SEcure Decentralized Intelligent Data MARKetplace

D5.2 Integrated releases of the SEDIMARK platform. First version

Document Identification	
Contractual delivery date:	31/03/2024
Actual delivery date:	23/04/2024
Responsible beneficiary:	WINGS
Contributing beneficiaries:	WINGS, UC, ATOS, LINKS, SURREY, EGM, SIE, NUID UCD, INRIA
Dissemination level:	PU
Version:	1.0
Status:	Final

Keywords:

Independent scenarios PoCs, integrated releases, data flows, development process, software components integration, PoC specification, functional/non-functional requirements, continuous integration, continuous delivery, use cases, SEDIMARK platform

This document is issued within the frame and for the purpose of the SEDIMARK project. This project has received funding from the European Union's Horizon Europe Framework Programme under Grant Agreement No. 101070074. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

Document Information

Document Identification			
Related WP	WP5	**Related Deliverables(s):**	SEDIMARK_D5.1
Document reference:	SEDIMARK_D5.2	**Total number of pages:**	81

List of Contributors	
Name	**Partner**
Panagiotis Vlacheas Grigorios Koutantos	WINGS
Alberto Carelli Andrea Vesco Luca Giorgino	LINKS
Pablo Sotres Juan Ramón Santana Jorge Lanza Luis Sánchez	UC
Maxime Costalonga	ATOS
Tarek Elsaleh	SURREY
Marianne Marot Franck Le Gall Romain Manganni Sacha Bydon	EGM
Diarmuid O'Reilly-Morgan Erika Duriakova Honghui Du Elias Tragos	NUID UCD
Gabriel-Mihail Danciu	SIE
Maroua Bahri	INRIA

Document History			
0.1	01/03/2024	WINGS	First draft of ToC
0.11	22/03/2024	UCD	Contribution to Chapter 3.5
0.12	22/03/2024	LINKS	Contribution to "Participant Onboarding" Section
0.2	01/04/2024	WINGS	Merge inputs from partners
0.3	03/04/2024	WINGS	Updates to Section 3 and Section 4
0.31	03/04/2024	LINKS	Contribution to "Cross-check of High Priority Requirements"
0.4	10/04/2024	UC	Fill in Section 2, remove Section 5
0.5	12/04/2024	INRIA, WINGS	Update Section 3.2 and Section 4, Section 3.5 and general formatting
0.51	17/04/2024	SIE, UC, WINGS	Update Section 4
0.6	18/04/2024	WINGS	Cleaned version for internal review
0.8	22/04/2024	WINGS	Final version to be submitted to ATOS for Quality Review
0.9	24/04/2024	ATOS	Quality Review Form
1.0	23/04/2024	ATOS	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	**Who (Partner short name)**	**Approval date**
Reviewer 1	Juan Echevarria (SDR)	20/04/2024
Reviewer 2	Maxime Costalonga (ATOS) Cesar Caramazana Zarzosa (ATOS)	20/04/2024
Quality manager	María Guadalupe Rodríguez (ATOS)	23/04/2024
Project Coordinator	Miguel Ángel Esbrí (ATOS)	24/04/2024

Table of Contents

Document Information 2
Executive Summary 12
1 Introduction 13
1.1 Purpose of the document 13
1.2 Relation to other work packages and tasks 13
1.3 Structure of the document 13
2 First integrated release of SEDIMARK platform 15
3 Independent scenarios PoCs 17
3.1 Participants onboarding 17
3.1.1 High-level description 17
3.1.2 Step-by-step definition and data flows 18
3.1.3 Specifications of involved components 19
3.1.4 Integration specification 19
3.1.5 Results 20
3.1.6 Guidelines for deployment and execution 21
3.1.7 Software licences 30
3.2 Data quality improvement 30
3.2.1 High-level description 30
3.2.2 Step-by-step definition and data flows 31
3.2.3 Specifications of involved components 32
3.2.4 Integration specification 35
3.2.5 Results 35
3.2.6 Guidelines for deployment and execution 37
3.2.7 Software licences 39
3.3 Offering lifecycle 40
3.3.1 High-level description 40
3.3.2 Step-by-step definition and data flows 41
3.3.3 Specifications of involved components 42
3.3.4 Integration specification 43
3.3.5 Results 43
3.3.6 Guidelines for deployment and execution 44
3.3.7 Software licences 44

3.4 Asset (Data) exchange 44
3.4.1 High-level description 44
3.4.2 Step-by-step definition and data flows 45
3.4.3 Specifications of involved components 46
3.4.4 Integration specification 47
3.4.5 Results 47
3.4.6 Guidelines for deployment and execution 48
3.4.7 Software licences 49
3.5 AI-related scenarios 49
3.5.1 High-level description 49
3.5.2 Step-by-step definition and data flows 50
3.5.3 Specifications of involved components 52
3.5.4 Integration specification 53
3.5.5 Results 53
3.5.6 Guidelines for deployment and execution 62
3.5.7 Software licences 63
3.6 GUIs 63
3.6.1 High-level description 63
3.6.2 Step-by-step definition and data flows 64
3.6.3 Specifications of involved components 65
3.6.4 Integration specification 66
3.6.5 Results 67
3.6.6 Guidelines for deployment and execution 67
3.6.7 Software licences 67
3.7 Open data enabler 68
3.7.1 High-level description 68
3.7.2 Step-by-step definition and data flows 69
3.7.3 Specifications of involved components 69
3.7.4 Integration specification 70
3.7.5 Results 71
3.7.6 Guidelines for deployment and execution 71
3.7.7 Software licences 71
4 Cross-check of high-priority requirements 72
5 Conclusions 80
6 Bibliography 81

List of Tables

Table 1 Assessing quality and some different transformations on input data. 36
Table 2 Functional requirements of the architecture related to the different PoCs 72
Table 3 Non-functional requirements of the architecture related to the different PoCs.....78

List of Figures

Figure 1 General view of SEDIMARK first iteration Proof-of-Concept scenarios 15
Figure 2 Involved components in the Participant Onboarding PoC 18
Figure 3 Successful completion of onboarding of a participant 21
Figure 4 Home page (provisional) for participant onboarding 22
Figure 5 Setup of the communication between user Connector and Issuer. 23
Figure 6 Generation of a DID: before creation (left) and after (right) 24
Figure 7 DID Document recorded onto the DLT 25
Figure 8 Request of VC to the Issuer 26
Figure 9 Credential Subject information 27
Figure 10 Metamask browser extension upon receiving VCs 28
Figure 11 Issuer operations 29
Figure 12 Example of JWT encoding for a VC 29
Figure 13 Decoding of the JWT in the plain VC 30
Figure 14 involved components in the data quality improvement PoC. 31
Figure 15 Sample screenshot 36
Figure 16 Detail of components involved in offering lifecycle scenario 40
Figure 17 Offering Lifecycle 41
Figure 18 Overview of component involvement throughout offering life-cycle stages 42
Figure 19 API Test Suite for Offering Lifecycle (Creation) 44
Figure 20 Detail of components involved in asset exchange scenario 45
Figure 21 SEDIMARK connector during scenario execution 48
Figure 22 AI related scenario component interaction 50
Figure 23 AI-related scenario data flows 52
Figure 24 Predictions using XGBoost 54
Figure 25 Scatter plot using XGBoost 55
Figure 26 Predictions using Neural Network 56
Figure 27 Scatter plot using Neural Network 57
Figure 28 MSE validation loss over iterations for the decentralized machine learning model trained on Node 0 and Node 1. Average loss is calculated for the global model. 58
Figure 29 Validation MAE over iterations for the decentralized machine learning model trained on Node 0 and Node 1. Average loss is calculated for the global model. 59
Figure 30 Predictions using Federated Learning on test set 60
Figure 31 Scatter plot using Federated Learning on test set 61
Figure 32 GUIs scenario components 64

Figure 33 Marketplace architecture 66
Figure 34 Open data scenario components 68
Figure 35 Open data enabler working principle 70

List of Acronyms

Abbreviation / Acronym	Description
ABAC	Attribute-Based Access Control
AI	Artificial Intelligence
API	Application Programming Interface
BSD	Berkeley Source Distribution
CI/CD	Continuous Integration and Continuous Delivery/Continuous Deployment
CKAN	Comprehensive Knowledge Archive Network
CSV	Comma Separated Values
DAG	Directed Acyclic Graph
DID	Decentralized Identifier
DLT	Distributed Ledger Technology
DSP	Dataspace Protocol
Dx,y	Deliverable number y belonging to WP x
EC	European Commission
EDC	Eclipse Dataspace Components
EUPL	European Union Public Licence
FAIR	Findable Accessible Interoperable Reusable
FT	Fungible Tokens
GNU	Gnu's Not Unix
GPL	General Public Licence
GUI	Graphical User Interface
gRPC	gRPC Remote Procedure Call
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
H-REQ	High-priority Requirements
ID	Identity Document

Abbreviation / Acronym	Description
IDSA	International Data Spaces Association
IOTA	Internet of Things Application
ISC	IOTA Smart Contract Framework
IP	Internet Protocol
ISCP	IOTA Smart Contract Protocol
KPIs	Key Performance Indicators
JS	JavaScipt
JSON	JavaScript Object Notation
JWT	JSON Web Token
L1/L2	Layer 1/ Layer 2
LAN	Local Area Network
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MAPE	Mean Absolute Percentage Error
ML	Machine Learning
MSE	Mean Square Error
MVP	Minimum Viable Product
NF	Non-Functional
NFT	Non-fungible Token
NGSI-LD	Next Generation Service Interfaces for Linked Data
OM	Offering Manager
PEP	Policy Enforcement Point
PoC	Proof of Concept
RDF	Resource Description Framework
ReLU	Rectified linear unit
RMSE	Root Mean Square Error
SC	Smart Contract
SPARQL	SPARQL Protocol and RDF Query Language

Abbreviation / Acronym	Description
SSI	Self-Sovereign Identity
SSR	Server-Side Rendering
TBD	To be defined
TLS	Transport Layer Security
ToC	Table of Contents
UC	Use Case
URL	Uniform Resource Locator
VC	Verifiable Credentials
WP	Work Package
YAML	Yet Another Markup Language

Executive Summary

The current document is the second deliverable of WP5 and reports the results of Task 5.2 activities regarding continuous platform integration. The actions adhere to the integration activities of the modules and components developed in other work packages (especially WP3 “Distributed data quality management and interoperability” and WP4 “Secure data sharing in a decentralized Marketplace”) based on the previously defined WP2 architecture. The target audience of the document is mainly the technical partners of SEDIMARK but it offers valuable insights to other stakeholders.

Before delving into the core of the deliverable, the document briefly analyses the implementation of the first integrated release of the SEDIMARK platform in terms of status, which includes deviations and challenges. Seven independent PoCs scenarios will ensure the proper functionalities of the platform. All technology providers are accountable for the various modules to which they are assigned based on a top-down integration plan outlined in the document SEDIMARK_D5.1 [1] too. Some architecture components are not included in the platform's first version because they are part of the platform's second and final releases.

All scenarios will be thoroughly analysed by their leaders on the following topics:

- High-level description: Scope of the scenario, relation to other modules or scenarios
- Step-by-step definition and data flows: Same approach as described in SEDIMARK_D5.1, including updates on the steps of each scenario, accompanied with possible figures, related datasets and dataflows applied.
- Specifications of involved components: Brief description of the components used (references or links to the respective deliverable could be used).
- Integration specification: Integration steps (inter-component communication, setup monitoring and logging, implement CI/CD).
- Results: Output of each scenario, available figures and screenshots, results analysis.
- Guidelines for deployment and execution: Guidelines, tips, or tricky points during the execution of the scenario.
- Software licences: Identify licences for each component per scenario.

It is important to verify that the requirement specifications defined in WP2 and Tasks T2.1-T2.4 are fulfilled at all implementation phases. For this purpose, a table is provided correlating all the high-priority requirements (H-REQ) that were promised to fulfill for the first version, with the PoCs. The aim is to monitor the status of fulfillment and in which PoC scenario they are addressed. The table covers both functional and non-functional requirements.

After the documentation of the first integrated release is completed, there is an overview of the subsequent versions of the SEDIMARK platform. The idea is to extend the current functionalities, fulfill the additional requirements and sophistication of components, and decrease the amount of coding as the versions progress. The integration will consider the timeplan for releasing the SEDIMARK integrated platform in three versions, as described in [1].

1 Introduction

1.1 Purpose of the document

The main objective of this deliverable is to unveil the first integrated release of the SEDIMARK platform while meticulously scrutinizing the implementation of its current version. Leveraging proof-of-concept scenarios previously examined in SEDIMARK_D5.1, this deliverable aims to offer a comprehensive analysis of the platform's functionalities. The overarching aim is to present a version of the platform that encapsulates all fundamental functionalities and addresses high-priority requirements. This initial release paves the way for subsequent iterations where features and functionalities will be expanded upon. By providing a detailed examination of the current implementation, this deliverable equips project partners with invaluable insights for forthcoming work. It serves as a blueprint for further development efforts, guiding the extension of features and functionalities in subsequent releases. While the primary audience for this deliverable is the project partners, it also offers value to other stakeholders with aligned interests. They can glean useful ideas for crafting appropriate methodologies or refining their own projects based on the insights and strategies outlined within.

