b7fb60",
  "iat": 1591966224,
  "exp": 1591966254,
  "aud": "EU.EORI.NLHAPPYCATTLE",
  "delegationEvidence": {
    "notBefore": 1541058939,
    "notOnOrAfter": 2147483647,
    "policyIssuer": "EU.EORI.NLHAPPYCATTLE",
    "target": {
      "accessSubject": "EU.EORI.NLSMARTSHEPHERD"
    },
    "policySets": [
      {
        "maxDelegationDepth": 1,
        "target": {
          "environment": {
            "licenses": [
              "ISHARE.0001"
            ]
          }
        },
        "policies": [
          {
            "target": {
              "resource": {
                "type": "Animal",
                "identifiers": [
                  "*"
                ],
                "attributes": [
                  "animalActivity"
                ]
              }
            }
          }
        ]
      }
    ]
  }
}
```

```
        },
        "actions": [
            "POST:Notification"
        ]
    },
    "rules": [
        {
            "effect": "Permit"
        }
    ]
}
]
```

32. The PEP/PDP checks if the policy is valid and whether it contains the necessary Permit rule.
33. If access is granted, the PEP/PDP forwards the notification to the Context Broker of HappyCattle.
34. The entity of the animal activity prediction was created at the Context Broker of HappyCattle. This can now be displayed on their Dashboard application.