

Federated Data Exchange Infrastructure Architecture

28 February 2025

opentunity



OPENTUNITYproject.eu

Deliverable details

Title	WP	Version
Federated Data Exchange Infrastructure Architecture	3	1

Contractual delivery date	Actual delivery date	Delivery type*	Dissemination**
Feb 2025 (M26)	Feb 2025 (M26)	R: Document, report	PU

*Delivery type: R: Document, report; DEM: Demonstrator, pilot, prototype; DEC: Websites, patent fillings, videos, etc; OTHER; ETHICS: Ethics requirement; ORDP: Open Research Data Pilot.

Dissemination Level: **PU: Public; **CO**: Confidential, only for members of the consortium (including the Commission Services)

Author(s)	Organization
Michalis Vasileiadis	QUE
Ioannis Mettos	QUE
Yannis Syrimpeis	QUE
Apostolos Papafragkakis	QUE

Version	Date	Person	Action	Status***
0.1	13/01/2025	Yannis Syrimpeis, Apostolos Papafragkakis	1 st Version	Draft
0.2	24/02/2025	Yannis Syrimpeis, Apostolos Papafragkakis, Ioannis Mettos, Michalis Vasileiadis	Ready for peer review	Draft
Final	27/02/2025	Yannis Syrimpeis, Apostolos Papafragkakis, Ioannis Mettos, Michalis Vasileiadis	Final version	Final

***Status: Draft, Final, Approved, Submitted (to European Commission).

Contributors (organization)

Pablo Bort (ETRA), Nicolas Stathopoulos (HYP), John Karakitsios (ICCS)

Keywords

Data Space, Federated Data Exchange Infrastructure, Trust services, Data Space connector, Federated Catalogue, Gaia-X Trust Framework

Executive Summary

This deliverable provides the initial structure and composition of OPENTUNITY's Federated Data Exchange Infrastructure (FDEI) Architecture, which is one of OPENTUNITY's innovations. The purpose of FDEI is to facilitate data exchange by delivering OPENTUNITY's federated data space infrastructure, ultimately creating a secure, trustworthy environment that promotes interoperability, sovereignty and trust. ¹D3.1 Federated Data Exchange Infrastructure Architecture was completed in Month 26 (M26) and describes the activities carried out under Work Package 3 (WP3), particularly Tasks 3.1, 3.2 and 3.4. It presents the progress achieved towards the realization of an operational and scalable federated data space.

In accordance to the project's execution timeplan, the consortium has achieved Proof of Concept of the major Data Space components, which are described in detail in this deliverable. Since the project is in the second reporting period, the comments from the 1st review have been considered, and those actions are also included in this report. Regarding T3.1, T3.2 and T3.4, two points were reported by the project's reviewers. Firstly, there was a recommendation to integrate SSH aspects more strongly into OPENTUNITY and the proposed actions on this are described in section 2.2. The second comment was the suggestion that the Federated data spaces approach should be closely coordinated with other relevant Horizon Projects, which are described in the specific Section of Interoperability of this document.

Copyright statement

The work described in this document has been conducted within the OPENTUNITY project. This document reflects only the OPENTUNITY Consortium view, and the European Union is not responsible for any use that may be made of the information it contains.

This document and its content are the property of the OPENTUNITY Consortium. All rights relevant to this document are determined by the applicable laws. Access to this document does not grant any right or license on the document or its contents. This document or its contents are not to be used or treated in any manner inconsistent with the rights or interests of the OPENTUNITY Consortium or the Partners detriment and are not to be disclosed externally without prior written consent from the OPENTUNITY Partners.

Each OPENTUNITY Partner may use this document in conformity with the OPENTUNITY Consortium Grant Agreement provisions.

¹ OPENTUNITY's Description of the Action

Index

INDEX	5
1 INTRODUCTION	10
1.1 Purpose of the document	10
1.2 Scope of the document	10
1.3 Structure of the document	10
2 WHAT IS A DATA SPACE	12
2.1 Evaluation of existing solutions	12
2.1.1 Introduction	12
2.1.2 The blockchain approach	12
2.1.3 The Data Space transition	12
2.2 Basic concepts	13
2.3 Data space fundamentals	14
2.3.1 Useful concepts and terminology	14
2.3.2 Data Space provisions	14
3 OPENTUNITY'S FDEI ARCHITECTURE	17
3.1 OPENTUNITY's FDEI key features and conceptual architecture	17
3.2 FDEI components	18
3.2.1 Trust services	18
3.2.2 Data space connector	18
3.2.3 Federated catalogue	18
3.3 Data Space deployment strategy	19
4 FDEI IMPLEMENTATION	20
4.1 Trust services	20
4.1.1 Access Levels	20
4.1.1.1 Participant level	20
4.1.1.2 End-user level	21
4.1.2 Centralized vs Decentralized components	21

4.1.3	Trust services subcomponents	22
4.1.3.1	Identity and Access Management	22
4.1.3.2	Wallet	25
4.1.3.3	Identity Provider (IdP)	26
4.1.3.4	Onboarding UI	28
4.1.3.5	Participant Template	31
4.1.3.5.1	Component breakdown	31
4.1.3.5.1.1	Task	31
4.1.3.5.1.2	Docker/compose	32
4.1.3.5.1.3	Bash	33
4.1.3.5.1.4	Caddy	33
4.1.3.5.1.5	EDC Connector	34
4.1.3.5.1.6	Credentials Manager	34
4.1.3.5.1.7	Walt.id Wallet	36
4.1.3.5.2	Directory structure	36
4.1.3.6	Participant Management UI	36
4.2	Data Space Connector	41
4.2.1	Available data space connector implementations	41
4.2.1.1	Eclipse Dataspace Components (EDC)	41
4.2.1.2	TRUE Connector	41
4.2.1.3	Fraunhofer Dataspace Connector (IDS reference)	42
4.2.1.4	TNO Security Gateway	42
4.2.2	Eclipse Dataspace Components (EDC)	42
4.2.3	Extensions developed	43
4.2.3.1	IAM/SSI Support	44
4.2.3.1.1	Key components	44
4.2.3.1.2	Authentication flow	44
4.2.3.2	Automatic contract offering generation (OpenAPI)	46
4.2.3.2.1	Key Components	47
4.2.3.2.2	Operating steps	48
4.3	Federated Catalogue	49
4.3.1	Available Federated Catalogue implementations	49
4.3.1.1	IDS Metadata Broker	49

4.3.1.2	XFSC Federated Catalogue	50
4.3.2	Next steps	50
4.4	Opentunity Ontology/Data Model	51
4.4.1	Gaia-X Trust Framework Ontology overview	51
4.4.2	Key Features of the Gaia-X Trust Framework Ontology	51
4.4.3	Purpose and Objectives	52
4.4.4	Evolution and Openness	52
4.4.5	Trust Framework usage in Opentunity	52
4.5	Participant onboarding plan	56
4.5.1	Initial onboarding group	56
4.5.2	Collecting information	57
4.5.3	Verifying compatibility and request for adaptations	58
4.5.4	Future platform onboarding	59
4.6	Components deployment plan	59
4.6.1	Centralized Components	59
4.6.2	Decentralized Components	60
4.6.3	Deployment Timeline	61
4.6.4	Key Considerations	61
4.7	Testing and evaluation plan	61
4.7.1	Testing and Evaluation Methods	61
4.7.1.1	Unit testing	62
4.7.1.2	Integration testing	62
5	INTEROPERABILITY	63
5.1	The crucial role of interoperability	63
5.2	Assessment of other projects dataspace	64
6	CONCLUSION	66
7	REFERENCES AND ACRONYMS	67
7.1	References	67
7.2	Acronyms	68

8 ANNEX I RELATED UCS, REQUIREMENTS AND SGAM

DIAGRAMS	69
8.1 UC 1.2 Non-Intrusive Load Monitoring.	69
8.2 UC 1.8 HEMS/BEMS DR optimization and local flexibility management.	71
8.3 UC 1.9 Initialization of HEMS/BEMS Demand Response strategy.	75

List of figures

Figure 1: T3.1, T3.2 and T3.4 Timeline	10
Figure 2: Core functions of a data space (figure courtesy of DATA CELLAR Horizon project)	13
Figure 3: Data Space provisions	15
Figure 4: Conceptual architecture	17
Figure 5: OPENTUNITY Data Space deployment strategy	19
Figure 6 IAM (KeyCloak) login screen	24
Figure 7 OPENTUNITY realm roles	24
Figure 8 Participant groups in KeyCloak	25
Figure 9 Example users in the Hypertech participant group	25
Figure 10 Walt.Id HTTP API	26
Figure 11 Part of the IdP OpenAPI Specification	27
Figure 12 Generation of API Token	28
Figure 13 Regeneration of API Key	28
Figure 14 Participant registration	29
Figure 15 Onboarding UI login screen	29
Figure 16 Participant configuration	30
Figure 17 The user is downloading the preconfigured participant template.	31
Figure 18 The credential-manager OpenAPI specification in Swagger format	35
Figure 19 The participant template directory structure	36
Figure 20 The participant manager login screen	37
Figure 21 Wallet unlocking	37
Figure 22 Wallet contents	38
Figure 23 Local connector status and catalogue	39
Figure 24 Remote connector catalog retrieval	40
Figure 25 End-to-end dataspace service call test	40
Figure 26 The connector needs to be authenticated against a counterparty	45
Figure 27 Authenticating a counterparty	46
Figure 28 Flow diagram of the custom-built OpenAPI EDC extension	48
Figure 29 Example of a Participant self-description	54
Figure 30 Example of a service offering self-description	55
Figure 31 Participant information is combined into a JSON-LD that conforms to the Gaia-X trust framework ontology, signed by the IdP and submitted to the federated catalogue	56
Figure 32 Red arrows show the initial participants of the dataspace	57
Figure 33 Example of filled questionnaire	58

Figure 34: Interoperability model of New European Interoperability Framework.....	63
Figure 35: UC1.2 SGAM Diagram.....	71
Figure 36: UC 1.8 SGAM Diagram.....	74
Figure 37: UC1.9 SGAM Diagram	77

List of tables

Table 1: Data Space authorization levels.....	20
Table 2: Centralized vs Decentralized components	21
Table 3: Pros and Cons of IDS Metadata Broker	50
Table 4: Pros and Cons of XFSC Catalogue.....	50
Table 5: UC 1.2 - Application of NILM for consumer's energy awareness	70
Table 6: UC1.8 - HEMS/BEMS DR optimization and local flexibility management	74
Table 7: UC1.9 - Initialization of HEMS/BEMS Demand Response strategy	76

1 Introduction

1.1 Purpose of the document

This deliverable presents the work carried out in Tasks 3.1 - Federated Data Exchange Infrastructure design and delivery, Task 3.2 - Management of data space participants and data/service offerings and Task 3.4 - Integration with other platforms. It is the 1st version of the Federated Data Exchange Infrastructure (FDEI) Architecture and includes the FDEI architecture, as well as documentation of the services and interfaces developed to support the management of data space participants and data service offerings.

1.2 Scope of the document

The scope of this deliverable is to describe in detail the activities carried out to achieve the objectives of the above-mentioned tasks. As presented in the timeline in Figure 1, at this stage of the project, the focus was on the development of the key components of the Data Space, specifically the Trust services and the Connector.

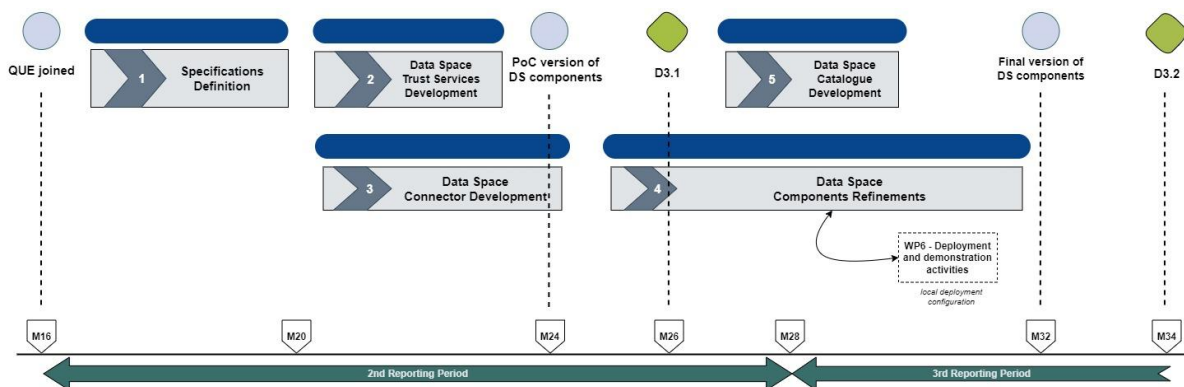


Figure 1: T3.1, T3.2 and T3.4 Timeline

Apart from describing the developed components, the document also provides a basic description of the Data Spaces including an evaluation of alternative solutions, the architecture and design principles, the evaluation and status of the Federated catalogue, the Data Space onboarding plan and the interoperability strategy.

1.3 Structure of the document

The document is organized into the following sections:

- Section 1 introduces the deliverable.
- Section 2 provides a basic description of Data Spaces and outlines alternative approaches.
- Section 3 describes the overall architecture design and includes a short description of the Federated Data Exchange Infrastructure components.
- Section 4 is the core part of the deliverable in which the Federated Data Exchange Infrastructure Implementation is described. It covers the implementation of Data Space components, the Gaia-X Trust Framework, the user onboarding plan, and the integration and deployment strategy.

- Section 5 outlines the interoperability strategy, highlighting the role of interoperability and its importance.
- Section 6 concludes the document and describes the next steps.

2 What is a Data Space

2.1 Evaluation of existing solutions

2.1.1 Introduction

The OPENTUNITY Federated Data Exchange Infrastructure (FDEI) needs to enable interoperable, trustable, and sovereign data exchange between participants. An integral step towards the establishment of this infrastructure was the examination of existing solutions that align with the principles of data sovereignty, interoperability, and trust. Initially the project was envisioned to follow a blockchain-based paradigm, but following an amendment, the decision was made to transition to a Data Space approach due to its stronger alignment with OPENTUNITY's objectives. The transition from blockchain-based solutions to a data space model aligns with EU priorities on data sovereignty and interoperability.

2.1.2 The blockchain approach

At the beginning of the project, the aim was to use a specific public blockchain infrastructure and design an open, secure, and trusted platform for data exchange between energy assets and systems using Decentralized Identifiers (DIDs) and Verifiable Credentials (VCs). Blockchain was chosen as the technology for OPENTUNITY's Data Exchange. Blockchain offered benefits such as decentralization, resistance, and immutability.

Although blockchain provided foundation benefits of decentralization and security, its limitations in scalability, regulatory compliance, and integration in complex energy ecosystems made OPENTUNITY choose a more versatile Data Space architecture. This transition better aligns with the project's overarching goals of promoting interoperability, trust, and data sovereignty within the European energy market.

2.1.3 The Data Space transition

In OPENTUNITY's decision to the Data Space transition approach the involvement of QUE Technologies, a new partner with extensive experience in Data Spaces played a crucial role. QUE is a key technology partner in Data Cellar's ² federated data space. QUE's know-how in developing decentralized, self-sovereign data-sharing environments directly benefits OPENTUNITY by applying the best practices and lessons learned from Data Cellar. Furthermore, Data Cellar's participation in the INT:Net³ initiative provides a framework for OPENTUNITY to align with interoperability and governance standards. Following an amendment to the project, OPENTUNITY transitioned to a Data Space approach.

² <https://datacellarproject.eu/>

³ <https://intnet.eu/>

2.2 Basic concepts

A data space is a decentralized ecosystem that enables secure and controlled data sharing among multiple stakeholders. It allows participants to maintain data sovereignty while facilitating interoperability and collaboration.⁴

The main objective of data spaces is to address the issues that complicate data sharing between organizations. A data space brings together different organizations who all agree to follow the same guidelines when it comes to sharing data safely and transparently⁵. Trust and sovereignty are key aspects of data spaces, and dataspace participants are required to adhere to the same high-level standards and guidelines of data storage and sharing⁶. The core functions of a data space are depicted in Figure 2⁷.

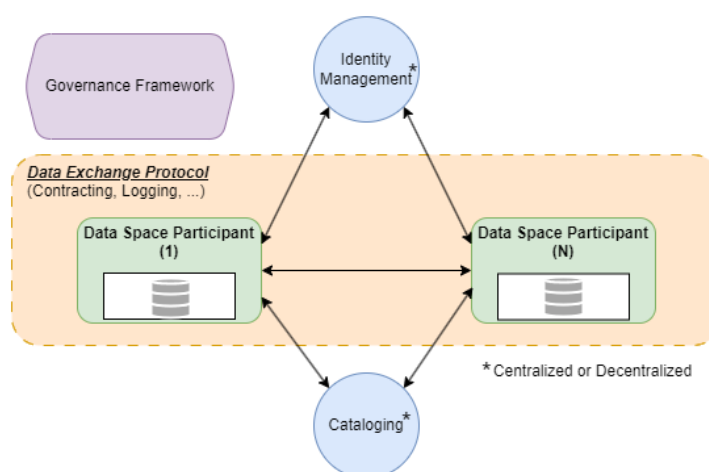


Figure 2: Core functions of a data space (figure courtesy of DATA CELLAR Horizon project)

In Figure 2, apart from the governance framework and identity management that shape a data space, all participants follow a common data exchange protocol. This protocol lays out the needs for data exchange, which includes aspects like contracting and logging.

In a data space, data are stored close to where it was generated and stay with the original owner. There is no central storage since centralization goes against data sovereignty, which is crucial for data space. Instead, each participant uses their own storage and chooses what data to share and under what policies. This type of distributed storage needs ways to organize and find data and services. Organizing involves systematically arranging and listing data and information resources using a suitable framework, which helps create services that let participants easily find what they want.

From an end-user perspective, and especially for those that are unfamiliar, data spaces seem complex. To address any potential concerns on data security, control and usage⁸ we are

⁴ <https://www.sciencedirect.com/science/article/pii/S2352340924009314>

⁵ <https://dssc.eu/space/Glossary/176554052/2.+Core+Concepts>

⁶ <https://gaia-x.eu/what-is-gaia-x/about-gaia-x/>

⁷ Data Cellar Horizon2020 project

⁸ <https://link.springer.com/article/10.1007/s11747-022-00845-y>

supplementing the rollout of the technical solutions developed in OPENTUNTY with questionnaires and interviews. This approach will help us understand user needs and identify any barriers or attitudes that might hinder the adoption of data space applications⁹. The insights gained will be used to:

- a) Adapt the OPENTUNTY tools to be more user-centric
- b) Identify potential new use cases
- c) Derive recommendations for the future design of data spaces for end users.

2.3 Data space fundamentals

2.3.1 Useful concepts and terminology

- **Participant:** A legal entity (legal person, organization etc.) that participates in the dataspace to exchange data or provide or interact with services
 - **Consumer:** A participant that receives data or uses services
 - **Producer:** A participant that shares data or provides services
- **Decentralized Identifier DID:** The unique, cryptographically secured identity of a participant.
- **Data plane:** The data layer that includes the exchange of actual data, i.e. energy usage data
- **Control plane:** The data layer that contains meta data and control messages. Includes all the data interaction that happens to establish a data plane channel, i.e. authentication, authorization, service discovery, access policies etc

2.3.2 Data Space provisions

Data spaces are changing data exchange, in a way ensuring trust, sovereignty, and interoperability without depending on centralized storage. The overall idea of a data space is that each participant has full ownership of their data, deciding who has access to it, for what purpose, and for how long.