1.2 Relation to other work packages and tasks

This deliverable is the outcome of Task 5.2 (Platform continuous integration) and is the continuation of the work done during the first year of the project especially in Task T5.1 (Integration and Evaluation plan and methodologies). SEDIMARK_D5.2 is a follow-up deliverable of SEDIMARK_D5.1, elaborating on the scenarios developed there. The work presented in the document is strongly related to the components and tools developed in WP3 and WP4, which involve the technical aspects of the platform's development and create the overall decentralised marketplace, based on the architecture design of Task T2.3 and the interfaces specified in Task T2.4. The output of SEDIMARK_D5.2 will also be used as input to the upcoming activities of the remaining tasks (Task T5.3, Task T5.4) of WP5 for the three integrated releases of the SEDIMARK platform which will be presented in three phases (M18-Mar. 2024, M27-Dec. 2024, M36-Sep. 2025) and analysed in the current deliverable SEDIMARK_D5.2 (Integrated releases of the SEDIMARK platform. First version), and forthcoming deliverables SEDIMARK_D5.3 (Integrated releases of the SEDIMARK platform. Second version) and SEDIMARK_D5.4 (Integrated releases of the SEDIMARK platform. Final version). These future deliverables will likely document further progress and refinements to the platform. This gradual platform deployment allows beneficiaries to gain valuable insights into performance and make any necessary adjustments or improvements.

1.3 Structure of the document

This document is structured into 5 major chapters:

- **Chapter 1** is the current chapter of the report and stands as an introduction to the next sections.
- **Chapter 2** presents a high-level description of the first integrated release of the SEDIMARK platform.

- **Chapter 3** is the core chapter of the document and focuses on the independent scenarios PoCs that will be implemented and their initial integration status.
- **Chapter 4** validates the correlation of the scenarios with the high-priority requirements of the architecture.
- **Chapter 5** concludes the document, summarizing the main outcomes and the future steps in alignment with the objectives and project roadmap.

2 First integrated release of SEDIMARK platform

The initial release of the SEDIMARK platform is a significant milestone in our project. We have decided to showcase the platform's capabilities through seven independent scenarios, each serving as a proof-of-concept for the implementation and results of the different concepts explored within the project. These scenarios aim to demonstrate the versatility and robustness of our architecture.

Figure 1 depicts the different scenarios, which will be described in Section 3, in an overlay layer on top of the system view of the SEDIMARK platform (originally explained in SEDIMARK_D2.2 and also used in SEDIMARK_D5.1). This provides a visual representation of how each component interacts within the system and contributes to each scenario. As it can be seen in the figure, even though these scenarios operate independently, they are often interconnected, demonstrating the synergy between different modules and components of the system. Altogether, the defined set of proof-of-concept scenarios serve the purpose of providing a comprehensive overview of the platform's functionalities.

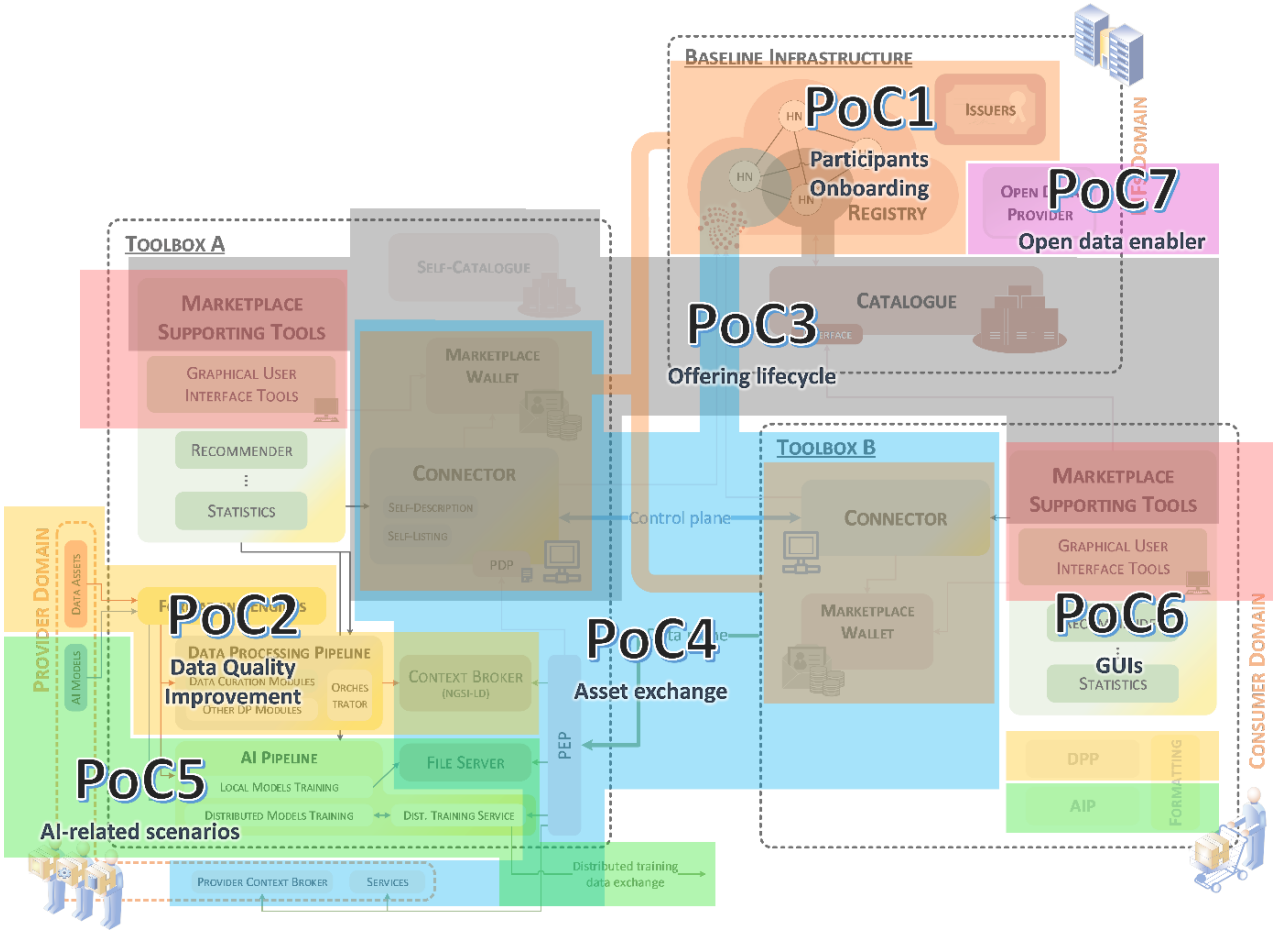


Figure 1 General view of SEDIMARK first iteration Proof-of-Concept scenarios

The scenario descriptions provide a clear and concise summary of their scope and relation to other modules or scenarios, often relying on the system view of the SEDIMARK platform. Furthermore, the proof-of-concept encompasses specific information, such as high-level descriptions, step-by-step instructions, detailed component specifications, guidelines for deployment or integration steps. The output of each scenario, including images or screenshots where available, is also analysed to provide insights into the practical results and implications.

As the SEDIMARK platform evolves and improves throughout successive iterations, we will incrementally enhance and refine its components to improve performance and functionality. In this sense, we will broaden the platform's capabilities by implementing new components in order to make the platform more versatile by serving a broader range of use cases and scenarios. With this in mind, the main goal is to focus on seamless integration, ensuring that the PoC scenarios run sequentially from the same set of interfaces, creating a more streamlined and cohesive workflow. This will make it easier for users to navigate and manage different scenarios.

3 Independent scenarios PoCs

In accordance with the outlined three-step plan detailed in SEDIMARK_D5.1, the first release of the integrated SEDIMARK platform is aimed at facilitating all functionalities tasked with meeting high-priority requirements. Instead of consolidating into an integrated platform right away, a series of fundamental scenarios have been sketched. These scenarios encompass critical procedures and events that the SEDIMARK platform is expected to support. To implement each scenario, a subset of the components under development in the project will be integrated, thereby presenting an initial version of the platform.

However, although the seven foundational scenarios are closely interconnected, they will be addressed independently in this initial version of the SEDIMARK platform. It is assumed that the prerequisites for any scenario being fulfilled elsewhere will already be provided.

The description of the seven scenarios will be focused on the following aspects:

- High-level description: Overview of the scenario's scope, emphasizing its connection to other modules or scenarios within the system architecture.
- Step-by-step definition and data flows: Similar approach as SEDIMARK_D5.1, provide updates on the steps of each scenario, incorporate figures or diagrams illustrating data flows, related datasets, and any specific dataflows applied during the scenario execution.
- Specifications of involved components: Brief descriptions of the components utilized in the scenario, referencing or providing links to the corresponding deliverables.
- Integration specification: Outline the integration steps required for the scenario, including inter-component communication protocols, setup for monitoring and logging, and implementation details for continuous integration and continuous deployment (CI/CD) pipelines.
- Results: Present the outputs generated from executing the scenario, potentially including images or screenshots to illustrate key findings. Analyze the results obtained, providing insights into the performance or functionality of the scenario.
- Guidelines for deployment and execution: Provide guidelines, tips, or tricky points during the execution of the scenario. Highlight any potential challenges or complexities that may arise and provide strategies for addressing them effectively.
- Software licences: Identify licences for each component per scenario (e.g. open source).

3.1 Participants onboarding

3.1.1 High-level description

The Participants onboarding scenario encompasses the steps necessary for becoming a proper user of the SEDIMARK Marketplace. Through this procedure, either Providers and Consumers are enabled to acquire an account making them able to interact with each other within the Marketplace.

The Onboarding process is a necessary procedure that takes care of the generation and registration of appropriate digital identities of new users who want to profit from the features and services offered by the Marketplace.

The account is realized through the credentials owned by the new user in the form of Decentralized Identities and Verifiable Credentials (VC), according to the Self-Sovereign Identity (SSI) model. The account will be employed to use different functionalities of the marketplace by the Trust Layer in conjunction with authentication and authorization policies.

Figure 2 shows the system view of the key components involved in this scenario. It has to be pointed out that the procedure follows the same steps in the case of an account for a Provider (Toolbox A) or for a Consumer (Toolbox B). The depicted decentralized infrastructure (Baseline Infrastructure), described in SEDIMARK_D4.1, is actually partitioned into two interacting layers: Layer 1 (L1) which is the IOTA ledger and Tangle and Layer 2 (L2) which is the IOTA Smart Contract Framework (ISC). The Issuer, belonging to the SEDIMARK domain, creates the VC for the user requesting access to the Marketplace.

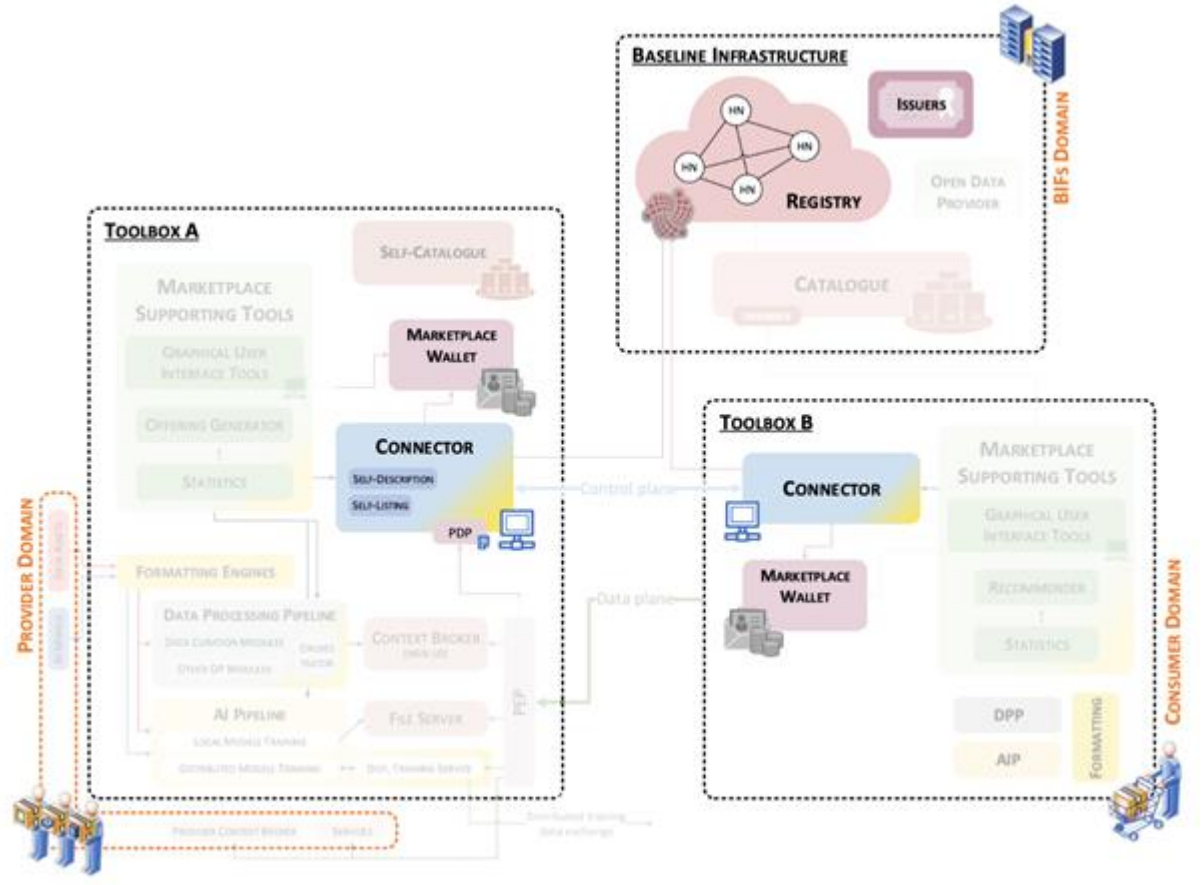


Figure 2 Involved components in the Participant Onboarding PoC

This PoC contributes to the Security and Trust domain by employing the distributed ledger and by giving a basic form of access to the functionalities available in the Marketplace. The implementation of this scenario enables an external user to become part of the SEDIMARK Ecosystem, to allow interactions with other actors involved and to benefit from the services offered by the Marketplace, according to proper authorization mechanisms.

