Data Spaces Business Alliance

Unleashing the Data Economy

BDV BIG DATA VALUE ASSOCIATION

FIWARE FOUNDATION

gaia-x

INTERNATIONAL DATA SPACES ASSOCIATION

Technical Convergence

Discussion Document

Version 1.0.1

2022-09-26

Table of Contents

Copyright information

Publisher

Data Space Business Alliance - DSBA

https://data-spaces-business-alliance.eu/

Release Notes

The goal of the Data Space Business Alliance is to provide a common view on data spaces including a common technical framework based on the scope of work of the partners and the technical alignment achieved between its members: BDVA/DAIRO, FIWARE Foundation, Gaia-X, IDSA. This is a joint process that must include the acceptance of all members of the different organizations.

Each organization stays autonomous and can at all time take decisions which differ from this document or complement it.

Each organization stays committed to inform the rest of the DSBA members and work on future alignment to improve technical convergence.

Changelog

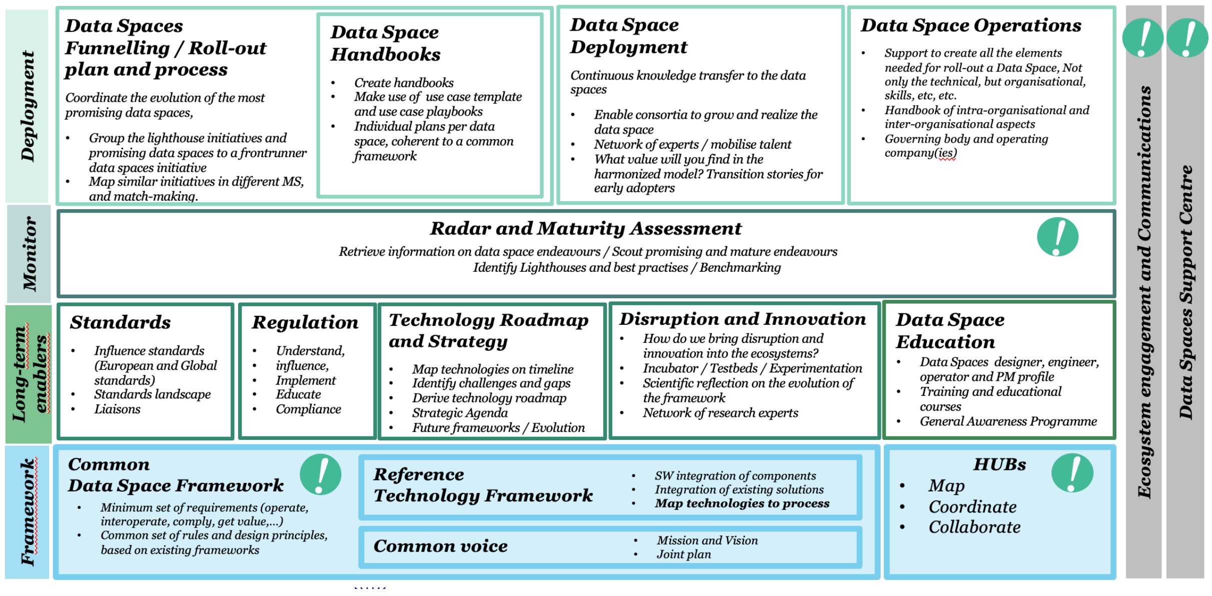
Version	Date	Description	Editor
1.0	2022-09-21	First public version of the Discussion Paper	See Authors below
1.0.1	2022-09-26	- Adding the Changelog. - Gaia-X is not meant to be a standardization organization (see section 6, 2nd paragraph). Removing the statement.	Pierre Gronlier Sebastan Steinbuss

1 Introduction

1.1 Background

Data spaces are viewed as key to achieving sovereign, interoperable and trustworthy data-sharing across businesses and societies – a key step to the Data Economy of the future. In September 2021, the Big Data Value Association (BDVA), FIWARE Foundation, Gaia-X and the International Data Spaces Association (IDSA) decided to join forces and formed the **Data Spaces Business Alliance (DSBA)** aimed at driving the adoption of data spaces across Europe and beyond.

Members of the DSBA came with a 100-days Implementation Plan that is described in the following picture.


Deployment
Monitor
Long-term enablers
Framework
Ecosystem engagement and Communications
Data Spaces Support Centre

Data Spaces Funnelling / Roll-out plan and process

Coordinate the evolution of the most promising data spaces,

- *Group the lighthouse initiatives and promising data spaces to a frontrunner data spaces initiative*
- *Map similar initiatives in different MS, and match-making.*

Data Space Handbooks

- *Create handbooks*
- *Make use of use case template and use case playbooks*
- *Individual plans per data space, coherent to a common framework*

Data Space Deployment

Continuous knowledge transfer to the data spaces

- *Enable consortia to grow and realize the data space*
- *Network of experts / mobilise talent*
- *What value will you find in the harmonized model? Transition stories for early adopters*

Data Space Operations

- *Support to create all the elements needed for roll-out a Data Space, Not only the technical, but organisational, skills, etc, etc.*
- *Handbook of intra-organisational and inter-organisational aspects*
- *Governing body and operating company(ies)*

Radar and Maturity Assessment

Retrieve information on data space endeavours / Scout promising and mature endeavours
Identify Lighthouses and best practises / Benchmarking

Standards

- *Influence standards (European and Global standards)*
- *Standards landscape*
- *Liaisons*

Regulation

- *Understand,*
- *influence,*
- *Implement*
- *Educate*
- *Compliance*

Technology Roadmap and Strategy

- *Map technologies on timeline*
- *Identify challenges and gaps*
- *Derive technology roadmap*
- *Strategic Agenda*
- *Future frameworks / Evolution*

Disruption and Innovation

- *How do we bring disruption and innovation into the ecosystems?*
- *Incubator / Testbeds / Experimentation*
- *Scientific reflection on the evolution of the framework*
- *Network of research experts*

Data Space Education

- *Data Spaces designer, engineer, operator and PM profile*
- *Training and educational courses*
- *General Awareness Programme*

Common Data Space Framework

- *Minimum set of requirements (operate, interoperate, comply, get value,...)*
- *Common set of rules and design principles, based on existing frameworks*

Reference Technology Framework

- *SW integration of components*
- *Integration of existing solutions*
- ***Map technologies to process***

Common voice

- *Mission and Vision*
- *Joint plan*

HUBs

- *Map*
- *Coordinate*
- *Collaborate*

As part of this plan, members of the DSBA agreed to work towards defining a common reference technology framework, based on the technical convergence of existing architectures and models, leveraging each other's efforts on infrastructure and implementations. The goal is to achieve interoperability and portability of solutions across data spaces, by harmonizing technology components and other elements.

1.2 Implementation-driven Plan

In order to materialize the desired technical convergence, an implementation-driven plan is proposed around evolution through subsequent versions of a Minimum Viable Framework (MVF) enabling creation of data spaces.

A first version of the MVF will be the result of a first workstream (**workstream 1**) targeted to provide a minimum set of building blocks required to cover the three major technology pillars for creation of data spaces:

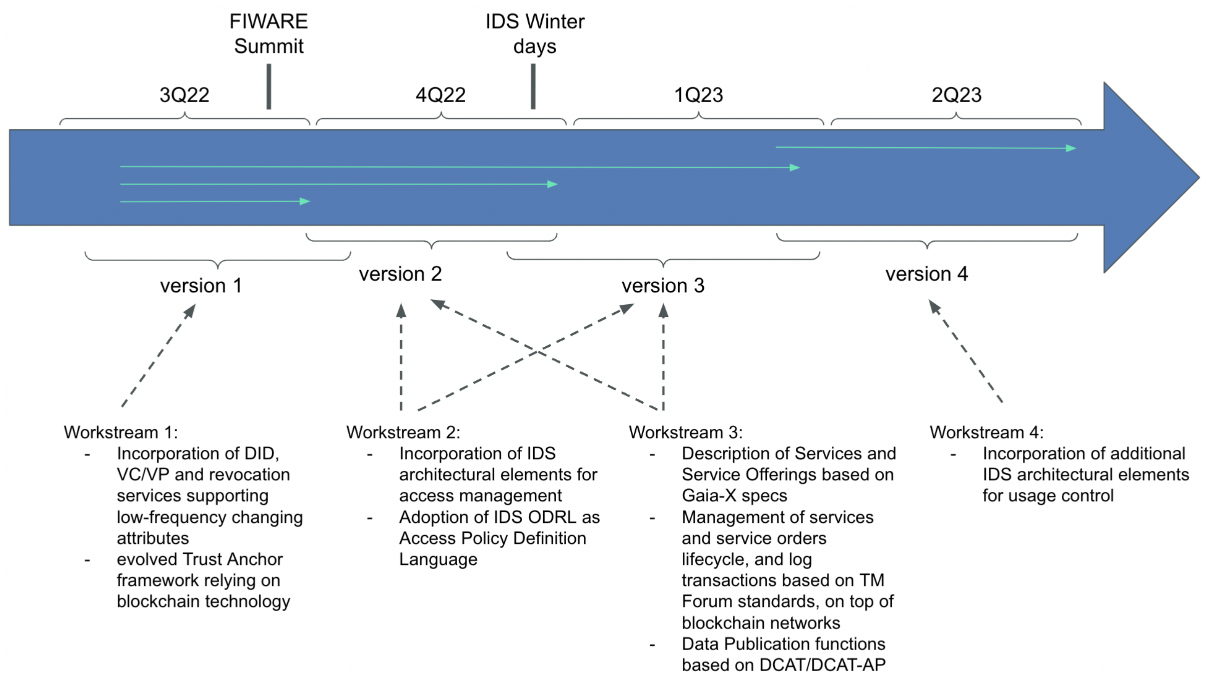
- **Data Interoperability**: NGSI-LD API and smart data models for actual data exchange will be adopted and extend the interoperability mechanisms of the IDS-RAM with a special focus on the IDS-Infomodel and the Vocabulary Hub.
- **Data Sovereignty and Trust**:
    - An eIDAS and EBSI -compatible Trust Anchor framework
    - A decentralized Identity and Access Management (IAM) framework based on:
        - A set of Verifiable Credential issuing protocols (Self-Issued OpenID Provider v2 (SIOPv2), via DIDComm channel, etc)
        - A set of verifiable presentation protocols (ex: OpenID Connect for Verifiable Presentations (OIDC4VP), Verifiable Presentation Request (https://w3c-ccg.github.io/vp-request-spec/), etc)
        - An ABAC (Attribute Based Access Control) framework comprising components implementing PEP, PDP, PAP/PMP, and PIP functions
- **Data value creation**: Centralized Service Catalogue and Marketplace functions based on TM Forum standards

Note this MVF will be just a starting point: a number of additional workstreams have been proposed addressing concrete topics that are relevant to achieve a technical convergence:

- **Workstream 2**: Incorporation of IDS Connector functions and support to ODRL for the definition of access/usage control policies
- **Workstream 3**: Shared Catalogue and Federated Marketplace services based on TM Forum standards and aligned with Gaia-X and IDS RAM specifications
- **Workstream 4**: Incorporation of additional IDS architectural elements for usage control

Once alignment within a given workstream is achieved, implementation of a new version of the MVF will be started. A detailed design using sequence diagrams describing the interactions between components of the technology framework will typically be produced at the beginning of each version, taking a concrete reference use case as the basis.

The following figure illustrates how the MVF will evolve through different versions, based on alignment achieved as a result of discussion within the different workstreams. Note that an Agile approach is taken, therefore it cannot be anticipated what version will correspond to the implementation of the results of which workstream. Versions will be initiated as soon as results from the discussion in a given workstream get consolidated and implementation can be initiated without disturbing developments of the version under way. Versions are expected to last between 3 and 6 months, otherwise they will be splitted. It can be anticipated that version 1 will be based on results of workstream 1, since a first draft of how DID and VC/VP can be incorporated has been proposed. Also that last version will be based on results of workstream 4, since technologies for supporting usage control are considered yet immature. However, version 2 and 3 will match results of workstreams 2 and 3 depending on the progress of alignment discussions within workstreams 2 and 3.


FIWARE Summit
IDS Winter days
3Q22
4Q22
1Q23
2Q23
version 1
version 2
version 3
version 4
Workstream 1:
- Incorporation of DID, VC/VP and revocation services supporting low-frequency changing attributes
- evolved Trust Anchor framework relying on blockchain technology
Workstream 2:
- Incorporation of IDS architectural elements for access management
- Adoption of IDS ODRL as Access Policy Definition Language
Workstream 3:
- Description of Services and Service Offerings based on Gaia-X specs
- Management of services and service orders lifecycle, and log transactions based on TM Forum standards, on top of blockchain networks
- Data Publication functions based on DCAT/DCAT-AP
Workstream 4:
- Incorporation of additional IDS architectural elements for usage control

2 Decentralized Trust Anchor and IAM Framework

2.1 Overview

Any Data Space requires a Trust Anchor Framework and associated Decentralized Identity and Access Management Framework to enable the trusted operation of the system without requiring a central entity intermediating in all interactions among participants. This is required to ensure trust in the information published on the data space by providers, as well as to enable customers to access the dataspace portal services, manage their profile and seamless login into federated marketplaces where they can benefit from a tailored experience.

The Trust Anchor Framework defines and enforces a set of rules that different organizations agree to follow to deliver one or more of their services. This includes legislation, standards, guidance, and other rules. By following them, all services and organizations using the Trust Anchor Framework can use their digital identities and attributes in a consistent and trusted manner. This makes it easier for organizations and users to complete interactions and transactions or share information with other participants.

We also present a Decentralized Identity and Access Management Framework based on Verifiable Credentials/Verifiable Presentations and leveraging the Trust Anchor Framework to provide an efficient, scalable, and Decentralized IAM that participants can use not only to interact with the data space and marketplaces, but they can also adopt for interactions between themselves and their product/service consumers.

The Trust Anchor Framework addresses the following issues:

- **ID Binding**: How to verify that a given identifier corresponds to a valid legal identity of an entity in the real world?

- **Proof of participation**: How to verify that the entity is trusted because it is a subscribed participant in a given ecosystem (e.g., to check the trust of the Shared Catalogue of Product Specifications and of Product Offerings)?
- **Proof of Issuing Authority**: How to check that the credentials presented by a participant have been issued by another entity that can be considered a Trusted Issuer of that type of credentials? This enables the verifier to put the right amount of trust in the facts attested by the Verifiable Credentials presented by a participant.

To enable transactional activity in the marketplace, the Decentralized Identity and Access Management Framework leverages on the above and provides an IAM system addressing additionally:

- **Identification**: How to verify that an identifier sent by a participant to another entity has been sent by the participant and not by an impostor that knows about the identifier? In addition, we need to cryptographically bind the identifier to the Verifiable Credentials sent by the participant so the facts attested in the credentials can be used for authentication and authorization.
- **Authorization**: How to use the attested facts in the Verifiable Credentials presented by a participant to perform advanced RBAC/ABAC access control and policy enforcement?

2.2 ID Binding

At the root of any trust framework there is the requirement to verify the identity of an entity in the real world and the assignment of some identifier that can be used later in representation of the real entity in the online processes. This association between an identifier (including some metadata) and the real identity of an entity is what we call **ID Binding**.

Please note that at this level, ID Binding states only who the entity is in the real world, not any additional properties that may be interesting for other purposes. For example, ID Binding establishes that the entity is a business incorporated in the EU, but it does not say what products it sells or the characteristics of the product, or the markets in which it operates, or in which data spaces it participates.

Many ecosystems assign a proprietary identifier to entities when they are onboarded in the ecosystem, creating silos of identifiers, and making very difficult the interoperability across ecosystems.

We propose to rely on identifiers already used in digital certificates issued by the Trust Service Providers (TSPs) authorized by the relevant European laws. The combination of digital certificates issued by TSPs, and Verifiable Credentials contributes to the legal validity and interoperability of the cross-border data-related transactions in the European Union facilitating the cross-border validation of eSignatures, eSeals, and more. Essentially, Verifiable Credentials and Presentations (including Product Specifications and Offerings) used in the ecosystem will be signed using digital certificates.

Some of the characteristics and advantages of using this type of ID Binding are described below.

2.2.1 Cross-border use of mutually recognised electronic identification means

We propose that during onboarding of a new member, the Data Space and its participants accept a digital certificate or seal if it is issued by any European TSP.

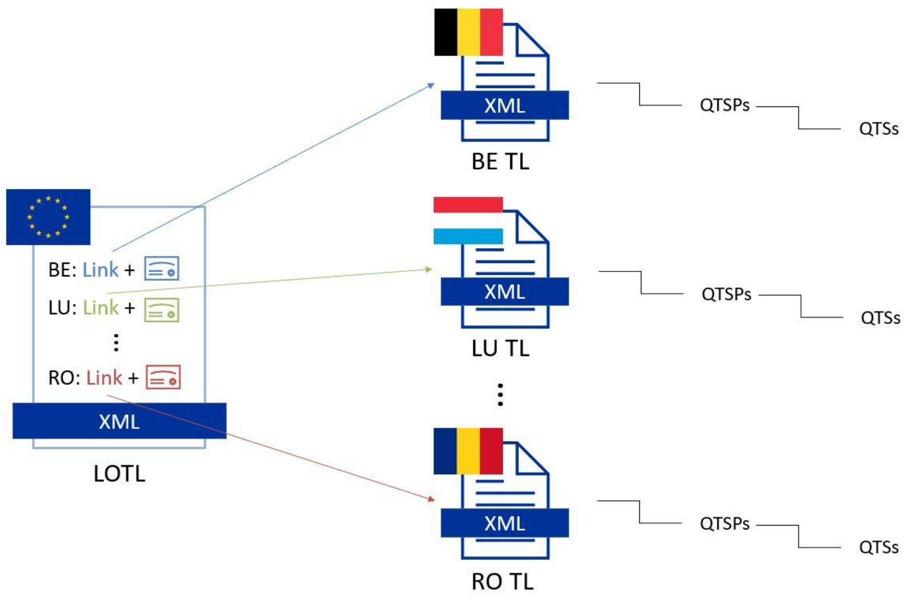
The Regulation on electronic identification and trust services for electronic transactions in the internal market (**eIDAS Regulation**) states that in order to contribute to their general cross-border use, it should be possible to use trust services as evidence in legal proceedings in all Member States. DOME is fully aligned with the objectives of the eIDAS regulation, specifically Article 17 of the eIDAS Regulation, says that Member States should **encourage the private sector** to voluntarily use electronic identification means under a notified scheme for identification purposes when needed for online services or electronic transactions. The possibility to use such electronic identification **enables the private sector to rely on electronic identification and authentication already largely used in many Member States at least for public services and to make it easier for businesses and citizens to access their online services across borders**.

In this way, interoperability of Verifiable Credentials across the public and the private sector can be achieved in large Digital Ecosystems (e.g., Data Spaces) across the EU.

Article 22 of the Regulation on electronic identification and trust services for electronic transactions in the internal market (**eIDAS Regulation**), obliges Member States to **establish, maintain and publish trusted lists**. These lists should include information related to the qualified trust service providers for which they are responsible, and information related to the qualified trust services provided by them.

In order to contribute to their general **cross-border** use, it should be **possible to use trust services as evidence in legal proceedings in all Member States.**

In practice, each Member State publishes its Trusted List, and the Commission publishes the List Of Trusted Lists (**LOTL**). There are different ways to access the lists, but one which is machine-processable (XML) is located at https://ec.europa.eu/tools/lotl/eu-lotl.xml which contains the addresses of each of the Trusted Lists published by each Member State. This is represented in the following figure:



We propose to use an integrated, easy and performant way to perform ID Binding based on the digital certificates provided by the EU TSPs, including transparent access to the consolidated list of TSPs when applications in the Data Space must perform ID Binding (typically during onboarding and verification of signatures of credentials).

2.2.2 ID Binding and the Verifiable Credential

An additional problem related to ID Binding when a participant sends online a Verifiable Credential to another participant is to make sure that the credential has been sent by an entity authorized to do so and not by an impostor (either because the sender is the subject of the credential or because it has been authorized by a proper mechanism recognised in the ecosystem). This is called ID Binding of the credential. Later in the document we describe and implement a simple approach to perform this using widely used public key cryptography and leveraging on the trust already provided by the digital certificates used in the ID Binding described above.

The detailed description can be found later in this document, but the essential characteristics are below.

We assume that the issuer of the Credential has an eIDAS certificate (we will use this term for a digital certificate or seal issued by a TSP in the EU Trusted List), and that the issuer is a participant in the Data Space (the concept of participation is elaborated in the next section).

The subject of the credential (either a natural or juridical person) wants to receive a credential from the issuer. The credential is an attestation of something that the issuer knows about the subject. That means that **there should be a previous close relationship between the issuer and the subject** and there is a pre-existing trusted identification mechanism that the issuer uses for everything related to the subject. It could be physical (e.g., a student going to the secretary to request the credential) or electronic (e.g., the student using the "traditional" identification mechanisms for accessing online the University services).

In other words, it is not possible that an issuer provides a credential to a subject that it does not know. Or better said, such a credential would be useless because the facts attested inside the credential cannot be really trusted.

In this context, the approach is the following:

1. The subject authenticates to the issuer with whatever mechanism has been used during the previous relationship. In the example of the Diploma, the student uses whatever identification mechanism was provided by the University when the student enrolled in the studies.
2. Once an authenticated session exists, the subject generates a pair of public-private keys and sends the public key to the issuer, keeping the private key private (that's the meaning of its name, obviously). The actual mechanism uses a digital signature of a challenge to ensure that the subject controls the private key associated to the public key sent to the issuer.
3. The issuer creates a credential with the relevant attestations and includes also the public key received from the subject as an additional attestation.
4. The issuer digitally signs the credential with its eIDAS certificate and makes the credential available to the subject. The specific mechanism to "send" the credential to the subject can be diverse. Given that there is a previous relationship, the credential could be sent inside the authenticated session when it is generated or made available in the "electronic office space" of the subject, or sent encrypted via email, or even printed in physical media and sent via mail, if required. The Verifiable Credential is essentially a file with data in JSON (or JSON-LD) format and digitally signed by the issuer. It can be sent from the issuer to the subject with whatever secure transmission mechanism that the issuer and subject have been using in the past, or if they want, with a new mechanism as will be described in this document.

With this mechanism, the receiver of the credential can have the same level of trust in the "normal" attested attributes inside the credential and in the public key inside the credential.

For example, if the receiver of a Diploma has a given level of trust in that a certain University

2.2.3 About identifiers for legal persons

We use W3C Verifiable Credentials with DIDs as identifiers. A DID is a simple text string consisting of three parts: **1)** the did URI scheme identifier which is the word "*did*", **2)** the identifier for the DID method which specifies the mechanism used for resolving a DID, and **3)** the identifier specific to that DID method.

As mentioned before, when using eIDAS digital certificates for identity binding, it does not make sense to "invent" identifiers or to promote the usage of different DID methods that are not well integrated with eIDAS certificates and that generate identifiers which are not in general legally recognised in the EU for economic transactions (e.g., that can be used in electronic invoices across the EU).

In general, an ecosystem may accept one or more DID methods and their associated DID resolution mechanisms (e.g., did:web, did:peer, etc.). For Legal Persons, we propose that one of the DID methods used is the following, using as identifiers the same identifiers that are already embedded in the eIDAS certificates that conform to the relevant ETSI standards.

ETSI EN 319 412-3 V1.2.1 (2020-07) "**Electronic Signatures and Infrastructures (ESI); Certificate Profiles; Part 3: Certificate profile for certificates issued to legal persons**" states:

> *"The subject field shall include at least the following attributes as specified in Recommendation ITU-T X.520:*
>
> - *countryName*
> - *organizationName*
> - ***organizationIdentifier** and*
> - *commonName"*

And regarding the *organizationIdentifier* attribute it says:

> *The **organizationIdentifier** attribute shall contain an identification of the subject organization different from the organization name. Certificates may include one or more semantics identifiers as specified in clause 5 of ETSI EN 319 412-1 [i.4].*

And the document referenced, ETSI EN 319 412-1 V1.4.2 (2020-07) "**Electronic Signatures and Infrastructures (ESI); Certificate Profiles; Part 1: Overview and common data structures**" states:

> *When the legal person semantics identifier is included, any present **organizationIdentifier** attribute in the subject field shall contain information using the following structure in the presented order:*
>
> - *3 character legal person identity type reference*
> - *2 character ISO 3166 [2] country code*
> - *hyphen-minus "-" (0×2D (ASCII), U+002D (UTF-8)) and*
> - *identifier (according to country and identity type reference)*
>
> *The three initial characters shall have one of the following defined values:*
>
> 1) *"VAT" for identification based on a national value added tax identification number.*
> 2) *"NTR" for identification based on an identifier from a national trade register.*

3) "PSD" for identification based on national authorization number of a payment service provider under Payments Services Directive (EU) 2015/2366 [i.13]. This shall use the extended structure as defined in ETSI TS 119 495 [3], clause 5.2.1.

4) "LEI" for a global Legal Entity Identifier as specified in ISO 17442 [4]. The 2 character ISO 3166 [2] country code shall be set to 'XG'.

5) Two characters according to local definition within the specified country and name registration authority, identifying a national scheme that is considered appropriate for national and European level, followed by the character ":" (colon).

Other initial character sequences are reserved for future amendments of the present document. In case "VAT" legal person identity type reference is used in combination with the "EU" transnational country code, the identifier value should comply with Council Directive 2006/112/EC [i.12], article 215.

That means that any eIDAS digital certificate issued by TSPs to legal persons compliant with the ETSI standards including an organizationIdentifier attribute can be used to trivially derive a DID from the ETSI standard identifier by applying the following rule:

did:elsi:organizationIdentifier

Examples:

- **International Data Spaces**: *did:elsi:VATDE-325984196*
- **Gaia-X**: *did:elsi:VATBE-0762747721*
- **FIWARE Foundation**: *did:elsi:VATDE-309937516*
- **TNO**: *did:elsi:LEIXG-724500AZSGBRY55MNS59*

Where:

- "did" is the W3C did uri scheme.
- "elsi" stands for ***E****TSI* ***L****egal* ***S****emantic* ***I****dentifier*, which is the acronym for the name for this type of identifier used in the ETSI documents.
- "organizationIdentifier" is the exact identifier specified in the ETSI standard, and that can evolve with the standard to support any future requirement.

In this way, there is a **bidirectional mechanism to derive DIDs from the eIDAS digital certificate** and inversely.

Proving the control of an ELSI DID, as required by W3C Verifiable Credentials implementations is trivial using the associated digital certificate: including the certificate with any signature can do that. By the way, this means that any existing digital signature of any type of document (not only Verifiable Credentials) is already compliant with this DID method specification, just by making a trivial translation.

In other words: any legal person can have a standard eIDAS certificate with an automatically associated DID identifier complying with the ELSI did method specification. There is no need to invent new identifiers or have a central entity in a Data Space assign identifiers to participants.