⁹ <https://library.oapen.org/bitstream/handle/20.500.12657/58395/1/978-3-030-98636-0.pdf#page=347>

Data Spaces facilitate:

- **Data Sovereignty** is one of the core principles of data spaces where each data owner has full authority over their data. Unlike the cloud or central storage paradigms, data in a data space remains to be stored by its original owner. This means that the owner still has the right to define who can locate, access,

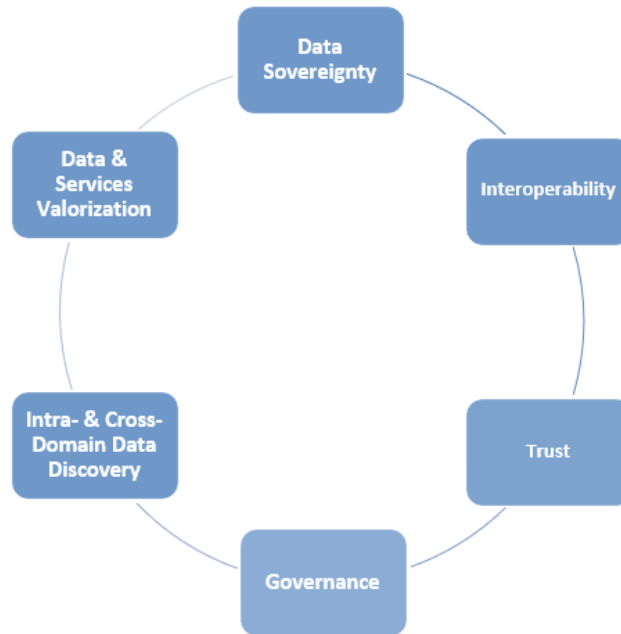


Figure 3: Data Space provisions

- and use their data and under what context and duration. No one is permitted to access or process the data unless they receive permission from the owner, hence guaranteeing privacy, security, and regulation. Sharing of data is only done after the consent of the owner has been granted, enhancing trust and self-governance in digital spaces as well as inter-industry cooperation.
- **Interoperability** at the level of data spaces is achieved through a structured strategy that offers, across systems and organizations, the smooth and seamless exchange of data. This takes into view a three-tier perspective: communication, syntax, and semantics. Communication has to do with data spaces based on common mechanisms and protocols, upon which the colleagues agree beforehand for secure and efficient interactions between participants. The syntax level guarantees an agreed-upon structure for every exchange of data, which allows different systems to analyze and process the data in a uniform way through standardized data-exchange protocols. Finally, at the semantic level, ontological alignment leverages the relevant available metadata and vocabulary hubs, thus ensuring that whatever the domain, the interpretation of the data remains the same. Aiming in this way at the integration of these three layers puts data spaces in a harmonized ecosystem where participants can effortlessly share, comprehend, and utilize data to foster cross-sector partnerships and innovation
- **Trust** in data spaces is built on identity verification, metadata validation, and decentralized security mechanisms, ensuring that all participants interact with confidence and transparency. Rather than relying on centralized identity management or third-party authorities, the schemes pertaining to data spaces are based on Self-Sovereign Identities, where an entity is able to manage and prove its own independent digital identity. The subject matter of candidates on SSI revolves around DIDs and VCs/VPs being the decentralized identifiers and verifiable credentials/verifiable presentations, respectively, which provide a secure and tamper-proof manner of identifying users and establishing a link to open data sources. Every interaction in the

data space reflects the proof of who one is and that whoever one deals with is validated; data exchange can only occur among authorized participants. These open and standardized identity schemes, then, combined with secure data exchange mechanism, provide an ecosystem that is transparent, trustworthy, and resilient

- **Governance** of data spaces is established through a pre-agreed set of rules and policies that all members must follow, thus offering an orderly, transparent, and compliant data-sharing environment. This would include rules for onboarding that define how participants join the data space along with operational interactions that set benchmarks on how data are to be accessed, shared, and used. Policy enforcement mechanisms ensure that data usage is in compliance with the agreed terms and conditions, allowing very fine-tuned control over access rights while keeping the security and compliance. Further, governance frameworks also have features concerning provenance and traceability which allow for monitoring and verification of all transactions and transactions involving data. This should also include support for billing, charging models, and smart contracts so that exchange, while secure and compliant, is also seen to be just and accountable among all participants
- **Intra- & Cross-Domain Data Discovery** is necessary to enable synergies, data openness and reuse in data spaces. Conventionally structured mechanisms help users in data spaces to find the available datasets and services in a pool of participants, query them, and synchronize access with other datasets. That means users will have no access to those datasets of his/her domain but at different domains where the principal control of access and use will lie. Accordingly, another big part that will fuse into this process is a federated catalogue. It basically acts like a decentralised metadata repository through which participants could search and index and request datasets without undergoing the problem of maintaining the meta-information in a central location. These capabilities will improve interoperability, efficiency, and cross-sector collaboration, therefore maximising the value of shared data while ensuring security and compliance
- **Data & Services Valorization** in data spaces is preeminently determined by Data Space Marketplaces that provide new options for the monetization and utilization of the data. These marketplaces the emerge of direct distribution channels for data which will allow the participants to share, trade, or license their datasets and services securely while ensuring that they remain in control of their use. Based on the valorization of existing data and services, organizations can be enabled to unleash new streams of revenues while optimizing process efficiencies and driving innovation. From collaboration within and across sectors, data space allows the building of added-value solutions across industries that foster synergies between stakeholders. This structured approach effectively reconceptualizes data from a passive depletion to a dynamic resource through maximizing it for economic and social returns

3 OPENTUNITY's FDEI Architecture

3.1 OPENTUNITY's FDEI key features and conceptual architecture

A complete Data Space implementation according to the standards set by many collaborative efforts, such as IDSA and Gaia-X is comprised of a multitude of subcomponents. These components are not all mandatory and each Data Space implementation is tailored to suit each case's requirements. Nevertheless, all Data Spaces that are built according to the recognized standards conform to and reinforce the principles of data sovereignty, interoperability, trust, and governance.

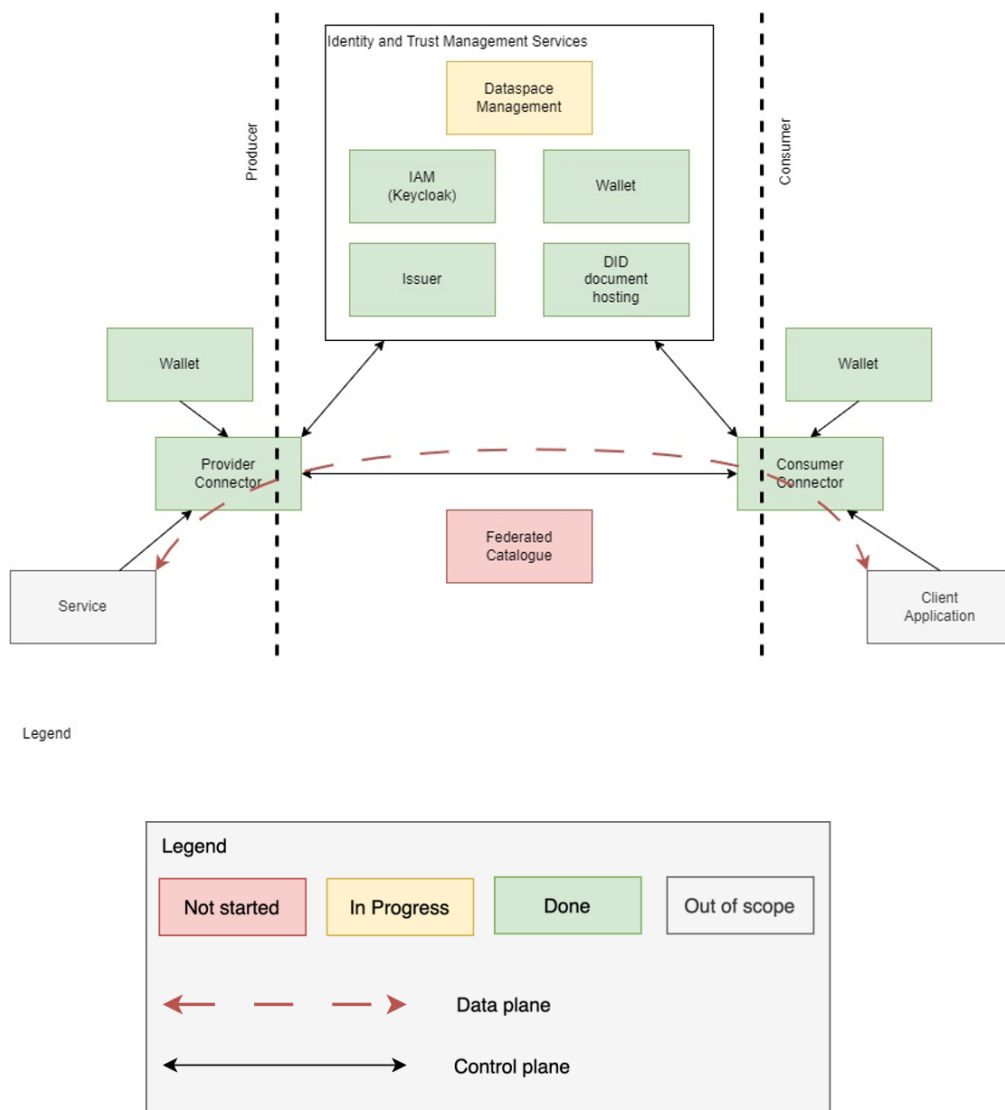


Figure 4: Conceptual architecture

As such, the OPENTUNITY Data Space has encompassed these features and, as a result, implemented the relevant subcomponents. More particularly, the OPENTUNITY Data Space implementation spans across three main components:

- Trust Services
- Data Space connector

- Federated catalogue

Each of those main building blocks is not monolithic and is built on top and around both custom software implementations as well as readily available third-party and open-source packages. The FDEI conceptual architecture is presented in Figure 4.

3.2 FDEI components

3.2.1 Trust services

In OPENTUNITY's Federated Data Exchange Infrastructure (FDEI), Trust Services are major components to ensure security and trust in data exchange. Self-Sovereign Identity (SSI) makes that possible by enabling actors to maintain full control of digital identities; using Decentralized Identifiers (DIDs) they can control the identification process autonomously and establish trust through a decentralized, tamper-immune process. Enhancements to Identity and Access Management (IAM) allow actors to authenticate, authorize, and act in relation to data while still maintaining control of its credentials.

3.2.2 Data space connector

The connector is a software module that implements and handles the data exchange protocol; it is integrated to the infrastructure of each participant and acts as a logical gatekeeper. The connector with its supporting extensions ensures that APIs, and other types of interfaces exposed by Data Space participants, are properly offered to the data space with respect to the requirements of the control plane of the data exchange protocol. The actual data transfer will reuse existing communication protocols (e.g. API over HTTPS). All data space participants that are presented below are equipped with a connector implementing the necessary functionalities and interact with other participants of the data space via this software component.

The connector is a software component that enables all interactions between data space participants and thus is involved in UC 1.2, UC 1.8 and UC 1.9. More information in *Annex I Related UCs, requirements and SGAM diagrams*.

3.2.3 Federated catalogue

The catalogue is a federated service which facilitates data discovery within a data space. This component interacts with all data space participants to collect the Verifiable Credentials/Presentations (GAIA-X Credentials) of all entities/service offerings/data products available in the data space. Each data space participant consumes the catalogue service to publish their offerings and discover products offered by other data space participants, depending on the published access and usage policies.

3.3 Data Space deployment strategy

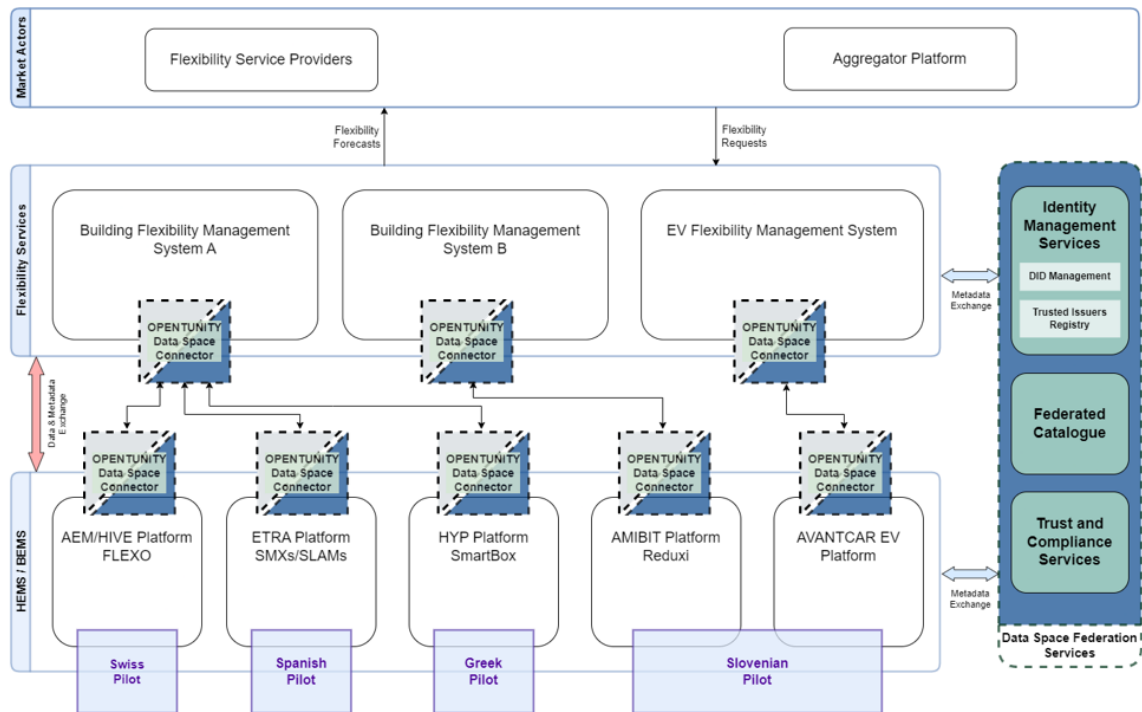


Figure 5: OPENTUNITY Data Space deployment strategy

The OPENTUNITY Data Space is going to serve the purpose of exchanging energy data and control commands in a federated manner, ensuring trust between the parties as well as conforming to the OPENTUNITY governance model. Use of the data space also allows more participants to be added in the future, with relative ease and consistency.

The above diagrams depict the 3 layers of the deployment of OPENTUNITY: Market actors, Flexibility services and HEMS/BEMS. As demonstrated, the data space components (i.e. the data space connector) will be integrated with the Flexibility Services and the HEMS/BEMS platforms, to enable the data exchange between the two layers. Each party will deploy a data space connector (part of the participant template that will be detailed in the next section of this documents). This is the crucial component that acts as a gateway between systems and orchestrates the data exchange in a federated and trusted manner.

The Data Space Federation Services on the right (comprised of the centralized components of the infrastructure) act as the authorities needed to support the connectors in their goal to provide a trusted data exchange. The Data Space Federation Services can be used – apart from the connector – by all onboarded participants and can be seen as a central clearing house that ensures security, trust, governance and discoverability.

4 FDEI Implementation

4.1 Trust services

The Trust Services component of the OPENTUNITY Data Space ensures that the participants can reliably identify each other and trust the integrity and the security of data exchanges. The main functional requirements of the component are:

- **Identity and Trust Management:** Each participant and user is assigned a unique digital identity, managed by the Trust Services. This ensures that all entities are verifiable and authenticated before they can participate in data exchanges. The Trust Services components are responsible for generating identities, signing Verifiable Credentials and Presentations and validating proofs.
- **User and Participant Onboarding:** The Trust Services include components that enable and support the onboarding of the participant. These include both the technical infrastructure and the accompanying UIs.

4.1.1 Access Levels

Access and authorization in the OPENTUNITY Data Space targets two different types of entities: participants and end-users. Participants are legal entities that are involved in the Data Space by either sharing data or receiving data (or both). End-users are actual physical users that need to be authenticated to perform various tasks, mainly administrative and maintenance.

Due to their nature, these two types of entities require a different approach regarding their authorization and identity management and verification. These different approaches mainly stem from their technical characteristics. In the Data Space context we can define two authorization levels, the Data Space level (participants) and the End-user level.

Data space level	End-user level
Using Self-Sovereign Identification (SSI) for participant authentication	JWT token-based login and Verifiable Credentials and Presentations (SSI)
Credentials (VC/VP/DID) are stored in a Participant Wallet	The IAM Server (KeyCloak) is used to generate and verify the JWT tokens
Stored credentials are used by the connectors to identify and authorize the counter-party	

Table 1: Data Space authorization levels

4.1.1.1 Participant level

The participant (or legal participant) is an organization that participates in the Data Space. To participate in a Data Space, in the current technical implementation context, means to have the required software and accompanying provisioning to be able to share data with, or retrieve data from, the Data Space - i.e. other Data Space participants.

Each interaction with the Data Space (using the Data Space connector component as we will see later) requires the identity of the participant (as configured) to be verified. For example, when requesting for data from a provider connector, the consumer connector presents, along with the data request, its credentials. The provider connector will need to verify this credential in order to make sure that the data interaction should be allowed. Part of this verification happens with the help of the centralized Identity Management component.

The credentials used to authenticate and authorize participants are based on the **Decentralized Identity (DID)** and **Self-Sovereign Identity (SSI)** technologies.

4.1.1.2 End-user level

The end-user, on the other hand, is a physical user that is accessing a Data Space related component, through the UIs that have been developed for the OPENTUNITY Data Space. Since the data interconnections facilitated by the OPENTUNITY Data Space are to be used by the platform's software clients, operations involving end-users have to do with onboarding the participant and maintaining and monitoring the related software and services.

To authenticate end-users the **JWT token-based** approach was implemented, since it is an industry standard for such use cases.

4.1.2 Centralized vs Decentralized components

One of the key dataspace principles is the preservation of data sovereignty by the data owner. Therefore, the Data Space cannot comprise only centralized components, since parts of the identification logic and the entirety of the data need to be located on a closed participant system. The implemented Data Space design facilitates this by providing two groups of components: centralized and decentralized.

The centralized components are the backbone of the Data Space. These are deployed once, are maintained by the Data Space administrator team and are used by all participants, i.e. they provide services to all the participants. The decentralized components, on the other hand, are deployed by each participant and are private, serving only one "part" of the Data Space.

The fact that a component exists in the shared/centralized deployment does not exclude it from being used as part of the decentralized group of components, albeit as a different instance. Some components – such as the wallet – are deployed in both the central OPENTUNITY infrastructure and the participant premises, since their functionality is required in both environments.

Centralized components	Decentralized components
Credentials-Manager	Wallet (Walt.Id) *
Onboarding Portal	Credentials-Manager *
IAM service (KeyCloak)	Participant Management UI *
Credential Issuer	Data Space Connector *

Table 2: Centralized vs Decentralized components

4.1.3 Trust services subcomponents

As mentioned before, the Trust Services component of the OPENTUNITY Data Space comprises a multitude of software components. These components are either based on third-party open-source software or custom developed exclusively for OPENTUNITY. Where the implementation is based on a third-party software package, the rationale for the latter's selection is provided. In the case of custom developed modules, the technical design and implementation details are presented.

4.1.3.1 Identity and Access Management

Identity and Access Management (IAM) is a collection of policies, technologies and processes that, in combination, ensure that the right individuals have access to a software system. An IAM system, usually a software package provides the following functionality:

- **Identification:** Identifying a user or software system
- **Authentication:** Verifying the user/software credentials (passwords, API keys, MFA etc.)
- **Authorization:** Grant or restrict access to protected systems
- **Access Control:** Enforce policies such as Role-Based Access Control (RBAC) or Attribute-Based Access Control (ABAC)
- **Audit:** Logging and recording of activity

There are many IAM third-party software packages that come in many forms. Open-source, enterprise or cloud-based products exist and most cover all the required functionalities. In the case of OPENTUNITY an open-source, on-premises solution is required and as such the following options exist:

- **Keycloak:** Keycloak is an open-source software product to allow single sign-on with identity and access management aimed at modern applications and services. Until April 2023, this WildFly community project was under the stewardship of Red Hat, who used it as the upstream project for their Red Hat build of Keycloak. In April 2023, Keycloak was donated to the CNCF and joined the foundation as an incubating project. Keycloak supports various protocols such as OpenID, OAuth version 2.0 and SAML and provides features such as user management, two-factor authentication, permissions and roles management, creating token services, etc. It is possible to integrate Keycloak with other technologies, such as front-end frameworks like React or Angular, as well as containerization solutions like Docker.¹⁰
- **WSO2 Identity Server:** WSO2 Identity Server is a modern identity and access management solution for on-premises or cloud environment. It enables organizations to deliver trusted digital experiences to all types of users. WSO2 Identity Server is available as open source under the free Apache 2 software license. Typical organizations can scale WSO2 Identity

¹⁰ <https://en.wikipedia.org/wiki/Keycloak>

Server to support millions of customers without incurring additional subscription licensing costs. WSO2 Identity Server is a developer-friendly access management solution. It offers templates for adding apps and authentication methods, a visual editor to preview the login experience at build time, SDKs, extensive documentation, and more.¹¹

- **FreeIPA:** FreeIPA is an integrated security information management solution combining Linux (Fedora), 389 Directory Server, MIT Kerberos, NTP, DNS, Dogtag (Certificate System). It consists of a web interface and command-line administration tools. FreeIPA is an integrated Identity and Authentication solution for Linux/UNIX networked environments. A FreeIPA server provides centralized authentication, authorization and account information by storing data about user, groups, hosts and other objects necessary to manage the security aspects of a network of computers. FreeIPA is built on top of well-known Open Source components and standard protocols with a very strong focus on ease of management and automation of installation and configuration tasks.¹²

The IAM solution choice for OPENTUNITY is **KeyCloak**. KeyCloak is an extremely popular open-source software that exhibits the following strong points, among others:

- **Broad protocol support out of the box:** Keycloak supports a wide range of standard protocols (OpenID Connect, OAuth 2.0, SAML 2.0) natively, making it highly interoperable with modern applications.
- **Ease of use:** Keycloak provides a user-friendly web-based Admin Console that simplifies configuration, user management, and policy setup.
- **Strong community:** As a Red Hat-backed project and a Cloud Native Computing Foundation (CNCF) incubating project, Keycloak benefits from a large, active community and enterprise-grade stability.
- **Cost:** Keycloak is completely free with no hidden licensing costs, unlike some competitors.
- **Lightweight:** Keycloak runs efficiently with modest resources.