3.1.2 Step-by-step definition and data flows

The mandatory step to "use" the Marketplace is to register a new digital identity according to the SSI model.

An external user, who desires to be part of the SEDIMARK Marketplace, needs to instantiate its own Connector and generate a new identity to get the credentials from the Issuer.

More in detail, the steps of such a procedure are the following. The new user needs to:

1. Create its own DID (Decentralized Identifier).
   a) Generate a set of cryptographic keys (public and private).
   b) Build the DID-Document embedding the public keys onto the DID-Document.
   c) Publish the DID-Document onto the Distributed Ledger.
2. Request the Verifiable Credentials (VCs) to the Issuer of the SEDIMARK Marketplace.
3. Store the received VCs locally and securely.

It has to be pointed out that the Issuer, before releasing the VC, performs additional steps:

1. Retrieves the DID-Document from the distributed ledger.
2. Verifies the DID ownership, i.e., performs a challenge-response with the user.

In the case of a successful verification, the Issuer:

1. Creates the VC.
2. Signs the VC with its own Private Key.
3. Communicates to the user the VC requested.

If the verification fails, the Issuer does not return any VC to the user.

3.1.3 Specifications of involved components

From the architectural point of view, there are two core layers involved in this scenario.

On the one hand, there is the Trust Layer, defined in SEDIMARK_D2.2, focusing on the security domain of SEDIMARK. The onboarding procedure realizes in fact a form of authentication mechanism that allows to access the services offered by the Marketplace only to its users.

On the other hand, the Interaction Layer is also heavily involved. This layer deals with the interactions with the DLT (Distributed Ledger Technology), described in SEDIMARK_D4.1 [2], which in turn is realized in the decentralized Infrastructure, detailed in SEDIMARK_D3.3 [3]. The various steps in the onboarding scenario interact with the DLT, which is employed to store the data containing the public user information, i.e., the DID document. Such requirements enable third parties and other entities to access the document for implementing the verifications steps needed in the SSI paradigm to allow secure communication and proper policy check.

The other involved component is the Frontend or GUI, found in the Service Layer, which allows a friendly interaction for the user with the rest of the Marketplace functionalities. At the moment of writing this deliverable, the GUI employed is not in its final status but is instead an ongoing work. For the sake of the PoC demonstration, the GUI is separated from the general interface, which is discussed in SEDIMARK_D4.5 [4].

3.1.4 Integration specification

At the time of writing of this deliverable, such PoC is software available as a web application. The main software for a new user desiring to take part in the SEDIMARK Marketplace, realizing the application logic of the onboarding procedure, is the DLT Enabler module residing at the Connector.

For the sake of simplicity and the target of this PoC, we do not distinguish among Connector and DLT Enabler in the current Section.

Nevertheless, in order to have a standalone PoC, other services must be instantiated as well. The following steps can be considered as pre-requirements to run this scenario:

- Start local development environment to partially recreate the final decentralized infrastructure with functionality of both L1 and L2 locally).
- Start the Issuer.
- Start the Frontend for the DLT Enabler and the Issuer.

These steps build the infrastructure and the services that will be employed for the onboarding procedure.

The previous steps are necessary for this PoC. It has to be pointed out that after the whole Marketplace is finalized and the components are integrated, there would be no need to reproduce such steps anymore.

As a requirement, the user must have a cryptocurrency wallet installed. The wallet of choice is Metamask, which will be used for interacting with the Smart Contracts (SC) and for buying and selling assets available on the SEDIMARK Marketplace.

The user can install the wallet as a browser extension from Metamask [5].

From the perspective of the integration, the whole onboarding procedure realised in this PoC will be integrated into the Marketplace. The decentralised identity functionality is deeply nested into the core business logic of the secure Marketplace, spanning from purchases of assets to browsing of the offerings. At the time of writing, this procedure does not interact directly with other scenario. However, authenticating an user of the SEDIMARK Marketplace is a fundamental pre-requirement for enabling and interacting with other scenarios described in this document and for every other service of the Marketplace.

All the components mentioned above should co-exist and must interact with each other, as well as with other components of the Marketplace. In particular, the DLT Enabler should be part of the final SEDIMARK Connector.

The functionalities offered by the frontend components of this PoC should be instead merged with the general Marketplace GUI/Frontend to have one single uniform interface for the SEDIMARK End-Users.

3.1.5 Results

At the completion of this scenario, the user is able to gain a digital identity according to the SSI model for the SEDIMARK Marketplace.

The completion of the flow of the Participant onboarding is shown in Figure 3:

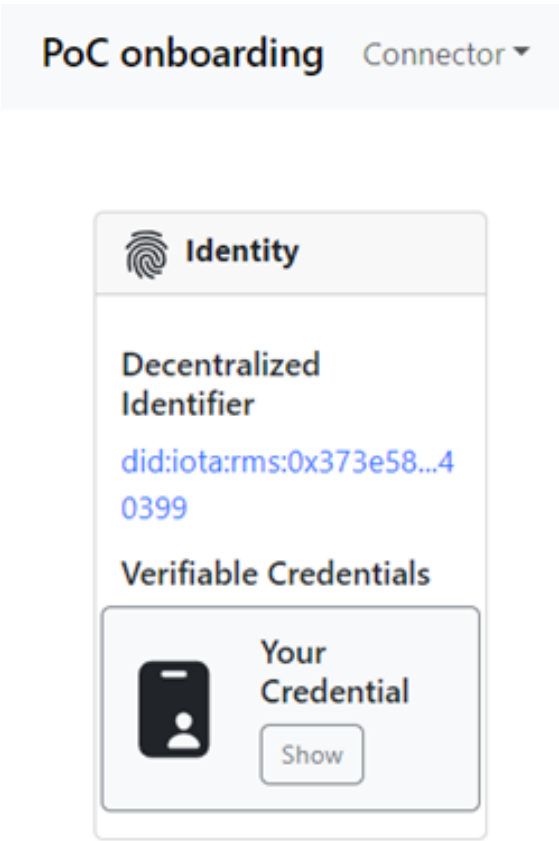


Figure 3 Successful completion of onboarding of a participant

The VC obtained is stored locally and securely together with the private key of the identity generated. Additional elements stored are the DID, the DID-Document, and the public key of the user. The DID Document is also stored onto the DLT to be retrieved by other parties (e.g., the Issuer, Asset Provider).

After this procedure, a new user is able to access the functionalities offered by the SEDIMARK Marketplace.

3.1.6 Guidelines for deployment and execution

To perform the onboarding procedure, a user shall instantiate its own Connector/DLT-Enabler.

With respect to the Onboarding procedure, the Connector is mainly composed of a database for locally storing the identity of the user (i.e., the credentials) with other personal information and a backend service.

After the successful boot-up of the Connector, the user can open the browser at the address https://localhost:5174/issuer to find the interface for the onboarding. The page is shown in Figure 4.

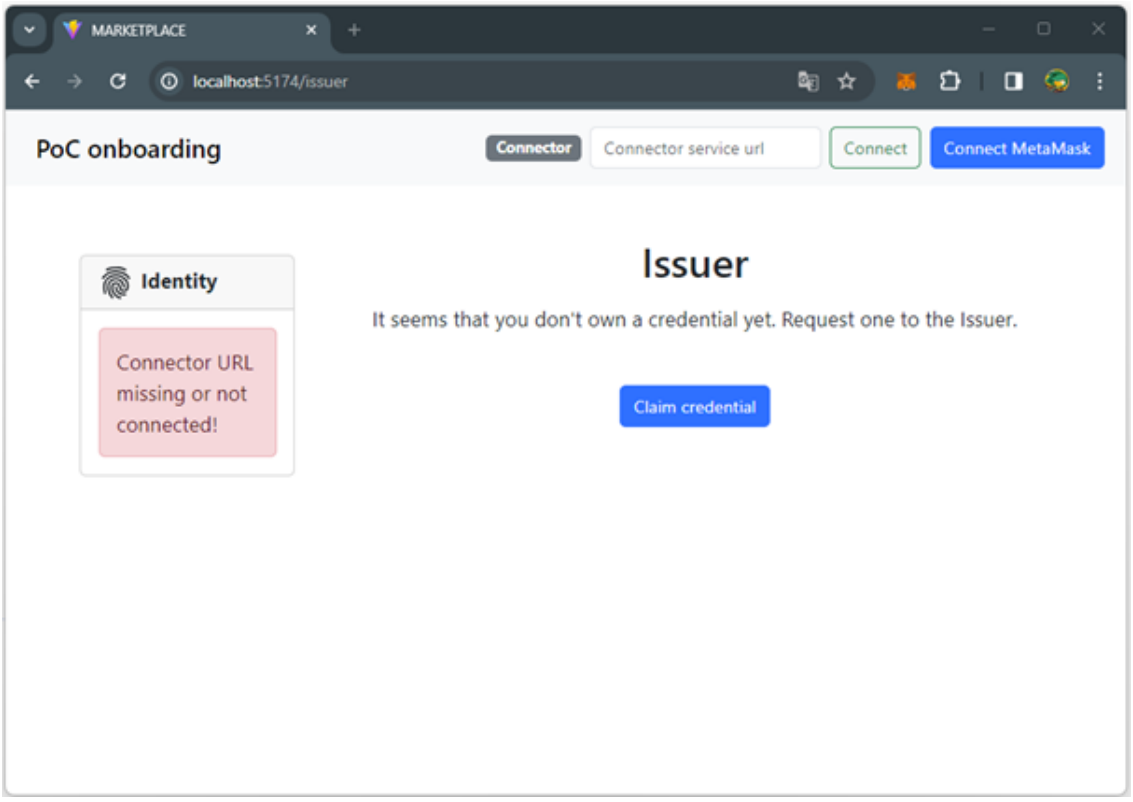


Figure 4 Home page (provisional) for participant onboarding

The user must establish a communication with the Issuer of the credentials, as shown in Figure 5, for contacting it later on.

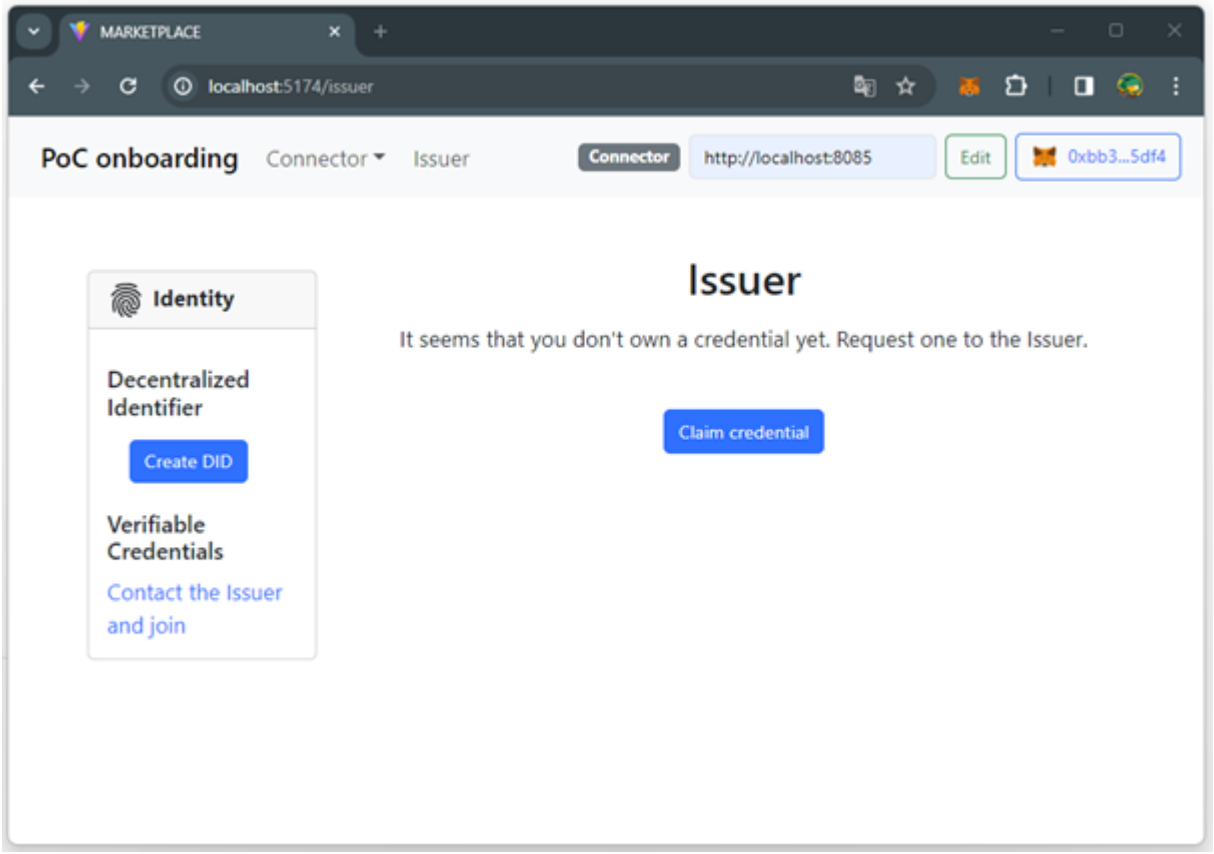


Figure 5 Setup of the communication between user Connector and Issuer.