2.2.4 About identifiers for natural persons

In principle, we could use the same approach for natural persons as for legal persons. The ETSI standards referenced above also cover natural persons and they define a "Natural Person Semantic Identifier".

However, legal persons are completely different to natural persons, especially from the point of view of privacy (look at the GDPR to see some differences). It is for those privacy reasons that we propose a different approach to the identifiers of natural persons participating in a sharing ecosystem like a Data Space.

The problems and solutions are discussed in detail later in this document, after legal persons are covered sufficiently.

2.2.5 About identifiers for connectors, gateways and application context

In addition to identifiers for legal and natural persons, identifiers are required for IDS-connectors or more in general gateways respectively in the application context. Such software components require identification on a similar basis. From an organizational perspective the application context must be linked with legal and/or natural persons identifiers to clarify the delegation of power to the application context. Such identifiers could be realized as X.509 certificate or as DID. The current version of the IDS-RAM describes the use of X.509 certificates. The use of DIDs should be described in the IDS-RAM based on the results of this document.

A valid identifier should contain at least:

- Issuer distinguished name
- Subject distinguished name
- Serial number
- Version information
- Serial number
- Validity information

2.3 Proof of participation

When verifying a Verifiable Credential/Presentation, we must address the following:

1. How do we know that the **issuer** of the Verifiable Credential is a **participant** in the concrete ecosystem (e.g., a given Data Space) where we are also participants?
2. How do we know that the **subject** of the Verifiable Credential is a **participant** in the concrete ecosystem (e.g., a given Data Space) where we are also participants?

We propose to use a **Trusted Participant List** including the identities and associated metadata of all legal persons participating in the concrete ecosystem. The Trusted Participant List is updated during the onboarding process of an entity and is managed by one or more collaborating trusted entities. Please note that this list is different from the EU Trusted List with the identities of TSPs authorized to issue digital certificates/seals in the EU.

There are different ways to implement the Trusted Participant List but in any case, the users of the Trusted Participant List should not be aware of the technology used to implement it. The users of the Trusted Participant List just use an API to query the list on verification, and the maintainers use a different API to register and update the list.

This way, it is completely possible to use a mix of centralized and decentralized technology without the users noticing it. Or to migrate transparently from one technology to another depending on the requirements of the specific ecosystem.

Having said that, we propose that one of the implementations uses a federated set of interoperable blockchain networks for the maintenance of the Trusted Participant List, providing a decentralized, hyper-replicated, efficient and resilient mechanism for querying the list. Anyone can create a replica of the information using centralized systems if they wish.

We propose to base the API in the one defined by EBSI for Trusted Lists of different types. For example:

GET /participants and ***GET /participants/{did}*** to get the list of participants or to check a given participant if we have its DID, respectively.

There are several other APIs to get attributes/metadata associated to the participants, and APIs to maintain the list, used by the entity or entities responsible for the list. The full specification is later in this document.

We propose to follow this principle:

> *If something we need is already in EBSI, just use it. Otherwise define it trying to be as consistent as possible with EBSI, unless there is no chance to do so.*

2.4 Proof of Issuing Authority

Given that anyone can have access to the technology needed to create Verifiable Credentials and anybody can issue credentials and digitally sign them with their eIDAS digital certificate, the problem is how a verifier knows that the Verifiable Credentials received from the subject have been issued by an entity which is entitled or authorized to issue that type of credential.

The primary mechanism to solve this problem is the use of **Trusted Issuer Lists** (there may be several lists, one per domain or type of credential). A **Trusted Issuers List** is a register of trusted public entities which can issue Verifiable Credentials belonging to a given domain or of a given type. It is assumed that an entity must be first in the Trusted Participant List before it appears in the Trusted Issuers List. This list includes the identifiers, public keys for verification of signatures and their accreditations in the form of Verifiable Credentials/Presentations from third parties, enabling the entity to issue credentials of a given type. All information in the registry is validated and signed by trusted legal entities of the corresponding domain (Conformity Assessment Bodies and third-party auditors).

Using Trusted Issuers Lists (there may be several lists, one per domain or type of credential) is the simplest mechanism. However, in very complex ecosystems with many entities issuing credentials of different types, the management of Trusted Issuer Lists can be difficult to scale. For those ecosystems we can use the combination of Trusted Issuer Lists with the "*chaining*" of Verifiable Credentials, like the certificate chaining used with traditional X.509 digital certificates:

- At the root of the trust hierarchy there is a set of Trusted Issuers Lists as described above, containing the primary trusted entities in the ecosystem.
- The entities in those Trusted Issuers Lists can issue special Verifiable Credentials to other entities, authorising them to be also **Trusted Issuers, even if they are not included in a Trusted Issuer List**. The signature of the special Verifiable Credential attests that the subject of the credential is explicitly authorized by the signer to issue a given type of credentials (usually a subtype of the parent type, but not necessarily; the specific rules have to be defined in the corresponding governance model for the domain/ecosystem). This mechanism can be also used by those Trusted Issuers not in Trusted Lists if we need several levels in the hierarchy, though usually two or three layers (including the root Trusted Issuers List) should be enough to handle large ecosystems.

There is a trade-off in choosing one or another mechanism. Having all Trusted Issuers in one or more Trusted Issuers Lists makes verification very simple: the verifier of a credential just checks once in the Trusted Issuers List corresponding to the type of certificate. Checking a credential using the chained mechanism is more involved: the verifier has to check the chain of signatures for the relevant Verifiable Credentials until it reaches an issuer which is in a Trusted Issuers List for the domain (or if no issuer is in any Trusted Issuers List, then the Verifiable Credential should be rejected).

The mechanisms can be combined and are not exclusive or all-or-nothing in an ecosystem. Depending on the requirements/complexity of a domain in an ecosystem, one domain can use just Trusted Issuers List while another domain can use a chained mechanism. It is even possible to start with only a Trusted Issuers List and transition seamlessly to the chained mechanism if the domain complexity grows beyond some limit, decided by the governance rules of the domain.

2.5 Identification and Authorisation

We propose to use OpenID Connect for Verifiable Presentations (OIDC4VP) and Self-Issued OpenID Provider v2 (SIOPv2), which leverages the proven, robust and secure standards of OpenID Connect protocols to:

- Transport Verifiable Credentials/Presentations in the flows of OpenID Connect, so Relying Parties can use well known mechanisms to issue and receive Verifiable Credentials.
- Enable all participants (via SIOPv2) to send identity data and Verifiable Credentials to other participants without the requirement for big and centralized Identity Providers as it is unfortunately common in implementations of standard OpenID Connect.

This way we implement a distributed, fault-tolerant, trustful and efficient IAM system avoiding the existence of centralized Identity Providers (IdPs). Using widely implemented standards like OIDC and W3C Verifiable Credentials provides a very low barrier of entry to participants implementing IAM.

Using OIDC for transporting Verifiable Credentials enables integration of the attested data inside the credential for sophisticated and flexible **Authorization** schemes. Participants implementing this **Decentralized Identity and Access Management Framework** can use credential data for advanced RBAC/ABAC access control and policy enforcement.

Furthermore, the IAM Framework can be used by participants not just to interact with the data space and marketplaces but they can adopt it and use it for peer-to-peer interactions between participants in the ecosystem without the involvement of central entities (except for initial onboarding and certification processes).

3 Shared Catalogue and Marketplace Services

3.1 Overview

Data spaces should provide support for the creation of multi-sided markets where participants can generate value out of sharing data. This requires the adoption of common mechanisms enabling the description of services for accessing data or linked to applications processing data, the description of offerings associated with those services, the publication and discovery of both services and service 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 use a service, according to certain service offering.

The proposed approach will take the form of a Decentralized Open Marketplace Ecosystem (DOME) based on the federation of marketplaces, all of them connected to a commonly shared digital catalogue of cloud and edge services and service offering descriptions.

Cloud and edge services can be further classified as:

- data services, providing access to data
- application (app) services, which gather and process data, and typically deliver data results
- cloud or edge infrastructure services, supporting the deployment and execution of data/app services

Cloud and edge infrastructure service providers, in turn, can be classified as cloud/edge IaaS providers or cloud/edge Platform service providers (in this latter case, providing a platform targeted to solve either the integration of several data/app services linked to a given application domain, like smart cities or smart farming, or the integration of certain type of data/app services, e.g., AI services)

Each of the federated marketplaces in the referred DOME will be a marketplace provided by an independent marketplace provider or a marketplace connected to the offering of a given cloud / edge infrastructure service provider (IaaS or platform provider). Besides these marketplaces, A DOME global portal would implement functions through which cloud/edge service providers may register their product offerings and end customers can discover offered products.

DOME will rely on the adoption of common open standards for the description of cloud and edge services and service offerings as well as their access through a shared catalogue.

Following subsections elaborate on the roles that organizations can play with respect to DOME as well as some details of the technical architecture. Further clarification might be required, please visit the outlook section for this.

3.2 Roles of organizations in the ecosystem

Six different roles can be played by organizations involved in the ecosystem linked to DOME as illustrated in figure 1: cloud and edge service providers, marketplace providers, customers, the operators of the DOME technical infrastructure, third parties capable of integrating and offering their services complementing those implemented in the DOME technical infrastructure, and members of governance and supervisory bodies.

The following subsections will introduce the mentioned roles.

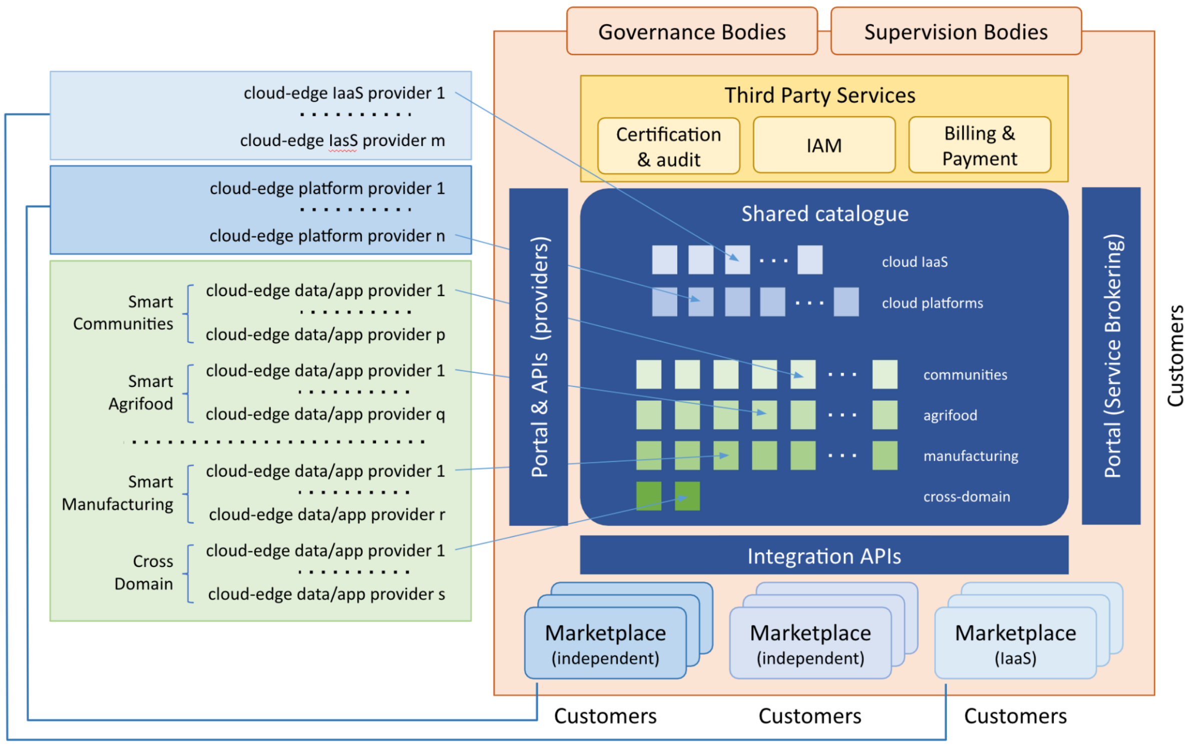

Governance Bodies
Supervision Bodies
Third Party Services
Certification & audit
IAM
Billing & Payment
Shared catalogue
cloud IaaS
cloud platforms
communities
agrifood
manufacturing
cross-domain
Portal & APIs (providers)
Portal (Service Brokering)
Customers
Integration APIs
Marketplace (independent)
Marketplace (independent)
Marketplace (IaaS)
Customers
Customers
Customers
cloud-edge IaaS provider 1
cloud-edge IasS provider m
cloud-edge platform provider 1
cloud-edge platform provider n
Smart Communities
cloud-edge data/app provider 1
cloud-edge data/app provider p
Smart Agrifood
cloud-edge data/app provider 1
cloud-edge data/app provider q
Smart Manufacturing
cloud-edge data/app provider 1
cloud-edge data/app provider r
Cross Domain
cloud-edge data/app provider 1
cloud-edge data/app provider s

Figure 1: High-level vision of DOME architecture, operating model and roles

3.2.1 Cloud and edge service providers

Cloud and edge service providers (IaaS, platform and app/data service providers) are organizations (public institutions or private companies) that offer service products that can be consumed by customers, such as other organizations or individuals. They access a DOME global portal where they can register and manage the description of specifications and offerings linked to their products. Product Specifications and Product Offerings associated with a given service from a service provider are stored in the Shared Service Catalogue that is the central part of DOME. A Product comprises a number Services and supporting Resources (e.g., an Air Quality monitoring product for a giving City may consist in an application offered as a Service from the Cloud and a number of computing resources on the Cloud plus a number of IoT devices for monitoring air quality deployed in the field).

The description of each Product Specification and Product Offering will be provided in a standard-based format prescribed in DOME. The description of a Product Specification will comprise information like the unique identifier of the product, its name, version, associated documentation, description of software services implementing the product functionality, description of resources required for execution of such software services (e.g., computing capacity, including disk storage, to be provisioned for serving each customer, or devices to be deployed on the edge), status within the lifecycle of the product (under testing, validated, active, obsolete, retired, ...), etc. On the other hand, the description of a Product Offering will comprise an unique identifier, a reference to the specification of the product being offered, lifecycle status, terms and conditions associated to its use, pricing model, associated agreements (e.g., list of Service Level Agreements that users can choose from), target market segment, kind of marketplaces through which the product can be offered, etc. Both, Product Specifications and Product Offering descriptions, will comprise a number of labels issued by certification agencies in connection with the service offered that certifies compliance with defined EU regulations or rules established by supervision authorities (e.g., GDPR regulations, established regulations for specific sectors like health, energy, finance, regulations for cloud services to be established in the EU Cloud Rulebook, ...), relevant standards (e.g., standards for interoperability) or best practices (e.g., Open Source Security Foundation Best Practices).

A Cloud and edge Data/App service provider could receive Product Orders from end customers directly or from those marketplaces federated in DOME which have incorporated the given Data/App service as part of their catalogue (see description of the role of Federated Marketplaces below). Similarly, a given Data/App service provider may receive payments through third-party payment service providers that have integrated their services directly with DOME and it has decided to rely on for direct processing of orders, but they can receive such payments also from the payment services implemented by the federated marketplaces through which service orders for the Data/App service were issued.

Through specific pages for providers of the DOME global portal, cloud and edge IaaS, Platform and data/app service providers can also monitor the evolution of their contracts for particular end users and generate different kinds of reports. In order to be able to access these specific pages for providers under the DOME global portal, each cloud and edge IaaS, Platform and data/app service provider has to be registered in the eIDAS service.

3.2.2 Federated Marketplaces

As illustrated in figure 1, different kind of marketplaces can be federated to DOME:

- Marketplace connected to an IaaS provider, which comprises a catalogue of cloud and edge data/app services which customers can pick and then easily deploy on top of the computing infrastructure supported by the given IaaS provider
- Marketplace connected to a Platform provider which comprises a catalogue of cloud and edge data/app services which customers can pick and easily activate, integrated with the rest of applications on top of the provided Platform.
- Independent Marketplace, which comprises a catalogue of cloud and edge data/app services which are not tied to any particular IaaS or Platform provider

Examples of Marketplaces connected to Platform providers would be marketplaces connected to specific application domains, like Smart Cities or a Smart Farming, or marketplaces connected to specific technology frameworks, like a Spark-based platform for development of AI apps, or a Grafana-based platform for development of dashboard apps. In the case of a marketplace connected to a specific Smart City platform, the catalogue may comprise apps for Smart Parking, Smart Air Monitoring or Smart Waste Management, for example. Note that each data/app service may be hosted on a different IaaS cloud or servers and it does not need to be the same where the Platform is hosted. In the case of a Smart Farming Platform, the catalogue may comprise apps for Smart Field Watering, Smart Pesticide Spreading or Smart Silo. Similarly, a marketplace connected to a Spark-based platform may comprise applications for predictive maintenance of vehicles, or Weather predictions. Some of the data/app services can be provided by the Platform provider (e.g., Integrated Command and Control system in connection with Smart City Platforms, or Smart Farm Management system in connection with Smart Farming). Some of them may be already active by default for all customers, otherwise may require acquisition through the marketplace.

Note that a given cloud/edge service may be visible in multiple marketplaces. On the other hand, a given marketplace may only comprise a subset of the cloud and edge services listed in the DOME shared catalogue (e.g., the Marketplace connected to a concrete Smart City platform based on FIWARE will only include data/app services relevant for cities that implement the NGSI-LD standard for integration).

Also note that cloud and edge data/app services will always be visible and could be directly procured once discovered through the DOME global portal. However, federated Marketplaces will typically bring a personalized user experience to their target customers, and also a different implementation of their own rating, billing and payment processes, even though they may rely on payment and billing services offered by third parties through DOME.

3.2.3 Customers

European public and private customers looking for trusted cloud and edge services will interact with DOME following one of the two following paths:

- In a very first step, accessing the DOME global portal and leveraging service brokering functions of the DOME technical infrastructure to discover IaaS or platform providers which, together with their associated marketplaces can bring to them the best personalized experience. Afterwards, interacting directly through the marketplace associated with the IaaS and platform of their choice, picking the concrete cloud and edge data/app services offerings that are published through the marketplace catalogue which therefore can be seamlessly integrated with their selected IaaS/Platform to support processes of their organization.
- Accessing the DOME global portal to find cloud and edge services directly, placing and managing orders of selected services via mechanisms those

services expose through the DOME infrastructure, and conducting payments via payment systems which are supported directly by the service provider or are offered by third parties integrated with the DOME infrastructure that have been accepted by the selected service provider.

While the second path will be feasible, it is envisaged that the first path will be more optimal, since the consumer will benefit from a more rich and comprehensive service that IaaS/Platform providers can offer.

3.2.4 Operators of the DOME technical infrastructure

For a given data space or federated set of data spaces, a number of companies will act as operators of the DOME technical infrastructure, ensuring the proper functioning of DOME, including security aspects.

3.2.5 Third parties integrating/offering complementary services

DOME will provide means for integration of Third-party services, like for example:

- Services from certification and audit agencies which will help to validate the reliability, security, and sovereignty of certain cloud services by checking/verifying their compliance with predetermined market-wide certifications.
- IAM service providers offering services aligned with open standards for IAM adopted in DOME, bringing participants the ability to securely manage identities and access to specific cloud and edge data/app services.
- Billing and Payment service providers working as gateways that rely on transaction logs registered in the federated blockchain network infrastructure underlying DOME to provide secure, transparent and trustful billing to consumers and payment to providers.

For all these three kinds of third-party services, or additional ones, DOME represents a new source of revenue, as it gives them access to a new market (the cloud and edge service providers and the customers). On the other hand, they may represent potential sources of revenue for securing the sustainability of DOME.

3.2.6 Members of governance and supervisory bodies

Last but not least, DOME will define suitable governance and supervisory bodies that will oversee development of the ecosystem around DOME ensuring fulfillment of its objectives. These bodies will typically incorporate operators of the DOME infrastructure, representatives of the organizations using DOME, and other relevant stakeholders including, when relevant, representatives of public bodies.

3.3 Basic Information Model

DOME relies on a subset of TM Forum Open API recommendations with regards to the definition of its underlying information model as well as the implementation of APIs that support marketplace federation and the generation of logs during the lifecycle of cloud and edge services and service offerings.

DOME supports the concept of Product (Offering) Catalog which is a collection of Product Offerings published by Providers through a set of specific Distribution Channels (e.g., federated marketplaces or DOME itself) and targeted to Market Segments[1]. Here, we use the term Product to align with the terminology used in TM Forum recommendations which refers to the particular instantiation for a Customer of a set of related Services (e.g., a set of software services offered through a RESTful API on a particular endpoint) and required Resources (e.g., the concrete devices that had to be deployed on the consumer premises or storage and VM capacity that needed to be allocated in the cloud to support execution of software services associated to the Product). A Product complies with a given Product Specification which in turn refers to the Service Specifications and Resource Specifications that Services and Resources realising the Product have to comply with.

Product instances as well as the associated services and resources instances are registered in the Product, Service and Resource Inventory when they are instantiated.

When a Customer wishes to get a Product instance materialized, it has to issue a Product Order. Only when that Product Order has been successfully completed, a Product instance gets materialized for the Customer. That Product instance is bound to a contract between the Customer and the Provider of the associated Product Offering.

A Product Offering comprises elements such as a reference to the corresponding Product Specification, the agreements that govern usage of the served product, a productOfferingPrice, the marketSegment it is targeted to, channels through which it can be offered, and other aspects which characterize the products created when the Product Offering is procured. Note that there may be one or more Product Offerings around the same Product Specification (e.g., associated with different prices or targeted to different market segments).

Each time a Resource or Service associated with a Product is used, a Usage Log is created, which typically is used to calculate how much can be charged to Customers and paid to Providers.

[1] Note that in TM Forum recommendations the term "Product" is used instead of "Service". This is because marketplaces relying on the specifications can be defined for different kind of products, including cloud and edge services but also any kind of physical or digital products/assets.

The described information model is represented in the following figure:

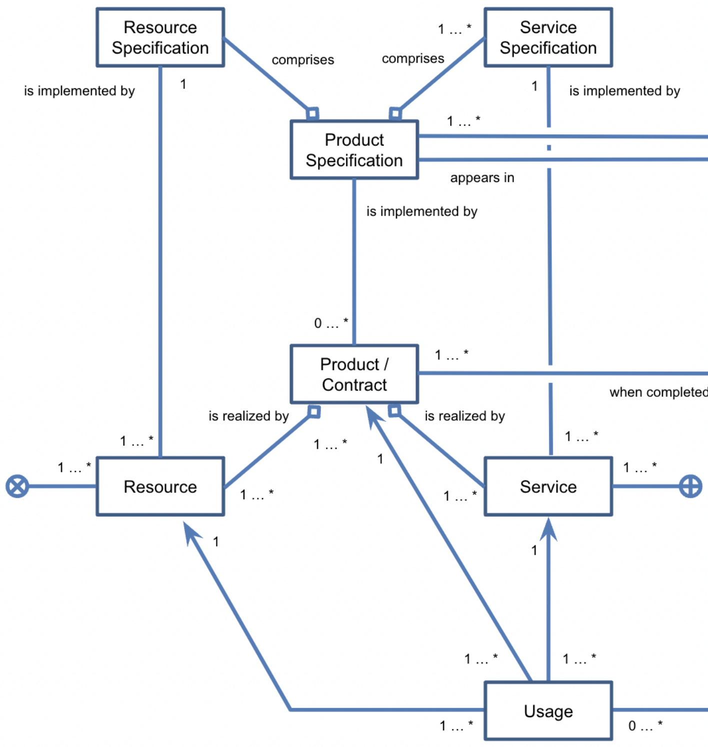


Figure 2 - TM Forum information model applied in DOME

3.4 Shared Catalogue and Transactions Ledger (Distributed Persistent Layer)

At the heart of the technical architecture of DOME is the DOME Distributed Persistent Layer which manages storage of, and access to, information associated with:

- the Shared Catalogue of Product Specifications (including the specifications of associated services and supporting resources) and Product Offerings defined by cloud and edge service providers
- Product Orders and Product instances along their lifecycle, as well as information about actual Usage of Products

The DOME Distributed Persistent Layer can be implemented on top of a number of interconnected national blockchains (like Alastria or HashNet) compatible with the European Blockchain Service Infrastructure (EBSI). As illustrated in Figure 3, each cloud and edge service provider, federated marketplace, and the DOME Global Portal backend itself implements an access node to the DOME Distributed Persistent Layer that implements the standard TM Forum APIs defined for the implementation of Marketplace functions.

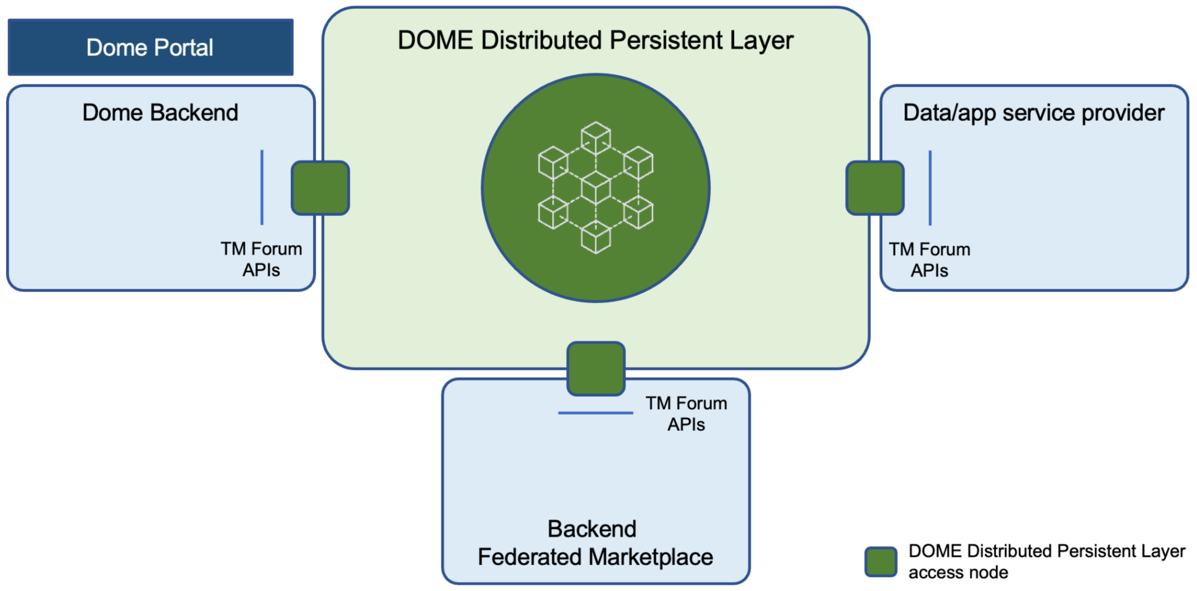

Dome Portal
Dome Backend
TM Forum APIs
DOME Distributed Persistent Layer
Data/app service provider
TM Forum APIs
TM Forum APIs
Backend Federated Marketplace
DOME Distributed Persistent Layer access node

Figure 3 - High-level architecture of the DOME Distributed Persistent Layer

When a Product Offering is created through the DOME Global Portal, for example, information about it has to be stored in the DOME Distributed Persistent Layer. This is achieved by invoking the specific operation for creating a Product Offering entity of the TM Forum Catalog Management API (TMF620 recommendation) that the Distributed Persistent Layer access node implemented in the DOME Global Portal backend supports. Part of this information is stored in the blockchain and, consequently, becomes replicated in all other nodes connected to the DOME Distributed Persistent Layer while rest of the information will be stored “off-chain” within the access node, which will typically also store a local copy of the information stored in the blockchain to support local queries in a more efficient manner. What part of the information will be stored in the blockchain and what part of information will be stored only “off-chain” is still to be decided. In any case, any access node will be able to access information stored “off-chain” based on information stored in the blockchain, provided it owns the necessary credentials that grant them access to the nodes where such “off-chain” data is stored.