¹¹ <https://wso2.com/identity-server/>

¹² <https://www.freeipa.org/About.html>

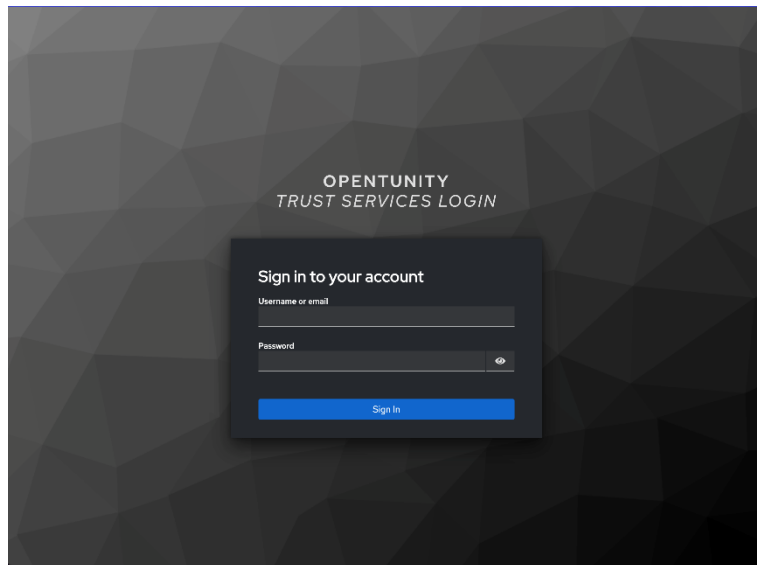


Figure 6 IAM (KeyCloak) login screen

The Data Space roles defined as per project requirements are consumer and provider and as such, corresponding roles have been created in KeyCloak in the OPENTUNITY realm. A participant is allowed to have both roles.

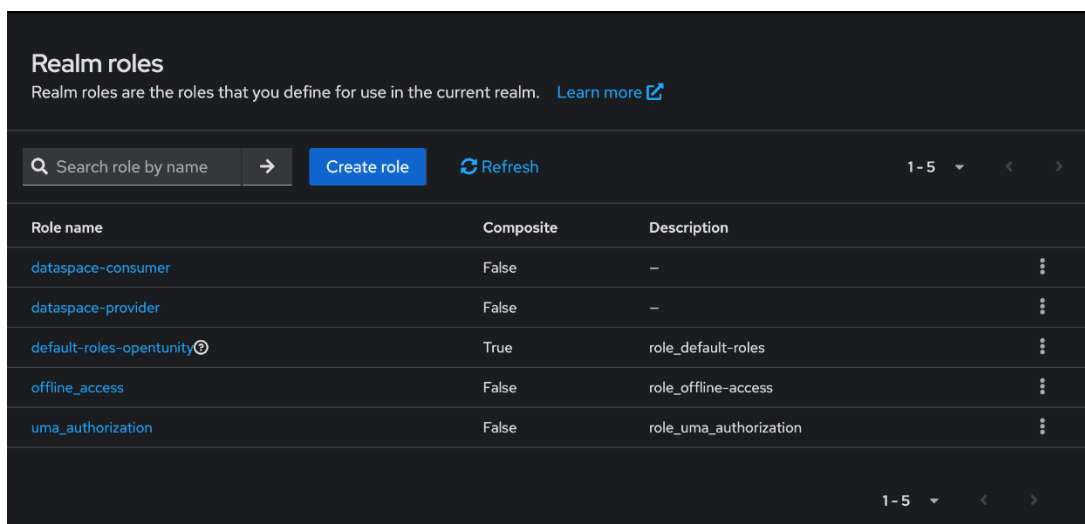


Figure 7 OPENTUNITY realm roles

During participant onboarding, a group is created (for each participant). Participant users are added to the participant group as they are created.

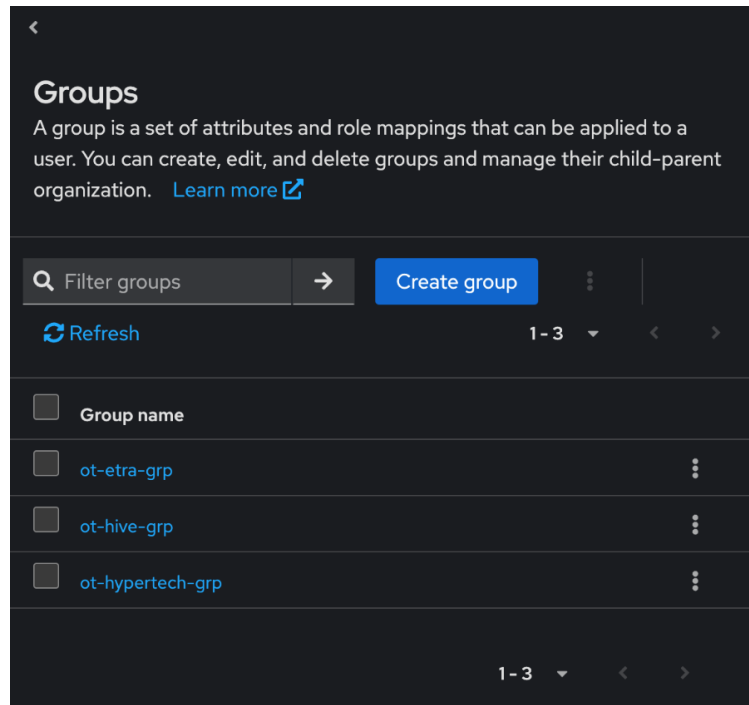


Figure 8 Participant groups in KeyCloak

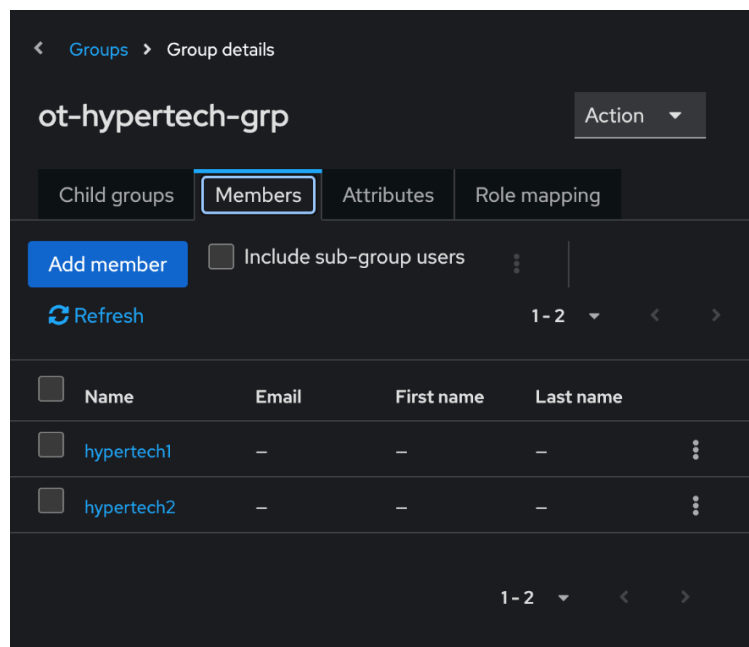


Figure 9 Example users in the Hypertech participant group

4.1.3.2 Wallet

In order to securely create, use and validate verifiable credentials and presentations (as part of the Self-Sovereign Identities functionality in the OPENTUNITY Data Space), a wallet software is needed. This software is deployed in the relevant environment and should satisfy the following functional and non-functional requirements:

- Store **Decentralized Identifiers** (DID) and **Verifiable Credentials**
- Allow users to securely manage their identity data on their devices (or servers)
- Share credentials and DIDs where required
- **Issue** Verifiable Credentials (VC) and Presentations (VP)
- **Verify** Verifiable Credentials and Presentations
- Support the **OIDC4VCI** protocol for exchange of VCs/VPs
- **Open-Source** license

Since the SSI market is still emerging, while a lot of products are available, no single dominant player exists. Competitors vary in their focus—some prioritize enterprise adoption (Evernym, Sphery), others individual empowerment (SelfKey), and some developer tools (Veramo, Trinsic).

The selected product is **Walt.Id**, which satisfies all of the above requirements, both functional and non-functional. Furthermore it provides an easy-to-use and ease-to-integrate HTTP API. Walt.Id is deployed via docker and in the OPENTUNITY Data Space context is both a centralized (OPENTUNITY infrastructure) and decentralized (participant premises) component.

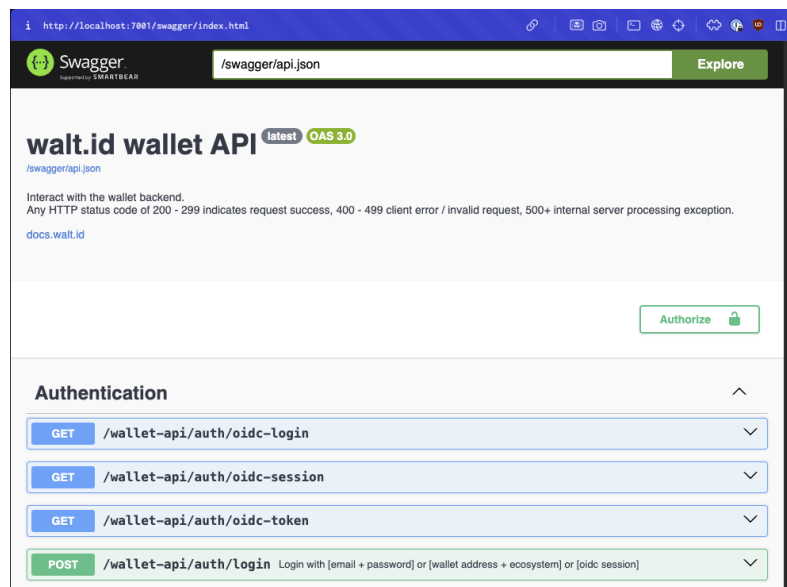


Figure 10 Walt.Id HTTP API

4.1.3.3 Identity Provider (IdP)

The Identity Provider (IdP) component is a custom-built centralized component that enables the OPENTUNITY Data Space to leverage Self-Sovereign Identities in order to facilitate trust between the participants through their connectors. It is being developed using Python and provides an HTTP API for the data space to use.

Its functionality is split between functional requirements which can also be seen from the individual endpoints that it provides.

- **Onboarding:** Supports the onboarding process by collecting user information and facilitating the generation of the pre-configured participant template
- **Participants:** Manages the participants
- **Wallet:** Help to create and manage Walt.Id wallets
- **Issuer:** Endpoint that issues Verifiable Credentials/Presentations
- **Verifier:** Endpoint that verifies VC/VPs
- **Catalogue:** Allows the publishing of Participant and Service Offering VC/VPs to the Federated Catalogue

Issuer			^
POST	/issuer/api_token	Update Hashed Key	🔒 ↓
POST	/issuer/vc/DataCellarCredential	Generate Credential Offer	🔒 ↓
POST	/issuer/vc/TermsAndConditions	Issue Vc Terms And Conditions	🔒 ↓
POST	/issuer/vc/LegalRegistrationNumber	Issue Vc Legal Registration Number	🔒 ↓
POST	/issuer/vc/LegalParticipant	Issue Vc Legal Registration Number	🔒 ↓
POST	/issuer/vp/sign	Vp Issuer Sign	🔒 ↓
POST	/issuer/sd/sign	Sd Issuer Sign	🔒 ↓
Verifier			^
POST	/verify/proof	Verify Credential Signature	↓
Gaia-x			^
POST	/gaiax/compliance	Get Gaiax Compliance	🔒 ↓
Catalogue			^
POST	/catalogue/participant/register	Register Participant To Catalogue	🔒 ↓

Figure 11 Part of the IdP OpenAPI Specification

The IdP is a dockerized application and depends on the wallet and KeyCloak. It supports two levels of authentication: **Credential** and **API Key**.

Credential authentication is based on a **bearer token** generated by KeyCloak, as it is integrated with the OPENTUNITY IAM service in order to authenticate the user of the API. This is used when users use the **onboarding UI** to register and get their preconfigured participant template.

The API Key authentication is used by the participant template components in order to request issuance or verification of credentials. The API Key is generated from the IdP and is distributed via the preconfigured participant template. The API Key is shared when created and is stored hashed in the IdP. This means that if the API Key is lost it cannot be recovered and has to be regenerated and updated in the participant template configuration.

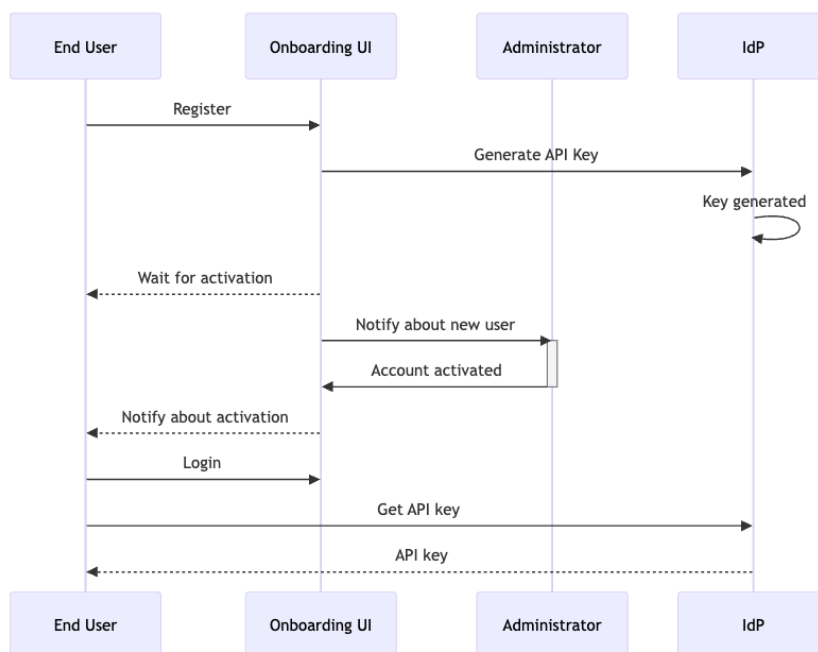


Figure 12 Generation of API Token

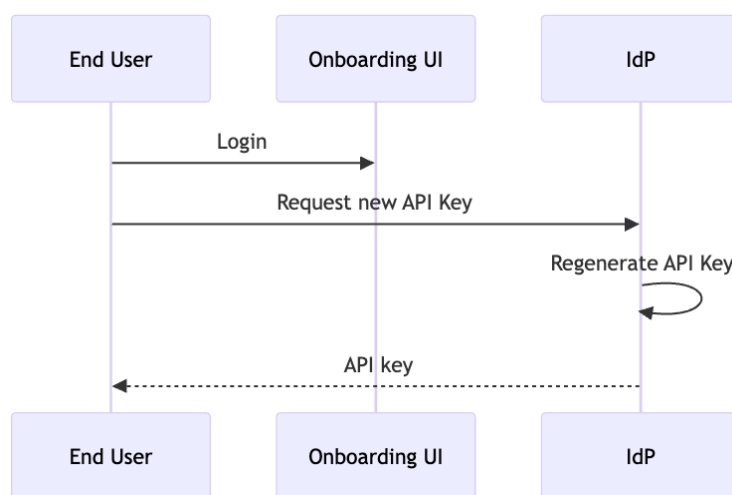


Figure 13 Regeneration of API Key

4.1.3.4 Onboarding UI

A participant joins the OPENTUNITY Data Space by going through the onboarding process that is facilitated by the custom-built Onboarding Portal. This is a web application developed using **Python**

and **React**. The Onboarding Portal is deployed using Docker and is part of the centralized components of the Data Space.

A user registers to the Onboarding Portal via an open, publicly accessible form and has to go through an approval and activation process, where an OPENTUNITY administrator manually reviews and admits him.

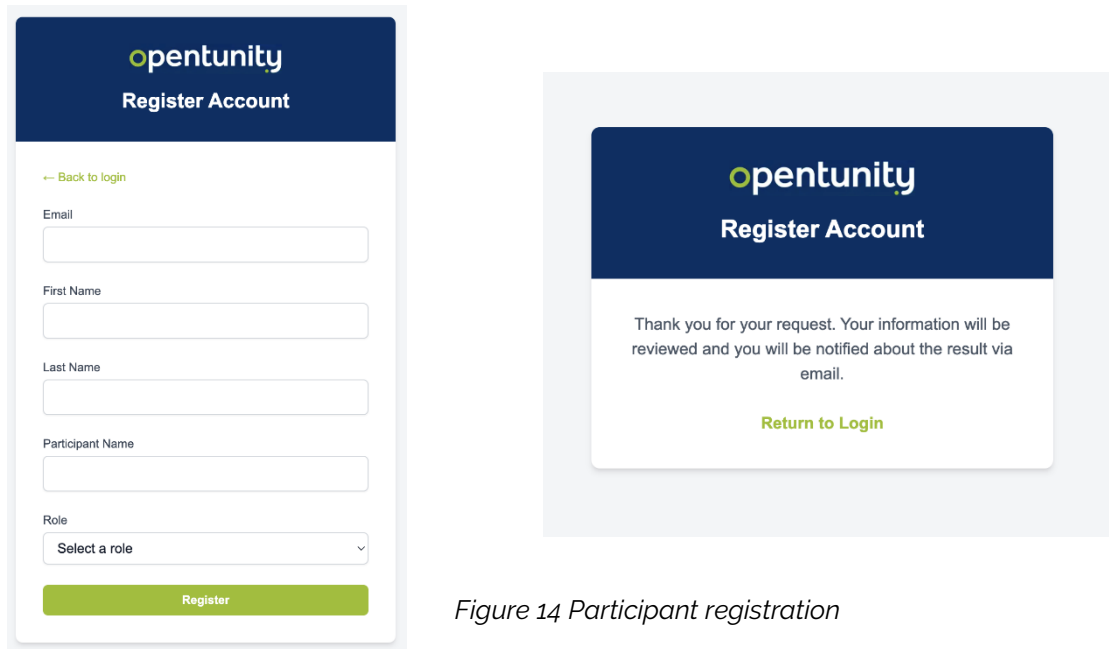


Figure 14 Participant registration

After the user is activated by the administrator, they can then log into the system, where they will proceed with entering the needed information in order to deploy their participant template, the software bundle that technically enables their organization to act as an OPENTUNITY Data Space participant.