Following the steps detailed in Section 3.1.2, the new user shall first generate its own DID, as figured in Figure 6.

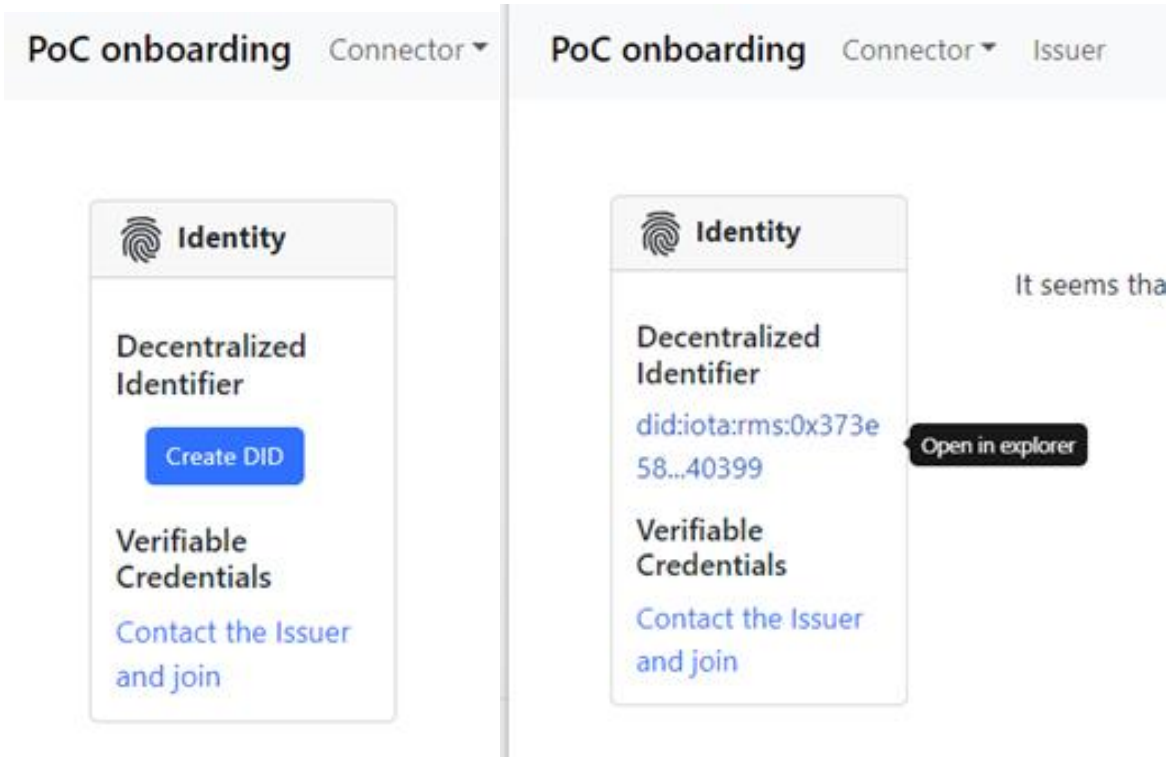


Figure 6 Generation of a DID: before creation (left) and after (right)

During this step, the cryptographic material needed (e.g., the keypair, DID, etc.) is generated and securely stored locally. Finally, this step triggers the communication with the DLT that is employed to store the newly generated DID-Document. Such DID-Document can be retrieved directly from the DLT, as shown in Figure 7.

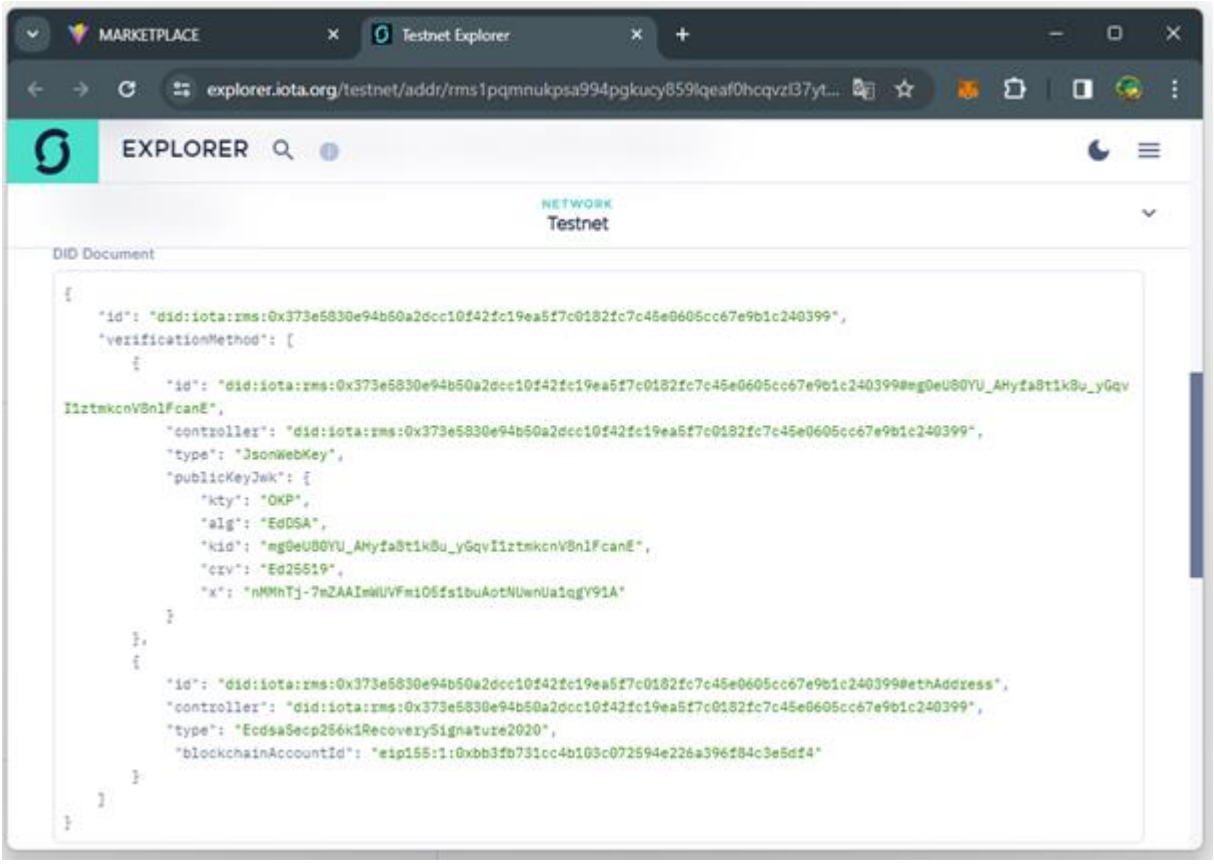


Figure 7 DID Document recorded onto the DLT

After creating its DID, the user requests the credentials for accessing the Marketplace to the Issuer, as shown in Figure 8.

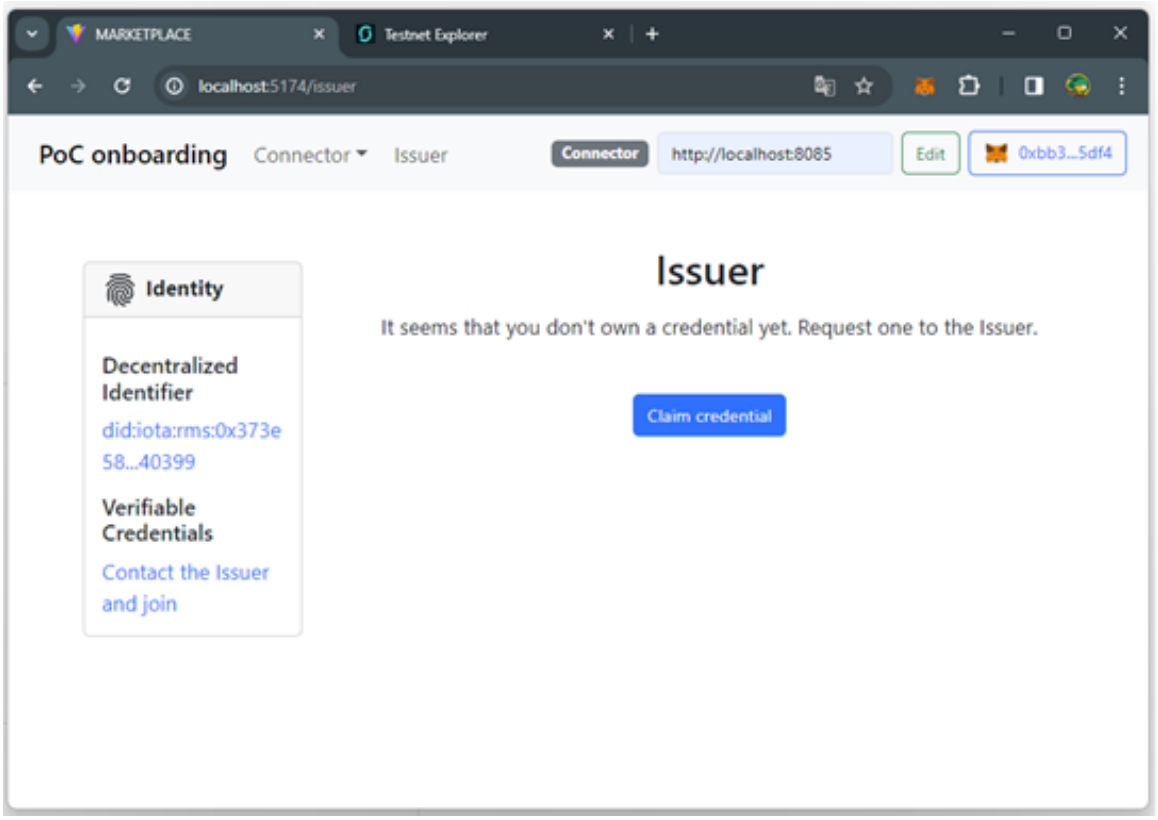


Figure 8 Request of VC to the Issuer

The user should fill in the information defined in the Credential Subject, as shown in Figure 9. For this PoC, the personal information is simplified and limited to “Name” and “Surname”.

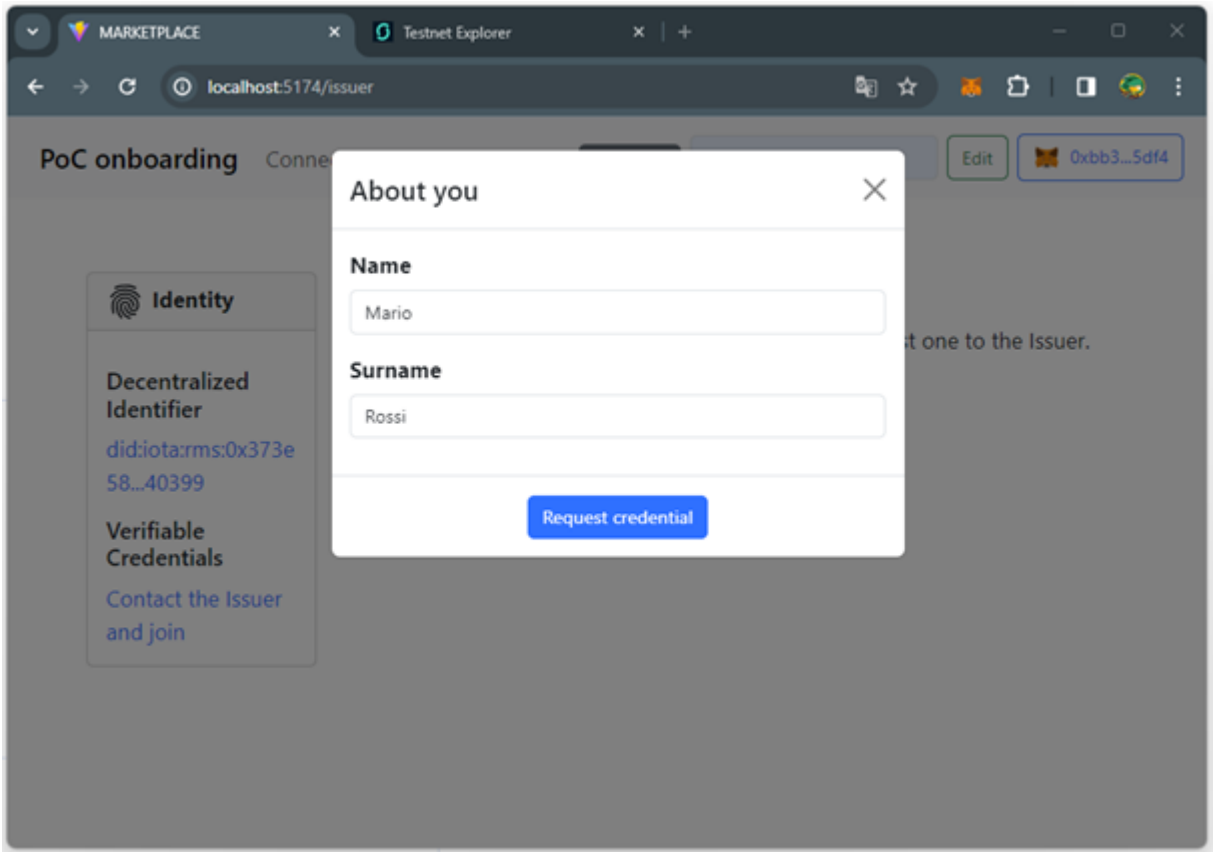


Figure 9 Credential Subject information

To verify the DID ownership, the Issuer asks to the user to sign a challenge. This operation triggers the Metamask wallet on the user's side, as shown in Figure 10.