Aligned with Gaia-X specifications, the description of Product Specifications and Product Offerings will be represented in the form of Verifiable Credentials/Presentations (VC/VP) compliant with W3C standard specifications[2], some

[2] W3C Verifiable Credentials Data Model v1.1, W3C Decentralized Identifiers (DIDs) v1.0

of which will take the form of labeled certifications (verifiable credentials issued by certification and audit agencies). These VC/VPs are also stored in the DOME Distributed Persistent Layer.

The DOME Distributed Persistent Layer brings transparency and trust to all participants since all transactions linked to the creation of Product Specifications, Product Offerings, Product Orders and Product Instances as well as their evolution over time or the generation of Usage Logs will be stored in a blockchain. This allows, for example, cloud data/app service providers to audit when their services have been procured and through which marketplace (any of the federated ones or directly DOME). Similarly, it will allow a given marketplace provider to audit when a given data/app service that was procured through its marketplace has been used. Last but not least, Usage Logs can be used by third party Charging/Billing/Payment gateways integrated with DOME which may be offered to cloud and edge service providers which do not want to implement a charging/billing/payment system on their own. They can also be used to generate verifiable credentials regarding operations of a service provider which can later be used as “passport” in front of investors or funding agencies.

3.5 Services providers journey

Figure 4 describes the journey that cloud and edge service providers will go through when interacting with DOME.

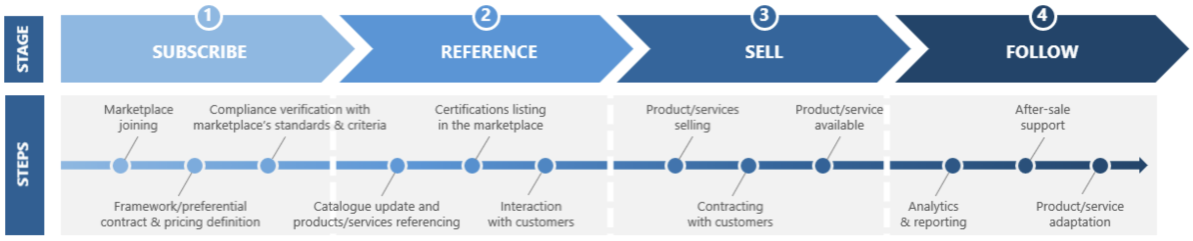

STAGE
1 SUBSCRIBE
2 REFERENCE
3 SELL
4 FOLLOW
STEPS
Marketplace joining
Compliance verification with marketplace's standards & criteria
Certifications listing in the marketplace
Product/services selling
Product/service available
After-sale support
Framework/preferential contract & pricing definition
Catalogue update and products/services referencing
Interaction with customers
Contracting with customers
Analytics & reporting
Product/service adaptation

Figure 4 - Service providers journey in DOME

Next we explain each of the stages, providing some details about what goes on within each stage from a technical perspective:

3.5.1 Stage 1 - Subscribe

The ‘subscription’ stage comprises all the steps followed by any given cloud and edge service provider since it joins DOME until it publishes its Product Offerings. This consists of three steps that the provider performs via DOME (either through APIs or the Portal) – 1) registration as a cloud/edge service (product) provider, 2) registration of the specifications of products it offers (defined as combination of services and associated resources) as well as definition of the basic characteristics of product offerings around registered product specifications like market segments the offering is targeted to (useful later on to fine tune discovery services), sale channels through which the offering will be visible (e.g., type of federated marketplaces in addition DOME), or terms and conditions, including information about the different pricing models supported, and 3) verification of the compliance with DOME's basic standards and criteria.

All these steps will imply registration and management of information linked to entities described in the information model previously described in Figure 2 using TM Forum APIs that the DOME Distributed Persistent Layer supports. As an example, registration of a given cloud/edge service provider would mean creation of a Party playing the role of Provider using the TM Forum Party Management API (TMF632 recommendation). Similarly, registration of Product Specifications and Product Offerings will be performed using the TM Forum Product Catalog Management API (TMF620 recommendation) which in turn will rely on the the TM Forum Service Catalog Management API (TMF633 recommendation) and the TM Forum Resource Catalog Management API (TMF634 recommendation) since products are made out of the combination of services and supporting resources. Cloud and edge service providers can perform these operations programmatically using the TM Forum APIs that their access nodes to the DOME Distributed Persistent Layer support or via de DOME Global Portal (whose backend, on the other hand, uses TM Forum APIs supported by the DOME Distributed Persistent Layer). Compliance verification of a given Product Specification or Product Offering will imply the transition of their status (one of the attributes these kinds of entities export) into "active" status.

An IaaS or Platform service provider that has implemented a marketplace connected to its services also relies on the TM Forum Party Management, Product Catalog Management, Service Catalog Management and Resource Catalog Management APIs (TMF632, TMF620, TMF633 and TMF634 recommendations) that their access nodes to the DOME Persistent Layer offer in order to register data/app service providers, as well as Product Specifications and Product Offerings of those data/app service providers that have been registered through their marketplaces instead of directly through DOME. This is why we say that these marketplaces connected to IaaS or Platform service providers are federated to DOME: no matter how a data/app service provider registers, directly or through a federated marketplace, its Product Specifications or Product Offerings will end up registered in the DOME Share Catalogue (part of the DOME Distributed Persistent Layer).

Cloud and edge service providers get notified when information relevant to them is stored in the DOME Persistent Layer. Such notifications are received through their access nodes to the DOME Persistent Layer. Thus, for example, when a specific data/app service provider registers a given product offering (associated with a given product specification defined as combination of services and resources) in DOME, it will be offered the possibility of register the product offering just in DOME or in DOME as well as any of the marketplaces connected to IaaS or Platform service providers federated with DOME. In the latter case, these IaaS or Platform service providers will receive a notification through the access node to the DOME Persistent Layer they implement. This way, a provider has only to register a data/app service once in DOME and get visible through the catalogue of all federated marketplaces it allows to work as sales channels. Note that additional compliance verification may be performed at the level of each of the federated marketplaces. For example, support of a NGSI-LD interface by the data/app service being registered may be verified by marketplaces associated with Platform services that are based on FIWARE.

Figure 5 illustrates interactions that take place during the Subscribe stage.

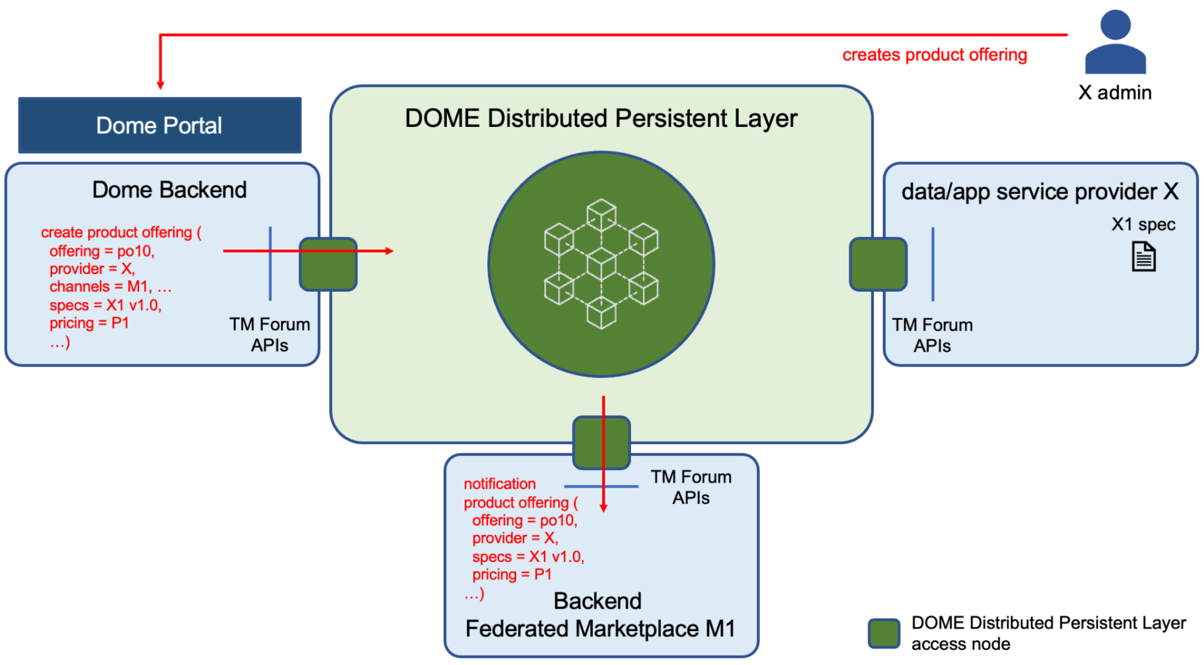


Figure 5 - interactions among components during subscribe stage

3.5.2 Stage 2 - Reference

Once a Product Offering becomes "active" it becomes visible to other users of DOME (typically end customers as well as IaaS and Platform service providers in the case of data/app services, since they may be interested to incorporate those data/app services in their respective catalogs). The provider of the Product Offering may establish visibility rules that determine who can get access to the offering.

Cloud and edge service providers may refer to web pages of the DOME global portal describing their product offerings once incorporated in the DOME Catalog. This way being able to promote them in front of potential customers.

A Cloud and edge service provider is able to update characteristics of its Product Offerings as well as corresponding Product Specifications (or specifications of associated services and resources). Those updates can be formulated through TM Forum APIs supported by the access nodes to the DOME Distributed Persistent Layer or via the DOME global portal. These updates will not only get registered in the DOME Product Catalog (becoming then visible through the DOME Global Portal to other direct users) but will be propagated to federated marketplaces in which the given Product Offerings / Specifications that got updated were also registered. This propagation will take place through notifications that federated marketplaces will receive through the access nodes to the DOME Distributed Persistent Layer they implement.

Through search and browsing capabilities that the DOME Global Portal will implement, customers will be able to easily find the specific product they are looking for. In its most basic format, the DOME global portal will allow customers to launch product offerings and product specifications searches, leveraging category filters and tagging functions, some tags connected to Verifiable Credentials (VCs) describing them. These functions will also be accessible via API enabling integration of more sophisticated customer applications.

Beyond basic search functions, DOME will implement more advanced features with the goal of connecting consumers with relevant services as quickly as possible. As an example, it will be possible to implement a search algorithm which would match customer search queries with keywords from relevant product listings. Even more advanced search functions may leverage additional information (such as product ratings or click-through rates) to prioritise/rank the results of search queries and improve the customer experience. Finally, search algorithms could also be specific to the customer's sector to provide results that take into account the customer's particularities.

Figure 6 illustrates interactions that take place during the Reference stage.

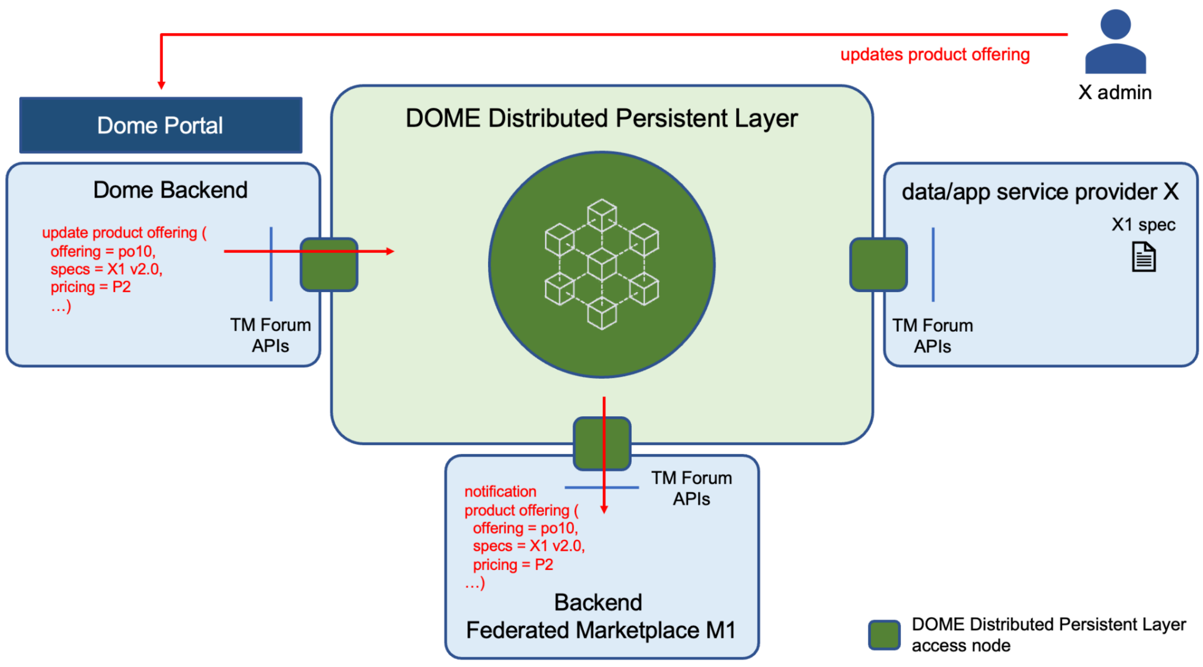

updates product offering
X admin
Dome Portal
DOME Distributed Persistent Layer
Dome Backend
update product offering (offering = po10, specs = X1 v2.0, pricing = P2 …)
TM Forum APIs
data/app service provider X
X1 spec
TM Forum APIs
notification product offering (offering = po10, specs = X1 v2.0, pricing = P2 …)
TM Forum APIs
Backend
Federated Marketplace M1
DOME Distributed Persistent Layer access node

Figure 6 - interactions among components during reference stage

3.5.3 Stage 3 - Sell

There are two ways in which a given Product offered by a cloud/edge service provider can be procured: either directly after discovery through DOME or through marketplaces associated with IaaS or Platform service providers where the corresponding Product Offering has also been registered. When a given customer discovers a cloud/edge service Product Offering it is interested in, both possibilities are offered.

In the first case, procurement may be performed either via the DOME Global Portal or programmatically. In both cases, the creation of a Product Order will be ultimately requested using the TM Forum Product Ordering Management API (TMF622 recommendation) that the Distributed Persistent Layer access node implemented in the DOME Global Portal backend supports.

In the second case, typically associated with procurement of data/app services, the customer will be redirected to the marketplace of its choice, through which the procurement process will be handled. At a given moment, the creation of a Product Order will be performed via invocation of the TM Forum Product Ordering API supported by the DOME Distributed Persistent Layer access node linked to the selected marketplace. Note that this Product Order will also become visible not only in the federated marketplace but also at the DOME Global Portal.

In any of the two cases, a Product Order is created within the DOME Distributed Persistent Layer and the given cloud/edge provider will receive a notification about creation of the Product Order it should handle. This notification will be received through their corresponding DOME Distributed Persistent Layer access node.

Note that many customers will end up consuming services through the portals of federated marketplaces the DOME global portal will guide them to. This is because these portals are expected to provide a better tailored user experience (UX). However, the federation of marketplaces with DOME will mean that all relevant transactions will be registered in the DOME Distributed Persistent Layer and therefore become visible at the DOME Global Portal, this way ensuring transparency and giving higher trust to both customers and data/app service providers.

Figure 7 illustrates interactions that take place during Product Ordering.

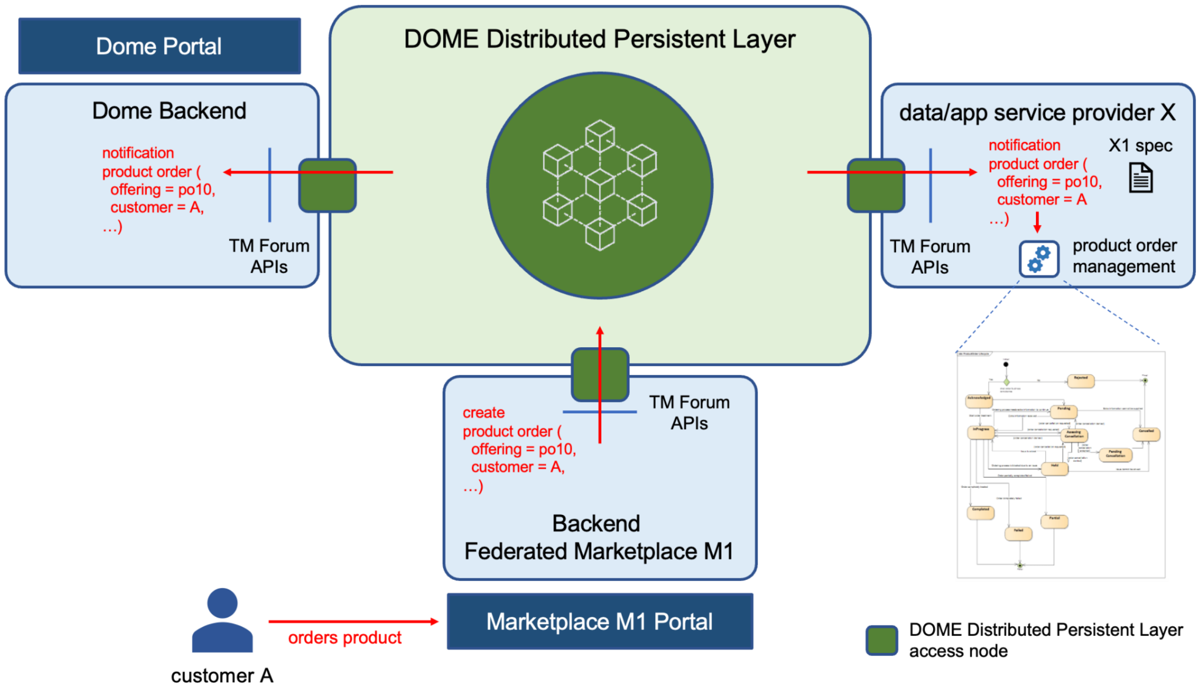
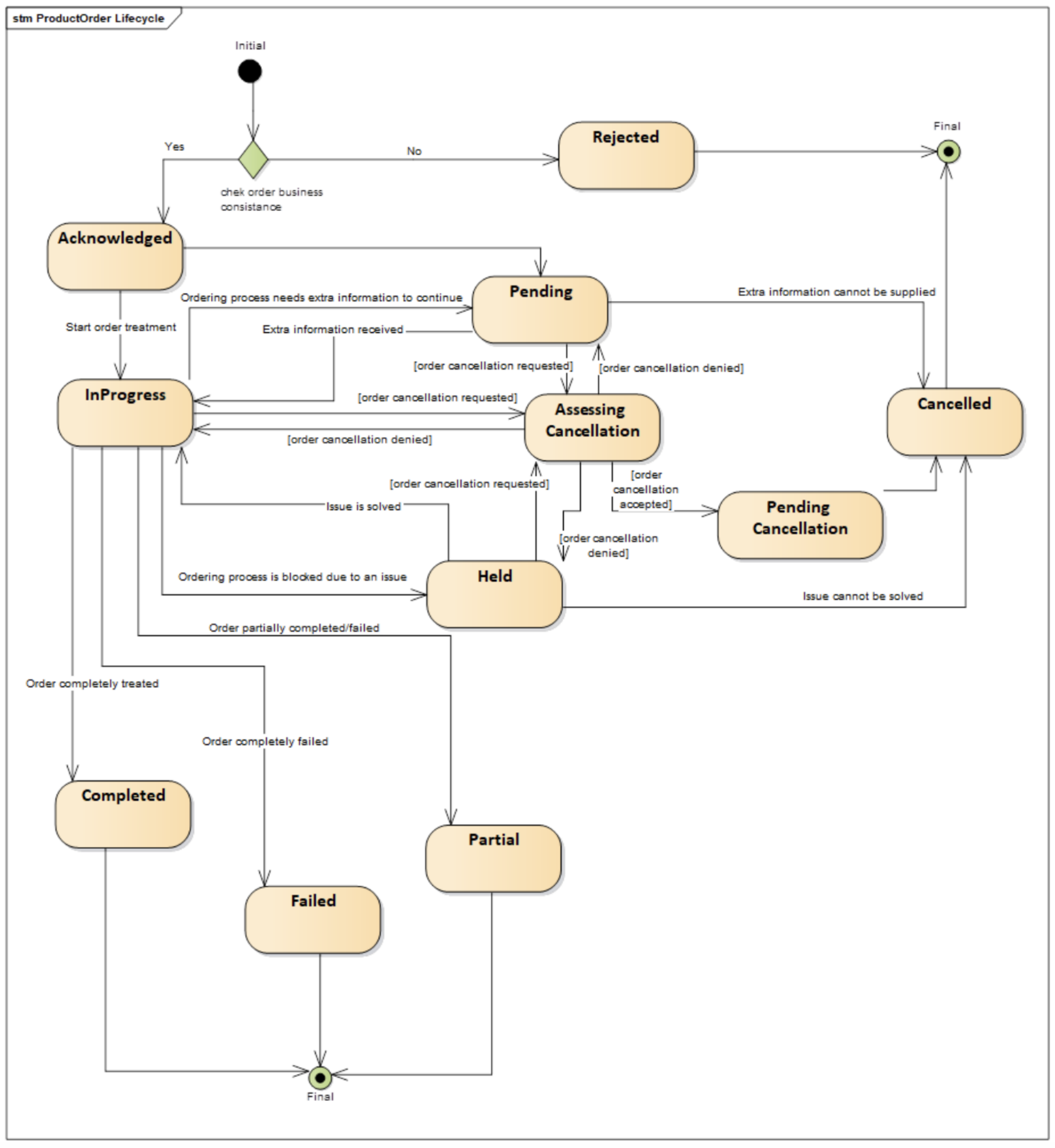


Figure 7 - interactions among components during Product Ordering

Figure 8 illustrates the different states a Product Order will go through since it is issued by a given customer and it gets completed. Such states will be reflected as values of the attribute “state” that any Product Order will support. The defined lifecycle complies with TM Forum specifications but will be revised based on feedback from first deployment and pilots of DOME.



stm ProductOrder Lifecycle
Initial
Yes
No
chek order business consistance
Rejected
Final
Acknowledged
Pending
Ordering process needs extra information to continue
Extra information cannot be supplied
Start order treatment
Extra information received
[order cancellation requested]
[order cancellation denied]
InProgress
[order cancellation requested]
Assessing Cancellation
Cancelled
[order cancellation denied]
[order cancellation requested]
[order cancellation accepted]
Issue is solved
Pending Cancellation
[order cancellation denied]
Ordering process is blocked due to an issue
Held
Issue cannot be solved
Order partially completed/failed
Order completely treated
Order completely failed
Completed
Partial
Failed
Final

Figure 8 - Lifecycle of Product Orders

Once a Product Order is completed, a contract between the customer and the service provider is established so that terms and conditions defined in the Product Offering start to apply. As a result, the customer becomes a Trusted Issuer of Verifiable Credentials relevant to the product business logic (see section on “Trust Anchor and Decentralized Identity and Access Management (IAM) Framework” below). The service provider will then create a Product instance using the TM Forum Product Inventory Management API (TMF637 recommendation) that its access node to the DOME Distributed Persistent Layer supports. As a result, the Product instance will become visible to the customer, either through the DOME global portal or the federated marketplace through which the originating Product Order was issued.

Figure 9 illustrates the different states a Product instance will go through since it is created, right after the originating Product Order was completed, until it is terminated. Such states will be reflected as values of the attribute "state" that any Product instance will support. The defined lifecycle complies with TM Forum specifications but will be revised based on feedback from first deployment and pilots of DOME.

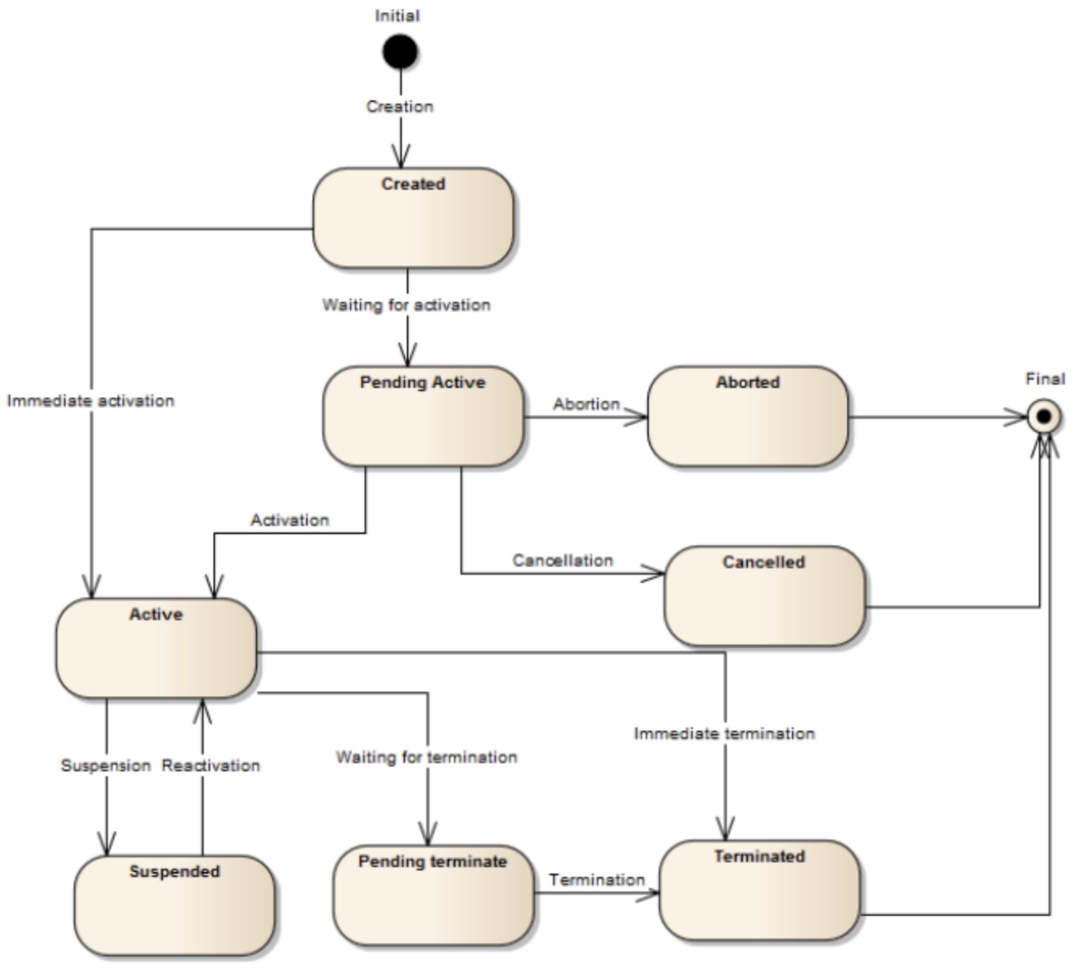


Figure 9 - Lifecycle of a Product (instance)

Note that the creation of a Product instance does not necessarily mean that its component services and required resources get automatically provisioned and activated. There may exist a period from the time at which a Product is created until it actually can be used by the customer that ordered it. This is for example the case in connection to products which require deployment of resources in the field (e.g., an app for air quality monitoring which requires deployment of several IoT devices in the field). It is also the case when manual configuration and/or integration testing with products from third parties is required. Once everything is ready for actual usage, the state of the Product becomes "Active". This will be the point at which access to the service will be permitted, or logs for the initial charging will be generated in connection to one-payment or subscription fee pricing models. It will also be the point at which

Usage logs will start to be generated, bringing the basis for the monitoring of services as well as the support to pay-per-use pricing models.