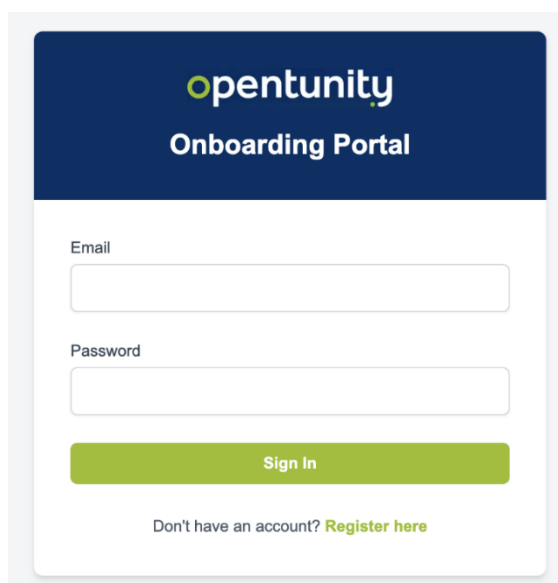


Figure 15 Onboarding UI login screen

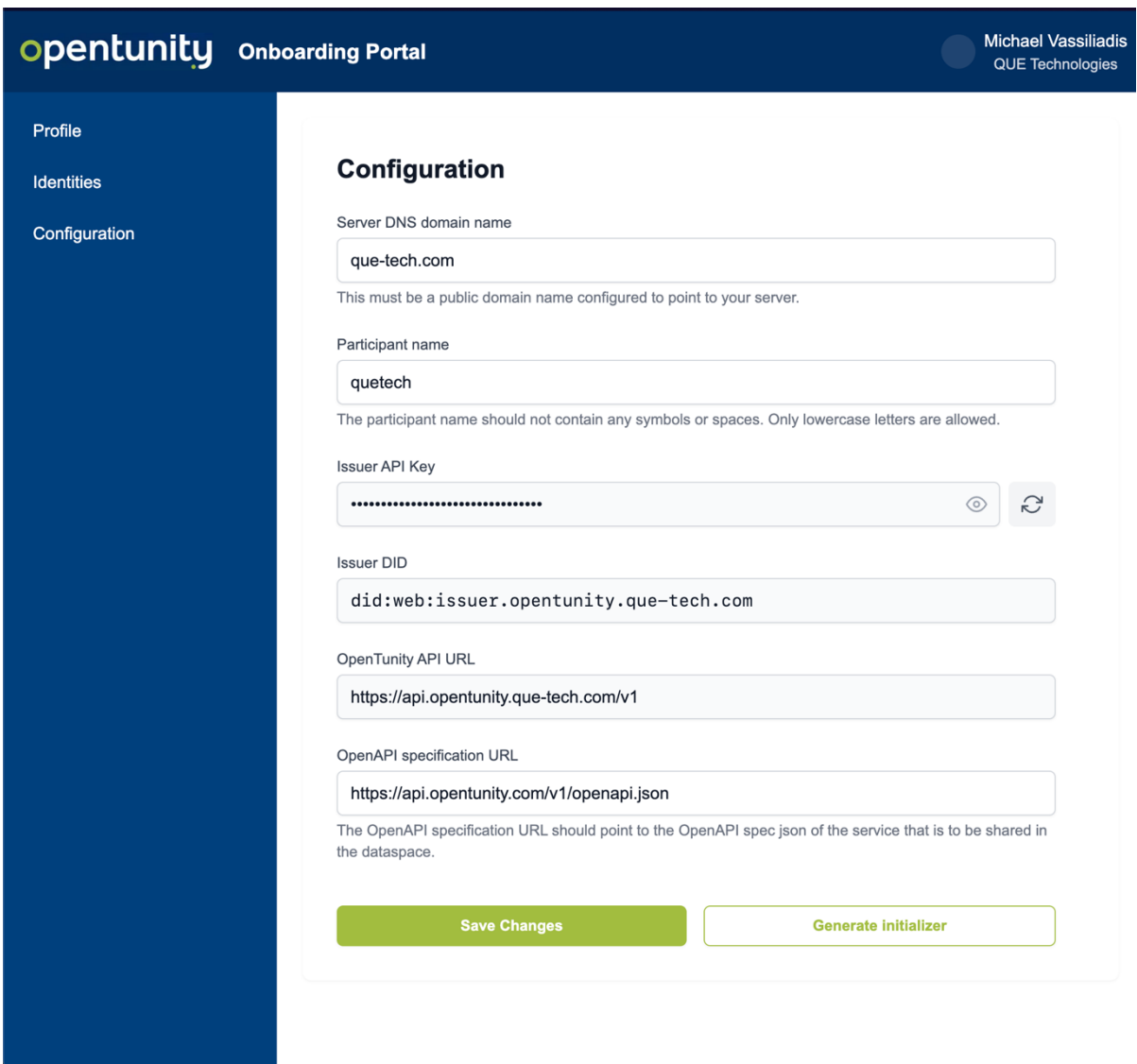


Figure 16 Participant configuration

In the configuration screen shown in *Figure 16* the user has to enter technical details needed for the software in order to bootstrap the participant and connect to the dataspace. The information needed is:

- **Server DNS domain name:** The domain name where the connector will be reachable
- **Participant Name:** The name of the participant. The full URL of the connector will be: `https://<PARTICIPANT_NAME>.<SERVER_DNS_DOMAIN_NAME>`
- **Participant role:** The role of the participant. Could be provider, consumer or both.
- **Backend OpenAPI specification URL:** If the participant is a provider (or both provider and consumer), this is the URL where the OpenAPI specification of the service - that is to be shared with the dataspace - is located.
- **Backend OpenAPI authentication method and details:** If the OpenAPI specification needs authentication, these should be filled with the correct authentication method and credentials, in order or the connector to gain access.

After filling the required information the system is ready to generate a preconfigured software bundle, the **participant template**. This bundle – which will be described in the next section – contains

all components needed to be deployed in order for an entity to become a participant in the OPENTUNITY Data Space. Clicking "Generate initializer" will commence the process, that will ultimately lead to the download of the archive file.

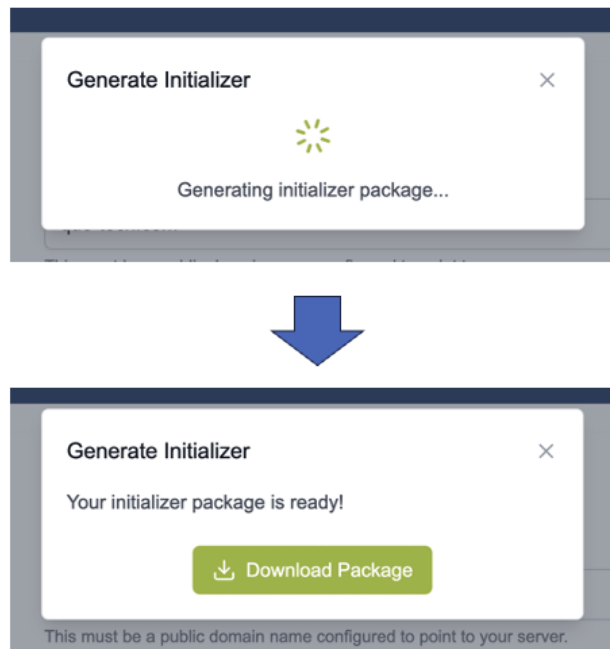


Figure 17 The user is downloading the preconfigured participant template.

4.1.3.5 Participant Template

Deploying the software required to participate in a data space can be a daunting task. A multitude of components, each with its specific configuration, are needed; testing and verifying their interactions, both between each other as well as with the centralized components of the data space is a very technical task that requires very particular knowledge that no participant should have by default. For this reason, to make onboarding easier and more streamlined, the **participant template** was created.

The **participant template** is a collection of software components, along with configuration and scripts that help bootstrap the environment needed for the participant. This package is obtained from the **Onboarding portal**, along with instructions about how to use it, and makes deploying a participant a matter of transferring the file into a server and executing a single command.

4.1.3.5.1. Component breakdown

4.1.3.5.1.1 Task

Task is both a task runner and build tool that tries to simplify and make it more straightforward to declare and run tasks within software development projects. Task provides an alternative to existing tools like Make with a newer, simpler solution. Task is a statically compiled binary written in Go and uses a simple to read YAML-based configuration file, typically `Taskfile.yml`, to specify

tasks. Such tasks can be building, testing, deploying software packages, or any other repetitive task that in need for automation. Some of Task's most interesting features include ease of installation (download a binary or install via package managers like Homebrew), cross-platform compatibility (runs on Linux, macOS, and Windows thanks to an embedded Go shell interpreter), and ease of integration into continuous integration pipelines. It supports the declaration of tasks in terms of commands, dependencies, variables, and conditions, and is therefore very flexible. Task also provides some higher-level functionality like file watching (to execute tasks when files change), environment variable handling, and inclusion of other Taskfiles for modular installs.

In the participant template, Task is used to orchestrate the entire process of setting up the participant and then executing everything needed to start the software, mostly starting the required containers. For normal operation a single task command is needed:

```
task start-all
```

This starts the entire software stack for the data space participant. In case that manual configuration is needed, for example if the configuration was not generated by the Onboarding Portal, or if there is a custom deployment needed, the Taskfile of the participant template provides granular control via individual commands:

- `start-proxy`: Starts the required reverse-proxy, that manages the APIs and services exposed by the software stack of the participant template.
- `setup-participant`: Reconfigures the participant software by asking the user to input various configuration parameters.
- `create-network`: Creates the internal docker network.
- `generate-certs`: Generate the certificates needed to support the participant DID.
- `import-certs`: Imports pre-generated certificates.
- `create-legalparticipant`: Creates the legal participant verifiable credential. Contacts the IdM for generation and signing.
- `start-all` (within the participant template): Starts the containers
- `stop-all` (within the participant template):

4.1.3.5.1.2 Docker/compose

Docker is an open-source software that helps automate the deployment, scaling, and management of applications using containers. Containers are lightweight and portable units that package an application along with its dependencies (libraries, configuration files, etc.) so it can run across different environments, like development, testing, or production systems. Essentially, Docker makes it easy to build, ship, and run applications by isolating them in these containers.

Docker Compose, on the other hand, is a tool that works with Docker to create and manage multi-container applications. It uses a YAML file, usually called `docker-compose.yml`, where you indicate the services, networks, and volumes your application needs. With one command, you can start, stop, or rebuild all the defined services, thus orchestrating a more complex application with

interrelated containers—a web server, database, and caching system, for instance—without having to manage each one manually.

The participant template makes use of the following images:

- `caddy:2.7`
- `opentunity/credentials-api`
- `postgres:14`
- `nginx`
- `waltid/wallet-api:1.0.2410150830-SNAPSHOT`
- `opentunity/edc-connector`

4.1.3.5.1.3 Bash

Bash, or "Bourne Again Shell," is a command-line shell and Unix-like operating system scripting language. It is a continuation of the original Bourne Shell (`sh`), written in the late 1980s by Brian Fox for the GNU Project. Bash is both an interactive shell—where one can type commands to interact with the operating system—and a scripting tool to automate tasks by writing strings of commands in files called shell scripts. Using Bash, you can do things such as navigating the file system (e.g., `ls`, `cd`), managing processes (e.g., `ps`, `kill`), file management (e.g., `cp`, `mv`, `rm`), and chaining commands together or redirecting them (`>`). It is highly configurable, with features such as variables, loops, conditionals, and functions, which are both appropriate for mundane tasks and complex automation.

The participant template contains the following bash scripts:

- `./install/install_docker.sh`
Installs docker to the current system
- `./install/install_taskfile.sh`
Installs task to the current system
- `./deploy/setup-participant.sh`
Configures the participant template
- `./participant-template/scripts/prepare-certs.sh`
Prepares the (preexisting) certificate for use from the participant template. Uses **openssl** to export the private and public keys in **pkcs12** and **x509** formats.

4.1.3.5.1.4 Caddy

Caddy¹³ is an open-source web server written in the Go programming language and regarded as simple, secure, and modern. Caddy is distinct from traditional web servers like Apache or Nginx since it is simpler and has more built-in features, and most importantly, automatic HTTPS. At its core, Caddy

¹³ <https://github.com/admpub/caddy>

serves HTTP, HTTPS, and even more modern protocols like HTTP/3 and QUIC, making it a modern choice for hosting websites and applications. Its most striking feature is auto-TLS certificate management with Let's Encrypt, meaning it secures your with HTTPS by default without needing you to set up or renew your certificate manually. It does so transparently—be it public domains, localhost, or internal IPs—using a self-managed certificate authority where needed.

Caddy is extremely configurable, mostly by means of a human-readable file known as the `Caddyfile`, although it also has JSON and a RESTful API available for dynamic configuration.. The Caddyfile deployed for the participant template reverse proxy exposes the following ports/services:

- **OpenAPI (Swagger) for the wallet API:** `/swagger/*`
- **The wallet API:** `/wallet-api/*`
- **DID document hosting:** `/.well-known/*`
- **Verifiable Credential hosting:** `/vc/*`
- **Verifiable Presentation hostign:** `/vp/*`
- **Credentials manager API:** `/api/v1/*`
- **OpenAPI (Swagger) for the credentials manager:** `/docs/*`
- **Participant Manager Web UI:** `/*`

4.1.3.5.1.5 EDC Connector

The EDC Connector is deployed in order to allow the participant software to communicate with the rest of the data space. Its design, implementation and use is described in detail in Section 4.2

4.1.3.5.1.6 Credentials Manager

The credentials manager is a helper application that was developed for the OPENTUNITY Data Space. It provides all required functions that manipulate verifiable credentials, DIDs and notarization (via the IdP). These functions are provided via an HTTP API.

Its main functions are the following:

- **Wallet management:** Handles digital wallet creation, authentication, and management
- **DID operations:** Creates and manages DIDs, particularly `did:web` identifiers
- **Credential management**
 - Issues verifiable credentials (VCs)
 - Manages verifiable presentations (VPs)
 - Handles credential verification
 - Supports credential exchange

The credentials manager is developed with Python. It uses several popular Python libraries and integrates with other data space components, namely:

- The local `Walt.id` service

- The Gaia-X compliance service
- The IdP issuer service

A docker-based deployment using docker compose has been implemented. The PM2 process manager is utilized to keep track of the service process. Supported authentication methods are JWT-based authentication and API key authentication (to access the IdP).

Furthermore, an SQLite database is used for local data persistence, environment-based configuration (via .env file) and dual environment support (development and production).

Auth		^
POST	/login Login	▼
DIDs		^
GET	/dids Get Dids	🔒 ▼
POST	/dids/create/web Create Did	🔒 ▼
DELETE	/dids/{did} Delete Did	🔒 ▼
Credentials		^
GET	/credentials Get Credentials	🔒 ▼
GET	/credentials/{credentialId} View Credential	🔒 ▼
DELETE	/credentials/{credentialId} Delete Credential	🔒 ▼
Credential exchange		^
POST	/credentials/useOfferRequest Accept Credential Offer	🔒 🔒 🔒 ▼
POST	/credentials/matchCredentialsForPresentationDefinition Match Credentials For Presentation Definition	🔒 ▼
OpenTunity		^
POST	/vc/TermsAndConditions Get Terms And Conditions	🔒 ▼
POST	/vc/LegalRegistrationNumber Get Legal Registration Number	🔒 ▼
POST	/vc/LegalParticipant Get Legal Participant	🔒 ▼
POST	/vp/self_sign Vp Self Sign	🔒 ▼
POST	/vp/issuer_sign Vp Issuer Sign	🔒 ▼
POST	/catalogue/participant/register Register Participant To Catalogue	🔒 ▼
Verifier		^
POST	/verify/proof Verify Credential Signature	▼

Figure 18 The credential-manager OpenAPI specification in Swagger format

4.1.3.5.1.7 Walt.id Wallet

The Walt.id wallet is deployed in order to store and retrieve DIDs and Verifiable Credentials. Its use is described in 4.1.3.2.

4.1.3.5.2. Directory structure

This is the directory structure within the participant template. The root directory contains a README .md file with usage instructions.

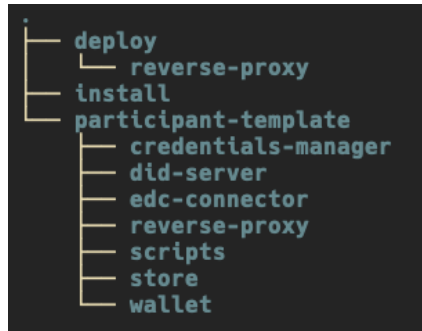


Figure 19 The participant template directory structure

4.1.3.6 Participant Management UI

The Participant Management UI is a tool that allows participant administrators to monitor and test their participant template deployment. The most important functionalities it provides are the following:

- **Connector monitoring:** Enables monitoring and manipulation of the local data space connector. Shows the local catalogue and the state of the connector
- **Wallet management:** Allows to view the contents of the local wallet (DIDs, VCs/VPs), recreate and delete credentials and DIDs.
- **Configuration viewer:** Shows the current participant template configuration parameters.

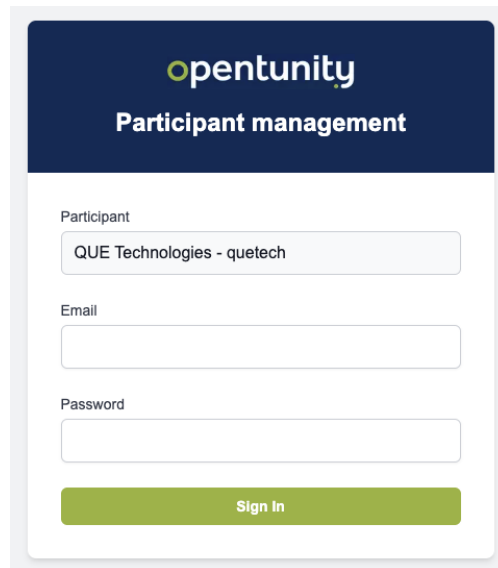


Figure 20 The participant manager login screen

The Participant Management UI requires the user to login. This is a **JWT-based** credential authentication that is supported by the – centralized – IAM service (**KeyCloak**).

The wallet (as per Walt.id design) requires a PIN to be unlocked. This is a second layer of security (the first being the **JWT-based** credential authentication required to log into the Participant Management UI) in order to further secure the sensitive wallet contents.

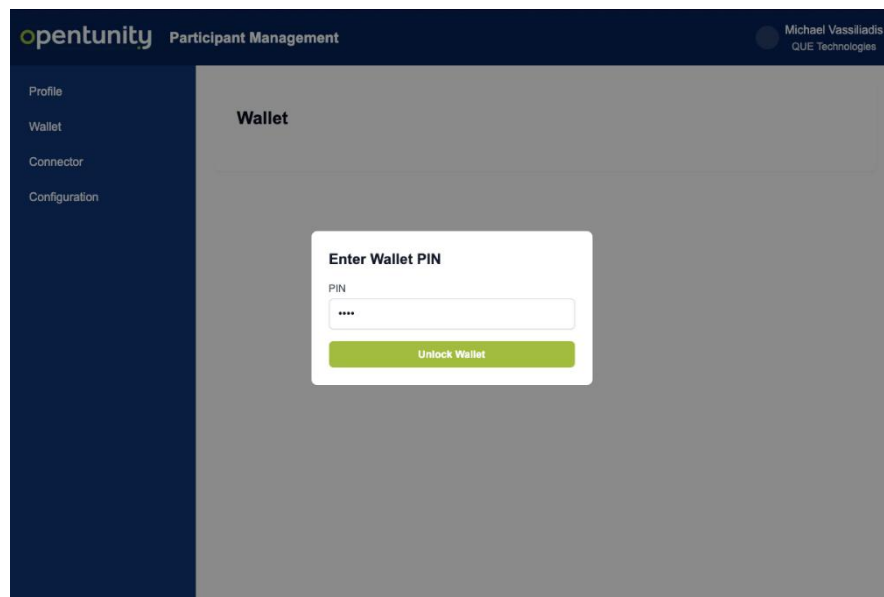


Figure 21 Wallet unlocking

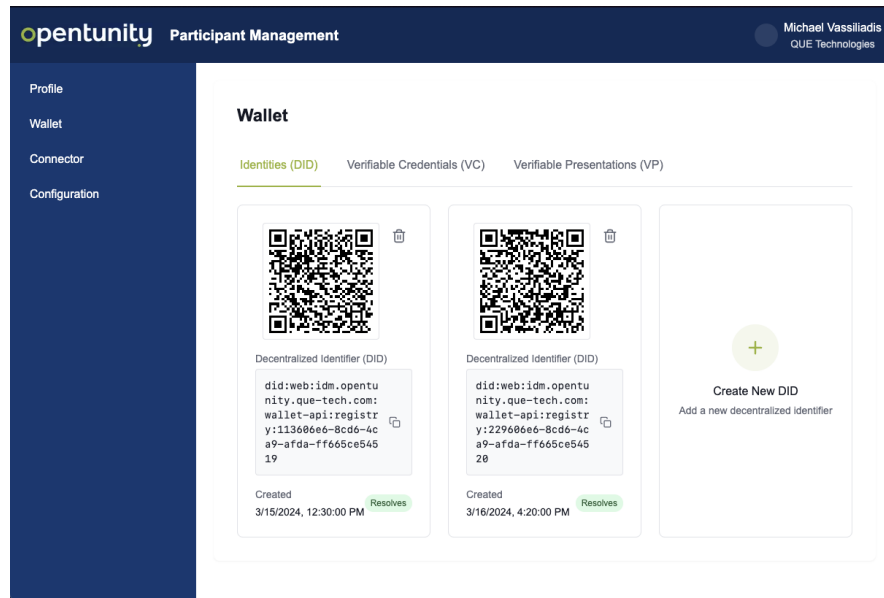


Figure 22 Wallet contents

After unlocking the wallet, the user can browse through the available DIDs, Verifiable Credentials and Verifiable Presentations. A Verifiable Credential (VC) is defined as a digitally signed assertion by a trusted party. This validates some set of characteristics, qualifications, and identities in a secure and authentic manner. A Verifiable Presentation (VP) is used to pass on one or more Verifiable Credentials. It provides participants to prove claims with privacy and control over their data. These contents can also be deleted and/or recreated, although these actions must be handled with care, since the participant template will cease to operate if the required credentials are missing.