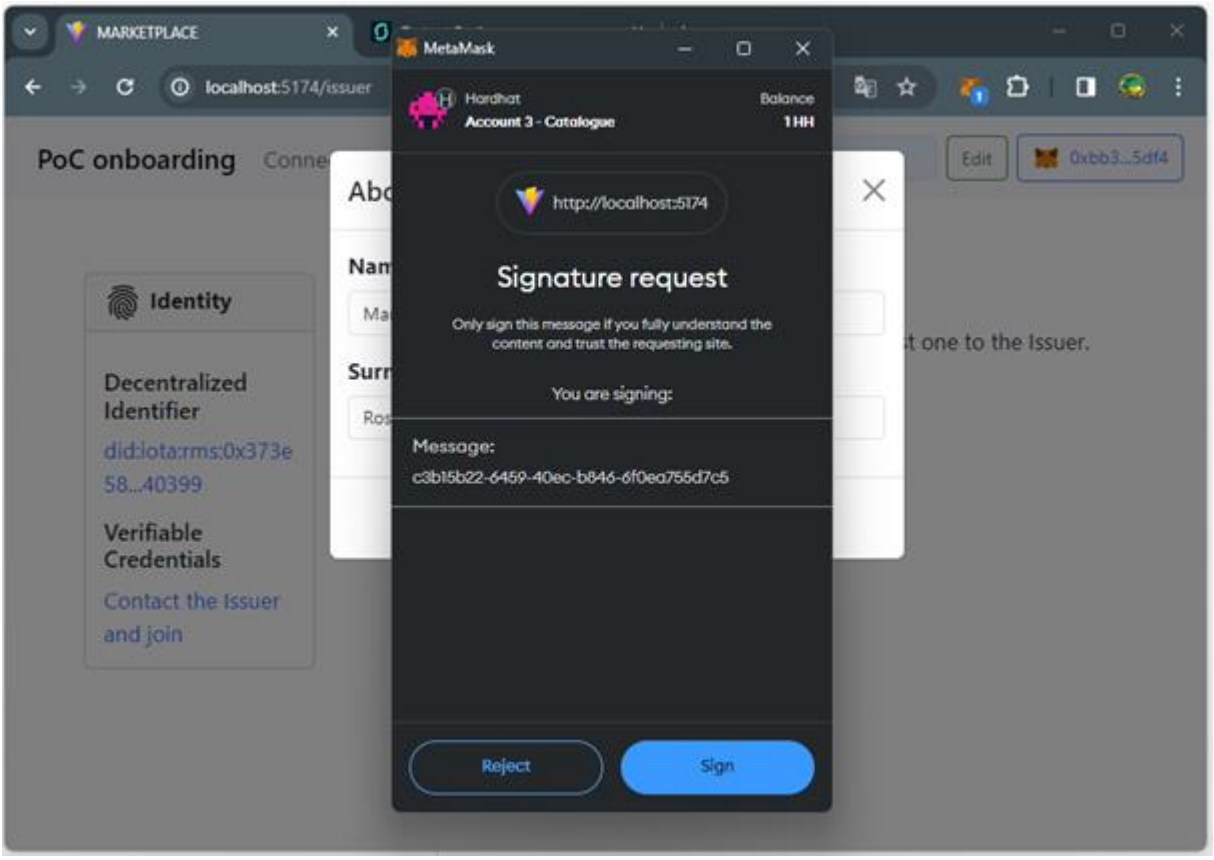


Figure 10 Metamask browser extension upon receiving VCs

After completing the necessary verification the Issuer generates and signs the VC, which is sent back to the user, as described in Figure 11.

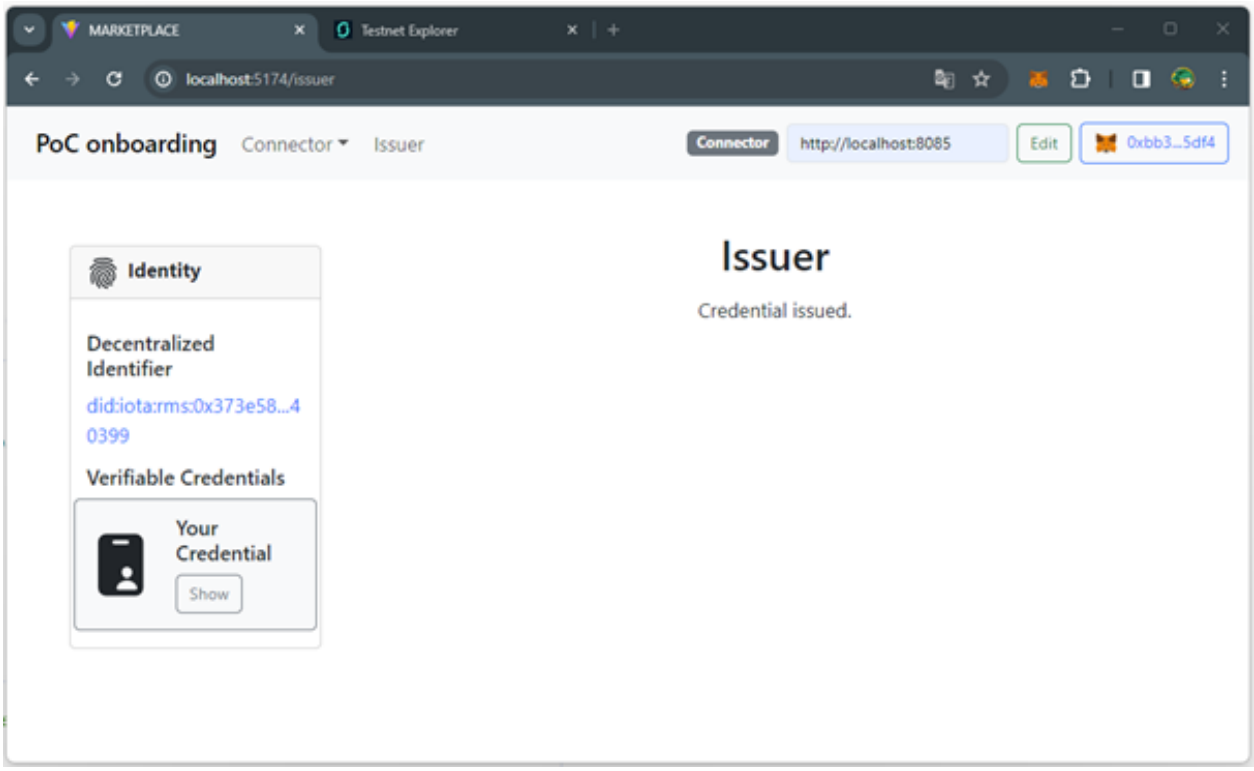


Figure 11 Issuer operations

Upon receiving the VC, the user completes the onboarding procedure.

The Verifiable Credential received is in JWT (JSON Web Token) format and it is securely stored in the Connector.

An example of VC encoded into a JWT is in the following Figure 12.

```
eyJraWQiOiJkaWQ6aW90YTpybXM6MHhhMTM2Yjk1YTFmMzc0ZWQwN2M1NDkzOTQ0ZmMwYWYyZj
M5NmU3YzExNTAzNDhhZmJhZGVlYmI5ZmVkNjExYzMxI3BjT0lUbnUxSTBqSm9HM3JJNnFaZGtr
VTVLaDUwNnctYnExc2t5QjBXOTAiLCJ0eXAiOiJKV1QiLCJhbGciOiJFZERTQSJ9.eyJleHAiO
jE3NDIwNTIyNjksImlzcyI6ImRpZDppb3RhOnJtczowGExMzZiOTVhMWYzNzRlZDA3YzU0OTM
5NDRmYzBhZjJmMzk2ZTdjMTE1MDM0OGFmYmFkZWViYjlmZWQ2MTFjMzEiLCJuYmYiOjE3MTA0M
jk4NjksImp0aSI6Imh0dHBzOi8vZXhhbXBsZS5tYXJrZXQvY3JlZGVudGlhbHMvMSIsInN1YiI
6ImRpZDppb3RhOnJtczowDM3M2U1ODMwZTk0YjUwYTJkY2MxMGY0MmZjMTllYTVmN2MwMTgyZ
mM3YzQ1ZTA2MDVjYzY3ZTliMWMyNDAzOTkiLCJ2YyI6eyJAY29udGV4dCI6Imh0dHBzOi8vd3d
3LnczLm9yZy8yMDE4L2NyZWRlbnRpYWxzL3YxIiwidHlwZSI6WyJWZXJpZmlhYmxlQ3JlZGVud
GlhbCIsIk1hcmtldHBsYWNlQ3JlZGVudGlhbCJdLCJjcmVkZW50aWFsU3ViamVjdCI6eyJuYW1
lIjoiTWFyaW8iLCJzdXJuYW1lIjoiUm9zc2kiLCJ1c2VyT2YiOiJTRURJTUFSSyBtYXJrZXRwb
GFjZSJ9fX0.Z1Y7H1tYIQ8nRq0dxxkM9lvi4pKoulrgFsiMx94cmIY8zrBAyOV42Eo7mlZNIp3
HlGQLSAm6WXh6_-IcfF5ZDg
```

Figure 12 Example of JWT encoding for a VC

The JWT can be decoded to an understandable format to read the information it contains. Figure 13 shows this optional step (e.g., using service such as JWT [6]).

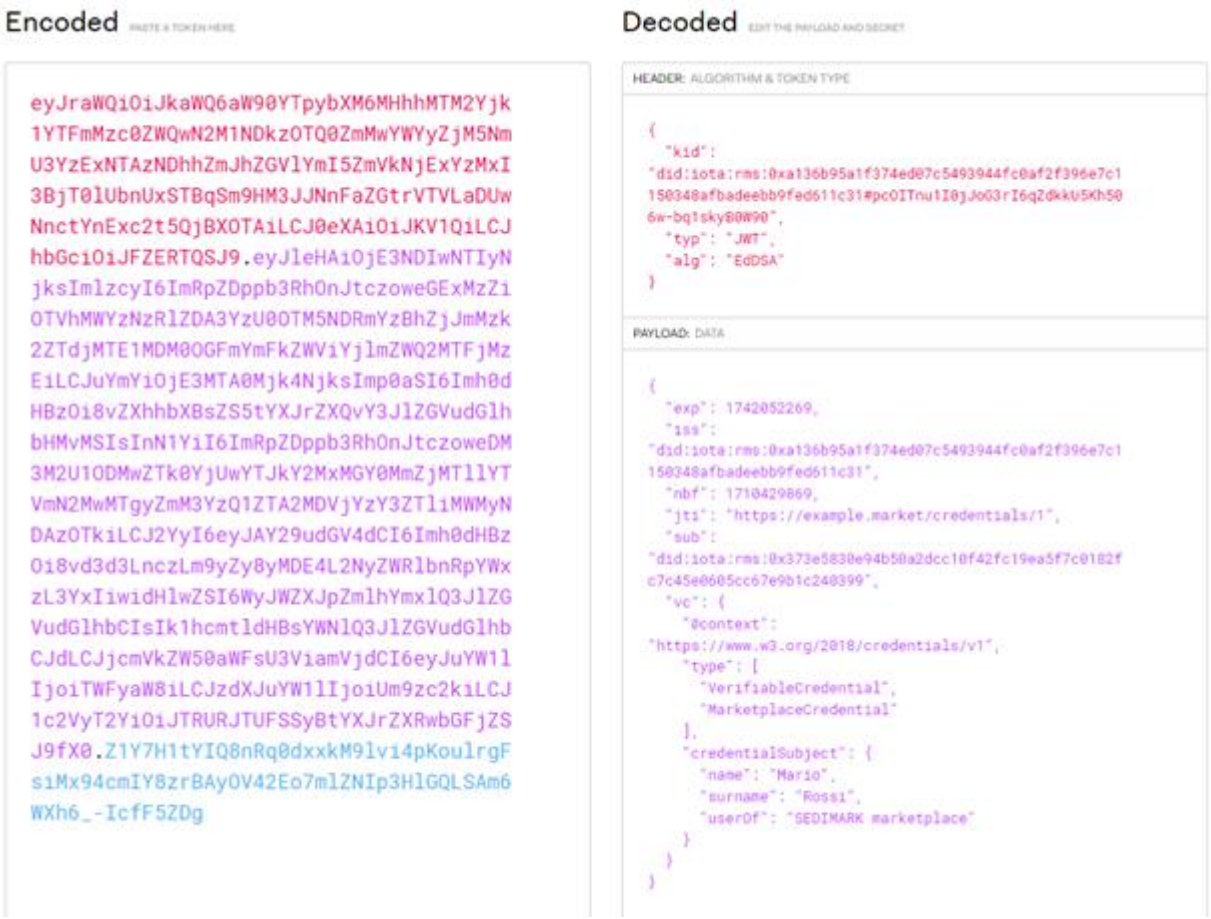


Figure 13 Decoding of the JWT in the plain VC

3.1.7 Software licences

The license of the software developed for realizing the Participants onboarding PoC is released under GNU General Public License v3.0 or later (GPL-3.0-or-later).