Cloud and edge service providers registered in DOME will commit to register Usage logs in the DOME Distributed Persistent Layer, using the TM Forum Usage Management API (TMF635 recommendation) that its access node to the DOME Distributed Persistent Layer supports. Those Usage logs will ultimately be recorded in the blockchain associated with the DOME Distributed Persistent Layer but multiple logs will be condensed into a block for performance reasons.

Figure 10 illustrates interactions that will take place during the lifecycle of a Product instance, particularly at the time of its creation as the result of completing a Product Order by a particular customer, and its activation for that customer.

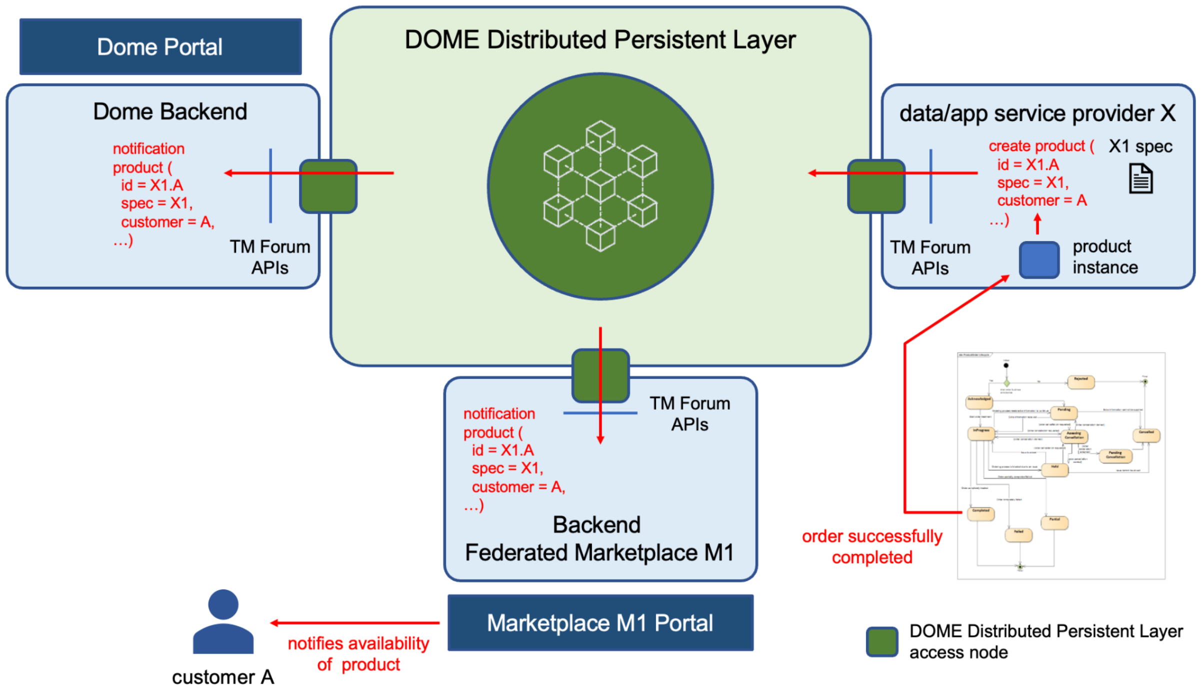

Dome Portal
Dome Backend
notification product (id = X1.A spec = X1, customer = A, ...)
TM Forum APIs
DOME Distributed Persistent Layer
data/app service provider X
create product (id = X1.A spec = X1, customer = A ...)
X1 spec
TM Forum APIs
product instance
notification product (id = X1.A spec = X1, customer = A, ...)
TM Forum APIs
Backend Federated Marketplace M1
Marketplace M1 Portal
order successfully completed
customer A
notifies availability of product
DOME Distributed Persistent Layer access node

Figure 10 - Interactions during the lifecycle of Product instances

3.5.4 Stage 4 - Follow

Once providers have sold their service, they will need to be able to monitor consumption, provide after-sale support and leverage the experience to innovate and continuously strengthen their service offering.

Reporting and analytics features will be offered via the DOME Global Portal that providers and customers can consume. Notably, the marketplace can provide users the option to personalize their reports or to export data to outside platforms via connectors and APIs. Since relevant information for fueling these reporting and analytic tools is accessible through the DOME Distributed Persistent Layer, more advanced versions can be offered as Third-Party services that get access to the APIs that the DOME Distributed Persistent Layer offers.

Note that customers will mostly end up consuming services through the portals of federated marketplaces the DOME global portal will guide them to. These portals are expected to incorporate their own reporting and analytic functions meeting the needs of customers (particularly to support the different stages of their journey).

3.6 Customers journey

Figure 11 describes the journey that consumers of cloud and edge services offered through DOME will go through when interacting with DOME.

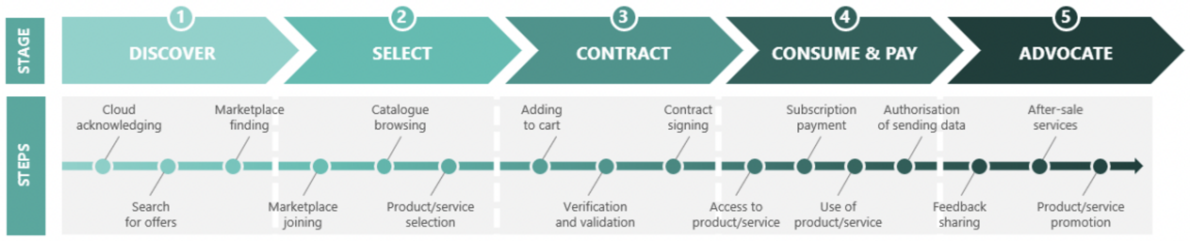


Figure 11 - Consumers journey in DOME

Those customers who approach the DOME ecosystem for the first time or wish to check other marketplaces different than the one they are already using, will connect to the DOME global portal searching for offerings. They may end selecting and contracting individual data/app cloud or edge services directly through DOME which may require use of third party payment services integrated with DOME. However, in other cases they will look for the marketplaces, connected or not to an IaaS, Platform provider or Individual Marketplace Provider that may better solve their overall needs. When a customer selects a given marketplace then they will further interact with its corresponding portal, which will typically mean they will enjoy a more personalized user experience through that portal interface, including the payment of services. Note that, despite further interactions will then be bilateral with the marketplace, DOME will bring trust to the relationships established between customers and app/data cloud and edge services, because both can audit transactions as they are logged in the DOME Distributed Persistent Layer. Satisfied customers will become the best ambassador of DOME and federated marketplaces based on a satisfactory experience.

3.7 Interoperability with Data Publication Platforms

Some of the cloud or edge data services registered in DOME may bring access to static data or near real-time data resources available through RESTful APIs (e.g., IoT data). DOME will integrate data publication functions enabling the exposure of such data resources in compliance with DCAT specifications defined by W3C and DCAT-AP recommendation by the EC. This way, data resources linked to data services offered through DOME can be harvested through external Data Publication platforms (e.g., the European Data Portal) .

4 Detailed workflows based on a common reference use case

4.1 Overall description of the reference use case

In order to better visualize and understand the details of the descriptions in the previous chapters, we define a highly detailed reference use case with technical descriptions that can be generalized to other use cases, always taking into account the different nature of different use cases. This section brings an overall description of the reference use case through which we will specify how the different technical building blocks for supporting data spaces will be integrated and be used together.

The reference use case implements a scenario where a data service provider offers a service on a public marketplace, so that service consuming parties can acquire access to this offering. Furthermore, these consuming parties can delegate the access to the acquired service offering to their users (e.g., customers).

In this use case, the provider is a packet delivery company, supporting creation and management of packet delivery orders and 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 services of the packet delivery company through the Data Space Marketplace, and delegate the access to its customers.

In the reference use case, several parties are involved, each hosting its own infrastructure. Namely:

- Data Space Marketplace: Public marketplace for creating service offerings and acquiring access to them

- Trust Anchor: fulfills 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.
- 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)

The following figure depicts the overall architecture of the reference use case. 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 4.2 Parties involved.

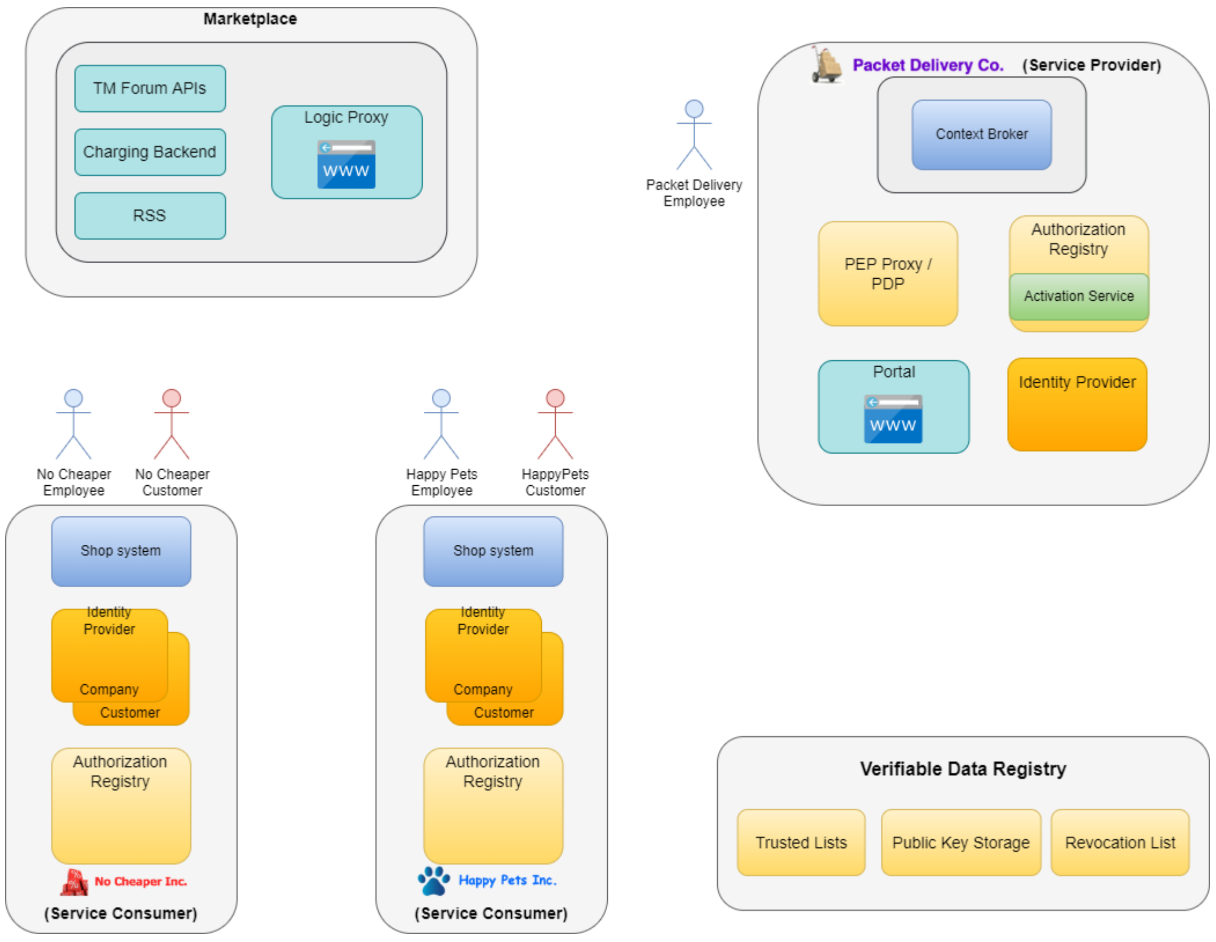


Figure 4.1: Overall architecture

4.2 Parties involved

4.2.1 Data Service Provider: Packet Delivery Company

Packet Delivery Company (PacketDelivery for short) 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 the PacketDelivery 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).

PacketDelivery 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)

PacketDelivery 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-ld/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-ld/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.

PacketDelivery 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 PacketDelivery 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 a 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, PacketDelivery 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 because it has acquired the Standard Packet Delivery service which does not allow to change those values.

4.2.2 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. 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 act 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
  - We will describe later in this document the detailed process to perform this operation

4.2.3 Data Service Consumer: NoCheaper Ltd

NoCheaper Ltd (NoCheaper for short) 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

4.2.4 Marketplace

The 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.

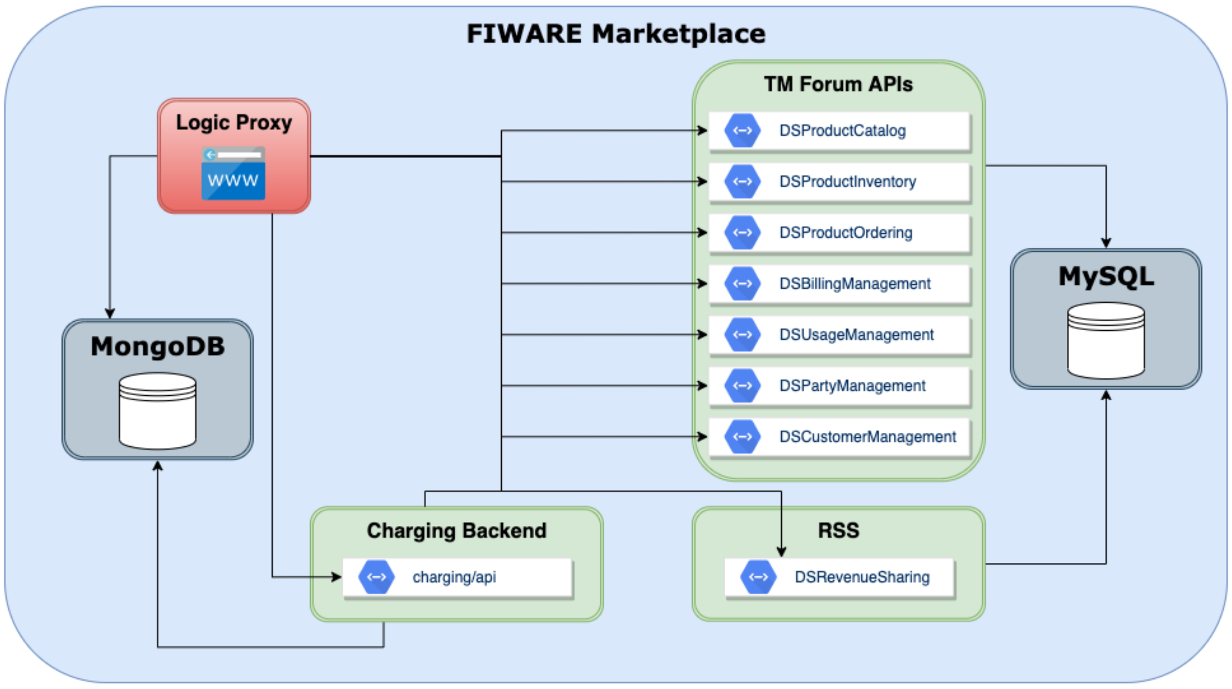


Figure 4.2: FIWARE Marketplace architecture overview

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.

4.2.5 Trust Anchor Framework

The system uses Verifiable Credentials and participants are identified via DIDs (described in "Decentralized Identifiers (DIDs) v1.0"). In order to enable an efficient and decentralized verification of the credentials and identities of participants, a blockchain-based Trust Framework has to be implemented to avoid central entities intermediation in all authentication flows.

The trust framework is basically composed of two things:

1. A list of the identities of trusted organizations stored in the blockchain, together with associated information for each entity.
2. A process to add, modify and delete the trusted entities, implementing a concrete governance model.

The trust framework is designed to be largely decentralized and represents the trust relationships in the real world. Here we describe a possible approach to implementing a blockchain-based trust framework which is very decentralized and at the same time simple, secure and robust.

The identities of the legal persons involved in the ecosystem are registered in a common directory implemented in the blockchain following a hierarchical scheme very similar to the DNS (Domain Name Service) schema in the Internet.

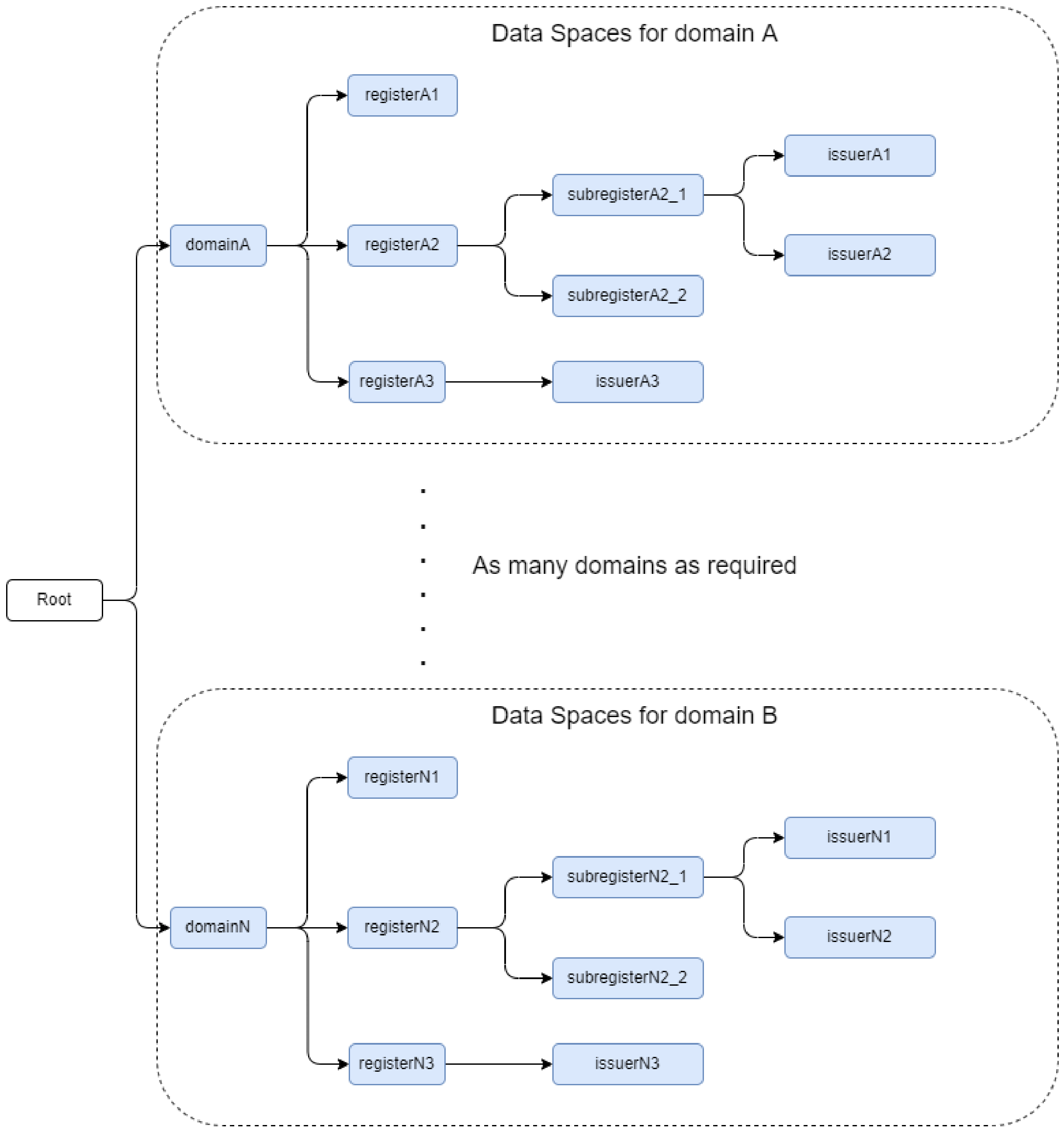
Essentially, once an entity is registered in the system, it is completely autonomous for adding other entities that are managed as child entities.

In this way, trust is delegated according to a well defined, transparent, auditable and public scheme. Any participant can get trusted information about the current trust structure of the ecosystem and also the events that led the system to the current situation. For example, what entity registered another entity, when it was done and what attributes were assigned to the child entity by the parent entity.

The scheme is flexible enough to implement as many levels as required by the actual governance model of the ecosystem. In very simple cases. it can have just two levels, where there is only one entity registering all other participants in the ecosystem.

In general, the system can be made very decentralized. However, there is one centralized element: the root of trust at the top of the hierarchy should be a trusted entity (or federation of entities) in the ecosystem that is the one responsible for bootstrapping the system. Depending on the concrete governance framework of the ecosystem, this may be the only mission of the root entity, possibly including the monitoring and oversight of the ecosystem. Typically it should be a regulatory body, a public administration or a neutral organization which is accepted as fully trusted by all participants in the ecosystem.

The approach for a single blockchain network is described in the following figure (the scheme is easily extensible to different blockchain networks where we want to establish trust across them so entities in one network can interact with entities in another network in a trusted way.



As can be seen in the figure, the Trust Framework in a given blockchain is not really a flat list, but a hierarchical structure, implemented as a Smart Contract:

- There is a special organization (or set of organizations) which is at the root of the hierarchy. This entity is called the Trusted Registration Authority (TRA) in EBSI, or Trusted Anchors in other contexts. We will use the term Trusted

Anchors in the following description. The essential characteristic is that this is the most trusted entity/entities in the ecosystem.

- This root entity is responsible for registering the identities of some other trusted entities. For example, in a country with several regions with autonomous competencies to manage universities, the Ministry of Education could register in the blockchain the identities of the regional institutions which are responsible for managing the universities in each of their regions.
- Once this is done, each of the regional institutions can register the identities of dependent entities, like universities.
- The hierarchy can have several levels. For example, a university can be big and have several organizational units with some autonomy, maybe distributed geographically. It can create sub-identities and register them as child nodes in the blockchain.

4.2.5.1 Registering identities in the ecosystem

A new identity can only be registered by an existing identity. The only exception is the Trust Anchors entity which is the one deploying the Smart Contract to the blockchain and so it has special privileges. On deployment, the Smart Contract allows the registering of the Trust Anchor's identity and associated information.

Every legal entity identity in the system has assigned a domain name, in a similar way to what happens with Internet domains. When a new identity is created, it is assigned a name and it is automatically set up as a sub-domain depending on the parent identity. The only exception is that the root domain (Trust Anchors) has an empty name.

For example, the entity ***issuerA1*** in the diagram above has a full domain name of ***domainA.registerA2.subregisterA2_1.issuerA1*** and it is uniquely identified by its full domain name.

In this example, ***domainA*** is a top-level domain which should have been added to the system by the Trust Anchors entity.

It should be clear that an organization can be registered in the blockchain only because its parent entity has registered it. No other entity in the Trust Framework can have performed the registration, not even the parent of the parent entity.

An organization is responsible for all its child entities, represented as child nodes in the Trust Framework.

A third party external to the framework with read access to the blockchain can see the whole trust structure, including the immutable history of events from the initial bootstrapping of the ecosystem that led to the current status. This provides an incredible transparency to the ecosystem.

4.2.5.2 Verifying identities: the Universal Resolver

To be useful to all participants, the Trust Framework requires a component that implements a public API (non-authenticated) which can be used by any participant to verify identities: the Universal Resolver. The Universal Resolver resolves Decentralized Identifiers (DIDs) across many different DID methods, based on the W3C DID Core 1.0 and DID Resolution specifications. A reference implementation of the Universal Resolver is available from the Decentralized Identity Foundation Identifiers & Discovery Working Group.

DID resolution is the process of obtaining a DID Document for a given DID. This is one of four required operations that can be performed on any DID ("Read"; the other operations being "Create", "Update", and "Deactivate"). The details of these operations differ depending on the DID method. Building on top of DID resolution, DID URL dereferencing is the process of retrieving a representation of a resource for a given DID URL. Software and/or hardware that is able to execute these processes is called a DID resolver.

The process of DID resolution is needed during the SIOP flows when we have to obtain the public keys associated with an entity and be able to verify its signature over some data used in the exchange of information. The public key is part of the DID Document that is obtained after performing DID Resolution. See the document Decentralized Identifier Resolution (DID Resolution) for more information.

Ideally, there should be many instances of the Universal Resolver running in the ecosystem, because having just one instance increases the risk of centralisation. In particular, any legal entity that wants to reduce dependencies from third parties as much as possible (to be self-sovereign) would like to operate its own instance of a Universal Resolver on top of its own blockchain node connected directly to the blockchain network. Alternatively, an entity may wish to rely on a third party that they trust to perform DID resolution.

4.3 Verifiable Credentials in the ecosystem

In this section we describe the different types of credentials that are needed for the functionalities in the ecosystem.

4.3.1 Employee of Packet Delivery

Packet Delivery issues credentials to some of its employees, so they can access the Marketplace, either to create offerings or to purchase offerings.

This credential is used by an employee of Packet Delivery to prove to a third party that she is entitled to use some services provided by the third party on behalf of the employed company (Packet Delivery in this case). In other words, the credential is used as a mechanism for Packet Delivery to delegate its access control rights to one or more of its employees.