Local Connector
Remote Connector

Connector ID
113606e6-8cd6-4ca9-afda-ff665ce54519

Type
dcat:Catalog

Connector URL

Status
● Running

Connector Catalogue ↻ Refresh

ID	NAME	TYPE	ODRL POLICY
GET-energy-consumption-data	GET /energy/consumption/data	dcat:Dataset	View Policy ↗
POST-energy-efficiency-metrics	POST /energy/efficiency/metrics	dcat:Dataset	View Policy ↗
GET-energy-forecast-weekly	GET /energy/forecast/weekly	dcat:Dataset	View Policy ↗
PUT-energy-threshold-settings	PUT /energy/threshold/settings	dcat:Dataset	View Policy ↗
GET-energy-peak-hours	GET /energy/peak/hours	dcat:Dataset	View Policy ↗
POST-energy-optimization-rules	POST /energy/optimization/rules	dcat:Dataset	View Policy ↗

Figure 23 Local connector status and catalogue

The Participant manager also allows the user to connect to a remote connector and retrieve its catalog, as well as initiate a contract negotiation and call a remote service via the data space mechanisms. This is a useful tool for developers that are adapting their tools/clients to work via the data space.

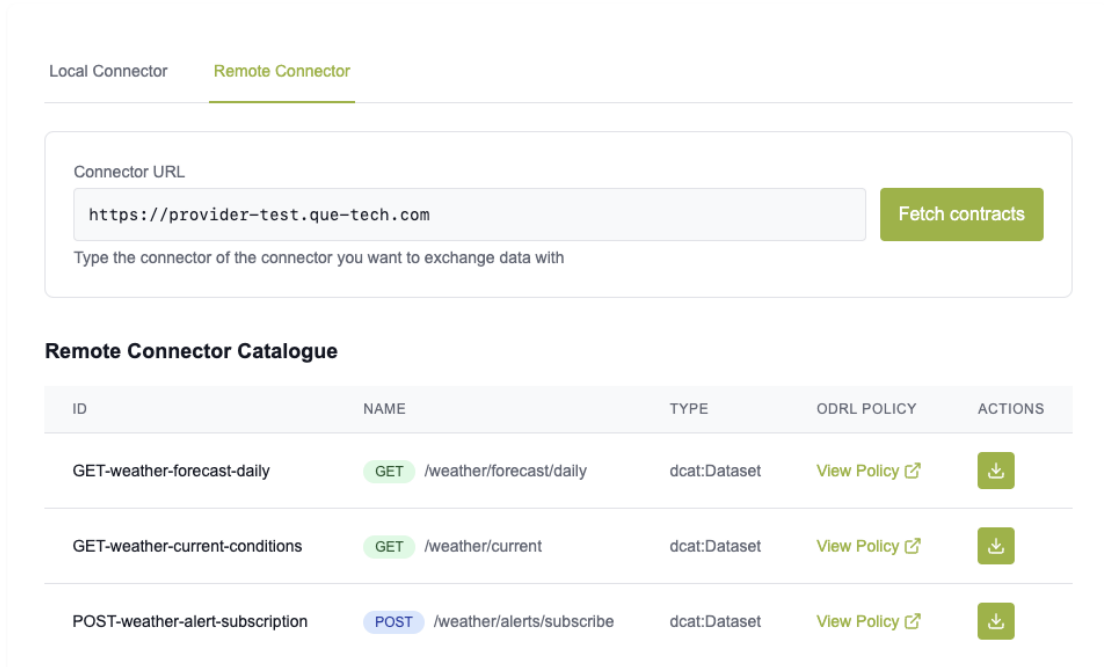


Figure 24 Remote connector catalog retrieval

The remote connector catalogue results are identical to the local connector. Each contract offering contained in the connector is listed along with the buttons that view the ODRL policy of the contract, the button (download icon) to commence the contract negotiation and finally (if successful) actually call the remote service.



Figure 25 End-to-end dataspace service call test

The Participant Manager UI has been developed in **Python** (backend) and **React** (front-end). It is deployed using docker (compose) and is fully integrated with the participant template for configuration and deployment.

4.2 Data Space Connector

4.2.1 Available data space connector implementations

4.2.1.1 Eclipse Dataspace Components (EDC)

The Eclipse Dataspace Components (EDC) project is an open-source framework under the Eclipse Foundation, designed for sovereign, inter-organizational data exchange. It aligns with the International Data Spaces (IDS) standard and the Gaia-X Trust Framework.

Key features include:

- Separation of the control plane (responsible for contract negotiation and policy enforcement) and the data plane (handling data transfer).
- An extensible architecture that supports multiple protocols, such as HTTPS, REST, and S3.
- Offered in the form of Java Libraries that allow the rapid creation of a variety of data space related components and tools.
- Detailed documentation covering all aspects of connector design, implementation and extension

The EDC framework is an active initiative, widely used in projects like Catena-X (automotive) and Mobility Data Space.

4.2.1.2 TRUE Connector

The TRUE Connector is developed by Engineering, a leading digital transformation company based in Italy. The connector is an open-source solution designed to enable self-determined data sharing while ensuring compliance with regulations such as GDPR. Initially focused on the manufacturing domain, the TRUE Connector has proven its versatility across diverse sectors including circular economy, energy, smart buildings, and agri-food domains¹⁴.

The TRUE connector has received the IDS Certificate¹⁵. The certification process, overseen by IDSA and conducted by the independent evaluation facility SQS, ensures adherence to the highest functionality and security standards.

The TRUE Connector is composed of three components:

- **Execution Core Container** (ECC), in charge of the data exchange through the IDS ecosystem using the IDS Information Model for the data representation and interacting with an external

¹⁴ <https://github.com/International-Data-Spaces-Association/true-connector>

¹⁵ https://github.com/International-Data-Spaces-Association/true-connector/blob/main/doc/IDSA_certificate_ENG_final.pdf

Identity Provider¹⁶. It is also able to communicate with an IDS Broker for registering and querying information.

- **Back-End (BE) Data Application**, represents a trivial data application for generating and consuming data on top of the ECC component.
- **Usage-Control (UC) Data Application**, a customized version of the Platoon base application for integrating Usage Control functionality. This version of Usage control application requires persistence layer, and it is included in this setup, it is H2 in memory database, with file persistence, but if required, it can be changed with PostgreSQL database.

4.2.1.3 Fraunhofer Dataspace Connector (IDS reference)

The Fraunhofer Dataspace Connector is an implementation of an IDS connector component following the IDS Reference Architecture Model. It integrates the IDS Information Model and uses the IDS Messaging Services for IDS functionalities and message handling. The core component in this repository provides a REST API for loading, updating, and deleting resources with local or remote data enriched by its metadata. It supports IDS conform message handling with other IDS connectors and components and implements usage control for selected IDS usage policy patterns.¹⁷

The Fraunhofer Dataspace Connector is no longer maintained (as of 2022). The code maintenance has been taken over from Sovity, a German company that offers dataspace solutions, in the form of Dataspace-As-A-Service products. Sovity also offers an on-premises community version of a dataspace connector, based on EDC¹⁸.

4.2.1.4 TNO Security Gateway

The Netherlands Organization for Applied Scientific Research (TNO)¹⁹ maintains a research-oriented Dataspace Connector, aligned with IDS principles. This is an experimental implementation focused on interoperability and policy enforcement. It is lightweight and customizable, but less mature and not as widely adopted for production use²⁰.

4.2.2 Eclipse Dataspace Components (EDC)

The **Eclipse Dataspace Components (EDC)** is the Data Space connector chosen for OPENTUNITY. It is a comprehensive framework (concept, architecture, code, samples) providing a basic set of features (functional and non-functional) that dataspace implementations can re-use and customize by leveraging the framework's defined APIs and ensure interoperability by design. It is

¹⁶ <https://github.com/Engineering-Research-and-Development/true-connector>

¹⁷ <https://github.com/FraunhoferISST/DataspaceConnector>

¹⁸ <https://github.com/sovity/edc-ce>

¹⁹ <https://www.tno.nl/nl/>

²⁰ <https://tsg.dataspac.es/>

powered by the specifications of the Gaia-X AISBL Trust Framework and the IDSA Dataspace protocol.²¹

The EDC is designed for developers who want to build dataspace implementations on an existing, standards-based framework and adapt it with their own solutions: Developers use the EDC to build data-sharing services for their customers.

The framework consists of a set of components and corresponding capabilities that are mandatory to implement a dataspace:

- Connector
- Federated Catalog
- Identity Hub
- Registration Service
- Data Dashboard (Management UI)

The project also provides repositories to support the onboarding of developers with simplified, essential samplings:

- Samples
- Demonstrator (MVD)

The EDC project is governed by the Eclipse Foundation. Its project management, implementation, communication, and compliance follow the corresponding roles and rules, based on the principles of openness, transparency, and meritocracy. EDC is architected as modules called extensions that can be combined and customized to create components that perform specific tasks. These components (the "C" in EDC) are not what is commonly referred to as "microservices." Rather, EDC components may be deployed as separate services or collocated in a runtime process. This section provides a quick overview of the key EDC components.

4.2.3 Extensions developed

It's accurate to say that EDC, at its core, is just a modular system that contributes features as extensions to a runtime. Runtimes are assembled to create components such as a control plane, a data plane, or an identity hub. The EDC module system provides a great deal of flexibility as it allows you to easily add customizations and target diverse deployment topologies, from small-footprint single-instance components to highly reliable, multi-cluster setups²².

In order to satisfy the OPENTUNITY data exchange requirements, **two EDC extensions have been developed:**

- SSI Support for VC/VPs
- OpenAPI automatic contract offering generation.

²¹ <https://projects.eclipse.org/projects/technology.edc>

²² <https://eclipse-edc.github.io/documentation/for-adopters/extensions/>

4.2.3.1 IAM/SSI Support

The SSI extension is a custom Identity and Management extension for the Eclipse Dataspace Components framework that implements a Verifiable Credentials-based authentication mechanism. The extension enables the OPENTUNITY EDC connector to use VCs and VPs for authentication and authorization between the data space participants. This is achieved by integrating with the wallet (Walt.Id) to manage and verify the credentials.

4.2.3.1.1. Key components

VCIdentityService

The `VCIdentityService` implements the following functionalities:

- Obtains client credentials by creating **Verifiable Presentations** that contain **Verifiable Credentials**
- Verifies **JWT tokens** containing VPs from remote participants
- Manages the verification of credentials by using a **trust anchor**

WaltIDIdentityServices

The `WaltIDIdentityServices` class handles all interactions with the WaltID wallet system, including:

- Wallet authentication
- Credential management
- Key management

4.2.3.1.2. Authentication flow

1. When a connector needs to **authenticate** as in Figure 26:
 - a. Retrieves the credentials from its Walt.Id wallet
 - b. Creates a VP (Verifiable Presentation) with these credentials
 - c. Signs the VP (Verifiable Presentation)
 - d. Send them to the remote side using a JWT token

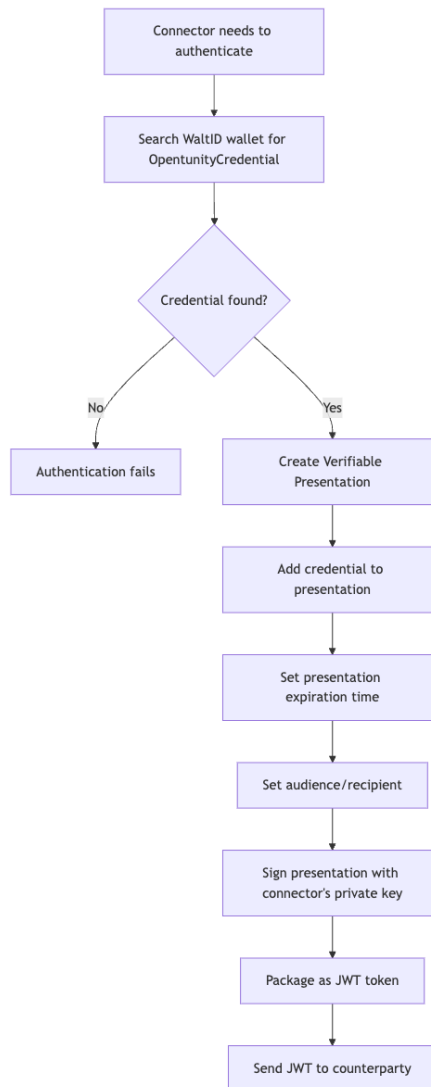


Figure 26 The connector needs to be authenticated against a counterparty.

2. When **authenticating a remote party** as depicted in Figure 27, the connector:
 - a. Verifies the JWT signature
 - b. Validates the VP structure
 - c. Verifies that the credentials are signed by an accepted Trust Anchor
 - d. Verifies that the VP holder matches the credential subject

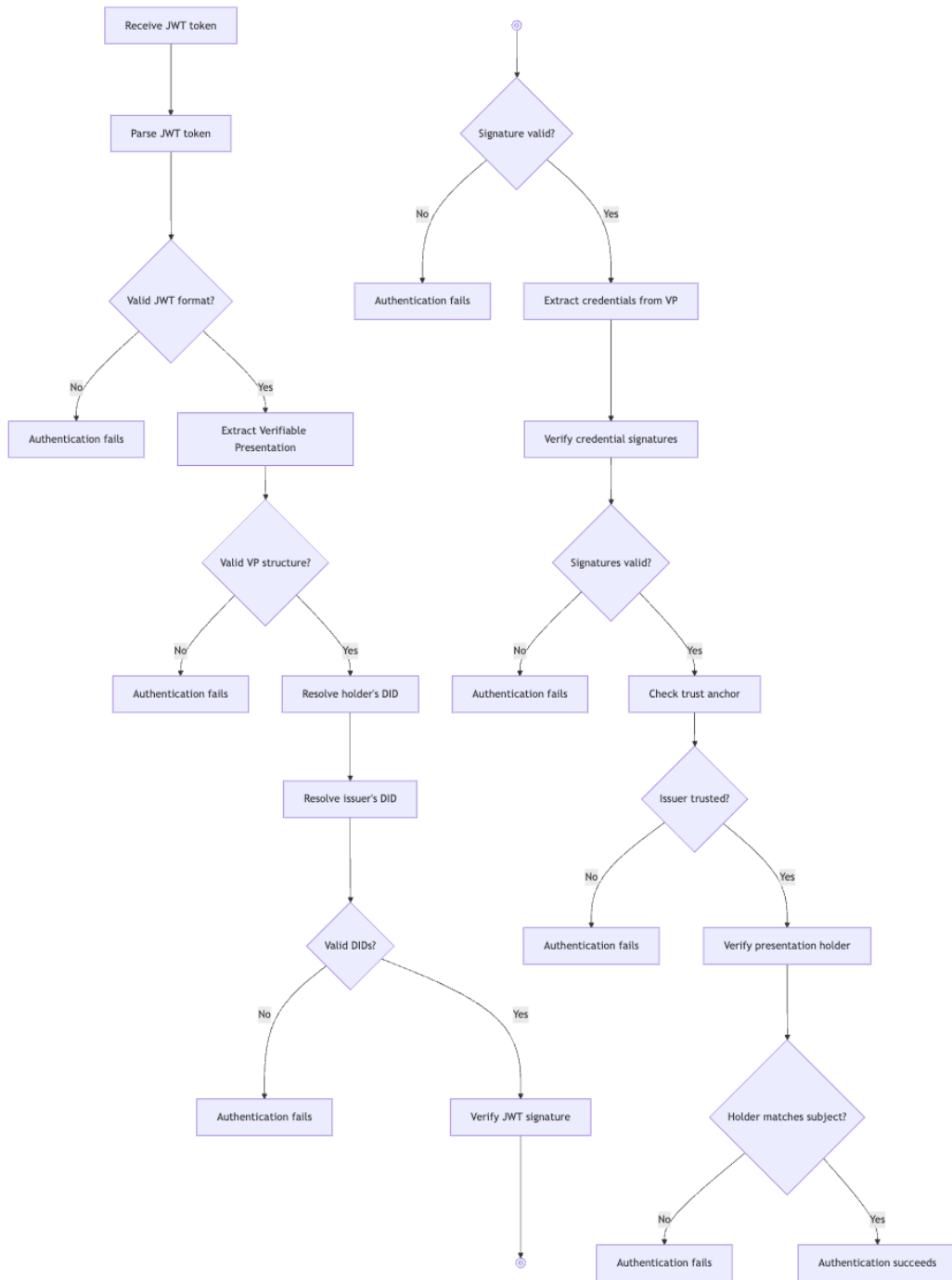


Figure 27 Authenticating a counterparty

4.2.3.2 Automatic contract offering generation (OpenAPI)

This is an EDC connector extension capable of interpreting an OpenAPI schema and generating a set of assets within the data space to represent the services provided by a participant component. The underlying idea is to enable participants to develop their own conventional HTTP APIs while the extension abstracts away the intricacies of exposing these HTTP APIs to the data space.

The extension, as is the case for all EDC extensions, is represented as a class that extends the **ServiceExtensions** base class from EDC. The **initialize** method of **ServiceExtensions** is overridden in order to provide the needed custom functionality. The extension serves as a bridge between a

dataspace connector and a private backend HTTP API. It leverages an **OpenAPI** specification (a standardized way to describe HTTP APIs) to:

- **Create assets:** Represent API endpoints as manageable resources in the connector.
- **Define policies:** Create and add access control rules to these assets.
- **Generate contract offerings:** Enable data sharing agreements based on the assets and policies.
- **Handle data transfers:** Configure the data plane to use HTTP decorators in order to proxy requests and utilize authentication.

As shown in Figure 28, the process is automated, relying on configuration settings and the OpenAPI specification to minimize manual actions.

4.2.3.2.1. Key Components

- **Data Plane**
 - Managed by the `DataPlaneInstance`, it handles data transfers between the dataspace and the backend API.
 - Configured with a public endpoint (e.g., `http://localhost:9291/public/`) for external access.
- **Assets**
 - Represent individual API endpoints (e.g., `GET /users`) as resources in the connector.
 - Each asset is tied to an `HttpDataAddress`, specifying how to proxy requests to the backend.
- **Policies**
 - Define access rules using constraints:
 - **Credential Constraints:** Require specific verifiable credentials (e.g., a credential type matching a pattern).
 - **Authorization Constraints:** Enforce additional authorization checks (optional).
 - Stored in the `PolicyDefinitionStore`.
- **Contract Definitions**
 - Link assets to policies, enabling negotiation and enforcement of data-sharing agreements.
 - Stored in the `ContractDefinitionStore`.
- **HTTP Decorators**
 - Enhance proxied requests with:
 - Contract details (for tracking agreements).
 - Authentication headers (for backend security).

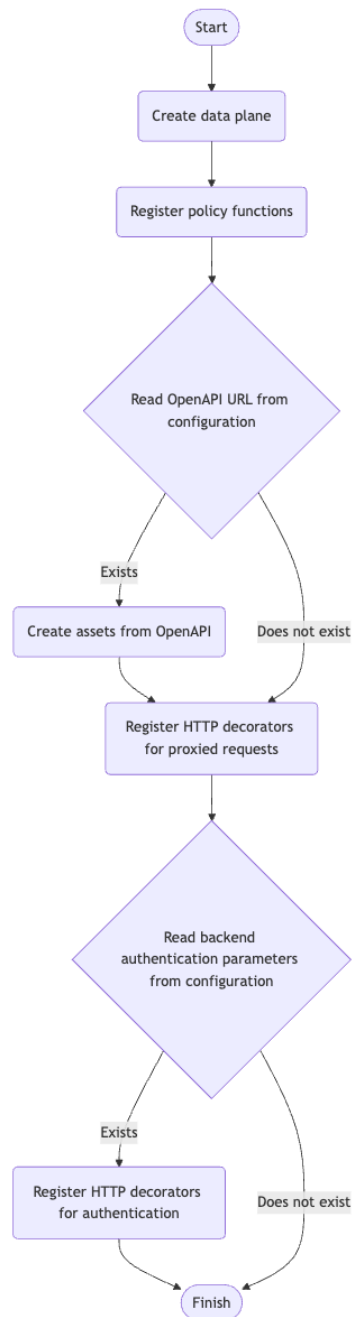


Figure 28 Flow diagram of the custom-built OpenAPI EDC extension

4.2.3.2.2. Operating steps

1. Initialization

- The `initialize` method begins the entire process, starting with data plane creation and policy registration.
- It ensures prerequisites like the data source are set up.

2. OpenAPI Processing

- If an OpenAPI URL is provided (e.g., `http://backend/api/openapi.json`), the extension reads the schema and iterates over its paths and operations.

- Example: For a `GET /users` endpoint, it creates an asset (`get-users`), a policy (e.g., requiring a specific credential), and a contract definition.

3. Data Sharing

- When a consumer requests access to an asset, the connector enforces the associated policy.
- Upon agreement, the data plane proxies the request to the backend API, applying any authentication headers.

4.3 Federated Catalogue

The **Federated Catalogue** (or **Meta-Data Broker** according to IDSA terminology) is a centralized component that serves the purpose of listing all resources (i.e. participants, service offerings and all other potential resource types) that can be found in the individual local connector catalogues that participate in the OPENTUNITY Data Space.

A properly populated Federated Catalogue helps with the discoverability aspect of a data space, by allowing all end-users and clients to query and find the available resources. Structured data models, usually in the form of RDF ontologies are typically used in order to provide a uniform and consistent way to store and query the information.