3.2 Data quality improvement

3.2.1 High-level description

This scenario implies the water valuation use case (France), leverages weather forecast data for Les Orres, France, from a broker, analyzing it for anomalies using a default-configured Anomaly Detection module, and then exporting the analyzed data back to the broker as a new entity. In this way, it contributes to the implementation of the data quality improvement functionalities of the SEDIMARK toolbox. It involves the Data Processing Pipeline, the Context Broker, and the File Server functional entities.

At this milestone of the project, and on the progression line from defined KPIs in Deliverable SEDIMARK_D5.1, the data quality of a dataset or data point is now annotated with metrics and semantically labelled, and successfully run through a processing pipeline. The augmented

dataset then integrated into the SEDIMARK data catalogue via a FIWARE NGSI-LD compliant innovative data model.

The data used within this step represent open datasets of water flow and water level data from flow meters installed up and downstream from a dam context, and atmospheric temperature from a local weather station (open access). The goal of this exercise is to predict the missing data at one of the stations which connects all others, i.e. the station at the dam, to infer when it will release water. The sensor at the dam does not supply real-time but deferred time data - i.e. a few months after the current time, contrary to the other sensors which transmit data in real-time.

The purpose of this general scenario is to implement this process as a distributed service with a federated learning process at the edge, locally. The components involved in that PoC are described in the following Figure 14.

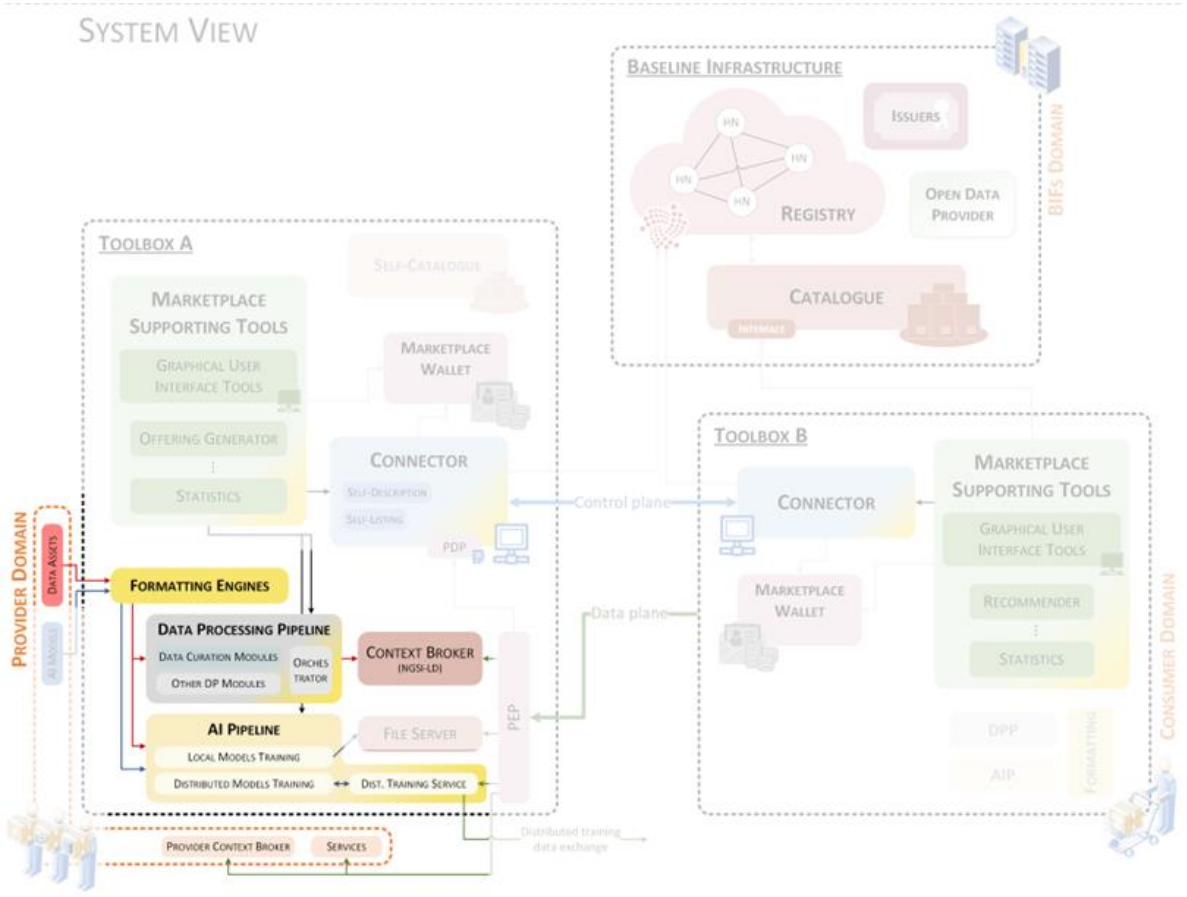


Figure 14 involved components in the data quality improvement PoC.

3.2.2 Step-by-step definition and data flows

The process is dockerized and run on a local server. The data are acquired from the Context Broker "Stellio" either as bulk datasets or as real-time streaming data for each sensor. The user (data provider or client) sets up a processing pipeline on these data using the data

cleaning modules, AI model training and prediction modules, semantic enrichment and data annotation modules from the SEDIMARK toolbox.

Data is loaded from a broker system, specifically targeting weather information identified by the entity ID urn:ngsi-ld:WeatherInformation:Forecasted:Hourly:France:Les_Orres.

A "DataSource" is created to encompass all the imported data. Following this, an Anomaly Detection module is instantiated with a default configuration to analyze the data for any irregularities.

A new entity is created in the same broker to hold the results of the analysis. The annotated data, which now includes detected anomalies, is exported back to the broker for access and use.

The data Orchestrator "Mage.ai" [8] is an open-source pipeline editor and manager used for creating pipelines and triggers, running and synchronizing processes, and monitoring and integrating data. The resulting output (ex. AI model parameters, transformed data, ⋯) is tagged, updated and sent back into the Context Broker for distributed storage via an API. MLFlow is used as a data model manager for stored AI models and hyperparameters in Minio.

3.2.3 Specifications of involved components

Currently, the scenario has been tested with 3 different processing pipelines, using a historical bulk dataset, demonstrating the data quality modules of the SEDIMARK toolbox used in varying levels of complexity:

- **Scenario A: Single variate data quality pipeline:** Consists of uploading data from either a CSV or json format from any sensor, and then ingesting into a NGSI-LD context broker using data adapters. The ingestion process serialises data using the NGSI-LD information model. From the context broker, data is then harmonized, profiled, deduplicated, imputated for any missing data (by method of interpolation), and outlier detection algorithms are applied. Data annotation and semantic enrichment of the dataset are then performed while the dataset progresses over the pipeline. Data annotation includes a Boolean property that annotates an anomaly, and a threshold score (based on the area under the curve method) generated by the decision score of the AI anomaly detection algorithm (based on the minimum covariance determinant method) reflecting the level of confidence in the detected anomaly. Additionally, another complementary property is the anomaly score, which indicates the degree of outlierness of a given data.

  Data quality annotation model adopted to model the data draws from the data quality assessment and encompasses several key properties [7] including:

```
{
  "id": <string>,
  "type": <string>,
  "source": <string>,
  "dateCalculated": <datetime>,
  "accuracy": <float>,
  "completeness": <float>,
  "timeliness": <float>,
  "precision": <float>,
  "outlier": {
    "isOutlier": <boolean>,
    "methodology": <string>,
    "outlierScore": <float>
  },
   "duplicate": {
     "isDuplicate": <boolean>,
     "foundMatches": [<string>]
  },
  "synthetic": {
    "isSynthetic": <boolean>,
    "methodology": <string>
  }
}
```

- **Scenario B: Single variate data prediction pipeline:** Consists of applying a data quality pipeline of data loading, formatting, and imputing missing data (interpolation), necessary for any AI model training, for a single water flow station at the dam and air temperature data from a local weather station. Data inference is then conducted using the Ludwig toolbox [9] built on top of TensorFlow [10] which allows users to train and test deep learning models. In this pipeline, we infer the last 20 values of the data time series for which their result is subsequently evaluated by comparing with the observed (target) values. The new dataset is then sent back into the Context Broker.
- **Scenario C: Multi-variate data prediction pipeline:** Consists of applying a data quality pipeline of data loading, formatting, and imputing missing data (interpolation method), necessary for any AI model training, for all 8 water stations, including the dam's water flow rate. Data inference is then conducted using the Ludwig toolbox for the last 3 values of the data. The accuracy of the predicted result is evaluated by comparing it with the observed (target) values. Once validated (the results are closest to the observed), the new dataset is sent back into the Context Broker.

The pipelines can be triggered when new data is recorded.

The code configurations are as follows:

Anomaly Detection Model Configuration

```
config = {

    "model" : 'pyod_mcd',

    "processing_options":'describe',
```

```
    "model_config" : {

        'threshold_type':'AUCP',

    },

    "data_type":'tabular'

}
```

The anomaly detection module represents a wrapper over several Python libraries, including PyCaret [11], Scikit-learn [12] and PyOD [13]. It offers flexibility in configuration, allowing users to adjust various parameters to suit their specific needs. One of the key parameters that can be adjusted is the threshold type, which influences how anomalies are treated.

Broker Connection Details

```
bucket ={'host': 'https://stellio-dev.eglobalmark.com',

        'url_keycloak':  'https://sso.eglobalmark.com/auth/realms/sedimark/protocol/openid-connect/token',

        'client_id': secret.client_id,

        'client_secret': secret.client_secret,

        'username': secret.username,

        'password': secret.password,

        'entity_to_load_from': 'urn:ngsi-ld:WeatherInformation:Forecasted:Hourly:France:Les_Orres',

        'entity_to_save_in': entity_id,

        'entitiy_id': entity_id,

        'link_context': 'https://easy-global-market.github.io/ngsild-api-data-models/sedimark/jsonld-contexts/sedimark.jsonld',

        'tenant': 'urn:ngsi-ld:tenant:sedimark',

        'time_query': 'timerel=after&timeAt=2023-08-01T00:00:00Z',

        'content_type': 'application/json'

}
```

The bucket is a Python dictionary designed to facilitate connectivity with the Stellio broker for data extraction purposes. Its configuration requires essential parameters for setting up and authentication, ensuring seamless interaction with the broker.

3.2.4 Integration specification

Apache NiFi [14] is used for data ingestion from original sources and serialization against a NGSI-LD information model.

Stellio NGSI-LD Context broker is used for storing and serving data through a Restful NGSI-LD API. It is associated with EGM Twin. Picks user interface for ingestion flow management as well as Grafana for visualisation dashboards.

MageAI is employed for both the creation and processing of the specified pipeline.

React and ReactFlow are utilized to present the pipelines in a manner that is accessible and user-friendly.

The Broker serves the purpose of supplying data to the MageAI pipeline, which is executed through the User Interface developed using React and ReactFlow.

PyOD is applied in the development of an Anomaly Detection algorithm, which operates on the data provided by the broker.

Lastly, Mage API facilitates the connection between the User Interface and the MageAI backend.

3.2.5 Results

The results of the computations are cleaned, annotated and semantically enriched datasets (or predicted datasets), which are stored back into the Context Broker with FAIR principles, accessible to the user for reading or further computations.

The scenarios described above can be refined by:

- Including the offering to the choice of the datasets.
- Adding the new datasets to the offering.
- Running the scenarios using the GUI

Work in progress consists of :

- Refining current pipeline processes to data streaming,
- Acquiring process performance metrics for individual parallel processes rather than global processes in order to assess precisely the carbon footprint of the implemented distributed storage.