The essential characteristics of such credential are:

- Nobody has tampered with its contents since it was issued, because the credential is digitally signed by the issuer, Packet Delivery.
- Proves that the issuer is Packet Delivery, because the public key that verifies the signature of the credential is cryptographically associated with the real-world identity of Packet Delivery registered in the Trust Framework.
- Optionally, it can prove that the credential was issued no later than a given time, because the credential was registered (timestamped) in the blockchain when it was issued. The term “notarisation” is commonly used for this action, but it is wrong, because the term is coming from anglo-saxon cultures where notaries are very different from the latin-germanic notary functions in the EU and many other countries in the world. We will use the term “timestamping”.

Please note that the date of timestamping can be greater than the date in the field “Issued at” included inside the credential. For example, the credential is created and signed at one time, but timestamped the next day (maybe to batch the operation with other credentials). The real requirement is that nobody can create a credential and timestamp as if it happened in the past. In other words, nobody can create credentials from the past. The verifiers have to check that the field inside the credential “Issued at” is not later than the timestamp (at least by a small leeway to account for clock synchronisation differences).

Also note that many credentials may not require timestamping, avoiding the overhead of the registration process. It all depends on the type of credential, the intended usage of the credential and the level of risk assumed. The employee credential discussed here is one example of credential that does not require timestamping with the same level of risk. The only thing that the verifier requires is that the holder can prove that at the time of usage of the credential (eg., login), the credential was issued by the employer (Packet Delivery in our case). Obviously, this does not require timestamping, because if the employee can present a credential when performing login, she can do so only if the credential was issued before.

From the above description we can derive the following trust properties for a verifier receiving a credential:

- The level of trust in the identity of the issuer of the credential depends on the level of trust of the verifier in the onboarding process implemented in the Trust Framework. The onboarding process associates the public key of the issuer with its real identity.

- The level of trust in the claims inside the credential depends on the level of trust that the verifier has with the issuer entity. For example, Packet Delivery could issue employee credentials to people who are not real employees. However, if this is the case the verifier has a strong non-repudiable mechanism to prove to third-parties (e.g., a court) that the issuer stated wrong facts.

From the above it follows that Packet Delivery can issue employee credentials which include some employee data (name, surname, etc.) and the verifier can have a given level of trust on those claims.

But this just proves that Packet Delivery attests that the data inside the credential (called claims) is true. It does not say anything about whether the person presenting the credential online is the same that is referred to in the claims. In other words, the person sending the employee credential to the verifier could be a different person from the employee.

This is the reason why the credential includes a public key as one of the claims associated with the employee (inside the “credentialSubject” object.

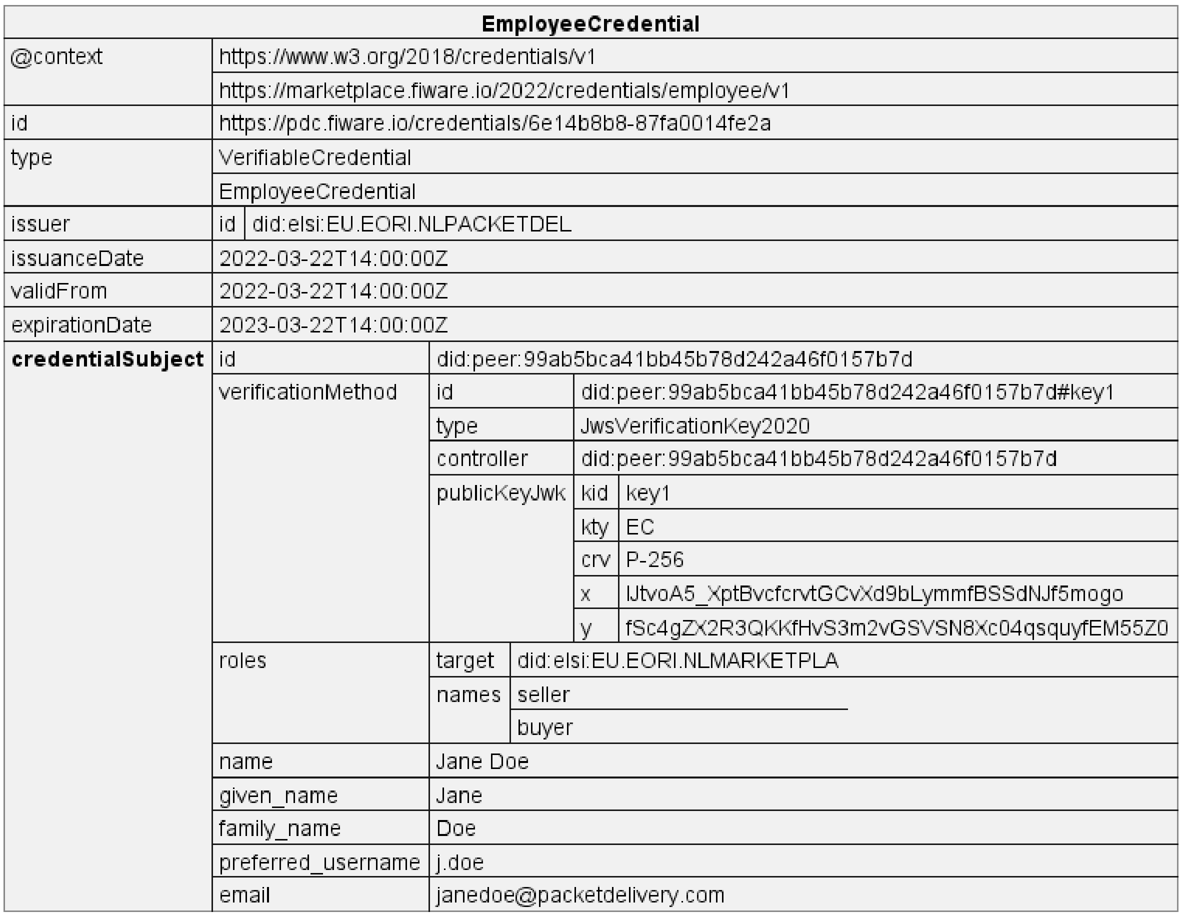
That public key corresponds to a private key that was generated in the employee device (PC or mobile) during the process of credential issuance. The process is explained in more detail later, but essentially:

- The employee generates a pair of public/private keys and sends the public key to the employer via an authenticated and encrypted channel (e.g., HTTPS). This channel can be the usual mechanism that employees use to connect to enterprise applications.
- The employer generates a credential with some employee data and includes the public key.
- The employer signs the credential and sends it to the employee using the same authenticated channel.

Below we present an example employee credential issued by Packet Delivery.

```
// Credential issued by PacketDelivery to its employees, providing access to
// Marketplace, either to create offerings or to purchase offerings.
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://marketplace.fiware.io/2022/credentials/employee/v1"
  ],
  "id": "https://pdc.fiware.io/credentials/6e14b8b8-87fa0014fe2a",
  "type": ["VerifiableCredential", "EmployeeCredential"],
  "issuer": {
    "id": "did:elsi:EU.EORI.NLPACKETDEL"
  },
  "issuanceDate": "2022-03-22T14:00:00Z",
  "validFrom": "2022-03-22T14:00:00Z",
  "expirationDate": "2023-03-22T14:00:00Z",
  "credentialSubject": {
    "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
    "verificationMethod": [
      {
        "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
        "type": "JwsVerificationKey2020",
        "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
        "publicKeyJwk": {
          "kid": "key1",
          "kty": "EC",
          "crv": "P-256",
          "x": "lJtvoA5_XptBvcfcrvtGCvXd9bLymmfBSSdNJf5mogo",
          "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
        }
      }
    ],
    "roles": [
        {
            "target": "did:elsi:EU.EORI.NLMARKETPLA",
            "names": ["seller", "buyer"]
        }
    ],
    "name": "Jane Doe",
    "given_name": "Jane",
    "family_name": "Doe",
    "preferred_username": "j.doe",
    "email": "janedoe@packetdelivery.com"
  }
}
```

The structure of the above credential can be visualized as follows:



EmployeeCredential				
@context	https://www.w3.org/2018/credentials/v1			
	https://marketplace.fiware.io/2022/credentials/employee/v1			
id	https://pdc.fiware.io/credentials/6e14b8b8-87fa0014fe2a			
type	VerifiableCredential			
	EmployeeCredential			
issuer	id	did:elsi:EU.EORI.NLPACKETDEL		
issuanceDate	2022-03-22T14:00:00Z			
validFrom	2022-03-22T14:00:00Z			
expirationDate	2023-03-22T14:00:00Z			
credentialSubject	id	did:peer:99ab5bca41bb45b78d242a46f0157b7d		
	verificationMethod	id	did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1	
		type	JwsVerificationKey2020	
		controller	did:peer:99ab5bca41bb45b78d242a46f0157b7d	
		publicKeyJwk	kid	key1
			kty	EC
			crv	P-256
			x	lJtvoA5_XptBvcfcrvtGCvXd9bLymmfBSSdNJf5mogo
			y	fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0
	roles	target	did:elsi:EU.EORI.NLMARKETPLA	
		names	seller	
			buyer	
	name	Jane Doe		
	given_name	Jane		
	family_name	Doe		
	preferred_username	j.doe		
	email	janedoe@packetdelivery.com		

The credential is of type "*EmployeeCredential*" and to enable access to the Marketplace the roles embedded in it can be "buyer", "seller" or both. The URL in the "@context" field points to the marketplace (https://pdc.fiware.io/2022/credentials/employee/v1), which defines the general requirements for an Employee Credential. However, participants in the ecosystem can extend it and of course use the roles and role names that they need for their own purposes.

The "*credentialSubject*" section in the credential has the following objects:

- "**id**", specified as a DID. For privacy reasons and given that this is a natural person, the DID used is the Peer Method as specified in the W3C Peer DID Method Specification. The method can be used independent of any central source of truth, and is intended to be cheap, fast, scalable, and secure. It is suitable for most private relationships between people, organizations, and things.
- "**verificationMethod**", which is a standard W3C VC object that specifies the Public Key associated with the DID of the employee. The binding between the

DID of the employee and the Public Key was performed at the moment of credential issuance by Packet delivery.

- "**roles**" is an array with one or more role specifications. Each specification defines a potential target entity that will receive the credential, and one or more names of roles defined by that target entity.
    - "**target**" is the DID of the entity that will receive the credential.
    - "**names**" is an array with one or more roles that the target entity recognizes and that will be used by the target entity to apply its own access control policies. In the example, we have used both "*buyer*" and "*seller*" roles as defined by the Marketplace. Other entities can define their own roles for their specific purposes. Names are made unique in the ecosystem thanks to the *target* property.
- The rest of the fields in the credential have the usual meaning in the standard W3C Verifiable Credential Data Model.

The "id" field at the top level is the identification of the credential, which can be used for revocation if that functionality is required. The basic requirements for the "id" field are that:

- it is unique in the scope where it is going to be used
- It is difficult to "guess" by a potential attacker who could for example revoke a given credential
- it is not related in any way with the personal data included in the credential, to minimise the risk of correlation

A UUID Version 4 complies with all those requirements but other schemas can be used.

4.3.2 Employee of Happy Pets (or No Cheaper)

The employee credential issued by Happy Pets and No Cheaper companies to its employees are virtually identical to the employee credential from Packet Delivery described above. The main difference is the set of roles assigned to the employee and specified in the "roles" claim.

```
// Credential issued by HappyPets to its employees, providing access
// to order creation in PacketDelivery.
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://happypets.fiware.io/2022/credentials/employee/v1"
  ],
  "id": "https://happypets.fiware.io/credentials/25159389-8dd17b796ac0",
  "type": ["VerifiableCredential", "EmployeeCredential"],
  "issuer": {
    "id": "did:elsi:EU.EORI.NLHAPPYPETS"
  },
  "issuanceDate": "2022-03-22T14:00:00Z",
  "validFrom": "2022-03-22T14:00:00Z",
  "expirationDate": "2023-03-22T14:00:00Z",
  "credentialSubject": {
    "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
    "verificationMethod": [
      {
        "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
        "type": "JwsVerificationKey2020",
        "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
        "publicKeyJwk": {
          "kid": "key1",
          "kty": "EC",
          "crv": "P-256",
          "x": "lJtvoA5_XptBvcfcrvtGCvXd9bLymmfBSSdNJf5mogo",
          "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
        }
      }
    ],
    "roles": [
        {
            "target": "did:elsi:EU.EORI.NLPACKETDEL",
            "names": ["P.Create"]
        }
    ],
    "name": "Jane Doe",
    "given_name": "Jane",
    "family_name": "Doe",
    "preferred_username": "j.doe",
    "email": "janedoe@packetdelivery.com"
  }
}
```

4.3.3 Customer of Happy Pets (or No Cheaper)

This credential is used by Happy Pets to delegate access control to customers that want access to services provided by Packet Delivery and that were purchased by Happy Pets in the past.

It follows the same model as with employee credentials except that:

- The credential should be issued by Happy Pets to customers using a secure and authenticated channel created as part of a previous customer onboarding process (KYC).
- The role included in the credential corresponds to the type of customer, with the role name defined and understood by the service provider, in this case Packet Delivery.

```
// Credential issued by HappyPets to a customer,
// providing access to Gold services at PacketDelivery.
{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://happypets.fiware.io/2022/credentials/employee/v1"
  ],
  "id": "https://happypets.fiware.io/credentials/25159389-8dd17b796ac0",
  "type": ["VerifiableCredential", "CustomerCredential"],
  "issuer": {
    "id": "did:elsi:EU.EORI.NLHAPPYPETS"
  },
  "issuanceDate": "2022-03-22T14:00:00Z",
  "validFrom": "2022-03-22T14:00:00Z",
  "expirationDate": "2023-03-22T14:00:00Z",
  "credentialSubject": {
    "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
    "verificationMethod": [
      {
        "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
        "type": "JwsVerificationKey2020",
        "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
        "publicKeyJwk": {
          "kid": "key1",
          "kty": "EC",
          "crv": "P-256",
          "x": "lJtvoA5_XptBvcfcrvtGCvXd9bLymmfBSSdNJf5mogo",
          "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
        }
      }
    ],
    "roles": [
        {
            "target": "did:elsi:EU.EORI.NLPACKETDEL",
            "names": ["P.Info.gold"]    // Or P.Info.standard
        }
    ],
    "name": "Jane Doe",
    "given_name": "Jane",
    "family_name": "Doe",
    "preferred_username": "j.doe",
    "email": "janedoe@packetdelivery.com"
  }
}
```

4.3.4 Role-based access

As can be seen in the above credentials, they contain claims specifying roles. The roles are not defined by the issuer of the credential, but by the provider that is going to receive the credential and perform authentication and authorization.

The provider defines a role having a certain name, and this role is mapped to a certain policy set representing the policies that the provider wants to enforce. 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 organization within the Authorisation Registry of the provider. Furthermore, the acquiring organization then can just assign these roles to their users by embedding the roles inside the Verifiable Credential issued to its users. When accessing the service, it is up to the PEP proxy/PDP component of the provider 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.

It is out of scope for this document to describe the actual policy language and engine used to perform the enforcement (ODRL, Rego, etc).

4.3.5 Deployment of components

In addition to the components described in section 4.2 Parties involved the following components are needed.

Verifiable Data Registry

In the form of a blockchain network that is used to implement the core technology of the Trust Framework for the ecosystem. Some entities participating in the ecosystem (not necessarily all of them) should operate blockchain nodes in order to create and operate collaboratively a suitable blockchain network that can implement the backbone of the Trust Framework.

Universal Resolver

The Universal Resolver resolves Decentralized Identifiers (DIDs) across many different DID methods, based on the W3C DID Core 1.0 and DID Resolution specifications. See 2.4.2.2 Verifying identities: the Universal Resolver for more details.

Credential Issuer and Verifier components

These components are normally implemented as extensions to existing components implementing the OIDC flows.

End-User wallet

The wallet component that the End-User employs to receive, hold and present Verifiable Credentials that have been issued to her. This component can be implemented as a native mobile application, a PWA application or even as a web app hosted by one or more highly trusted entities in the ecosystem.

4.4 Detailed workflows

4.4.1 Create Offering

We now describe the process of creating an offer. In the reference use case, Packet Delivery needs to perform it twice for creating the offerings for “Basic Delivery” and “Premium Delivery”, providing a different set of offering information. The process will be performed by an employee of the Packet Delivery company.

When using the SIOP flows with Verifiable Credentials it can be observed that the Marketplace does not have to query any other entity in the ecosystem to verify the credential because all the information needed is in the Verifiable Credential presented by the employee and in the Decentralized Verifiable Registry (implemented in our case using a blockchain network), accessed via the Universal Resolver.

In other words, the flows are essentially peer-to-peer and do not require any centralized IdP to be queried, providing an efficient, scalable, private and resilient framework.

4.4.1.1 Sequence description (Packet Delivery Co.)

The following gives a detailed description of the offer creation process. Figure 4.4.1.1a presents the different interactions in an architectural overview, whereas Figure 4.4.1.1b shows a detailed sequence diagram of the whole process.

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

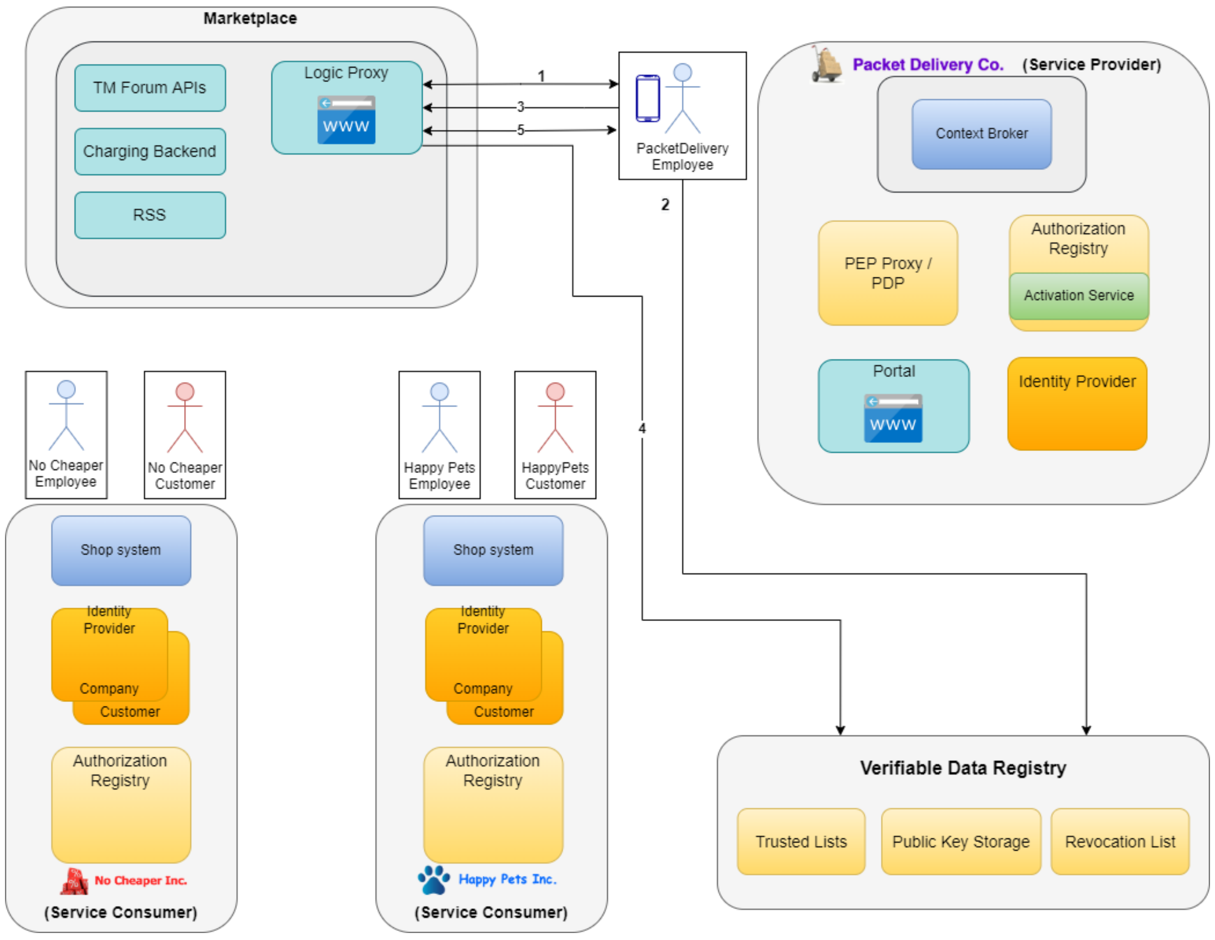


Figure 4.4.1.1a: Architecture diagram for step "Create Offering"

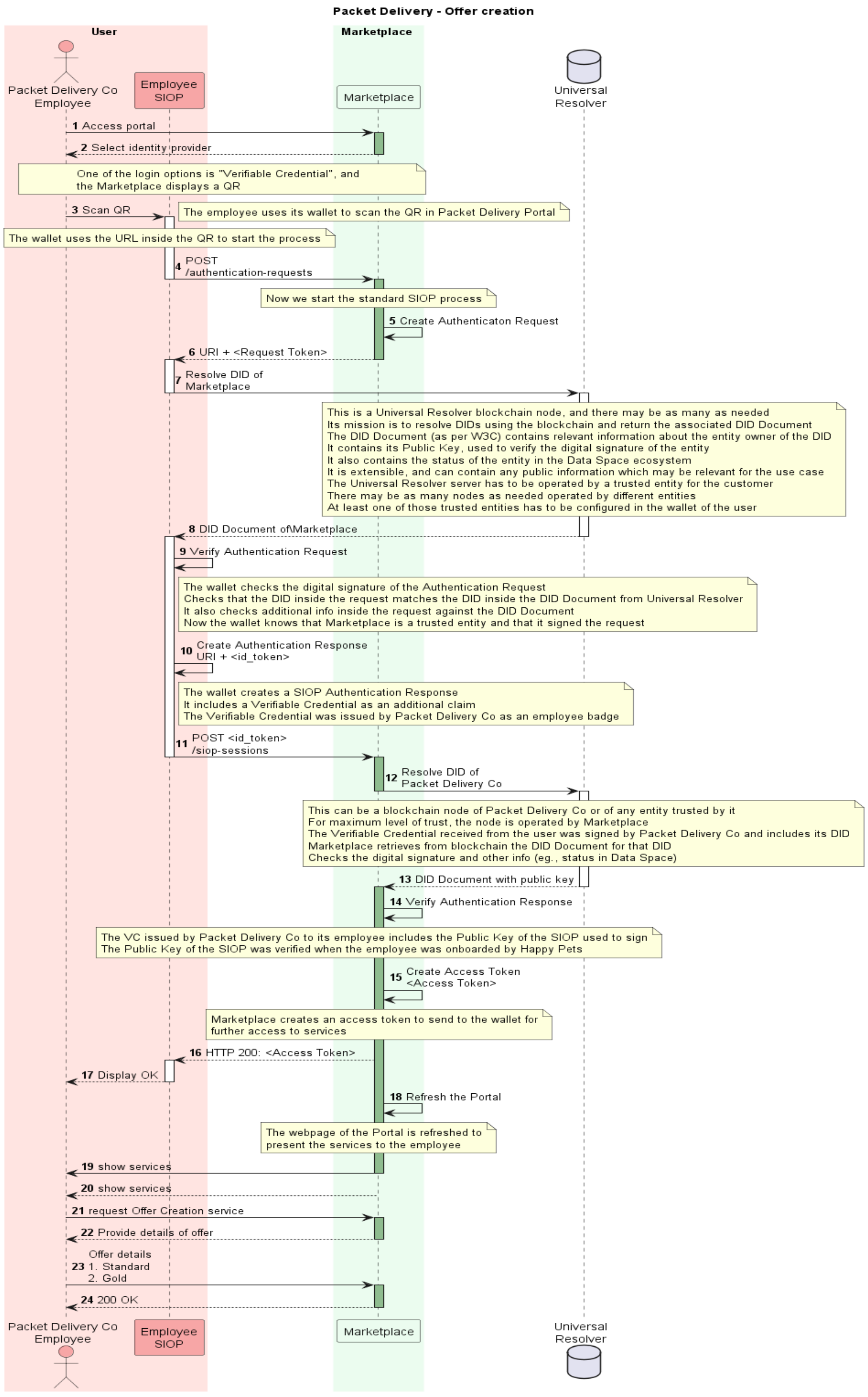

Packet Delivery - Offer creation
User
Marketplace
Packet Delivery Co Employee
Employee SIOP
Marketplace
Universal Resolver
1 Access portal
2 Select identity provider
One of the login options is "Verifiable Credential", and the Marketplace displays a QR
3 Scan QR
The employee uses its wallet to scan the QR in Packet Delivery Portal
The wallet uses the URL inside the QR to start the process
4 POST /authentication-requests
Now we start the standard SIOP process
5 Create Authenticaton Request
6 URI + <Request Token>
7 Resolve DID of Marketplace
This is a Universal Resolver blockchain node, and there may be as many as needed
Its mission is to resolve DIDs using the blockchain and return the associated DID Document
The DID Document (as per W3C) contains relevant information about the entity owner of the DID
It contains its Public Key, used to verify the digital signature of the entity
It also contains the status of the entity in the Data Space ecosystem
It is extensible, and can contain any public information which may be relevant for the use case
The Universal Resolver server has to be operated by a trusted entity for the customer
There may be as many nodes as needed operated by different entities
At least one of those trusted entities has to be configured in the wallet of the user
8 DID Document of\Marketplace
9 Verify Authentication Request
The wallet checks the digital signature of the Authentication Request
Checks that the DID inside the request matches the DID inside the DID Document from Universal Resolver
It also checks additional info inside the request against the DID Document
Now the wallet knows that Marketplace is a trusted entity and that it signed the request
10 Create Authentication Response URI + <id_token>
The wallet creates a SIOP Authentication Response
It includes a Verifiable Credential as an additional claim
The Verifiable Credential was issued by Packet Delivery Co as an employee badge
11 POST <id_token> /siop-sessions
12 Resolve DID of Packet Delivery Co
This can be a blockchain node of Packet Delivery Co or of any entity trusted by it
For maximum level of trust, the node is operated by Marketplace
The Verifiable Credential received from the user was signed by Packet Delivery Co and includes its DID
Marketplace retrieves from blockchain the DID Document for that DID
Checks the digital signature and other info (eg., status in Data Space)
13 DID Document with public key
14 Verify Authentication Response
The VC issued by Packet Delivery Co to its employee includes the Public Key of the SIOP used to sign
The Public Key of the SIOP was verified when the employee was onboarded by Happy Pets
15 Create Access Token <Access Token>
Marketplace creates an access token to send to the wallet for further access to services
16 HTTP 200: <Access Token>
17 Display OK
18 Refresh the Portal
The webpage of the Portal is refreshed to present the services to the employee
19 show services
20 show services
21 request Offer Creation service
22 Provide details of offer
23 Offer details 1. Standard 2. Gold
24 200 OK
Packet Delivery Co Employee
Employee SIOP
Marketplace
Universal Resolver

Figure 4.4.1.1b: Sequence diagram for step "Create Offering"

1. Packet Delivery employee accesses the Marketplace portal (provided by the BAE Logic Proxy), in order to login.
2. The Marketplace portal displays a list of Identity Providers for selecting the desired Identity Provider for login. One of the login options is "Login with Verifiable Credentials".
3. Packet Delivery Co employee selects the "Verifiable Credentials" login method, which causes the Marketplace portal to generate a QR containing the URL of the /authentication-requests endpoint of the Marketplace server.
4. The employee scans the QR with her mobile and the mobile calls the /authentication-requests endpoint.
5. This starts a standard SIOP (Self-Issued OpenID Provider) flow, where the Marketplace plays the role of Relying Party (RP in Open ID Connect terminology) and the mobile device of the employee as a Self-Issued IDP. In this step, Marketplace creates a SIOP Authentication Request. As a Self-Issued OP may be running as a native application or progressive web application (PWA), the RP may not have a network-addressable endpoint to communicate directly with the OP. We have to leverage the implicit flow of OpenID Connect to communicate with such locally-running OPs, as described in https://openid.net/specs/openid-connect-self-issued-v2-1_0.html.