In the case of the OPENTUNITY Federated Catalogue, the Gaia-X Trust Framework ontology has been chosen as the ontology for the Federated Catalogue. This allows the OPENTUNITY Data Space to be readily usable and interactable by a multitude of infrastructures that already are Gaia-X compliant, as long as access has been granted to them. The Federated Catalogue does not accept or contain actual data (i.e. does not work on the data space data layer), instead it only manipulates meta-data (i.e. the meta-data layer). Hence, there is no need to perform access-policy checks on the data and a connector is not needed to negotiate contracts. The Federated Catalogue should only provide an API that is authenticated with the IAM service on the user-access level using JWT tokens.

4.3.1 Available Federated Catalogue implementations

4.3.1.1 IDS Metadata Broker

The IDS (International Data Spaces) Metadata Broker is an implementation of a Broker, which is a registry for IDS Connector self-description documents. It intends to act as a reference for members of the International Data Spaces Association (IDSA) to help with the implementation of custom Broker solutions. Work on this repository closely aligns with the IDS Handshake Document, which describes the concepts of communication on the IDS in textual form.²³

²³ <https://github.com/International-Data-Spaces-Association/metadata-broker-open-core>

Pros	Cons
Standards compliant: As an IDS reference implementation the IDS Metadata Broker is fully compliant with all the IDS standards	No out-of-the-box Gaia-X compliance
Extendable: The software architecture allows extending and customizing the software	No recent/active development

Table 3: Pros and Cons of IDS Metadata Broker

4.3.1.2 XFSC Federated Catalogue

The **Cross Federation Services Components (XFSC) Federated Catalogue** is the reference implementation for **Gaia-X**. It provides full compliance with Gaia-X out-of-the-box, by being implemented with the **Gaia-X Architecture Document**²⁴ as a requirement specification.

The **XFSC Federated Catalogue** is built using **Java** and is an actively maintained project. It allows semantic metadata management by leveraging **Neo4J** as the backend storage and provides fully trusted services using **SSIs**. All added resources must be in the form of a signed **VP** which the XFSC Federated Catalogue verifies upon insertion. It provides a working deployment environment using **docker** and/or **kubernetes** (plain or **helm**).

Pros	Cons
Standards compliant: Full Gaia-X compliance. Comes with latest Gaia-X Trust Framework preinstalled	Complicated deployment. Relies on a multitude of sub-components.
Trust and Security: Validates all VPs proofs and trust anchors	
Actively maintained	

Table 4: Pros and Cons of XFSC Catalogue

The XFSC has been selected as the most suited basis for the OPENTUNITY Data Space Federated Catalogue implementation. It is the most actively developed and standards-compliant project, with a big community supporting it. It is widely used in other data space projects, like DATA CELLAR and OMEGA-X.

4.3.2 Next steps

After selecting the XFSC Federated Catalogue as the metadata broker solution for the Federated Data Exchange Infrastructure, the next steps include deploying and configuring it to suit the OPENTUNITY Data Space requirements. Work in the following areas is taking place:

- Deployment method.

²⁴ <https://docs.gaia-x.eu/technical-committee/architecture-document/24.04/>

- XFSC catalogue configuration.
- Ontology (Gaia-X Trust Framework Ontology) tweaking and usage in OPENTUNITY.
- Possible extensions of the ontology to cover OPENTUNITY specific use cases and requirements.
- Trust anchor selection (in tandem with the Trust Services configuration).
- Usage documentation (API).

4.4 Opentunity Ontology/Data Model

4.4.1 Gaia-X Trust Framework Ontology overview

The Gaia-X Trust Framework Ontology is one of the building blocks of the Gaia-X initiative, an EU initiative to create a federated, secure, and interoperable data infrastructure that enables data sovereignty, transparency, and trust. The ontology is a formal, machine-readable model of entities, relationships, and concepts in the world of Gaia-X, allowing for consistent description and interaction among participants, services, and resources. It backs the Gaia-X Trust Framework, the regulations and norms that define membership in this community, which creates interoperability and trust through the utilization of verifiable credentials and linked data.

At its core, the Gaia-X ontology is built upon semantic web technologies, in particular the Resource Description Framework (RDF), which organizes information in triples (subject-predicate-object). This allows for a flexible, extensible framework for encoding complex relationships between entities like participants (e.g., providers, consumers), service offerings, resources (e.g., software, hardware), and policies. The ontology is designed to be both human- and machine-readable, which will facilitate automation and reasoning—essential for a scalable, decentralized data economy.

4.4.2 Key Features of the Gaia-X Trust Framework Ontology

- **Core Concepts and Classes:** The ontology defines various core classes such as Participant, Service Offering, Resource, Node, and Contract. These represent the fundamental constructs of the Gaia-X world. For example, a Participant can be either a legal person (e.g., a business entity) or a natural person, while a Service Offering consolidates resources and assets provided by a participant. Relationships between such classes are defined in terms of properties, e.g., providedBy (between a service and a participant) or hosts (between a node and resources).
- **Integration with Existing Standards:** Gaia-X takes advantage of the Open Digital Rights Language (ODRL) ontology to define policies and contractual terms. ODRL provides a shared vocabulary for authoritatively declaring permissions, prohibitions, and obligations, which are exploited by the Gaia-X Policy Reasoning Engine to enforce contracts and automatically verify compliance. The ontology supports lightweight semantic standards like RDFS (RDF Schema) and a constrained subset of OWL (Web Ontology Language) to ensure web ecosystem depth without excessive complexity.
- **Self-Descriptions:** Gaia-X entities are represented in terms of "self-descriptions," which are formally defined metadata files within the ontology. Self-descriptions are cryptographically signed as Verifiable Credentials (VCs) to ensure authenticity and

trustworthiness. For instance, a Gaia-X compliant cloud service provider makes available a self-description of its service offerings that is validated against the schema of the ontology by a Gaia-X Digital Clearing House (GXDCH).

- **Trust and Compliance:** The ontology facilitates the Gaia-X Trust Framework by defining what constitutes trust and how it is sustained. Trust Anchors—those accredited by Gaia-X—issue attestations confirming claims made in self-descriptions. The system uses SHACL shapes, which are computed from the ontology, to enforce consistency and conformity across all self-descriptions
- **Technical Implementation:** Gaia-X specifies and manages its ontology with the Linked Data Modeling Language LinkML. With LinkML, the ontology is defined in a variety of different representations, like JSON-LD for linked data as well as SHACL for validation, rendering it extremely versatile across different use cases. The ontology is versioned and published on the Gaia-X Registry, and participants can view current definitions and make changes to respond to evolving standards.

4.4.3 Purpose and Objectives

The primary purpose of the ontology is to promote interoperability and trust among different ecosystems and data spaces. Unlike traditional organizational frameworks, it is not limited to a single context but is open and shared with participants having the option to reuse ontologies and ease interactions with others. By structuring information in the format of a knowledge graph, it makes advanced reasoning—new associations or insights based on current data—possible, which is necessary to drive activities like contract negotiation or service discovery.

For example, a customer who wants to get a service may request a Gaia-X Catalogue to narrow down providers by attributes like data protection regimes (e.g., GDPR compliance) or degrees of trust. The ontology ensures that all parties that join in are using the same "language," facilitating cross-border and cross-sector data exchange.

4.4.4 Evolution and Openness

The Gaia-X Trust Framework Ontology is a building block that structures the Gaia-X ecosystem so that data and services may be described, attested to, and traded in a trusted, interoperable manner. The Gaia-X ontology is not static; it constantly evolves with input from the open-source community and is in alignment with future standards like DCAT (Data Catalog Vocabulary). Open development takes place, with modules (e.g., core, resource, contract) being released publicly to repositories like GitLab, promoting collaboration and code optimization.

4.4.5 Trust Framework usage in Opentunity

OPENTUNITY is using the Gaia-X Trust Framework Ontology as a template for structuring and standardizing participant and service descriptions to be directly imported into the federated catalogue to be searched and queried by the users as shown in Figure 29 and Figure 30. We are using its semantic model in this ontology to create machine readable self-descriptions. The self-descriptions provide details about the participating parties, including data providers, consumers, and services they offer. For instance, a HEMS/BEMS platform would have a self-description with information pertaining to its name, qualifications, and compliance of data protection regimes (e.g.,

GDPR), while a data offering such as a dataset would have information such as access policy, resource dependency, and trust attributes. These RDF-based self-descriptions, conforming to standards such as ODRL for policy statement, are then consumed by the federated catalogue and provide a clear view of the data space offerings.

The **Gaia-X Trust Framework** extends OPENTUNITY by embedding trust within the data space. The use of Verifiable Credentials within the ontology ensures that all assertions — whether on a participant's legal status or a service's technical capabilities — are validated by **Trust Anchors**, so that the catalogue can be relied upon as a trusted source by users. While users browse the catalogue to locate datasets or services, the knowledge graph structure of the ontology, together with reasoning capabilities, supports sophisticated searches, such as filtering offers according to compliance requirements or trust levels. Another benefit stems from the framework's **extensibility**. In case OPENTUNITY needs to accommodate new types of resources or policies in the future, the modularity of the ontology and the fact that it is open source allows it to be adapted without hindering compatibility. Lastly, with Gaia-X, we are not just optimizing discovery and interaction but also aligning the built environment with a vision of sovereign, interoperable data spaces, expanding its impact and scope.

```

{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://registry.gaia-x.eu/development/api/trusted-shape-registry/v1/shapes/jsonld/trustframework#"
  ],
  "@id": "https://example.com/participants/datacorp",
  "@type": [
    "gx:Participant",
    "ot:OpenTunityParticipant"
  ],
  "gx:legalName": "DataCorp SA",
  "gx:legalRegistrationNumber": {
    "@type": "gx:LegalRegistrationNumber",
    "gx:registrationNumber": "EL123456789",
    "gx:registrationType": "VAT"
  },
  "gx:headquarterAddress": {
    "@type": "gx:Address",
    "gx:countryCode": "EL",
    "gx:streetAddress": "Papanikolaou 123",
    "gx:postalCode": "12345",
    "gx:locality": "Athens"
  },
  "gx:description": "A company providing data analytics services within the Gaia-X ecosystem.",
  "gx:termsAndConditions": {
    "@id": "https://example.com/datacorp/terms",
    "gx:policy": {
      "@type": "odrl:Policy",
      "odrl:permission": [{
        "odrl:target": "https://example.com/datacorp/services",
        "odrl:action": "use",
        "odrl:constraint": {
          "odrl:and": [{
            "odrl:leftOperand": "gx:compliance",
            "odrl:operator": "odrl:eq",
            "odrl:rightOperand": "GDPR"
          }
        ]
      }
    ]
  }
},
  "gx:trustAnchor": {
    "@id": "https://trust-anchor.gaia-x.eu/attestation/datacorp",
    "gx:attestationDate": "2025-02-01T10:00:00Z",
    "gx:issuer": "https://trust-anchor.gaia-x.eu"
  },
  "credentialSubject": {
    "@id": "https://example.com/participants/datacorp",
    "gx:legalStatus": "active"
  },
  "issuer": "https://trust-anchor.gaia-x.eu",
  "issuanceDate": "2025-02-01T10:00:00Z",
  "proof": {
    "type": "Ed25519Signature2018",
    "created": "2025-02-01T10:05:00Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "https://trust-anchor.gaia-x.eu/keys/1",
    "signatureValue": "exampleSignatureValueHere"
  }
}

```

Figure 29 Example of a Participant self-description

```

{
  "@context": [
    "https://www.w3.org/2018/credentials/v1",
    "https://registry.gaiax.eu/development/api/trusted-shape-registry/v1/shapes/jsonld/trustframework#",
    "https://www.w3.org/ns/odrl.jsonld"
  ],
  "@id": "https://example.com/services/datacorp-analytics",
  "@type": "gx:ServiceOffering",
  "gx:name": "DataCorp Analytics Service",
  "gx:description": "A cloud-based data analytics service for processing large datasets with real-time insights.",
  "gx:providedBy": {
    "@id": "https://example.com/participants/datacorp"
  },
  "gx:termsAndConditions": {
    "@id": "https://example.com/datacorp-analytics/terms",
    "gx:policy": {
      "@type": "odrl:Policy",
      "odrl:permission": [
        {
          "odrl:target": "https://example.com/services/datacorp-analytics",
          "odrl:action": "odrl:use",
          "odrl:constraint": [
            {
              "odrl:leftOperand": "gx:purpose",
              "odrl:operator": "odrl:eq",
              "odrl:rightOperand": "research"
            },
            {
              "odrl:leftOperand": "gx:compliance",
              "odrl:operator": "odrl:eq",
              "odrl:rightOperand": "GDPR"
            }
          ]
        }
      ],
      "odrl:prohibition": [
        {
          "odrl:target": "https://example.com/services/datacorp-analytics",
          "odrl:action": "odrl:commercialize"
        }
      ]
    }
  },
  "gx:dataResource": {
    "@id": "https://example.com/resources/analytics-dataset",
    "@type": "gx:DataResource",
    "gx:format": "application/json",
    "gx:accessType": "digital"
  },
  "gx:endpoint": {
    "@type": "gx:Endpoint",
    "gx:endpointType": "OpenAPI",
    "gx:endpointURL": "https://api.datacorp.com/v1/analytics/swagger.json",
    "gx:version": "1.0.0",
    "gx:documentation": "https://docs.datacorp.com/analytics"
  },
  "gx:serviceCategory": "DataAnalytics",
  "gx:trustAnchor": {
    "@id": "https://trust-anchor.gaiax.eu/attestation/datacorp-analytics",
    "gx:attestationDate": "2025-02-10T14:00:00Z",
    "gx:issuer": "https://trust-anchor.gaiax.eu"
  },
  "credentialSubject": {
    "@id": "https://example.com/services/datacorp-analytics",
    "gx:serviceStatus": "active"
  },
  "issuer": "https://trust-anchor.gaiax.eu",
  "issuanceDate": "2025-02-10T14:00:00Z",
  "proof": {
    "type": "Ed25519Signature2018",
    "created": "2025-02-10T14:05:00Z",
    "proofPurpose": "assertionMethod",
    "verificationMethod": "https://trust-anchor.gaiax.eu/keys/1",
    "signatureValue": "exampleSignatureValueHere"
  }
}

```

Figure 30 Example of a service offering self-description

The process begins by having a Participant along with its information, such as Organization Name, Address, VAT, etc. A JSON-LD that conforms to the Gaia-X Trust Framework ontology is created as in Figure 31. The type of the document is `gx:Participant` and `ot:OpenTunityParticipant`. The document is passed to the **IdP issuer** in order to create a signed **Verifiable Credential** out of it. This

Verifiable Credential is then submitted to the **Federated Catalogue** in order to become searchable and discoverable by any entities that have access to the Catalogue, whether from within the OPENTUNITY data space or external projects that have gained access.

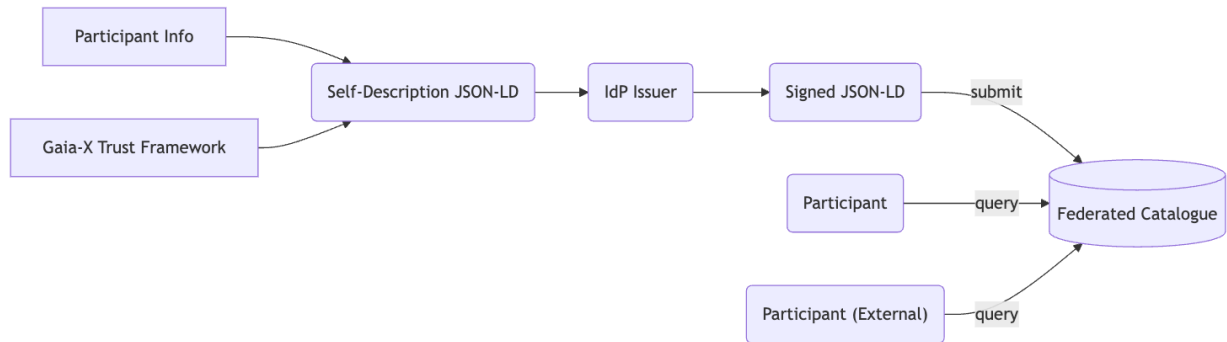


Figure 31 Participant information is combined into a JSON-LD that conforms to the Gaia-X trust framework ontology, signed by the IdP and submitted to the federated catalogue

4.5 Participant onboarding plan

Onboarding is the process of adding a participant to the OPENTUNITY Data Space. This process involves a multi-faculty approach; participants need to be aligned in multiple levels: organizational, legal and technical. The main focus of this section of the document concerns the technical level of onboarding.

The first step of the process is the identification of the participants that wish/should use the OPENTUNITY Data Space infrastructure and processes to exchange data. After that, technically onboarding a participant involves the following steps:

- Making the services and/or data compatible with the data space
- Making the services and/or data accessible to the data space
- Registering the participant using the OPENTUNITY Data Space Onboarding Portal
- Deploying the required software

4.5.1 Initial onboarding group

The targets for onboarding in the OPENTUNITY Data Space are parties that are required to exchange data with other counterparties. These entities are expected to offer technical services to the project, in the form of services and/or datasets and the data space has been optimized to serve such cases.

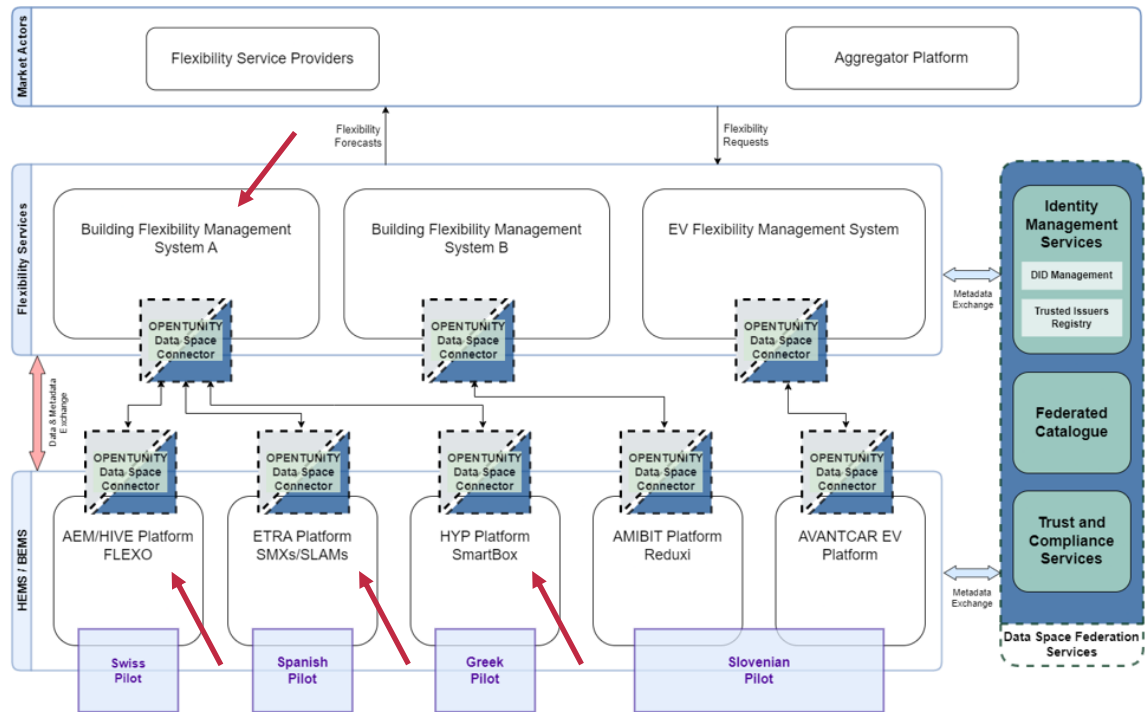


Figure 32 Red arrows show the initial participants of the dataspace

Some platforms and HEMS/BEMS systems are already technically integrated, which means that the addition of a data space layer between them would not achieve any advantage. As such, the initial group of onboarded entities is the following:

- Hypertech's Building Flexibility Management System (BFMS A in Figure 32)
- AEM/HIVE Platform (FLEEXO)
- ETRA Platform (SMXs/SLAMs)
- HyperTech Platform (SmartBox)

4.5.2 Collecting information

Collecting information and discussing with the potential participants were crucial to the development of the data space components and protocols. This is an ongoing process that will be finalized when the data space is up-and-running with all entities onboarded.

The collection of information and discussions regarding the technicalities and potential problems in the offered services is important for the implementation of the data space. Service protocols, authentication mechanisms and data models have to be agreed upon and translated into functional and technical requirements for the subsequent development of the data space components. For the third-party components, these functional requirements create the selection criteria that dictate the final selection of software that was utilized in the solution.

To help gather all requirements and potential problems a questionnaire was created and distributed to the potential participants. This questionnaire, although brief, gave the data space development team an important base line for the protocols and methods that should be supported.