Dataset curation: the initial dataset of interest contains records spanning from 1960 to 2023, in a challenging format. Initially, the dataset is partitioned into two separate CSV files; the process implies the merge of the two datasets, obtaining a final dataset, suitable for future processing.
The dataset is evaluated, extracting from it the necessary information. This process, including the imputation of missing values, takes 5.57 seconds. For reference, the final dataset contains 23376 rows and 4 columns, while the initial one contains 119988 records. A sample screenshot is provided in Figure 15.

```
Time elapsed: 5.576550483703613 seconds
```

waterStation observedAt	X031001001	X045401001	X050551301	X051591001
1960-01-01 00:00:00	30100.0	10300	44000.0	4439.0
1960-01-02 00:00:00	29400.0	10300	44000.0	4439.0
1960-01-03 00:00:00	29400.0	10300	44000.0	4439.0
1960-01-04 00:00:00	27800.0	10300	42000.0	4439.0
1960-01-05 00:00:00	27800.0	10300	42000.0	4439.0

Figure 15 Sample screenshot

Process criterion:

Table 1 Assessing quality and some different transformations on input data.

Pipeline Block Name	Execution Time (seconds)	Short Description
jsonld_loader	56.16	Loads jsonld file into the pipeline
Create_folders	0.67	Creates output folders
Jsonld_transformer	10.875	Extracts some data from the given jsonld
Jsonld_exporter	0.75	Exports the jsonld to file
Broker_loader	19.884	Retrieves data from the Stellio broker for a given entity
Anomaly_detection	66.953	Detects anomalies and their score
Train_hydro_series	10.865	Training of water flow model
Predict_future_data	7.665	Testing the water flow predictions on simulated data
Predict_x050551301	3.75	Predicts on input data coming from the Stellio broker
Train_ludwig	249.15	Trains the model using alternative training model

3.2.6 Guidelines for deployment and execution

The pipeline is deployed using a containerised environment based on docker. Deployment is facilitated using tools such as Ansible for deployment automation and Kubernetes as managed infrastructure.

Once deployed, the Apache NiFi data ingestion provides a "low-code" user interface available within a web browser while the context broker interface is accessible through API calls using user-friendly tools such as postman or directly from the Mage.ai pipeline. Data access is secured through an authentication and authorization interface, based on Keycloak and integrated within Stellio. Installation procedures are available pnline in [15].

The pipeline orchestrator Mage.ai interface is accessible from a url. Once in the orchestrator, the user may select a pipeline to trigger or edit, from the list of pipelines presented. The user may then precise or modify parameters and input data, based on his needs and objectives. The orchestrator also allows the user to create new pipelines with the available modules at his disposition in the folders. He may as well modify it at code level.

Setting up a Kubernetes Cluster

Install a Kubernetes cluster for ease of deployment. In this case, k3s is chosen for its simplicity. To install k3s, execute the command:

```
curl -sfL https://get.k3s.io | sh -
```

Configuring the Cluster

Once k3s is installed, prepare the cluster configuration:

- Create a `.kube` directory in your home folder using `mkdir ~/.kube`.
- Copy the configuration file to this directory with:

  ```
  sudo cp /etc/rancher/k3s/k3s.yaml ~/.kube/config.yaml
  ```

  Note: This step requires sudo privileges.
- Change the permission of the copied configuration file:

  ```
  sudo chmod 600 ~/.kube/config.yaml
  ```
- Edit the `~/.bashrc` file, appending the following line to export the `KUBECONFIG` variable:

  ```
  export KUBECONFIG=~/.kube/config.yaml
  ```

```
alias desc="kubectl describe"

export KUBECONFIG=/home/ubuntu-machine/.kube/config.yaml
```

Alternatively, use your full home directory path in place of `~`.

Cluster Management Tools

Install **kubectl** from Kubernetes' official documentation.

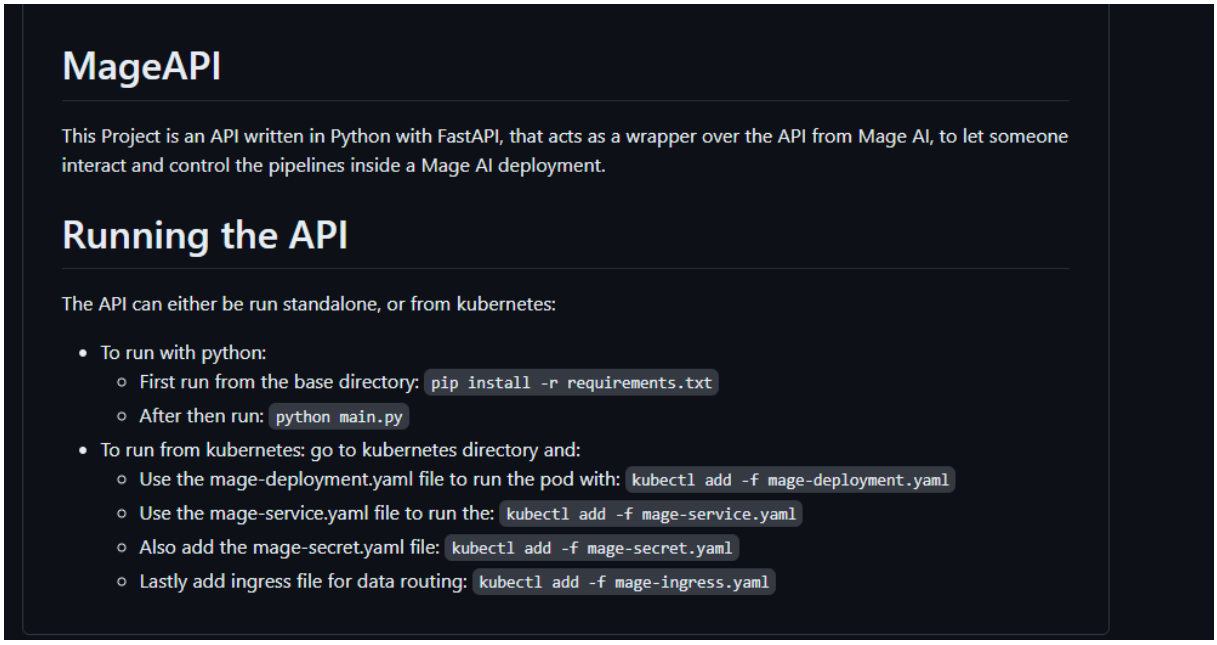
Install **Helm** from Helm's installation guide (using the script option).

Setting Up Nginx Ingress

Follow the tutorial on GitHub for Nginx Ingress to install Nginx Ingress using Helm. This will handle request routing within the cluster. (https://github.com/kubernetes/ingress-nginx/blob/main/charts/ingress-nginx/README.md)

Deploying the API

Deploy the API that interacts with Mage AI by following instructions from Sedimark's MageAPI repository on GitHub. (https://github.com/Sedimark/MageAPI)



Preparing Mage AI Environment

Copy the `mage_data` folder from the above repository into your Linux home directory.

Also, copy the `values.yaml` file to the home directory and edit the file at line 140, replacing `<user>` with your username.

```
137   extraVolumes:
138     - name: mage-fs
139       hostPath:
140         path: /home/<user>/mage_data
```

To deploy Mage AI, execute:

```
helm install --values ./values.yaml my-mageai mageai/mageai
```

Installing Dependencies for Mage AI

Identify the Mage AI pod by running `kubectl get pods`.

```
NAME                                        READY   STATUS    RESTARTS        AGE
models-api-deployment-78f6dd6996-8ds9s      1/1     Running   2 (2d1h ago)    3d5h
go-api-deployment-7976d89bbc-h9b44          1/1     Running   4 (2d1h ago)    3d6h
ingress-nginx-controller-69f9df69db-pdl6h   1/1     Running   5 (2d1h ago)    17d
mage-api-sedimark-deployment-766f584b68-mtwmf   1/1     Running   12 (2d1h ago)   16d
mageai-67b6b9c4b7-q9ql4                     1/1     Running   18 (14h ago)    17d
```

Install the necessary dependencies within the Mage AI deployment by executing:

```
kubectl exec -it <pod_name> -- pip install -r /home/src/default_repo/requirements.txt
```

Followed by:

```
kubectl exec -it <pod_name> -- pip install ./sedimark_dqp-dqp_develop
```

Replace `<pod_name>` with the actual name of the Mage AI pod.

Accessing Mage AI

To view Mage AI, determine the service port by running `kubectl get svc`. Access Mage AI via `http://localhost:<port>`, using the credentials `admin@admin.com` for username and `admin` for password.

```
ubuntu-machine@ubuntumachine-System-Product-Name:~/kubernetes$ kubectl get svc
NAME                                 TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)                      AGE
kubernetes                           ClusterIP      10.43.0.1       <none>          443/TCP                      21d
ingress-nginx-controller-admission   ClusterIP      10.43.24.68     <none>          443/TCP                      17d
ingress-nginx-controller             LoadBalancer   10.43.219.102   <pending>       80:30945/TCP,443:31855/TCP   17d
mage-api-sedimark-service            ClusterIP      10.43.222.115   <none>          49149/TCP                    17d
mageai                               LoadBalancer   10.43.33.255    192.168.8.15    6789:30121/TCP               17d
go-api-service                       ClusterIP      10.43.116.116   <none>          49151/TCP                    3d7h
models-api-service                   ClusterIP      10.43.87.53     <none>          49152/TCP                    3d6h
```

3.2.7 Software licences

Stellio context broker, MLFlow, mage.ai are available under Apache License Version 2.0

MinIO is dual licensed under GNU AGPL v3 and commercial license.

PyOD comes in two variants, the BSD 2-Clause and BSD 3-Clause.

3.3 Offering lifecycle

3.3.1 High-level description

This scenario contributes to establishment and management of the Offering lifecycle within the SEDIMARK ecosystem. The Offering will serve as the means of describing Assets available for exchange and used by the Catalogue. It is expected that Offerings will undergo changes based on the Providers preferences and therefore need to be considered.

Figure 16 provides a visual representation of the key components involved in this specific scenario. It illustrates how these components, as described in the system view of our architecture, interact and contribute to the overall functionality of the system. This diagram serves as a guide to understanding the practical implementation of the scenario, thus it is referenced across different subsections.

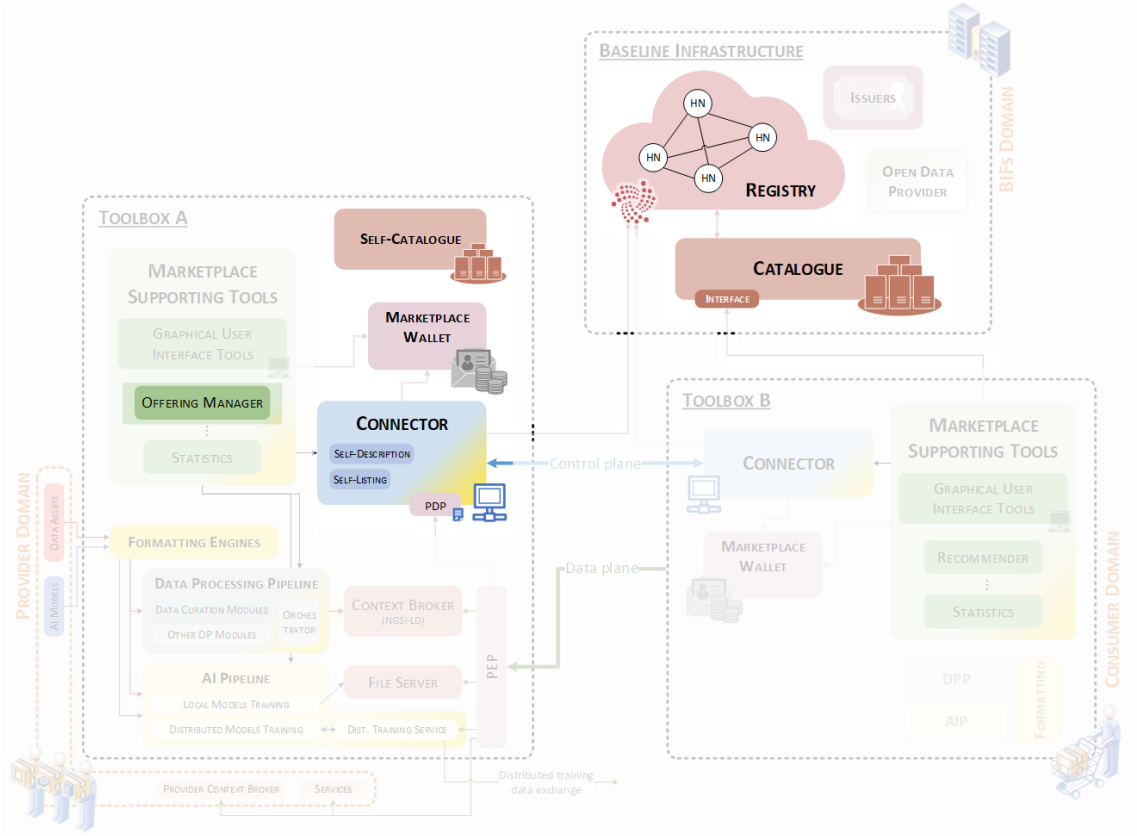


Figure 16 Detail of components involved in offering lifecycle scenario

The scenario will enable the publication of Offerings from Providers to the Marketplace. It will also enable Consumers to search and discover Offerings through a distributed Catalogue. This will be done through a defined set of interactions between modules within the SEDIMARK toolbox and the Baseline Infrastructure. The lifecycle is split into 2 main phases, Registration, and Discovery.