   The Authentication Request travels in the response to the HTTP GET request performed in the previous point, as a JWT signed by Marketplace. The decoded contents of the JWT may be:

```
openid://?
   scope=openid
   &response_type=id_token
   &response_mode=post
   &client_id=did:elsi:EU.EORI.NLMARKETPLA
   &redirect_uri=https://marketplace.fiware.io/siop_sessions
   &claims=... //the Marketplace would specify here what type of claims it wants
the employee to provide.  Those claims should be connected to roles of users in
the application, documented in the marketplace
   &registration={
      "subject_syntax_types_supported": ["did:key",
      "urn:ietf:params:oauth:jwk-thumbprint"]
   }
```

```
&nonce=n-0S6_WzA2Mj
```

6. The Authentication Request is returned to the employee wallet acting as SIOP. The SIOP flow uses a new response mode **post** which is used to request the SIOP to deliver the result of the authentication process to a certain endpoint. The parameter **response_mode** is used to carry this value.

   This endpoint where the SIOP shall deliver the authentication result is defined in the standard parameter **redirect_uri**.

7. In this step the employee verifies that the Marketplace is a trusted entity belonging to the ecosystem, by resolving the DID of the Marketplace which is received in the **client_id** parameter of the Authentication Request.

   To resolve a DID, the wallet sends a GET request to the **/api/did/v1/identifiers/did:elsi:EU.EORI.NLMARKETPLA** endpoint of one of several trusted servers implementing the Universal Resolver functionality. The Universal Resolver includes a blockchain node, and there may be as many as needed. Its mission is to resolve DIDs using the blockchain and return the associated DID Document. The DID Document (as per W3C) contains relevant information about the entity owner of the DID. It contains its Public Key, used to verify the digital signature of the entity. It also contains the status of the entity in the Data Space ecosystem. It is extensible and can contain any public information which may be relevant for the use case. The Universal Resolver server must be operated by a trusted entity for the customer. There may be as many nodes as needed operated by different entities. At least one of those trusted entities has to be configured in the wallet of the employee.

8. The wallet receives the DID Document of Marketplace, with trusted information about the entity, including the Public Key associated with the Private Key that Marketplace uses to digitally sign tokens. For example:

```
{
  "payload": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/v1"
    ],
    "id": "did:elsi:EU.EORI.NLMARKETPLA",
    "verificationMethod": [
      {
        "id": "did:elsi:EU.EORI.NLMARKETPLA#key-verification",
        "type": "JwsVerificationKey2020",
        "controller": "did:elsi:EU.EORI.NLMARKETPLA",
        "publicKeyJwk": {
          "kid": "key-verification",
          "kty": "EC",
          "crv": "secp256k1",
          "x": "V8XptJkb5wplYkExcTF4nkyYVp7t5H5d5C4UPqCCM9c",
          "y": "kn3nSPxIIvd9iaG0N4v14ceuo8E4PcLXhhGeDzCE7VM"
        }
      }
    ],
    "service": [
      {
        "id": "did:elsi:EU.EORI.NLMARKETPLA#info",
        "type": "EntityCommercialInfo",
        "serviceEndpoint": "https://marketplace.fiware.io/info",
        "name": "Packet Delivery co."
      },
      {
        "id": "did:elsi:EU.EORI.NLMARKETPLA#sms",
        "type": "SecureMessagingService",
        "serviceEndpoint": "https://marketplace.fiware.io/api/sms"
      }
    ],
    "anchors": [
      {
        "id": "redt.alastria",
        "resolution": "UniversalResolver",
        "domain": "marketplace.dataspace",
        "ethereumAddress": "0xbcB9b29eeb28f36fd84f1CfF98C3F1887D831d78"
      }
    ],
    "created": "2021-11-14T13:02:37Z",
    "updated": "2021-11-14T13:02:37Z"
  }
}
```

9. The DID Document includes one or more public keys inside the “verificationMethod” array. The keys are identified by the “id” field in each element of the array. The employee wallet uses the **kid** field that was received in the Authentication Request (in the protected header of the JWT) to select the corresponding Public Key and verify the signature of the JWT. It also verifies that the top-level “**id**” field in the DID Document ("did:elsi:EU.EORI.NLMARKETPLA") is equal to the **client_id** parameter of the Authentication Request.

10. The employee wallet creates an Authentication Response to be posted in the **redirect_uri** specified by Marketplace in step 5. The contents of the Authentication Response are described below.
11. The SIOP sends the authentication response to the endpoint passed in the **redirect_uri** authentication request parameter using a HTTP POST request using "application/x-www-form-urlencoded" encoding. The response contains an ID Token and a VP (Verifiable Presentation) token as defined in OpenID for Verifiable Presentations.

```
POST /siop_sessions HTTP/1.1

Host: marketplace.fiware.io

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


id_token=eyJ0 ... NiJ9.eyJ1c ... I6IjIifX0.DeWt4Qu ... ZXso

&vp_token=...

&state=af0ifjsldkj
```

The decoded **id_token** would be:

```
{
  "iss": "https://self-issued.me/v2",
  "aud": "did:elsi:EU.EORI.NLMARKETPLA",
  "iat": 1615910538,
  "exp": 1615911138,
  "sub": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
  "auth_time": 1615910535,
  "nonce": "n-0S6_WzA2Mj"
}
```

The **sub** claim is *did:peer:99ab5bca41bb45b78d242a46f0157b7d* which is the DID of the user and for privacy reasons it is not registered in any blockchain or centralized repository. It must be the same as the DID included in the Verifiable Credential that was issued by the Packet Delivery company when onboarding the employee and which travels in the authentication response.

The **vp_token** includes the Verifiable Presentation, which can be in two formats: **jwt_vp** (JWT encoded) or **ldp_vp** (JSON-LD encoded). The following example is using the JWT encoding:

```
{
   "format": "jwt_vp",
   "presentation":
   "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImRpZDpleGFtcGxlOmFiZmUxM2Y3MTIxMjA0
   MzFjMjc2ZTEyZWNhYiNrZXlzLTEifQ.eyJzdWIiOiJkaWQ6ZXhhbXBsZTplYmZlYjFmNzEyZWJjNmYxY
   zI3NmUxMmVjMjEiLCJqdGkiOiJodHRwOi8vZXhhbXBsZS5lZHUvY3JlZGVudGlhbHMvMzczMiIsImlzc
   yI6Imh0dHBzOi8vZXhhbXBsZS5jb20va2V5cy9mb28uandrIiwibmJmIjoxNTQxNDkzNzI0LCJpYXQiO
   jE1NDE0OTM3MjQsImV4cCI6MTU3MzAyOTcyMywibm9uY2UiOiI2NjAhNjM0NUZTZXIiLCJ2YyI6eyJAY
   29udGV4dCI6WyJodHRwczovL3d3dy53My5vcmcvMjAxOC9jcmVkZW50aWFscy92MSIsImh0dHBzOi8vd
   3d3LnczLm9yZy8yMDE4L2NyZWRlbnRpYWxzL2V4YW1wbGVzL3YxIl0sInR5cGUiOlsiVmVyaWZpYWJsZ
   UNyZWRlbnRpYWwiLCJVbml2ZXJzaXR5RGVncmVlQ3JlZGVudGlhbCJdLCJjcmVkZW50aWFsU3ViamVjd
   CI6eyJkZWdyZWUiOnsidHlwZSI6IkJhY2hlbG9yRGVncmVlIiwibmFtZSI6IjxzcGFuIGxhbmc9J2ZyL
   UNBJz5CYWNjYWxhdXLDqWF0IGVuIG11c2lxdWVzIG51bcOpcmlxdWVzPC9zcGFuPiJ9fX19.KLJo5GAy
   BND3LDTn9H7FQokEsUEi8jKwXhGvoN3JtRa51xrNDgXDb0cq1UTYB-rK4Ft9YVmR1NI_ZOF8oGc_7wAp
   8PHbF2HaWodQIoOBxxT-4WNqAxft7ET6lkH-4S6Ux3rSGAmczMohEEf8eCeN-jC8WekdPl6zKZQj0YPB
   1rx6X0-xlFBs7cl6Wt8rfBP_tZ9YgVWrQmUWypSioc0MUyiphmyEbLZagTyPlUyflGlEdqrZAv6eSe6R
   txJy6M1-lD7a5HTzanYTWBPAUHDZGyGKXdJw-W_x0IWChBzI8t3kpG253fg6V3tPgHeKXE94fz_QpYfg
   --7kLsyBAfQGbg"
}
```

Which decoded would be:

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiablePresentation"],
  "verifiableCredential": [
    {
      "@context": [
        "https://www.w3.org/2018/credentials/v1",
        "https://marketplace.fiware.io/2022/credentials/employee/v1"
      ],
      "id": "https://pdc.fiware.io/credentials/6e14b8b8-87fa0014fe2a",
      "type": ["VerifiableCredential", "EmployeeCredential"],
      "issuer": {
        "id": "did:elsi:EU.EORI.NLPACKETDEL"
      },
      "issuanceDate": "2022-03-22T14:00:00Z",
      "validFrom": "2022-03-22T14:00:00Z",
      "expirationDate": "2023-03-22T14:00:00Z",
      "credentialSubject": {
        "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
        "verificationMethod": [
          {
            "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
            "type": "JwsVerificationKey2020",
            "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
            "publicKeyJwk": {
              "kid": "key1",
              "kty": "EC",
              "crv": "P-256",
              "x": "lJtvoA5_XptBvcfcrvtGCvXd9bLymmfBSSdNJf5mogo",
              "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
            }
          }
        ],
        "roles": [
          {
            "target": "did:elsi:EU.EORI.NLMARKETPLA",
            "names": ["seller", "buyer"]
          }
        ],
        "name": "Jane Doe",
        "given_name": "Jane",
        "family_name": "Doe",
        "preferred_username": "j.doe",
        "email": "janedoe@packetdelivery.com"
      }
    }
  ]
}
```

12. Marketplace uses its own blockchain node or the one from a trusted entity implementing the Universal Resolver functionality to resolve the DID of Packet Delivery Co, which is inside the Verifiable Credential received in the Verifiable Presentation. This DID can be found in the "issuer" field of the "verifiableCredential" structure above.

    Resolution is performed sending a GET request to the Universal Resolver: **/api/did/v1/identifiers/did:elsi:EU.EORI.NLPACKETDEL**

    Marketplace could use a Universal Resolver operated by a different entity, but this would reduce the level of trust compared to using its own server directly connected to the blockchain network.

13. Marketplace receives the DID Document of Packet Delivery Co with trusted information about the company, including the Public Key associated with the Private Key that Packet Delivery Co used to digitally sign the Verifiable Credential that the employee has just sent inside a Verifiable Presentation as part of the authentication flow. **Using the Public Key and the DID inside the DID Document, it can verify the signature of the Verifiable Credential and that Packet Delivery Co is a trusted entity in the ecosystem and that it is active.**

14. The above is just for verification of the Verifiable Credential. In addition, Marketplace can also verify that the Verifiable Presentation including the Verifiable Credential is sent by the employee and not by a malicious agent. To do so, it uses the Public Key of the employee in the "verificationMethod" of the "credentialSubject" structure. That public key is cryptographically bound to the employee DID during the onboarding process that Packet Delivery Co performed with its employee.

15. Once all verifications have been performed, Marketplace creates an Access Token for the employee so she can use it to access services in the Marketplace server in the future.
16. The wallet (SIOP) receives the access token and saves it temporarily to be able to request services from Marketplace.
17. The wallet displays a success message to the employee.
18. The Marketplace server refreshes the page (it was the login page before) and displays the services available to the employee of Packet Delivery Co.

At this point the Packet Delivery Co employee is logged in on the Marketplace application. The user is now able to create catalogues, products and offerings.

At this moment, the Marketplace knows the following:

- That Packet Delivery Co belongs to the Data Space and can issue credentials of the type EmployeeCredential because it is included in the Trusted Issuers List and is active, because this info is in the DID Document retrieved in step 13.
- That Packet Delivery Co says that the user is one of its employees. This info is inside the Verifiable Credential that is digitally signed by Packet Delivery Co.

From this point on, the Marketplace can display to the user the services available to her and execute them if the user is entitled to do so. The Marketplace can use all the claims inside the credential to perform RBAC/ABAC access control and policy enforcement.

4.4.2 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.

4.4.2.1 Sequence description (Happy Pets Inc.)

The flows are exactly the same as the ones described in 2.3.4 Acquisition of Rights / Activation with the exception that the initial authentication of the Happy Pets or No Cheaper employees with regard to the Marketplace is performed using a Verifiable Credential issued to its employees by those companies. In the same way as described in 2.4.3 Create Offering, the policies delegated to its employees by Happy Pets and No Cheaper are embedded into the Verifiable Credentials. In this sense, the flows for authentication of the employees for Acquisition of Rights / Activation are exactly the same as described in 2.4.3.1 Sequence description (Packet Delivery Co.) with the appropriate company name changes and specific content of the policies.

For those reasons, we describe only the authentication process which replaces steps 1-14 in 2.3.4.1 Sequence description (Happy Pets Inc.) with steps 1-19 in the sequence diagram below.

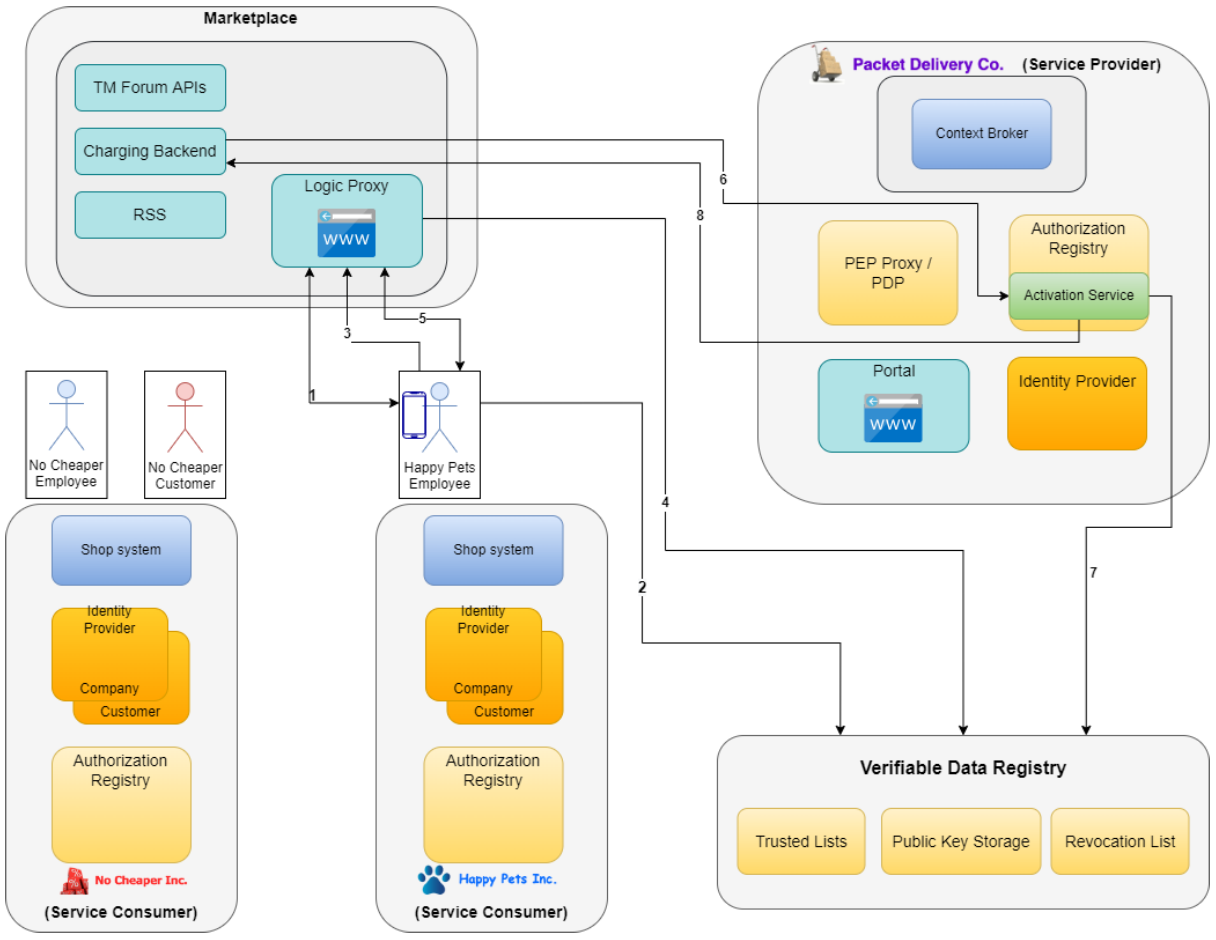
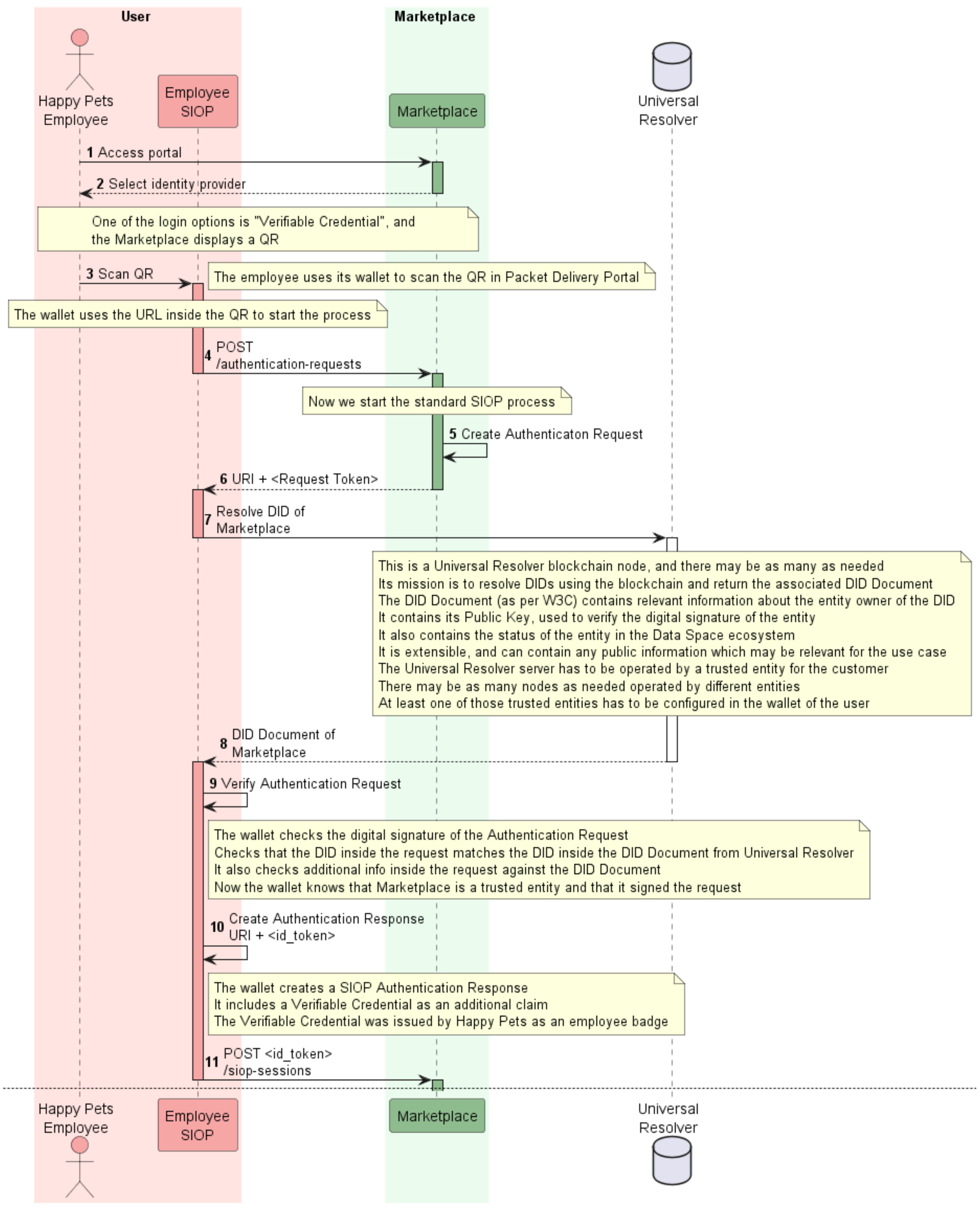


Figure 4.4.2.1a: Sequence diagram for step "Acquisition of Rights / Activation"

Packet Delivery - Acquisition of rights


User
Marketplace
Happy Pets Employee
Employee SIOP
Marketplace
Universal Resolver
1 Access portal
2 Select identity provider
3 Scan QR
4 POST /authentication-requests
5 Create Authenticaton Request
6 URI + <Request Token>
7 Resolve DID of Marketplace
8 DID Document of Marketplace
9 Verify Authentication Request
10 Create Authentication Response URI + <id_token>
11 POST <id_token> /siop-sessions
Happy Pets Employee
Employee SIOP
Marketplace
Universal Resolver

One of the login options is "Verifiable Credential", and
the Marketplace displays a QR

The employee uses its wallet to scan the QR in Packet Delivery Portal

The wallet uses the URL inside the QR to start the process

Now we start the standard SIOP process

This is a Universal Resolver blockchain node, and there may be as many as needed
Its mission is to resolve DIDs using the blockchain and return the associated DID Document
The DID Document (as per W3C) contains relevant information about the entity owner of the DID
It contains its Public Key, used to verify the digital signature of the entity
It also contains the status of the entity in the Data Space ecosystem
It is extensible, and can contain any public information which may be relevant for the use case
The Universal Resolver server has to be operated by a trusted entity for the customer
There may be as many nodes as needed operated by different entities
At least one of those trusted entities has to be configured in the wallet of the user

The wallet checks the digital signature of the Authentication Request
Checks that the DID inside the request matches the DID inside the DID Document from Universal Resolver
It also checks additional info inside the request against the DID Document
Now the wallet knows that Marketplace is a trusted entity and that it signed the request

The wallet creates a SIOP Authentication Response
It includes a Verifiable Credential as an additional claim
The Verifiable Credential was issued by Happy Pets as an employee badge

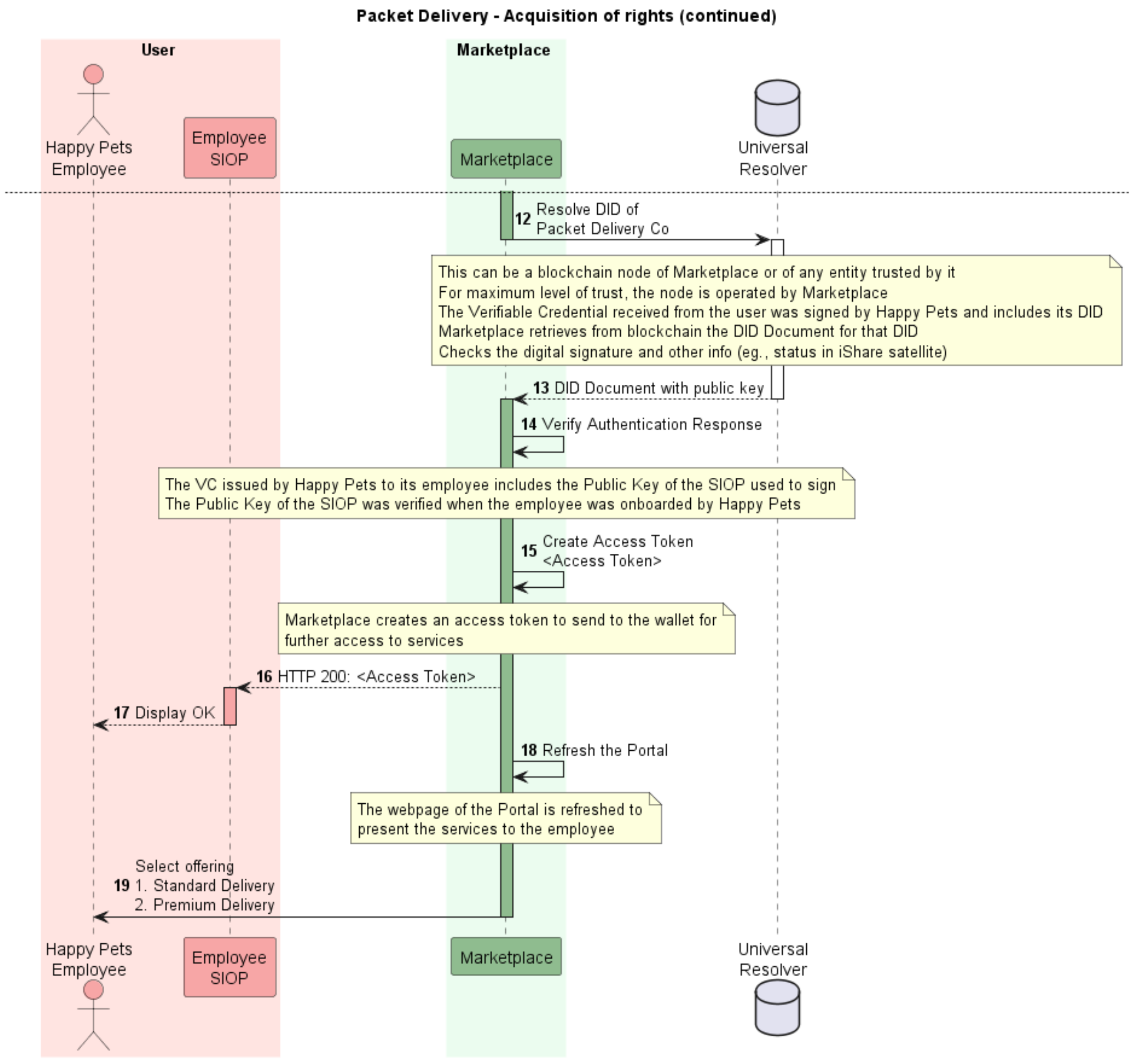

Packet Delivery - Acquisition of rights (continued)
User
Marketplace
Happy Pets Employee
Employee SIOP
Marketplace
Universal Resolver
12 Resolve DID of Packet Delivery Co
This can be a blockchain node of Marketplace or of any entity trusted by it
For maximum level of trust, the node is operated by Marketplace
The Verifiable Credential received from the user was signed by Happy Pets and includes its DID
Marketplace retrieves from blockchain the DID Document for that DID
Checks the digital signature and other info (eg., status in iShare satellite)
13 DID Document with public key
14 Verify Authentication Response
The VC issued by Happy Pets to its employee includes the Public Key of the SIOP used to sign
The Public Key of the SIOP was verified when the employee was onboarded by Happy Pets
15 Create Access Token <Access Token>
Marketplace creates an access token to send to the wallet for further access to services
16 HTTP 200: <Access Token>
17 Display OK
18 Refresh the Portal
The webpage of the Portal is refreshed to present the services to the employee
19 Select offering 1. Standard Delivery 2. Premium Delivery

Figure 4.4.2.1b: Sequence diagram for step "Acquisition of Rights / Activation"

1. The Happy Pets employee accesses the Marketplace portal (provided by the BAE Logic Proxy), in order to login.
2. Happy Pets employee is displayed a list of Identity Providers for selecting the desired Identity Provider for login. Happy Pets employee gets forwarded to a page for selecting the desired Identity Provider for login. One of the login options is "Verifiable Credentials" or something similar.
3. Happy Pets employee selects the "Verifiable Credentials" login method, which causes the Marketplace portal to generate a QR containing the URL of the /authentication-requests endpoint of the Marketplace server.
4. The employee scans the QR with her mobile and the mobile calls the /authentication-requests endpoint.

5. This starts a standard SIOP (Self-Issued OpenID Provider) flow, where the Marketplace IDP plays the role of Relying Party (RP in Open ID Connect terminology) and the mobile device of the employee as a Self-Issued IDP. In this step, Marketplace IDP creates a SIOP Authentication Request. As a Self-Issued OP may be running locally as a native application or progressive web application (PWA), the RP may not have a network-addressable endpoint to communicate directly with the OP. We have to leverage the implicit flow of OpenID Connect to communicate with such locally-running OPs, as described in https://openid.net/specs/openid-connect-self-issued-v2-1_0.html.

   The Authentication Request travels in the response to the HTTP GET request performed in the previous point, as a JWT signed by Packet Delivery company. The decoded contents of the JWT may be:

```
openid://?
   scope=openid
   &response_type=id_token
   &response_mode=post
   &client_id=did:elsi:EU.EORI.NLMARKETPLA
   &redirect_uri=https://marketplace.fiware.io/siop_sessions
   &claims=...
   &registration={
      "subject_syntax_types_supported": ["did:key",
      "urn:ietf:params:oauth:jwk-thumbprint"]
   }
   &nonce=n-0S6_WzA2Mj
```

6. The Authentication Request is returned to the employee wallet acting as SIOP. The SIOP flow uses a new response mode **post** which is used to request the SIOP to deliver the result of the authentication process to a certain endpoint. The parameter **response_mode** is used to carry this value.

   This endpoint where the SIOP shall deliver the authentication result is defined in the standard parameter **redirect_uri**.

7. In this step the employee verifies that the Marketplace is a trusted entity belonging to the ecosystem, by resolving the DID of the Marketplace which is received in the **client_id** parameter of the Authentication Request.

   To resolve a DID, the wallet sends a GET request to the **/api/did/v1/identifiers/did:elsi:EU.EORI.NLMARKETPLA** endpoint of one of several trusted servers implementing the Universal Resolver functionality. The Universal Resolver includes a blockchain node, and there may be as many as needed. Its mission is to resolve DIDs using the blockchain and return the associated DID Document. The DID Document (as per W3C) contains relevant information about the entity owner of the DID. It contains its Public Key, used to verify the digital signature of the entity. It also contains the status of the entity in the Data Space ecosystem. It is extensible and can contain any public information which may be relevant for the use case. The Universal Resolver server must be operated by a trusted entity for the customer. There may be as many nodes as needed operated by different entities. At least one of those trusted entities has to be configured in the wallet of the employee.

8. The wallet receives the DID Document of Marketplace, with trusted information about the entity, including the Public Key associated with the Private Key that Marketplace uses to digitally sign tokens. For example:

```
{
  "payload": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/v1"
    ],
    "id": "did:elsi:EU.EORI.NLMARKETPLA",
    "verificationMethod": [
      {
        "id": "did:elsi:EU.EORI.NLMARKETPLA#key-verification",
        "type": "JwsVerificationKey2020",
        "controller": "did:elsi:EU.EORI.NLMARKETPLA",
        "publicKeyJwk": {
          "kid": "key-verification",
          "kty": "EC",
          "crv": "secp256k1",
          "x": "V8XptJkb5wplYkExcTF4nkyYVp7t5H5d5C4UPqCCM9c",
          "y": "kn3nSPxIIvd9iaG0N4v14ceuo8E4PcLXhhGeDzCE7VM"
        }
      }
    ],
    "service": [
      {
        "id": "did:elsi:EU.EORI.NLMARKETPLA#info",
        "type": "EntityCommercialInfo",
        "serviceEndpoint": "https://marketplace.fiware.io/info",
        "name": "Packet Delivery co."
      },
      {
        "id": "did:elsi:EU.EORI.NLMARKETPLA#sms",
        "type": "SecureMessagingService",
        "serviceEndpoint": "https://marketplace.fiware.io/api/sms"
      }
    ],
    "anchors": [
      {
        "id": "redt.alastria",
        "resolution": "UniversalResolver",
        "domain": "marketplace.dataspace",
        "ethereumAddress": "0xbcB9b29eeb28f36fd84f1CfF98C3F1887D831d78"
      }
    ],
    "created": "2021-11-14T13:02:37Z",
    "updated": "2021-11-14T13:02:37Z"
  }
}
```

9. The DID Document includes one or more public keys inside the “verificationMethod” array. The keys are identified by the “id” field in each element of the array. The employee wallet uses the **kid** field that was received in the Authentication Request (in the protected header of the JWT) to select the corresponding Public Key and verify the signature of the JWT. It also verifies that the top-level “**id**” field in the DID Document ("did:elsi:EU.EORI.NLMARKETPLA") is equal to the **client_id** parameter of the Authentication Request.
10. The employee wallet creates an Authentication Response to be posted in the **redirect_uri** specified by Marketplace in step 5. The contents of the Authentication Response are described below.
11. The SIOP sends the authentication response to the endpoint passed in the **redirect_uri** authentication request parameter using a HTTP POST request using "application/x-www-form-urlencoded" encoding. The response contains an ID Token and a VP (Verifiable Presentation) token as defined in https://openid.net/specs/openid-connect-4-verifiable-presentations-1_0.html.

```
POST /siop_sessions HTTP/1.1
Host: marketplace.fiware.io
Content-Type: application/x-www-form-urlencoded

id_token=eyJ0 ... NiJ9.eyJ1c ... I6IjIifX0.DeWt4Qu ... ZXso
&vp_token=...
&state=af0ifjsldkj
```

The decoded **id_token** would be:

```
{
   "iss":"https://self-issued.me/v2",
   "aud":"did:elsi:EU.EORI.NLMARKETPLA",
   "iat":1615910538,
   "exp":1615911138,
   "sub":"did:peer:99ab5bca41bb45b78d242a46f0157b7d",
   "auth_time":1615910535,
    "nonce":"n-0S6_WzA2Mj",
}
```

The **sub** claim is *did:peer:99ab5bca41bb45b78d242a46f0157b7d* which is the DID of the user and for privacy reasons it is not registered in any blockchain or centralized repository. It must be the same as the DID included in the Verifiable Credential that was issued by the Happy Pets company when onboarding the employee and which travels in the authentication response.

The **vp_token** includes the Verifiable Presentation, which can be in two formats: **jwt_vp** (JWT encoded) or **ldp_vp** (JSON-LD encoded). The following example is using the JWT encoding:

```
{
    "format":"jwt_vp",
    "presentation":
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImRpZDpleGFtcGxlOmFiZmUxM2Y3MTIxMjA0
    MzFjMjc2ZTEyZWNhYiNrZXlzLTEifQ.eyJzdWIiOiJkaWQ6ZXhhbXBsZTplYmZlYjFmNzEyZWJjNmYxY
    zI3NmUxMmVjMjEiLCJqdGkiOiJodHRwOi8vZXhhbXBsZS5lZHUvY3JlZGVudGlhbHMvMzczMiIsImlzc
    yI6Imh0dHBzOi8vZXhhbXBsZS5jb20va2V5cy9mb28uandrIiwibmJmIjoxNTQxNDkzNzI0LCJpYXQiO
    jE1NDE0OTM3MjQsImV4cCI6MTU3MzAyOTcyMywibm9uY2UiOiI2NjAhNjM0NUZTZXIiLCJ2YyI6eyJAY
    29udGV4dCI6WyJodHRwczovL3d3dy53My5vcmcvMjAxOC9jcmVkZW50aWFscy92MSIsImh0dHBzOi8vd
    3d3LnczLm9yZy8yMDE4L2NyZWRlbnRpYWxzL2V4YW1wbGVzL3YxIl0sInR5cGUiOlsiVmVyaWZpYWJsZ
    UNyZWRlbnRpYWwiLCJVbml2ZXJzaXR5RGVncmVlQ3JlZGVudGlhbCJdLCJjcmVkZW50aWFsU3ViamVjd
    CI6eyJkZWdyZWUiOnsidHlwZSI6IkJhY2hlbG9yRGVncmVlIiwibmFtZSI6IjxzcGFuIGxhbmc9J2ZyL
    UNBJz5CYWNjYWxhdXLDqWF0IGVuIG11c2lxdWVzIG51bcOpcmlxdWVzPC9zcGFuPiJ9fX19.KLJo5GAy
    BND3LDTn9H7FQokEsUEi8jKwXhGvoN3JtRa51xrNDgXDb0cq1UTYB-rK4Ft9YVmR1NI_ZOF8oGc_7wAp
    8PHbF2HaWodQIoOBxxT-4WNqAxft7ET6lkH-4S6Ux3rSGAmczMohEEf8eCeN-jC8WekdPl6zKZQj0YPB
    1rx6X0-xlFBs7cl6Wt8rfBP_tZ9YgVWrQmUWypSioc0MUyiphmyEbLZagTyPlUyflGlEdqrZAv6eSe6R
    txJy6M1-lD7a5HTzanYTWBPAUHDZGyGKXdJw-W_x0IWChBzI8t3kpG253fg6V3tPgHeKXE94fz_QpYfg
    --7kLsyBAfQGbg"
}
```

Which decoded could be:

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiablePresentation"],
  "verifiableCredential": [
    {
      "@context": [
        "https://www.w3.org/2018/credentials/v1",
        "https://happypets.fiware.io/2022/credentials/employee/v1"
      ],
      "id": "https://happypets.fiware.io/credentials/25159389-8dd17b796ac0",
      "type": ["VerifiableCredential", "EmployeeCredential"],
      "issuer": {
        "id": "did:elsi:EU.EORI.NLHAPPYPETS"
      },
      "issuanceDate": "2022-03-22T14:00:00Z",
      "validFrom": "2022-03-22T14:00:00Z",
      "expirationDate": "2023-03-22T14:00:00Z",
      "credentialSubject": {
        "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
        "verificationMethod": [
          {
            "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
            "type": "JwsVerificationKey2020",
            "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
            "publicKeyJwk": {
              "kid": "key1",
              "kty": "EC",
              "crv": "P-256",
              "x": "lJtvoA5_XptBvcfcrvtGCvXd9bLymmfBSSdNJf5mogo",
              "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
            }
          }
        ],
        "roles": [
          {
            "target": "did:elsi:EU.EORI.NLPACKETDEL",
            "names": ["P.Create"]
          }
        ],
        "name": "Jane Doe",
        "given_name": "Jane",
        "family_name": "Doe",
        "preferred_username": "j.doe",
        "email": "janedoe@packetdelivery.com"
      }
    }
  ]
}
```

12. Marketplace uses its own blockchain node or the one from a trusted entity implementing the Universal Resolver functionality to resolve the DID of Happy Pets, which is inside the Verifiable Credential received in the Verifiable Presentation. This DID can be found in the “issuer” field of the “verifiableCredential” structure above.

    Resolution is performed sending a GET request to the Universal Resolver: **/api/did/v1/identifiers/did:elsi:EU.EORI.NLHAPPYPETS**

Marketplace could use a Universal Resolver operated by a different entity, but this would reduce the level of trust compared to using its own server directly connected to the blockchain network.

13. Marketplace receives the DID Document of Happy Pets with trusted information about the company, including the Public Key associated with the Private Key that Happy Pets used to digitally sign the Verifiable Credential that the employee has just sent inside a Verifiable Presentation as part of the authentication flow. **Using the Public Key and the DID inside the DID Document, it can verify the signature of the Verifiable Credential and that Happy Pets is a trusted entity in the ecosystem and that it is active.**
14. The above is just for verification of the Verifiable Credential. In addition, Marketplace can also verify that the Verifiable Presentation including the Verifiable Credential is sent by the employee and not by a malicious agent. To do so, it uses the Public Key of the employee in the “verificationMethod” of the “credentialSubject” structure. That public key is cryptographically bound to the employee DID during the onboarding process that Happy Pets performed with its employee.
15. Once all verifications have been performed, Marketplace creates an Access Token for the employee so she can use it to access services in the Marketplace server in the future.
16. The wallet (SIOP) receives the access token and saves it temporarily to be able to request services from Marketplace.
17. The wallet displays a success message to the employee.
18. The Marketplace server refreshes the page (it was the login page before) and displays the services available to the employee of Packet Delivery Co.

At this point the Happy Pets employee is logged in on the Marketplace application. The user is now able to use the services available to her.

At this moment, the Marketplace knows the following:

- That Happy Pets belongs to the Data Space and can issue credentials of the type EmployeeCredential because it is included in the corresponding Trusted Issuers List and is active, because this info is in the DID Document retrieved in step 13. Maintenance of this information is performed by the Trust Anchor entity (or entities) responsible for the Trusted Issuers List.
- That Happy Pets says that the user is one of its employees. This info is inside the Verifiable Credential that is digitally signed by Happy Pets.

From this point on, the Marketplace can display to the user the services available to her and execute them if the user is entitled to do so. The Marketplace can use all the claims inside the credential to perform RBAC/ABAC access control and policy enforcement.

4.4.2.2 Sequence description (No Cheaper Ltd)

The process is exactly the same as for the acquisition process for Happy Pets, except that the entities involved are No Cheaper Ltd and its employees. We do not provide a detailed flow to avoid repetition.

4.4.3 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.

4.4.3.1 Sequence description (Happy Pets Customer)

The following gives a detailed description of the process of changing the PTA attribute by the Happy Pets customer, when using Verifiable Credentials. Figure 4.4.3.1b 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.

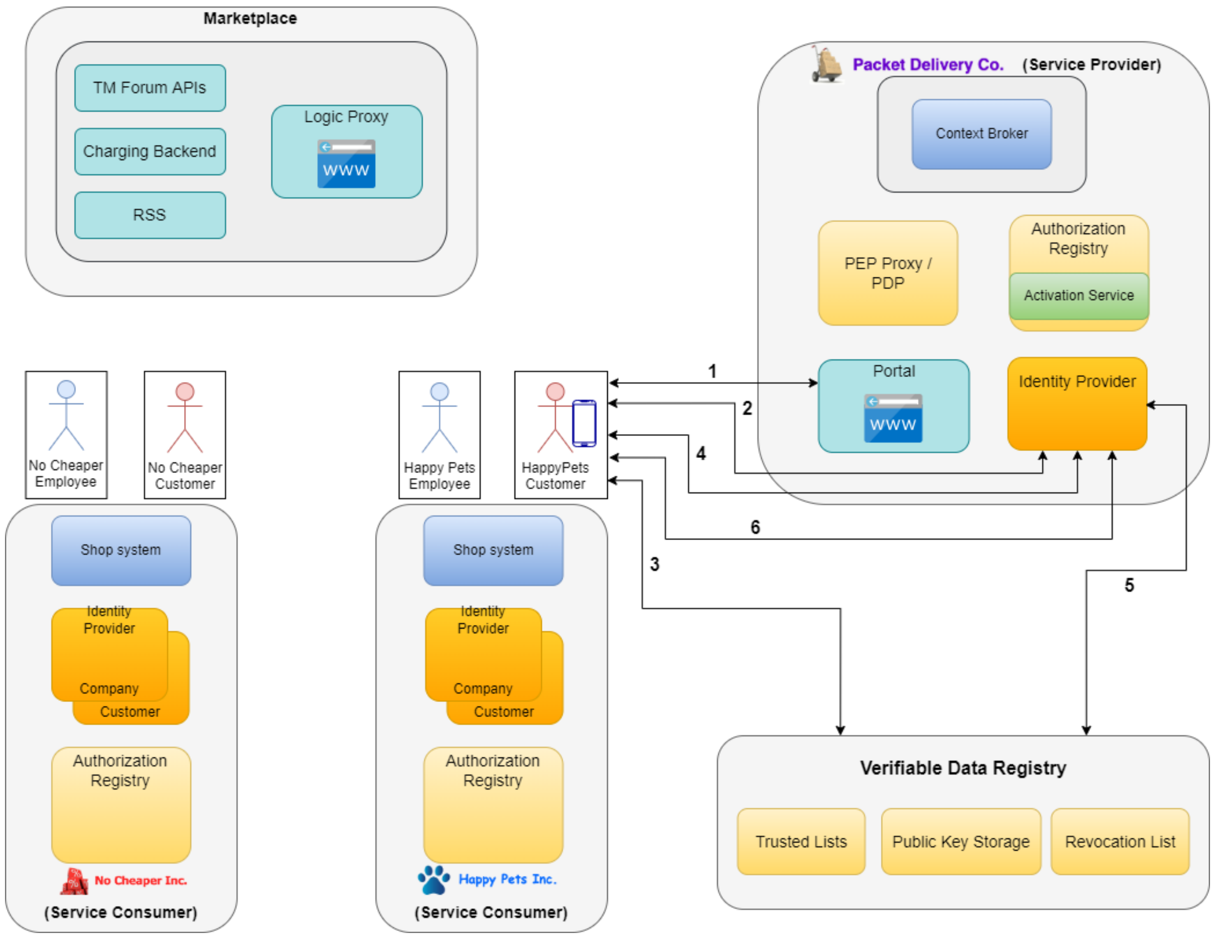
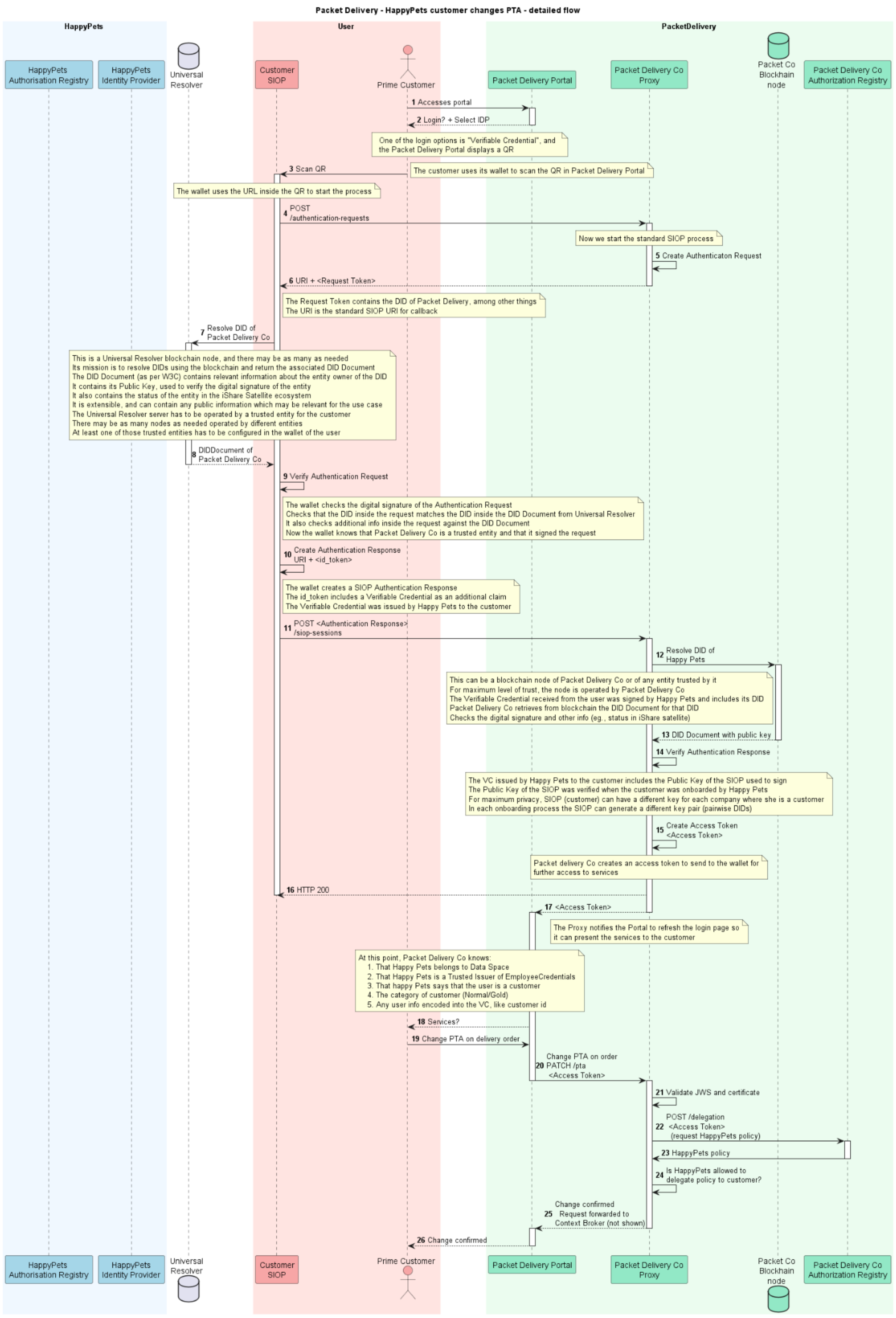


Figure 4.4.3.1a: Architecture diagram for step "Change PTA by Happy Pets customer"


Packet Delivery - HappyPets customer changes PTA - detailed flow
HappyPets
User
PacketDelivery
HappyPets Authorisation Registry
HappyPets Identity Provider
Universal Resolver
Customer SIOP
Prime Customer
Packet Delivery Portal
Packet Delivery Co Proxy
Packet Co Blockhain node
Packet Delivery Co Authorization Registry

1 Accesses portal

2 Login? + Select IDP

One of the login options is "Verifiable Credential", and the Packet Delivery Portal displays a QR

3 Scan QR

The customer uses its wallet to scan the QR in Packet Delivery Portal

The wallet uses the URL inside the QR to start the process

4 POST /authentication-requests

Now we start the standard SIOP process

5 Create Authenticaton Request

6 URI + <Request Token>

The Request Token contains the DID of Packet Delivery, among other things
The URI is the standard SIOP URI for callback

7 Resolve DID of Packet Delivery Co

This is a Universal Resolver blockchain node, and there may be as many as needed
Its mission is to resolve DIDs using the blockchain and return the associated DID Document
The DID Document (as per W3C) contains relevant information about the entity owner of the DID
It contains its Public Key, used to verify the digital signature of the entity
It also contains the status of the entity in the iShare Satellite ecosystem
It is extensible, and can contain any public information which may be relevant for the use case
The Universal Resolver server has to be operated by a trusted entity for the customer
There may be as many nodes as needed operated by different entities
At least one of those trusted entities has to be configured in the wallet of the user

8 DIDDocument of Packet Delivery Co

9 Verify Authentication Request

The wallet checks the digital signature of the Authentication Request
Checks that the DID inside the request matches the DID inside the DID Document from Universal Resolver
It also checks additional info inside the request against the DID Document
Now the wallet knows that Packet Delivery Co is a trusted entity and that it signed the request

10 Create Authentication Response URI + <id_token>

The wallet creates a SIOP Authentication Response
The id_token includes a Verifiable Credential as an additional claim
The Verifiable Credential was issued by Happy Pets to the customer

11 POST <Authentication Response> /siop-sessions

12 Resolve DID of Happy Pets

This can be a blockchain node of Packet Delivery Co or of any entity trusted by it
For maximum level of trust, the node is operated by Packet Delivery Co
The Verifiable Credential received from the user was signed by Happy Pets and includes its DID
Packet Delivery Co retrieves from blockchain the DID Document for that DID
Checks the digital signature and other info (eg., status in iShare satellite)

13 DID Document with public key

14 Verify Authentication Response

The VC issued by Happy Pets to the customer includes the Public Key of the SIOP used to sign
The Public Key of the SIOP was verified when the customer was onboarded by Happy Pets
For maximum privacy, SIOP (customer) can have a different key for each company where she is a customer
In each onboarding process the SIOP can generate a different key pair (pairwise DIDs)

15 Create Access Token <Access Token>

Packet delivery Co creates an access token to send to the wallet for further access to services

16 HTTP 200

17 <Access Token>

The Proxy notifies the Portal to refresh the login page so it can present the services to the customer

At this point, Packet Delivery Co knows:
1. That Happy Pets belongs to Data Space
2. That Happy Pets is a Trusted Issuer of EmployeeCredentials
3. That happy Pets says that the user is a customer
4. The category of customer (Normal/Gold)
5. Any user info encoded into the VC, like customer id

18 Services?

19 Change PTA on delivery order

20 Change PTA on order PATCH /pta <Access Token>

21 Validate JWS and certificate

22 POST /delegation <Access Token> (request HappyPets policy)

23 HappyPets policy

24 Is HappyPets allowed to delegate policy to customer?

25 Change confirmed Request forwarded to Context Broker (not shown)

26 Change confirmed

Figure 4.4.3.1b: 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, to login.
2. Happy Pets customer gets forwarded to a page for selecting the desired Identity Provider for login. One of the login options is "Verifiable Credentials" or something similar.
3. Happy Pets customer selects the "Verifiable Credentials" login method, which causes the Packet delivery company portal to generate a QR containing inside the URL of the /authentication-requests endpoint of the Packet Delivery company IDP.
4. The customer scans the QR with her mobile and the mobile calls the /authentication-requests endpoint.
5. This starts a standard SIOP (Self-Issued OpenID Provider) flow, where the Packet Delivery company IDP plays the role of Relying Party (RP in Open ID Connect terminology) and the mobile device of the customer as a Self-Issued IDP. In this step, Packet Delivery company IDP creates a SIOP Authentication Request. As a Self-Issued OP may be running locally as a native application or progressive web application (PWA), the RP may not have a network-addressable endpoint to communicate directly with the OP. We have to leverage the implicit flow of OpenID Connect to communicate with such locally-running Ops, as described in https://openid.net/specs/openid-connect-self-issued-v2-1_0.html.