(Título del documento)

+

If this is just an example for the NILM module (although it is general enough). Feel free to adapt it to your own characteristics and needs. If you need it, feel also free to cover more areas of your tool (not just the GUI).

Questionnaire: HEMS-BEMS

Greek pilot site -> HYP SmartBox

Functionalities/capabilities

- Monitoring of environmental conditions & energy consumption (total and per supported asset)
- Control of devices (actuators)
- Energy management

Type of loads/assets that are supported

- HVAC devices (e.g. heat pumps, resistive loads)
- DHW
- PV

Data collection

- Continuous, usual granularity 15 minutes
- Cleansing & normalization techniques
- Processing to provide required formats
- Secure data exchange and storage

Existing interfaces

- Communication between smart box and BFMS
- Metered data is available through rest API using an open API specification
- Downstream data (control signals) use message broker (AMQP)

Swiss pilot site -> HIVE FLEEXO & AEM platform

Functionalities/capabilities

Type of loads/assets that supports

Data collection

Existing interfaces

Spanish pilot site -> ETRA SMXs

Functionalities/capabilities

Type of loads/assets that supports

Data collection

Existing interfaces

Slovenian pilot site -> AMIBIT reduxi (optional)

Functionalities/capabilities

Type of loads/assets that supports

Data collection

Existing interfaces

HEMS - BEMS interaction with NILM

- What can we expect as an output of NILM?
 - Power consumption per asset?
 - Time granularity? Timeseries or aggregated consumption per day/week?
- Frequency of NILM execution?
 - How often are the abovementioned results obtained?
- Should NILM be used in all pilots?
 - What input is needed from NILM's side?
- Does the interface provide all necessary tools and features? Is there any item missing?
- Is the information and data presented clearly (this includes color palette)?

Figure 33 Example of filled questionnaire

In parallel with the completion and collection of the questionnaires, a pilot platform was chosen with which more detailed discussions were conducted, and an initial integration test of the Data Space connector was carried out. This participant was Hypertech's HEMS/BEMS platform. The results of this preliminary integration test were successful, albeit required some adjustments to the connector code and configuration.

4.5.3 Verifying compatibility and request for adaptations

The collection of the questionnaires resulted in some important insights. In general, it seems that all potential participants are generally aligned technically, with some variation in the authentication mechanisms used.

Insight #1: All services are using the HTTP API format

Insight #2: There is no common data model that could be extracted and used globally for the data exchanges in the OPENTUNITY Data Space

Insight #3: Backend API authentication mechanisms vary, with all of them falling into the following types:

- No authentication
- API Key authentication
- Username/password authentication

While "**no authentication**" and "**API Key authentication**" are adequate for data space connector use, the "**Username/Password**" authentication method is deemed inappropriate for the use case. As a consequence, discussions have commenced with the respective participants to adapt to a more machine-focused authentication protocol, such as **API Key**.

4.5.4 Future platform onboarding

The approach will be validated by its execution by the first participants and fine-tuned based on the lessons learned to improve methods for future use when new participants join the OPENTUNITY Data Space. The Onboarding Portal and Participant Template with their supporting documents ensure that the data space is scale-ready without any additional burden passed onto the program. The **Onboarding Portal** and the **Participant Template**, along with their documentation allow the data space to scale without any additional work.

4.6 Components deployment plan

The deployment strategy of the OPENTUNITY Data Space components focuses on easy installation, scalability, and performance across both the centralized and decentralized software components. The design is divided into two distinct categories of components with different deployment plans suited to the specific needs and infrastructure of its operation. In this section, the process of deployment, tools, and phases of each group are outlined in order to facilitate clear understanding and easy adoption for all interested stakeholders.

4.6.1 Centralized Components

The centralized components are the "backend" of the data space, providing core functionality and services accessible to all participating platforms. These components are deployed once centrally, hosted on virtual machines (VMs) or similar cloud-based environments managed by the project team.

Infrastructure Requirements

The centralized components require a hosting environment with sufficient compute, storage, and networking resources. Virtual machines are installed with operating systems supported by the software stack (e.g. Linux distributions), and network connectivity is established for secure communication with decentralized nodes.

Deployment Process

1. **Provision of the VMs** or cloud instances in the central infrastructure, ensuring they meet the specified resource requirements (e.g., CPU, RAM, disk space).
2. **Install necessary dependencies**, including runtime environments, libraries, and orchestration tools (e.g., Docker, Kubernetes) as dictated by the component architecture.
3. **Deploy the centralized software stack** using predefined configuration scripts or containerized images, ensuring consistency across instances.
4. **Configure** networking, firewalls, and security protocols (e.g., SSL/TLS certificates) to enable secure access and data exchange with decentralized components.
5. **Perform initial testing** to validate functionality, connectivity, and performance against project benchmarks.

Maintenance

Post-deployment, the centralized components will be monitored and updated as needed to ensure availability. Automated monitoring tools will track system health, and upgrades will be applied during scheduled maintenance windows.

4.6.2 Decentralized Components

The decentralized components are deployed individually at the premises of each participating platform, enabling localized processing and data sovereignty while integrating with the centralized infrastructure. To make this process easier and more consistent, a participant template has been developed as a standardized deployment toolkit.

- **Participant Template Overview:** The participant template is a software package comprising Task scripts (for task automation), Bash scripts, and Docker Compose files. This template allows participants to deploy the full suite of decentralized software in a single, user-friendly step, abstracting much of the complexity of manual configuration.
- **Infrastructure Requirements:** Each participating platform must provide a local environment (e.g., a server or VM) with Docker and basic command-line tools installed. The environment should allow network connectivity to the centralized components and meet minimum hardware specifications outlined in the template documentation.

Deployment Process:

1. **Preparation:** The participant receives the participant template package and documentation. They verify that their local environment meets the prerequisites (e.g. Task installed, Docker installed, sufficient disk space).
2. **Execution:** Using the Task command-line interface, the participant runs the primary deployment command (e.g. `task start-all`). This triggers the following automated steps:
 - Execution of Bash scripts to set up directories, permissions, and environment variables.
 - Deployment of Docker containers via Docker Compose, pulling necessary images from a designated registry (hosted centrally or locally cached).
 - Configuration of component settings, such as API endpoints for communication with the centralized infrastructure.

3. **Validation:** After deployment, the participant runs a provided verification script (e.g., task verify) to confirm that all services are operational and correctly integrated with the data space.
4. **Support and Troubleshooting:** The participant template includes a troubleshooting guide and log collection scripts to assist with diagnosing issues. Participants can escalate complex problems to the project support team if needed.

4.6.3 Deployment Timeline

- **Centralized Components:** Deployment will occur during the initial setup phase of the project, before the onboarding of participants. This ensures the central infrastructure is fully operational, ready to support decentralized nodes.
- **Decentralized Components:** Deployment will proceed in steps as participants join the data space. The participant template will be distributed, via the Onboarding Portal, to each platform, along with the required documentation, enabling users to execute the deployment themselves.

4.6.4 Key Considerations

- **Scalability:** The centralized infrastructure is designed to handle an increasing number of decentralized nodes, with resource scaling planned as participation grows.
- **Security:** Both deployment processes incorporate the best practices for secure communication (e.g., encrypted channels - TLS) and access control to protect the integrity of the data space.
- **Consistency:** The participant template ensures consistency in the deployments of the decentralized components, reducing the risk of configuration errors and simplifying maintenance for both the participant organization staff and the OPENTUNITY project team.

This deployment plan provides a flexible framework to bring the data space online. By using automation and standardized tooling, it minimizes deployment overhead while supporting participants to manage their local components effectively.

4.7 Testing and evaluation plan

In order to assess the efficiency and feature-completeness of the Federated Data Exchange Infrastructure of the OPENTUNITY project, a testing and evaluation plan has been put in place. The objectives of this plan are the following:

- Verify that all components function as intended
- Identify areas for improvement
- Identify items that will need special attention in the training and documentation material

4.7.1 Testing and Evaluation Methods

Due to the highly interconnected and externally integrated nature of the data space components, automation testing the complete end-to-end solution is not a realistic expectation. Nevertheless,

Carefully tailored and orchestrated manual testing should be conducted in order to verify all requirements, both functional and non-functional.

4.7.1.1 Unit testing

Unit testing involves each component being tested individually. Standard unit testing methodologies exist and broadly fall in the following categories:

- **In-code unit testing:** Special unit test code snippets are created. These code snippets, aim to provide input and expect specific output to the actual component code. This is important for assuring the vital internal component parts work as expected. Furthermore, it improves the maintainability of the software by eliminating regressions when modifying the code.
Tools/libraries used:
 - Python: **pytest**
 - Java: **Junit**
 - React/typescript: **Jest/React Testing Library**
- **External tool testing:** This method tests the external inputs and outputs of a component, usually by using automation tools. Depending on the nature of the component, a specific tool should be used. Given the fact that most components expose an HTTP API, **Postman** is a good choice for testing. **Postman** provides advanced scripting and automation facilities that can exhaustively test a component with external interfaces.

4.7.1.2 Integration testing

The multitude of components and deployments needed for a fully functional data space, such as the Federated Data Exchange Infrastructure of the OPENTUNITY project, make exceptionally hard to implement an automated and repeatable test suite. Nevertheless, a manual integration plan is being actively developed in parallel with the development of the individual components.

A script with specific, well-defined actions, each with its prerequisites, dependencies, inputs and expected outputs is being created. This script will allow validating the data space deployment as a whole, relatively quickly and consistently. Each action will have the following properties/details:

- Number
- Name
- Description
- Pre-requisites
- Steps to execute
- Expected outcome
- Extra information

Actions will be grouped in categories that are defined according to data space scenarios:

- Onboarding
- Participant template deployment
- Consumer: Retrieving data
- Provider: Sharing data

5 Interoperability

5.1 The crucial role of interoperability

Interoperability is needed to ensure that the global data ecosystems can last, be expanded, and are used productively over time. In some cases, even though exchanges can technically occur, non-interoperability can cause great strategic woes, often with one of the most prominent challenges: risks had that centralized control over the systems.

Especially in Data Spaces interoperability is critical, covering both intra-data-space and inter-data-space compatibility. European Commission's New European Interoperability Framework (EIF)²⁵ focuses on the "interoperability-by-design" approach. It outlines four levels of interoperability and includes a cross-cutting element of "Public Service Governance", as shown in Figure 32.



Figure 34: Interoperability model of New European Interoperability Framework

This approach encourages flexibility in implementation and integration choices while fostering interoperability. In the energy sector, its practical applications are explored by the Energy Interoperability Task Force²⁶. Such initiatives focus on finding the most effective ways to apply and implement the interoperability framework to support energy data exchanges and integrations.

The term "interoperability" describes the capacity of separate data spaces to access other datasets and their associated services without facing major technological hurdles that necessitate significant investments of bespoke adapters or communication bridges. It is to be expected that a certain amount of adaptation will need to occur - complete interoperability with both semantic and transfer of data technologies is not the aim. Instead of this, the aim is to develop the data spaces with a shared set of principles to allow interoperability.

²⁵ https://ec.europa.eu/isa2/sites/default/files/eif_brochure_final.pdf

²⁶ https://internationaldataspaces.org/wp-content/uploads/dlm_uploads/IDSA-Position-paper-Energy-interoperability-framework-v0.9-1.pdf

A set of System Use Cases (SUCs), i.e., generic use cases applicable throughout all projects involved in the creation of energy data spaces has been assembled in the framework of Interoperability Network for the Energy Transition (int:net)²⁷ umbrella EU project. These SUCs aim to address challenges that stem from the different approaches followed within different implementation contexts, bridging gaps and converging to a clear common set of guidelines and implementation principles. The set of SUCs are:

- SUC 1: Onboarding of participants – Identity Management.
- SUC 2: Data Discovery and push into the catalogue.
- SUC 3: Contracting.
- SUC 4: Data Exchange and Interoperability

The above set of SUCs ensures a common implementation approach with other Data Spaces. The outcomes of the discussions around those SUCs played a key role both during the design phase (architectural choices) as well as in the implementation of OPENTUNITY's FDEI. As a matter of fact, QUE being a key technical partner in the relevant project Data Cellar Horizon project, facilitates the knowledge transfer and interoperability across the two projects as well as other related project ecosystems.

This document is the initial version of the Federated Data Exchange Infrastructure Architecture and apart from the architecture, the main interoperability approach of the OPENTUNITY project is also described. The design decisions for the FDEI are based on the following assumptions:

1. Creating a sovereign data space
2. Following the interoperability definitions described in the European Commission's New European Interoperability Framework (EIF)
3. Creating reusable and proven technologies

Following the above, the OPENTUNITY Data Space meets interoperability requirements with other Horizon Data Space Projects, as it is GAIA-X compliant (Chapter 4.4.1 Gaia-X Trust Framework overview), and the next step is the assessment of other Data Spaces.

5.2 Assessment of other projects dataspace

To achieve "*Federated data spaces approach to be closely coordinated with other relevant Horizon projects*", the plan is to build on the extensive experience gained through the Data Cellar project. This strategy not only accelerates the evaluation process of other Data Spaces but also ensures OPENTUNITY's compliance with standardized methodologies and industry practices.

By adopting Data Cellar's best practices, OPENTUNITY aims to streamline the evaluation of other Horizon projects' Data Spaces, with a strong focus on interoperability, data sovereignty, and scalability. The involvement of QUE Technologies, a core partner in both OPENTUNITY and Data Cellar, ensures the seamless transfer of technical expertise and strategic insights. This synergy allows OPENTUNITY to implement a structured assessment methodology, already validated through the Data Cellar project, to conduct a comprehensive assessment of other Data Spaces.

²⁷ <https://intnet.eu/>

The assessment methodology consists of the following steps:

1. **Interoperability Assessment:** Examine the technical compatibility between other Data Spaces and OPENTUNITY's Federated Data Exchange Infrastructure (FDEI) to ensure smooth data integration and exchange.
2. **Compliance Verification:** Evaluate alignment with European standards and frameworks, such as GAIA-X and the Data Space Support Centre (DSSC) guidelines, to ensure full compliance with EU regulations on data privacy and security.
3. **Governance Review:** Analyze the governance models of other Data Spaces to assess their effectiveness in maintaining data sovereignty and enabling fair data sharing among participants.
4. **Use Case Alignment:** Identify shared use cases that could benefit from cross-data space collaboration, especially in areas like flexibility services, grid optimization, and demand response strategies.

Based on initial evaluations, OPENTUNITY will prioritize the assessment of the following Horizon projects, recognized for their advanced data space implementations:

- **Data Cellar** for best practices and technical standards.
- **OMEGA-X** for its cross-sectoral data discovery capabilities.

The insights gained from this assessment will shape OPENTUNITY's interoperability framework, enabling the integration of multiple data spaces into a cohesive ecosystem. This not only strengthens OPENTUNITY's data-sharing capabilities but also fosters cross-sectoral collaboration, promoting innovation and efficiency within the energy sector.

6 Conclusion

The work carried out in Tasks 3.1 - Federated Data Exchange Infrastructure design and delivery, Task 3.2 - Management of data space participants and data/service offerings and Task 3.4 - Integration with other platforms, have successfully demonstrated the Proof of Concept (PoC) for its Federated Data Exchange Infrastructure (FDEI), providing the fundamentals for secure, interoperable, and sovereign data exchange within OPENTUNITY.

Moving forward, the next steps will focus on the following key areas:

1. Integration of the Data Space Connector with OPENTUNITY's HEMS/BEMS and Flexibility Services platforms
2. Onboarding implementation
3. Federated Catalogue Development
4. Data Space Components Refinements

The results of the above-mentioned tasks will be included in the final version of the Federated Data Exchange Infrastructure Architecture Deliverable 3.2

7 References and acronyms

7.1 References

1. OPENTUNITY's Description of the Action
2. <https://datacellarproject.eu/>
3. <https://intnet.eu/>
4. <https://www.sciencedirect.com/science/article/pii/S2352340924009314>
5. <https://dssc.eu/space/Glossary/176554052/2.+Core+Concepts>
6. <https://gaia-x.eu/what-is-gaia-x/about-gaia-x/>
7. Data Cellar Horizon2020 project
8. <https://link.springer.com/article/10.1007/s11747-022-00845-y>
9. <https://library.oapen.org/bitstream/handle/20.500.12657/58395/1/978-3-030-98636-0.pdf#page=347>
10. <https://en.wikipedia.org/wiki/Keycloak>
11. <https://wso2.com/identity-server/>
12. <https://www.freeipa.org/About.html>
13. <https://github.com/admpub/caddy>
14. <https://github.com/International-Data-Spaces-Association/true-connector>
15. https://github.com/International-Data-Spaces-Association/true-connector/blob/main/doc/IDSA_certificate_ENG_final.pdf
16. <https://github.com/FraunhoferIISST/DataspaceConnector>
17. <https://github.com/Engineering-Research-and-Development/true-connector>
18. <https://github.com/sovity/edc-ce>
19. <https://www.tno.nl/nl/>
20. <https://tsg.dataspac.es/>
21. <https://projects.eclipse.org/projects/technology.edc>
22. <https://eclipse-edc.github.io/documentation/for-adopters/extensions/>
23. <https://github.com/International-Data-Spaces-Association/metadata-broker-open-core>
24. <https://docs.gaia-x.eu/technical-committee/architecture-document/24.04/>
25. https://ec.europa.eu/isa2/sites/default/files/eif_brochure_final.pdf
26. https://internationaldataspaces.org/wp-content/uploads/dlm_uploads/IDSA-Position-paper-Energy-interoperability-framework-v0.9-1.pdf

7.2 Acronyms

Acronym	Description
ABAC	Attribute-Based Access Control
API	Application Programming Interface
BEMS	Building Energy Management System
CNCF	Cloud Native Computing Foundation
DCAT	Data Catalog Vocabulary
DIDs	Decentralized Identifiers
ECC	Execution Core Container
EDC	Eclipse Dataspace Components
EIF	European Interoperability Framework
EU	European Union
FDEI	Federated Data Exchange Infrastructure
HTTPS	Hypertext Transfer Protocol Secure
IAM	Identity and Access Management
IDS	International Data Spaces
IDSA	International Data Spaces Association
RBAC	Role-Based Access Control
SGAM	Smart Grid Architecture Model
SHACL	Shapes Constraint Language
SSH	Social Sciences and Humanities
UI	User Interface
VCs	Verifiable Credentials
VP	Verifiable Presentation

8 Annex I Related UCs, requirements and SGAM diagrams

8.1 UC 1.2 Non-Intrusive Load Monitoring.

The applicability of Dataspaces in this use case relies on how data is exchanged between entities. NILM requires real-time and historical energy data from the involved households, stored in the corresponding AMI database or smart home gateway. This data will be accessed via the Dataspace connector, acting as a data consumer. After the algorithm performs disaggregation, the results will be published through an API endpoint offered in the Dataspace connector, acting as a data provider. The IDSA Dataspace protocol serves as the communication mechanism for data exchange between connectors.

UC 1.2	Application of NILM for consumer's energy awareness
Description	<p>Capturing the data from smart meters, the NILM will analyze the consumption of the linked meter and will disaggregate the consumption into the main consumption devices. The focus will be put on the main appliances like HVAC, lighting, refrigerator, TV, washing machine, dishwasher, dryer and electric boiler. The NILM algorithms may be embedded into IoT smart devices (mainly smart meters) or deployed in the cloud in order to run.</p> <p>The goal is to identify and analyze the energy consumption patterns of customers to facilitate the reduction of their energy consumption and detect anomalies that may mean an incorrect functioning of an asset. Linked to this, this NILM functionalities will also be the basis for estimating the actual Energy Efficiency Label of the asset based on EU regulation 2021/340 of 17 December 2020.</p>
Actors involved	<ul style="list-style-type: none"> - Consumer - EMS (Building Energy Management System or Home Energy Management System) - ESCO - AMI (Advanced metering infrastructure). - Smart home gateway. - NILM Module
Triggering Event	<p>An ESCO needs to understand the energy behaviour of a consumer without submetering.</p>
Pre-condition	<p>To receive the energy consumption data from the smart meters through Dataspace connectors periodically.</p>
OPENTUNITY innovations involved	<p>4. AI non-intrusive load monitoring algorithms as a low-cost technology for sensorless prosumers.</p>
Post-condition	<p>EMS shows (to both ESCO and consumer) the disaggregated consumption of its consumer and also provide relevant analysis linked to the energy behaviour (like when the different assets are functioning, abnormal behaviour of the assets or the assets that consumes the most) and energy efficiency of the assets.</p>
Basic Path: Dissagregation at the backoffice	

Step No.	Event	Description of process/ Activity	Info. exchanged	Actor producing the information	Actor receiving the information
1	Periodic	Data gathering	Home electrical measurements (aggregated)	Consumer (Smart meter)	AMI
2	Periodic	Data storage	Home electrical measurements (aggregated)	AMI through Dataspace connector	NILM module through Dataspace connector
3	Periodic	Energy Disaggregation	Home electrical measurements (disaggregated)	NILM module API offered in the Dataspace	EMS ESCO Consumer through Dataspace connector

Basic paths: Dissagregation at the edge

Step No.	Event	Description of process/ Activity	Info. Exchanged	Actor producing the information	Actor receiving the information
1	Periodic	Data storage	Home electrical measurements (aggregated)	Smart meter	Smart home gateway
2	Periodic	Energy Disaggregation	Home electrical measurements (disaggregated)	Smart home gateway through Dataspace connector	EMS ESCO Consumer through Dataspace connector

Realization

Main responsible partners (Author) - ETRA

Contributing partners

- IMPULSA
- AMIBIT
- AEM
- HIVE

Priority High

Table 5: UC 1.2 - Application of NILM for consumer's energy awareness

The only SGAM layer affected by Dataspace integration is the communication layer. The process of collecting inputs needed for NILM disaggregation and storing the disaggregation results involves using a Dataspace connector. This connector facilitates data policy acceptance, contract agreement negotiation, and acceptance, creating a link to access the data via a URL. Therefore, instead of accessing the database using a client or providing the results through an API, access is performed using a link. This link acts as a proxy to the real database or API endpoint, which is the only difference when using a Dataspace approach.

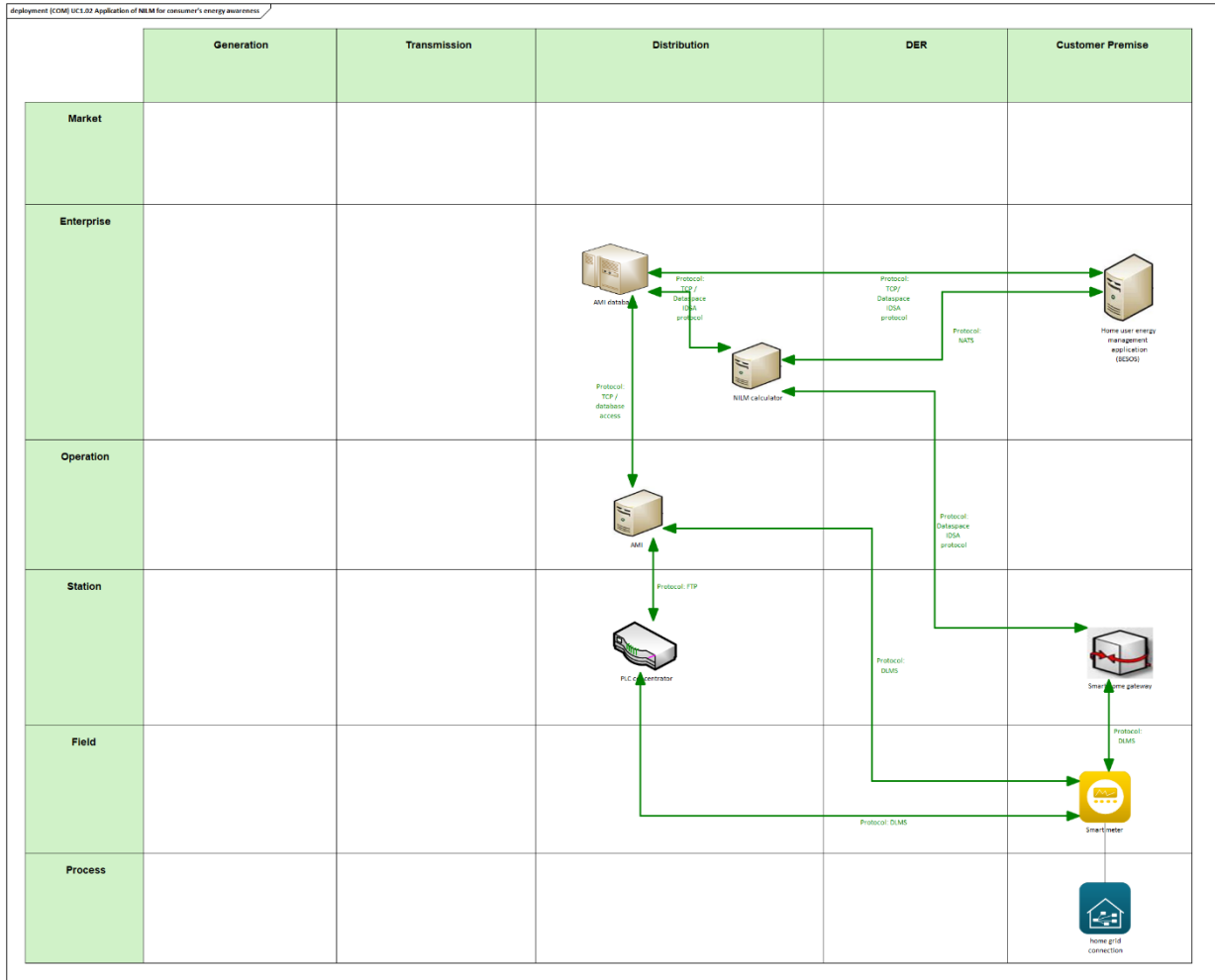


Figure 35: UC1.2 SGAM Diagram

8.2 UC 1.8 HEMS/BEMS DR optimization and local flexibility management.

UC 1.8	HEMS/BEMS DR optimization and local flexibility management
Description	<p>OPENTUNITY HEMS/BEMS flexibility management system (BFMS) is a cloud-based system responsible for the local flexibility management and the participation of the demo sites in Demand Response services. It consists of several software backend systems developed in a modular approach to provide specific functionalities and flexibility management over various building assets (i.e., HVACs, DHWs, EVs including charging points, storage, and other assets) for effectively unlocking and exploiting the distributed small-scale flexibility potential behind every residential/building connection point.</p> <p>HEMS/BEMS flexibility management system provides day-ahead (24h) flexibility forecasts - possibility for upwards or downwards regulation of a building/asset consumption - based on the existing energy resources inside the pilot premises and will co-optimize the operation of several appliances to deliver maximum flexibility without violating</p>

end-users' comfort boundaries. To achieve this goal, several optimization engines are embedded in the core functionalities of the flexibility system. Extracted flexibility forecasts will be provided to the main market actors including aggregators and the NODES' local flexibility markets through the proper interfaces and according to predefined or standardized flexibility semantics.

Upon a flexibility request from the DR market actors, the HEMS/BEMS flexibility management system integrated control system will respond by generating optimal control actions in order to deliver the requested load dispatching while monitoring the assets' consumption and the overall performance of the dispatched control signals. The final activation of the building assets will be performed through the local controllers within the pilot buildings.

Actors involved

- Aggregator
- EMS (from Buildings)
- Prosumer
- Flexibility Service Provider (FSP)
- Flexibility Market Operator (FMO)
- DSO

Triggering Event

- Requests for flexibility forecasts/profiles
- Flexibility requests for load dispatching.

Pre-condition

- Demo sites are equipped with HEMS/BEMS.
- Available energy resources inside the pilot premises to operate as flexibility assets.
- Reception of energy consumption data through Dataspace connectors.
- Integration of the backend systems with the deployed solutions inside the pilot sites.
- Existence and communication with the local controllers/actuators within the pilot buildings.
- Existence and data acquisition from the metering & sensing devices installed within the pilot buildings.
- Interfacing with the market actors.
- Building asset information retrieval for the plug 'n' play recognition system

OPENTUNITY innovations involved

3. HEMS and BEMS Flexibility and DR optimization including initial settings algorithms

Post-condition

Provide flexibility forecasts upon requests and respond to any flexibility request for load dispatching.

Basic Path

Step No.	Event	Description of process/ Activity	Info. exchanged	Actor producing the information	Actor receiving the information
1	Flexibility forecasts extraction	The flexibility management system identifies flexibility potential	Power metering data, indoor conditions sensing data & monitoring	EMS through Dataspace connector	BFMS

		from various assets	info from the assets		
2	Flexibility forecasts delivery	Extracted flexibility forecasts are sent to the market actors	Flexibility forecasts	BFMS	Aggregator, FMO, DSO
3	Flexibility request	Market actors send the flexibility requests for load dispatching based on provided forecasts	Flexibility request details (e.g., timing, required load dispatching)	Aggregator, FMO, DSO	BFMS
4	Flexibility response	Backend flexibility manager control system generates optimal control actions to deliver the requested load dispatching	Dispatched control actions, asset consumption, performance monitoring	BFMS through Dataspace connector	EMS/Local controllers within pilot buildings
5	Building asset activation	Local controllers within pilot buildings activate the building assets	Control actions	EMS /Local controllers within pilot buildings)	Building assets
6	Flexibility request acknowledgement	HEMS/BEMS flexibility management system sends an acknowledgement of dispatched flexibility	Acknowledgement information	BFMS	Aggregator, FMO, DSO

Exception paths

Step No.	Event	Description of process/ Activity	Info. Exchanged	Actor producing the information	Actor receiving the information
1a	Forecasted Flexibility not dispatched	End-user bypassed the dispatched control actions	Interaction similar to previous step 1.	End-user through Dataspace connector	BFMS

Realization

Main responsible partners (Author)

Add one or maximum two partners from this list:

Contributing partners

- HYP
- ETRA
- UL
- HEDNO
- EYPESA
- IMPULSA
- JR
- NODES
- SETUP
- AMIBIT
- EP
- AVANTCAR
- EL

- BLUE
- AEM
- HIVE
- SUPSI

Priority High.

Table 6: UC1.8 - HEMS/BEMS DR optimization and local flexibility management

The only SGAM layer affected by Dataspace integration is the communication layer. The process of collecting energy data from the HEMS/BEMS to be processed by the BFMS as well as the dispatching of control actions from the BFMS back to the HEMS/BEMS will take place using a Dataspace connector.

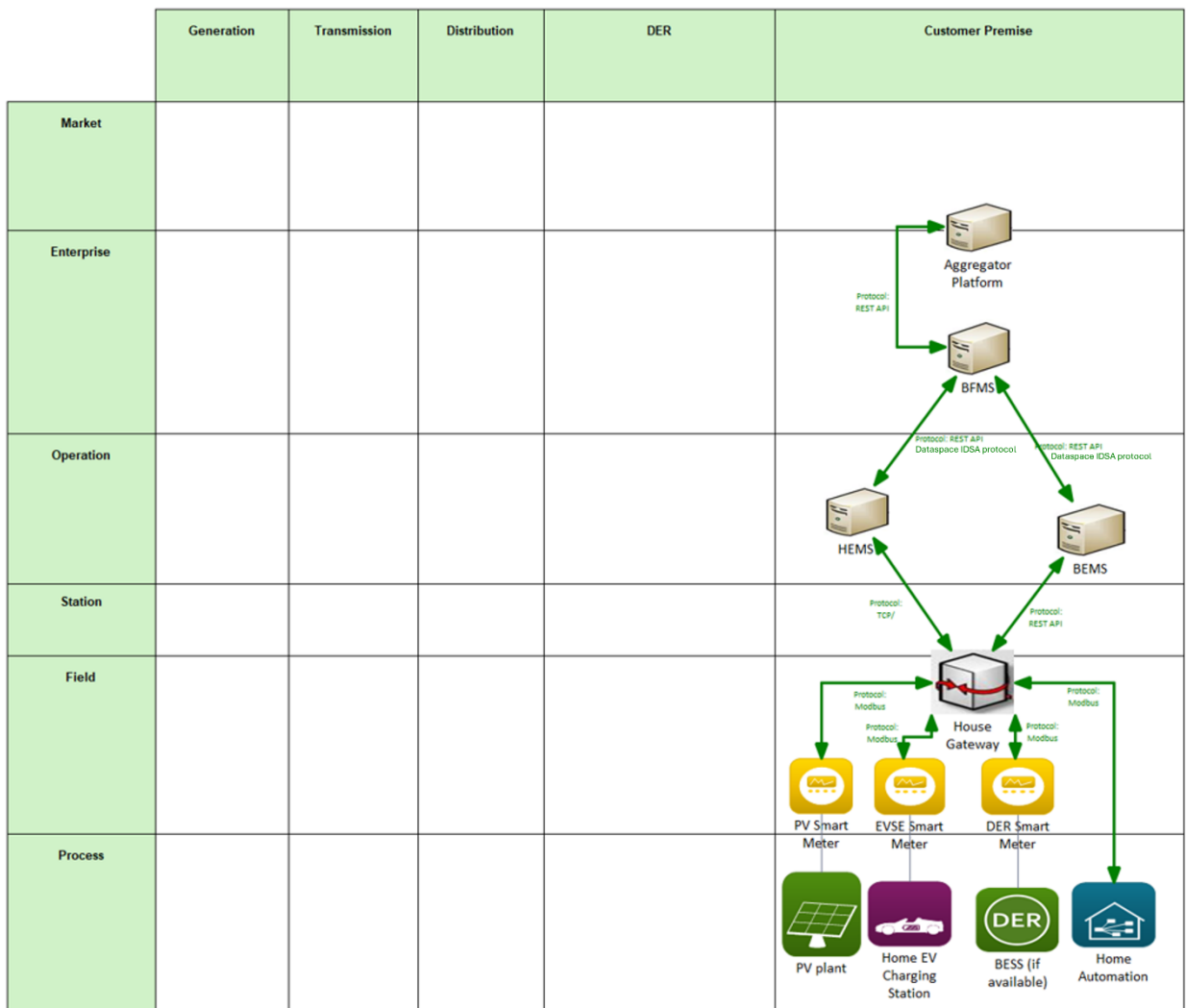


Figure 36: UC 1.8 SGAM Diagram

8.3 UC 1.9 Initialization of HEMS/BEMS Demand Response strategy.

UC 1.9	Initialization of HEMS/BEMS Demand Response strategy
Description	<p>End-users' comfort constraints must not be violated during the operation of the HEMS/BEMS systems while their participation in Demand Response campaigns should not interfere with their preferences and affect their daily routine. Therefore, the end-users are in the centre of the solution by actively participating in the system configuration during the initial phase of the demonstration activities. On the other hand, they can also bypass any control actuation on their own assets during the demand response campaigns.</p> <p>During HEMS/BEMS flexibility management system initialization the end-users will provide feedback in a user-friendly manner (e.g., through a UI), indicating their comfort preferences and daily schedules. All their inputs will be evaluated for understanding their DR capabilities and the degree of involvement in the DR campaigns. In addition, the initial settings algorithms will be triggered for fine-tuning the system and introducing the proper constraints in the optimization framework so that it can automatically generate demand response profiles and set the control optimization strategy without affecting the end-users. Many stakeholders will benefit from this innovation since they will have access to low volumes of flexibility usually occurring during peak hours but with high potential for large-scale aggregation.</p>
Actors involved	<ul style="list-style-type: none"> • BFMS • Prosumer
Triggering Event	<ul style="list-style-type: none"> • End-user participation • Received QR codes from the Plug 'n' play recognition system
Pre-condition	<ul style="list-style-type: none"> • Interface with the installed HEMS/BEMS systems • End-user Engagement • Optionally, integration with the Plug 'n' play recognition system
OPENTUNITY innovations involved	<p>3. HEMS and BEMS Flexibility and DR optimization including initial settings algorithms</p> <p>12. Plug and play recognition for flexibility devices.</p>
Post-condition	<ul style="list-style-type: none"> • Flexibility assets identification per prosumer • Generation of targeted Demand Response profiles • End-User comfort preservation

Basic Path					
Step No.	Event	Description of process/ Activity	Info. exchanged	Actor producing the information	Actor receiving the information
0 (optional)	Identification of flexibility assets & Flexibility Management System configuration	End-users scan QR codes of their meters and energy assets. Internal system configuration based on the	QR code information	Plug 'n' play recognition system,	HEMS/BEMS

		information retrieved from the flexibility assets. Depending on the collected data different modelling approaches and configuration settings will be applied.			
1	Initialization of comfort preferences	End-users share their comfort preferences and daily schedules	Comfort boundaries, daily schedules	Prosumer	BFMS
2	Optimization framework formulation	Initial settings algorithms formulate the optimization framework and introduce the appropriate constraints based on the end-users' inputs regarding comfort preferences & daily schedules	Flexibility asset data, Comfort boundaries, daily schedules	BFMS	BFMS
3	Demand response profiles	Generation of targeted, comfort-based demand response profiles	Flexibility forecasts	BFMS	Prosumer (HEMS/BEMS) Aggregator, FMO, DSO through Dataspace connector

Realization

Main responsible partners (Author) Add one or maximum two partners from this list:

- HYP

Contributing partners

- ETRA
- JR
- AMIBIT
- BLUE
- EP
- EL
- AEM
- IMPULSA

Priority High.

Table 7: UC1.9 - Initialization of HEMS/BEMS Demand Response strategy

The only SGAM layer affected by Dataspace integration is the communication layer. The process of collecting static configuration data as well as actual energy data from the HEMS/BEMS to set up the consumers' preferences and/or constraints at the BFMS will take place through data space connectors.

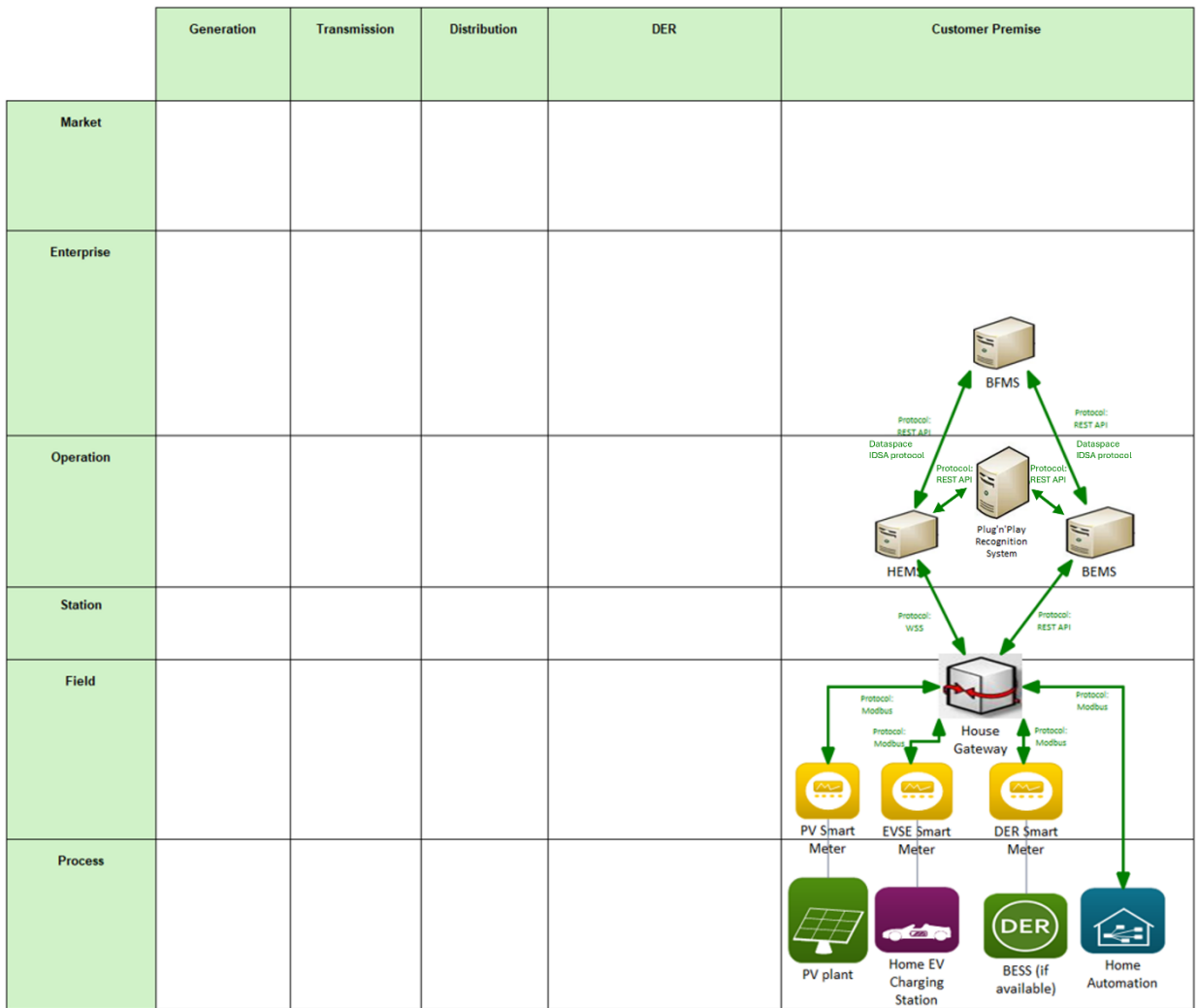


Figure 37: UC1.9 SGAM Diagram