   The Authentication Request travels in the response to the HTTP GET request performed in the previous point, as a JWT signed by Packet Delivery company. The decoded contents of the JWT may be:

```
openid://?
  response_type=id_token
  &response_mode=post
  &client_id=did:elsi:EU.EORI.NLPACKETDEL
  &redirect_uri=https%3A%2F%2Fidp-pdc.fiware.io%2Fsiop_sessions
  &scope=openid%20profile
  &state=af0ifjsldkj
  &nonce=n-0S6_WzA2Mj
  &registration=%7B%22subject_syntax_types_supported%22:%5B%22did%22%5D,
    %22id_token_signing_alg_values_supported%22:%5B%22RS256%22%5D%7
```

6. The Authentication Request is returned to the customer wallet acting as SIOP. The SIOP flow uses a new response mode **post** which is used to request the

SIOP to deliver the result of the authentication process to a certain endpoint. The parameter **response_mode** is used to carry this value.

This endpoint where the SIOP shall deliver the authentication result is defined in the standard parameter **redirect_uri**.

7. In this step the customer verifies that the Packet Delivery company is a trusted entity belonging to the ecosystem, by resolving the DID of the Packet Delivery company which is received in the **client_id** parameter of the Authentication Request.

   To resolve a DID, the wallet sends a GET request to the **/api/did/v1/identifiers/did:elsi:EU.EORI.NLPACKETDEL** endpoint of one of several trusted servers implementing the Universal Resolver functionality. The Universal Resolver includes a blockchain node, and there may be as many as needed. Its mission is to resolve DIDs using the blockchain and return the associated DID Document. The DID Document (as per W3C) contains relevant information about the entity owner of the DID. It contains its Public Key, used to verify the digital signature of the entity. It also contains the status of the entity in the Data Space ecosystem. It is extensible and can contain any public information which may be relevant for the use case. The Universal Resolver server must be operated by a trusted entity for the customer. There may be as many nodes as needed operated by different entities. At least one of those trusted entities has to be configured in the wallet of the user.

8. The wallet receives the DID Document of Packet Delivery company, with trusted information about the company, including the Public Key associated with the Private Key that Packet Delivery company uses to digitally sign tokens. For example:

```
{
  "payload": {
    "@context": [
      "https://www.w3.org/ns/did/v1",
      "https://w3id.org/security/v1"
    ],
    "id": "did:elsi:EU.EORI.NLPACKETDEL",
    "verificationMethod": [
      {
        "id": "did:elsi:EU.EORI.NLPACKETDEL#key-verification",
        "type": "JwsVerificationKey2020",
        "controller": "did:elsi:EU.EORI.NLPACKETDEL",
        "publicKeyJwk": {
          "kid": "key-verification",
          "kty": "EC",
          "crv": "secp256k1",
          "x": "V8XptJkb5wplYkExcTF4nkyYVp7t5H5d5C4UPqCCM9c",
          "y": "kn3nSPxIIvd9iaG0N4v14ceuo8E4PcLXhhGeDzCE7VM"
        }
      }
    ],
    "service": [
      {
        "id": "did:elsi:EU.EORI.NLPACKETDEL#info",
        "type": "EntityCommercialInfo",
        "serviceEndpoint": "https://packetdelivery.com/info",
        "name": "Packet Delivery co."
      },
      {
        "id": "did:elsi:EU.EORI.NLPACKETDEL#sms",
        "type": "SecureMessagingService",
        "serviceEndpoint": "https://packetdelivery.com/api"
      }
    ],
    "anchors": [
      {
        "id": "redt.alastria",
        "resolution": "UniversalResolver",
        "domain": "packetdelivery.ala",
        "ethereumAddress": "0xbcB9b29eeb28f36fd84f1CfF98C3F1887D831d78"
      }
    ],
    "created": "2021-11-14T13:02:37Z",
    "updated": "2021-11-14T13:02:37Z"
  }
}
```

9. The DID Document includes one or more public keys inside the “verificationMethod” array. The keys are identified by the “id” field in each element of the array. The customer wallet uses the kid field that was received in the Authentication Request (in the protected header of the JWT) to select the corresponding Public Key and verify the signature of the JWT. It also verifies that the top-level “id” field in the DID Document ("did:elsi:EU.EORI.NLPACKETDEL") is equal to the **client_id** parameter of the Authentication Request.
10. The customer wallet creates an Authentication Response to be posted in the **redirect_uri** specified by Packet Delivery company in step 5. The contents of the Authentication Response are described below.
11. The SIOP sends the authentication response to the endpoint passed in the **redirect_uri** authentication request parameter using a HTTP POST request using "application/x-www-form-urlencoded" encoding. The response contains

an ID Token and a VP (Verifiable Presentation) token as defined in https://openid.net/specs/openid-connect-4-verifiable-presentations-1_0.html.

```
POST /siop_sessions HTTP/1.1
Host: client.example.com
Content-Type: application/x-www-form-urlencoded

id_token=eyJ0 ... NiJ9.eyJ1c ... I6IjIifX0.DeWt4Qu ... ZXso
&vp_token=...
&state=af0ifjsldkj
```

The decoded **id_token** would be:

```
{
  "iss": "https://self-issued.me/v2",
  "aud": "did:elsi:EU.EORI.NLPACKETDEL",
  "iat": 1615910538,
  "exp": 1615911138,
  "sub": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
  "auth_time": 1615910535,
  "nonce": "n-0S6_WzA2Mj"
}
```

The **sub** claim is *did:peer:99ab5bca41bb45b78d242a46f0157b7d* which is the DID of the user and that is not registered in any blockchain or centralized repository. It must be the same as the DID included in the VP that was issued by the Happy Pets company when onboarding the customer and which travels in the authentication response.

The **vp_token** includes the Verifiable Presentation, which can be in two formats: **jwt_vp** (JWT encoded) or **ldp_vp** (JSON-LD encoded). The following example is using the JWT encoding:

```
{
    "format":"jwt_vp",
    "presentation":
    "eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsImtpZCI6ImRpZDpleGFtcGxlOmFiZmUxM2Y3MTIxMjA0
    MzFjMjc2ZTEyZWNhYiNrZXlzLTEifQ.eyJzdWIiOiJkaWQ6ZXhhbXBsZTplYmZlYjFmNzEyZWJjNmYxY
    zI3NmUxMmVjMjEiLCJqdGkiOiJodHRwOi8vZXhhbXBsZS5lZHUvY3JlZGVudGlhbHMvMzczMiIsImlzc
    yI6Imh0dHBzOi8vZXhhbXBsZS5jb20va2V5cy9mb28uandrIiwibmJmIjoxNTQxNDkzNzI0LCJpYXQiO
    jE1NDE0OTM3MjQsImV4cCI6MTU3MzAyOTcyMywibm9uY2UiOiI2NjAhNjM0NUZTZXIiLCJ2YyI6eyJAY
    29udGV4dCI6WyJodHRwczovL3d3dy53My5vcmcvMjAxOC9jcmVkZW50aWFscy92MSIsImh0dHBzOi8vd
    3d3LnczLm9yZy8yMDE4L2NyZWRlbnRpYWxzL2V4YW1wbGVzL3YxIl0sInR5cGUiOlsiVmVyaWZpYWJsZ
    UNyZWRlbnRpYWwiLCJVbml2ZXJzaXR5RGVncmVlQ3JlZGVudGlhbCJdLCJjcmVkZW50aWFsU3ViamVjd
    CI6eyJkZWdyZWUiOnsidHlwZSI6IkJhY2hlbG9yRGVncmVlIiwibmFtZSI6IjxzcGFuIGxhbmc9J2ZyL
    UNBJz5CYWNjYWxhdXLDqWF0IGVuIG11c2lxdWVzIG51bcOpcmlxdWVzPC9zcGFuPiJ9fX19.KLJo5GAy
    BND3LDTn9H7FQokEsUEi8jKwXhGvoN3JtRa51xrNDgXDb0cq1UTYB-rK4Ft9YVmR1NI_ZOF8oGc_7wAp
    8PHbF2HaWodQIoOBxxT-4WNqAxft7ET6lkH-4S6Ux3rSGAmczMohEEf8eCeN-jC8WekdPl6zKZQj0YPB
    1rx6X0-xlFBs7cl6Wt8rfBP_tZ9YgVWrQmUWypSioc0MUyiphmyEbLZagTyPlUyflGlEdqrZAv6eSe6R
    txJy6M1-lD7a5HTzanYTWBPAUHDZGyGKXdJw-W_x0IWChBzI8t3kpG253fg6V3tPgHeKXE94fz_QpYfg
    --7kLsyBAfQGbg"
}
```

Which decoded could be:

```
{
  "@context": ["https://www.w3.org/2018/credentials/v1"],
  "type": ["VerifiablePresentation"],
  "verifiableCredential": [
    {
      "@context": [
        "https://www.w3.org/2018/credentials/v1",
        "https://happypets.fiware.io/2022/credentials/employee/v1"
      ],
      "id": "https://happypets.fiware.io/credentials/25159389-8dd17b796ac0",
      "type": ["VerifiableCredential", "CustomerCredential"],
      "issuer": {
        "id": "did:elsi:EU.EORI.NLHAPPYPETS"
      },
      "issuanceDate": "2022-03-22T14:00:00Z",
      "validFrom": "2022-03-22T14:00:00Z",
      "expirationDate": "2023-03-22T14:00:00Z",
      "credentialSubject": {
        "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
        "verificationMethod": [
          {
            "id": "did:peer:99ab5bca41bb45b78d242a46f0157b7d#key1",
            "type": "JwsVerificationKey2020",
            "controller": "did:peer:99ab5bca41bb45b78d242a46f0157b7d",
            "publicKeyJwk": {
              "kid": "key1",
              "kty": "EC",
              "crv": "P-256",
              "x": "lJtvoA5_XptBvcfcrvtGCvXd9bLymmfBSSdNJf5mogo",
              "y": "fSc4gZX2R3QKKfHvS3m2vGSVSN8Xc04qsquyfEM55Z0"
            }
          }
        ],
        "roles": [
          {
            "target": "did:elsi:EU.EORI.NLPACKETDEL",
            "names": ["P.Info.gold"] // Or P.Info.standard
          }
        ],
        "name": "Jane Doe",
        "given_name": "Jane",
        "family_name": "Doe",
        "preferred_username": "j.doe",
        "email": "janedoe@packetdelivery.com"
      }
    }
  ]
}
```

12. Packet Delivery company uses its own blockchain node implementing the Universal Resolver functionality to resolve the DID of Happy Pets, which is inside the Verifiable Credential received in the Verifiable Presentation. This DID can be found in the "issuer" field of the "verifiableCredential" structure above.

    Resolution is performed sending a GET request to the Universal Resolver: **/api/did/v1/identifiers/did:elsi:EU.EORI.NLHAPPYPETS**

    Packet Delivery could use a Universal Resolver operated by a different entity, but this would reduce the level of trust compared to using its own server directly connected to the blockchain network.
13. Packet Delivery receives the DID Document of Happy Pets with trusted information about the company, including the Public Key associated to the Private Key that Happy Pets used to digitally sign the Verifiable Credential that the customer has just sent inside a Verifiable Presentation as part of the authentication flow. **Using the Public Key and the DID inside the DID Document, it can verify the signature of the Verifiable Credential and that Happy Pets is a trusted entity in the ecosystem.**
14. The above is just for verification of the Verifiable Credential. In addition, Packet Delivery company can also verify that the Verifiable Presentation including the Verifiable Credential is sent by the customer and not by a malicious agent. To do so, it uses the Public Key of the customer in the "verificationMethod" of the "credentialSubject" structure. That public key is cryptographically bound to the customer DID during the onboarding process that Happy Pets performed with the customer.
15. Once all verifications have been performed, Packet Delivery company creates an Access Token for the customer so she can use it to access services in Packet Delivery company in the future.
16. The wallet (SIOP) receives a successful reply to the POST request.

17. The Packet Delivery company proxy notifies the Packet Delivery portal that the customer is successfully authenticated, and the portal can display the services available to that customer. The browser of the user receives the Access Token created by Packet Delivery to enable it to request services without going through the previous authentication process. The Access Token is a standard OAuth access token that includes the information that Packet Delivery requires for accessing its services.

    At this point the Happy Pets customer is logged in on the Packet Delivery company portal/app and is presented with the possible services provided, including the option to change the PTA of its delivery orders.

    At this moment, the Packet Delivery company knows the following:

    - Happy Pets is a participant in the Data Space and that it is a Trusted Issuer of EmployeeCredentials because this info is in the DID Document retrieved in step 13. Maintenance of this information is performed by the Trusted Anchor entity(or entities) managing the Trusted Participants List and Trusted Issuers List.
    - Happy Pets says that the user is a customer. This info is inside the Verifiable Credential that is digitally signed by Happy Pets.
    - The category of the customer (and associated policies) with regards to the services offered by Packet Delivery company. This information is also in the Verifiable Credential presented by the customer.

18. The Happy Pets customer is presented with the possible services provided by Packet Delivery, including the option to change the PTA of its delivery orders.
19. 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.
20. 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 the Access Token generated in step 15, with information about the authorisation registry to retrieve policies from.

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

PATCH
https://umbrella.fiware.io/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.packetdelivery.com",
     "identifier": "EU.EORI.NLHAPPYPETS",
     "delegation_endpoint": "https://ar.packetdelivery.com/delegation",
   }
}
```

21. Packet Delivery company proxy received the request of step 19 for changing the PTA of a delivery order. The Access Token received from the customer ensures that she was assigned the delegation evidence with a policy for updating the PTA attribute of this specific delivery order (called issuance at **user level**). 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). The Verifiable Credential takes care of the user level policies.

    At first, the Packet Delivery company proxy validates the JWT which is part of the authorization header of the PATCH request.

22. 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. The proxy sends a request to the `/delegation` endpoint of the Packet Delivery company Authorization Registry.
23. The proxy receives the delegation evidence policy issued from Packet Delivery company to Happy Pets.

24. 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 Happy Pets is allowed to delegate the access to its customers. If the proxy received a valid policy, access would be granted on an **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.

25. 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 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. The Context Broker response is returned to the Packet Delivery company portal, in response to the request of step 26.

26. The successful change of the PTA is presented to the Happy Pets customer.

4.4.3.2 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 4.4.2 Acquisition of Rights / Activation, No Cheaper is just acquiring the standard service and therefore its customers will only be able to read attributes of delivery orders. This means that at the Packet Delivery authorisation registry, there is only a policy created allowing No Cheaper to only delegate GET access to delivery orders.

This scenario can be split 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 Verifiable Credential is issued 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 43, 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 Verifiable Credential is issued 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 62, 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.

4.4.3.3 Issuing tokens for Connectors / application context

In addtion to the section described above tokens (DAT Dynamic Access Token) for the application context must be issued containing the referencing connectors and their security profile. The details of issuing the tokens have to be described to act as an alternative to the current IDS-DAPS realization or to include this mechanisms as specified in IDS-G.

The DAPS issues the requested DAT, or an error response, as per RFC 6749. The Access Token ("the DAT") itself is a JWS adhering to RFC 9068, which in turn contains JSON-LD encoded data in addition to the standard claims, subject to the following additional constraints:

Field name	**additional constraints**
@context	Must be https:*//w3id.org/idsa/contexts/context.jsonld*
@type	Must be ids:DatPayload
securityProfile	Must be an instance of the ids:SecurityProfile class

The DAT MUST be signed using a digital signature scheme. It SHOULD be limited to a short time period (Recommendation: 1 hour). The default resource indicator to be used in the DAT includes idsc:IDS_CONNECTORS_ALL, which SHOULD be accepted by all connectors. Future revisions of this document may allow for mechanisms to specify connectors to be listed in the aud claim such as through RFC 8707.

Additional claims may optionally be present. This specification defines the following:

- **referringConnector** An optional URI of the subject. Is used to connect identifier of the connector with the self-description identifier as defined by the IDS Information Model. A receiving connector can use this information to request more information at a Broker or directly by dereferencing this URI.
- **transportCertsSha256** Contains the public keys of the used transport certificates, hashed using SHA256. The identifying X509 certificate should not be used for the communication encryption. Therefore, the receiving party needs to connect the identity of a connector by relating its hostname (from the communication encryption layer) and the used private/public key pair, with its IDS identity claim of the DAT. The public transportation key must be one of the transportCertsSha256 values. Otherwise, the receiving connector must expect that the requesting connector is using a false identity claim. In general, this claim holds an Array of Strings, but it may optionally hold a single String instead if the Array would have exactly one element.
- **extendedGuarantee** In case a connector fulfills a certain security profile but deviates for a subset of attributes, it can inform the receiving connector about its actual security features. This can only happen if a connector reaches a higher level for a certain security attribute than the actual reached certification asks for. A deviation to lower levels is not possible, as this would directly invalidate the complete certification level. In general, this claim holds an Array of Strings, but it may optionally hold a single String instead if the Array would have exactly one element.

Example

The following is an example of a sucessful response:

```
200 This is fine

Content-Type: application/json


{

  "access_token": "skdj54dkGjnb[...]lsl8723ijsdfuzticby_ch",

  "scope": "idsc:IDS_CONNECTOR_ATTRIBUTES_ALL",

  "token_type": "bearer",

  "expires_in": "3600"

}
```

The decoded DAT, including header and payload is shown below:

```
{
   "typ": "jwt+at",
   "kid": "somekid",
   "alg": "RS256"
}
.
{
   "iss": "https://daps.aisec.fraunhofer.de/v3",
                                                                              "sub":
"DD:CB:FD:0B:93:84:33:01:11:EB:5D:94:94:88:BE:78:7D:57:FC:4A:keyid:CB:8C:C7:B6:8
5:79:A8:23:A6:CB:15:AB:17:50:2F:E6:65:43:5D:E8",
   "nbf": 1516239022,
   "iat": 1516239022,
   "exp": 1516239032,
   "aud": ["idsc:IDS_CONNECTORS_ALL"],
   "scope": "idsc:IDS_CONNECTOR_ATTRIBUTES_ALL",
   "@context": "https://w3id.org/idsa/contexts/context.jsonld",
   "@type": "ids:DatPayload",
   "referringConnector": "http://some-connector-uri.com",
   "securityProfile": "idsc:BASE_SECURITY_PROFILE",
   "extendedGuarantee": "idsc:USAGE_CONTROL_POLICY_ENFORCEMENT",
   "transportCertsSha256": "bacb879575730bb083f283fd5b67a8cb..."
}
.
somesignature
```

Open aspects to be discussed

Currently, the connector is identified by the attributes of an X.509 certificate. The identifier based on DID has to be decribed. This is still open.

In IDS the security profile is validated by an external evaluation facility and provided to a central authority. The workflow of providing the claim directly as VC/VP needs to be described in detail.

5 Policy Definition Language

A Policy Definition Language is required to define and agree access and usage policies. The defined and agreed policies can be used directly or translated into an executable language, e.g. Rego.

We propose to use ODRL as an interoperable standard for the negotiation and acceptance of Access and Usage Policies including the policiy negotiation sequence as defined by IDSA in the IDS-RAM. Nevertheless, it might be required to translate ODRL policies into an executable language like Rego during runtime.

The enforcement of those policies can be realized as described in the IDS-RAM.

To be noted that under the condition that the various grammars allow it, the semantic interoperability between different executable policy engines could be achieved only if common controlled vocabularies are adopted.

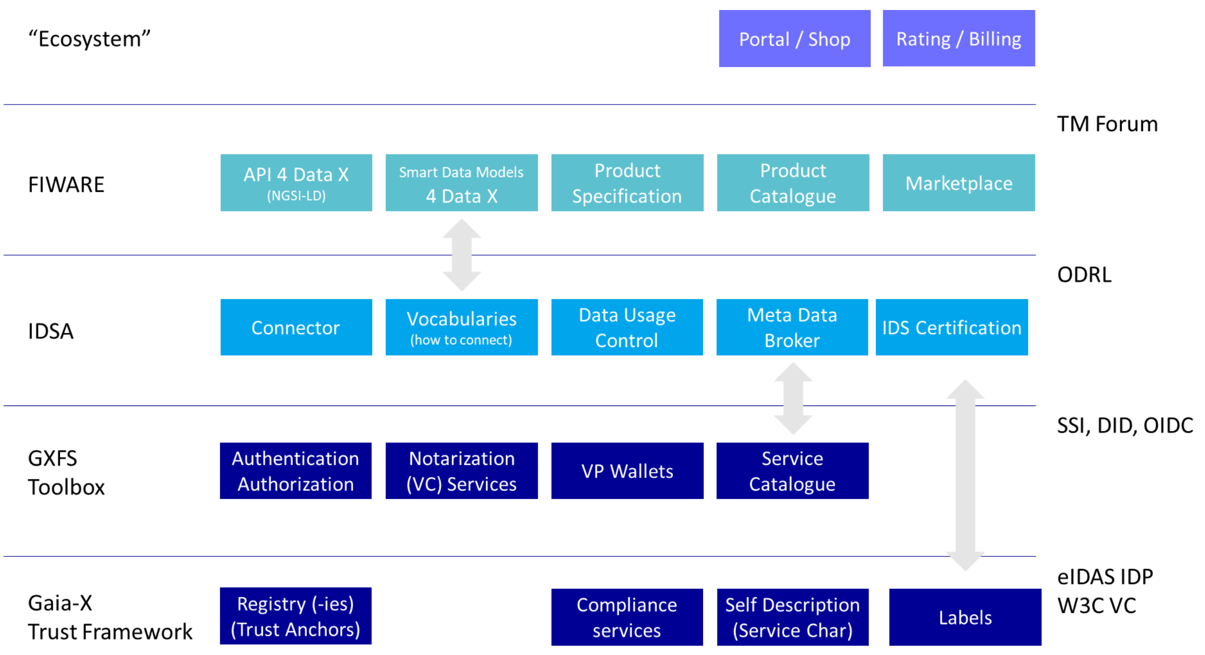
6 Outlook and next steps

The Architecture Coherence Workshop series of the Data Space Business Alliance DSBA discussed the alignment between the members of the alliance and came up with the detailed approach described in the sections above. While some aspects are quite clear and should be adopted by the initiatives, e.g. Identity Management mechanisms and the use of DID and VC/VP, other aspects need further clarification. The Workshop series will be continued to close those gaps. In order to provide a common framework for data space the DSBA will continue to further refine everything that is depicted in this document during the next months. It will also work towards development of some of the required components as open source. Additional support will be provided based on the Data Spaces Support Centre and the work conducted in the OpenDEI project. Furthermore the ongoing alignment on a common framework for data spaces under the DSBA is crucial to provide a robust foundation for all initiatives realizing data spaces.

In conjunction with the detailed description in the sections above, the DSBA has a common understanding on the roles of the members and their scope of work. Gaia-X is focusing on architectures and specifications for data spaces and IDSA is focusing on architectures and standard specifications for data spaces. In this respect, It is common understanding that the main focus of Gaia-X is the definition of a common set of rules to enable interoperable data-infrastructure and to anchor the result of the policies negotiation for data exchange down to the hardware for a full stack execution traceability,while the IDS Connectors realize distributed functionalities for participants in a data space to trust each other, and apply access and usage policies while sharing data, including the traceability of data accross organizational domains. FIWARE, on the other hand, provides a robust set of Open Source Bulding blocks that implements and contributes based on implementation experience to the work of Gaia-X and IDSA, also covering additional functionalities and components to put data spaces into practice.

Interoperability in data spaces is an important goal for the partners in the DSBA, but interoperability can be achieved on different layers, e.g., semantic interoperability, technical interoperability, contract interoperability. This is work in progress of the DSBA with the intend of maximize the number of layers, where interoperability is achieved.

The following picture depicts a layered approach and how architectures, implemententations (here including the Gaia-X Federation Services GXFS) provide a framework for data spaces. While some components and concepts rely and build on each other, others have a clear interface or provide a certain degree of freedom to add required configuration or extensions by the relevant Ecosystems, i.e. data space instances or federations.



The future work of the DSBA will focus on the following aspects. This is not meant to be a complete list, but listing items that have currently high priority.

- It is still not finally described how IDS Connectors fit into the described identity solution. Various options are still possible in an integrated approach or in a hybrid approach.
    - In an integrated approach, the Application Context in IDS Connectors would make use of DID and VC/VP in a similar manner for individuals and organizations.
    - While in an hybrid approach X.509 certificates and the Dynamic Attribute Provisioning Service (DAPS) would act in parallel to DID and VC/VP for individuals and organizations.
    - Additionally a mixed mode could also be described.
- The understanding of the interaction of Marketplaces with the Federated Catalog and the Meta-Data-Broker is not final, yet. There is a certain amount of

overlap, while a clear responsibility of each component is given. In the process of understanding and clarification some aspects are in the current focus:

  - The mapping of TMForum model and Gaia-X model exists in Slides (see appendix), but is not finalized. Additionally, the mapping with the IDS model needs to be realized.
  - The relationship of TMForum APIs and IDS contract negotiation sequence needs to be described. TMForum APIs can be used to support bilateral negotiations between two participants on which IDS contract negotiation could be based, but this is not described, yet.
  - Overall, the responsibilities, functionalities and requirements of each must be described clearly and distinguished from each other.

- The use of Policies for Access and Usage Control is well understood and a clear differentiation from the negotiation of policies and their execution is understood. Nevertheless, there is a clear need to describe the controlled vocabularies that realize those different aspects to provide the required reliability for the implementation.
- Finally, a big picture of the DSBA common framework is missing. The group has already identified an outline and has achieved a valid understanding, but the clear communication of the common vision is missing and should be addressed soon to provide a clear guideline for everyone building data spaces.

7 Authors and Contributors

Put your name and affiliation here:

- Gernot Böge, FIWARE Foundation
- Erik Cornelisse, TNO
- Simon Dalmollen, TNO
- Pierre Gronlier, Gaia-X AISBL
- Juanjo Hierro, FIWARE Foundation
- Maarten Kollenstaart, TNO
- Jörg Langkau, nicos AG
- Klaus Ottradovetz, ATOS
- Matthijs Punter, TNO
- Jesus Ruiz, FIWARE Foundation
- Anna Maria Schleimer, Fraunhofer ISST
- Sebastian Steinbuss, International Data Spaces Association IDSA
- Denis Wendland, FIWARE Foundation