The lifecycle consists of several stages: Creation, Registration, Population, Discovery, Update, Withdrawal (Figure 17).

3.3.2 Step-by-step definition and data flows

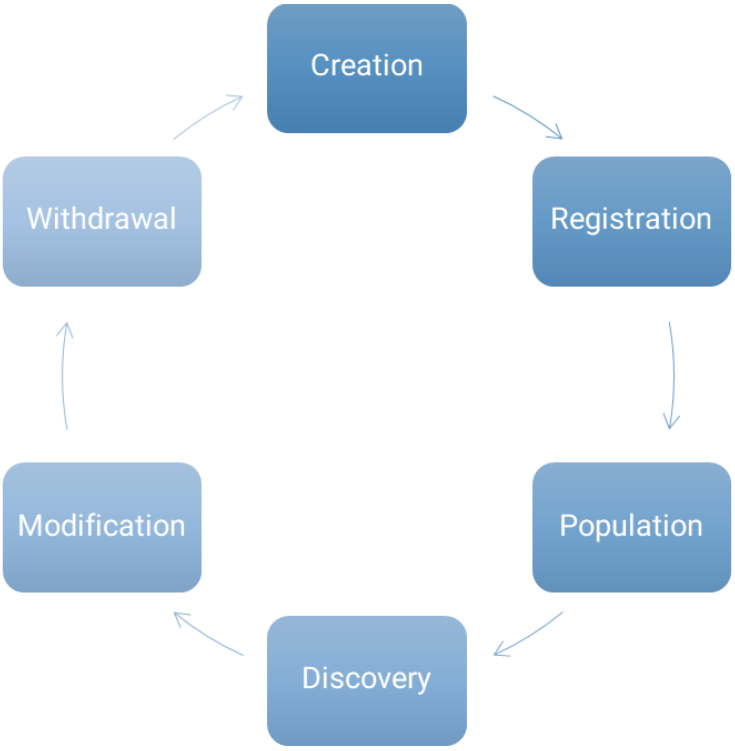


Figure 17 Offering Lifecycle

Creation

The first stage involves the creation of an Offering Description that conforms to the Marketplace Information Model. For this PoC, offering is created manually. In future iterations, the usage of a Marketplace Supporting Tool (i.e. Offering Generator) is foreseen. The created Offering format is validated for compliance using the tools provided by the Interoperability Enabler. Upon correct validation, the Offering is then stored on the Provider's Connector's Self-Listing, making it available through the Connector.

Registration

The second stage involves storing the endpoint of the Connector exposing the Offering description and the hash of the Offering description in the Registry. In practice, the connector interacts with a smart contract on the Registry (L2 ISC) in order to register the offering. This results in an NFT representing the offering, which includes a hash of the offering and a URL pointing to the full offering description in the self-listing. Registry does provide trustworthiness to the whole operation, and access to offering descriptions is distributed across the different provider's connectors. Anyone with the needed credentials (e.g. participants or catalogues) can use the registry to retrieve the existing offerings and query/crawl the corresponding self-listings in a trustworthy manner. Besides, interested parties (e.g. catalogues) can be made aware of new offering registration (i.e. notified) so they can index them.

Population

The third stage involves the Catalogue either proactively querying or subscribing to the Registry for new or updated Offerings. New Offering descriptions are retrieved through Connector's Endpoint and indexed based on Descriptions attributes or properties. Local Catalogues, if available, can also be used for storing subsets of the Catalogue.

Discovery

The fourth stage involves the creation of requests for Offerings of interest. The constructed Query is then sent to the Catalogue to obtain existing Offerings. Current implementation of the catalogue is based on the use of Triple-stores, so a SPARQL endpoint will be used to demonstrate the proof of concept.

Modification

The fifth stage considers that Offering descriptions need to be updated due to changes in provisions. This involves the retrieval of the Offering Description, modifying it, validating it for compliance, storing the description at the Self-Listing, and storing the new hash in the Registry.

Withdrawal

The sixth stage involves removing the Offering from the marketplace. A request is created to remove the Offering. The Offering is first checked if it is in the Self-Listing and Registry, and is then removed from the Self-Listing. The Registry is then updated that the Offering is no longer available, which is then flagged and can be removed/ignored by the Global Catalogue when it is re-populating.

3.3.3 Specifications of involved components

While Figure 15 depicts the components involved in this specific scenario within the broader context of the SEDIMARK system view architecture, Figure 18 complements it by showing how these components are involved at different offering life-cycle stages.

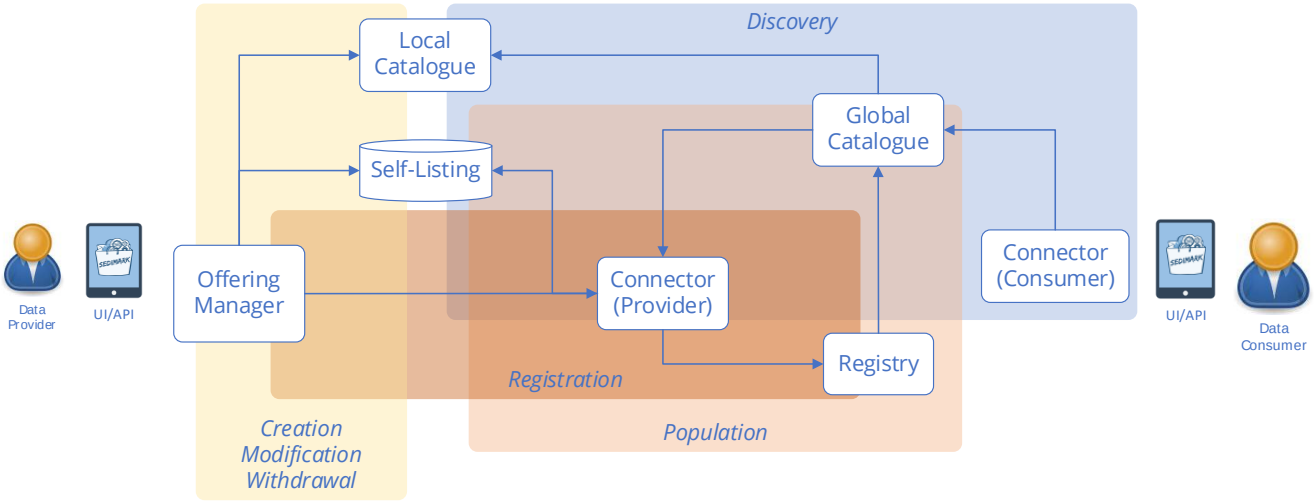


Figure 18 Overview of component involvement throughout offering life-cycle stages

Offering Manager

This component is responsible for managing Offerings submitted by the Provider. It exposes a RESTful API to the Provider for interaction. The Manager will validate a submitted Offering against the Marketplace Information Model.

Self-Listing

The Self-Listing is the primary repository for storing Participant Offerings validated by the Offering Manager.

Connector

Among other functionalities, the Connector provides the means to register a new Offering at the Registry.

Registry

The Registry acts as the means of registering Offering on a DLT that will be used for verification.

Global Catalogue

The Global Catalogue interacts with the Registry to discover new and updated Offerings, and then in turn retrieves the Offering descriptions from their respective Self-Listings.

Local Catalogue

The Local Catalogue will provide an interface for the Global Catalogue to query for Offerings offered by the hosting Provider or a subset of Offerings made available within the Marketplace instance.

3.3.4 Integration specification

The Offering Manager (OM) mainly orchestrates the process of creation, modification and withdrawal. The OM employs a set of interfaces that are based on the SPARQL Graph Store HTTP Protocol and the EDC management API. The Connector employs the DLT Enabler API for the Registration process. The Catalogue also employs the DLT Enabler API for discovering Offerings, and then retrieves the Offering Description from the Provider's Connector using the EDC management API.

This scenario interacts with the following ones:

- Onboarding scenario, as a DID might be needed to interact with the L2 ISC smart contract. Besides, some information from the participant Self-description might also be needed during this scenario.
- Marketplace GUIs for:
  - Offering discovery to browse the catalogue of offerings.
  - Offerring publication.
  - Offering management which is a dashboard to manage/edit offerings to be implemented.

3.3.5 Results

Results will be verified through a test suite for testing the API calls for each stage of the Offering Lifecycle (Figure 19).

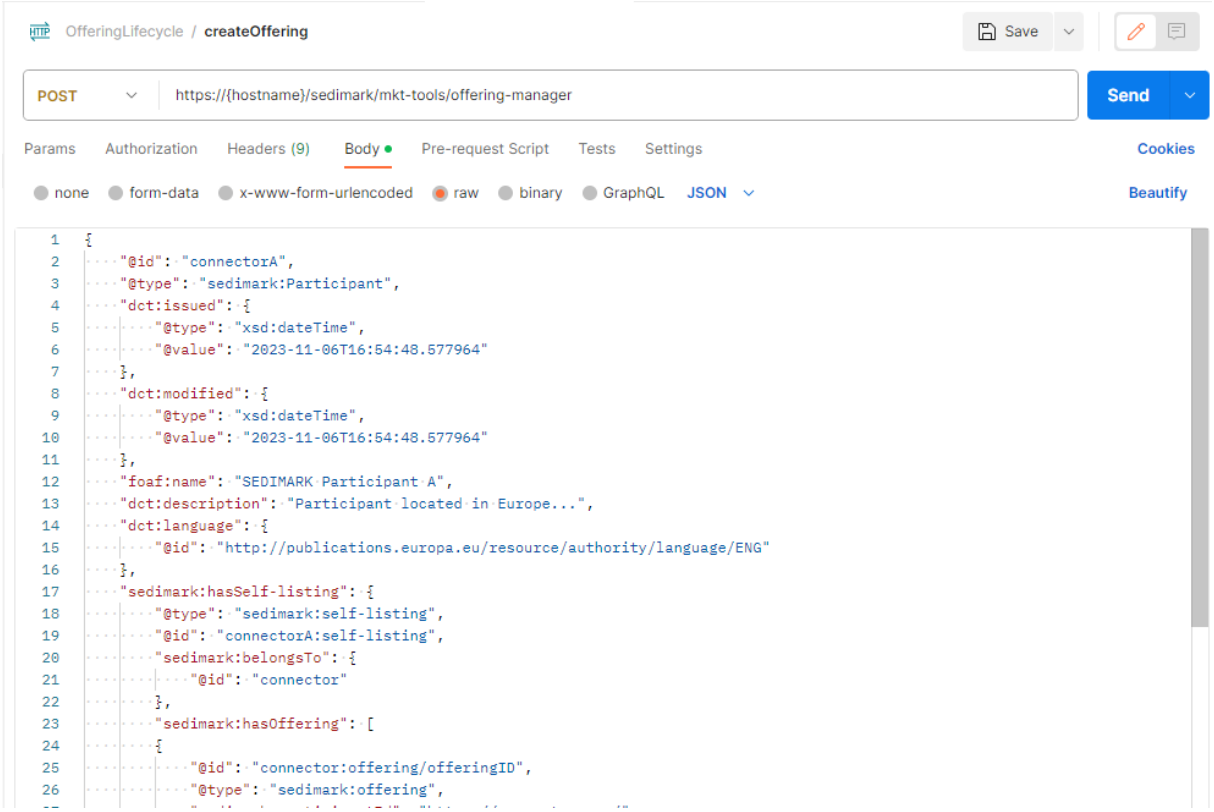


Figure 19 API Test Suite for Offering Lifecycle (Creation)

3.3.6 Guidelines for deployment and execution

Deployment and execution will rely on tools that include Maven for building and packaging, Docker and Docker Compose for containerised deployment, which will apply for the Offering Manager, Self-Listing and Catalogue. As for the Connector and Registry, sections on Data Sharing and Participant Onboarding provides details respectively.

3.3.7 Software licences

The source for the Offering Manager, Self-Listing and Local Catalogue is licenced under EUPL 1.2.

3.4 Asset (Data) exchange

3.4.1 High-level description

This scenario plays a central role in advancing the capabilities of the data space-related functionalities. This concept revolves around creating a marketplace where different assets can be bought, sold, and exchanged among various participants. This bridges the gap between providers and consumers, fosters collaboration, and enables efficient access to a wide range of assets.

The scenario encompasses two complementary planes:

- The control plane ensures the smooth governance of data transactions, with participants able to agree on access controls, pricing models, and licensing terms to protect their interests and maintain data privacy.
- The data plane, on the other hand, underpins the technical infrastructure that enables the exchange of assets. It employs different technologies like secure APIs, data streaming, and encryption to facilitate the seamless flow of information from provider to consumer and to ensure that data is exchanged efficiently, securely, and in compliance with regulatory frameworks.

Figure 20 provides a visual representation of the key components involved in this specific scenario. It illustrates how these components, as described in the system view of our architecture, interact and contribute to the overall functionality of the system. This diagram serves as a guide to understand the practical implementation of the scenario, thus it is referenced across different subsections.

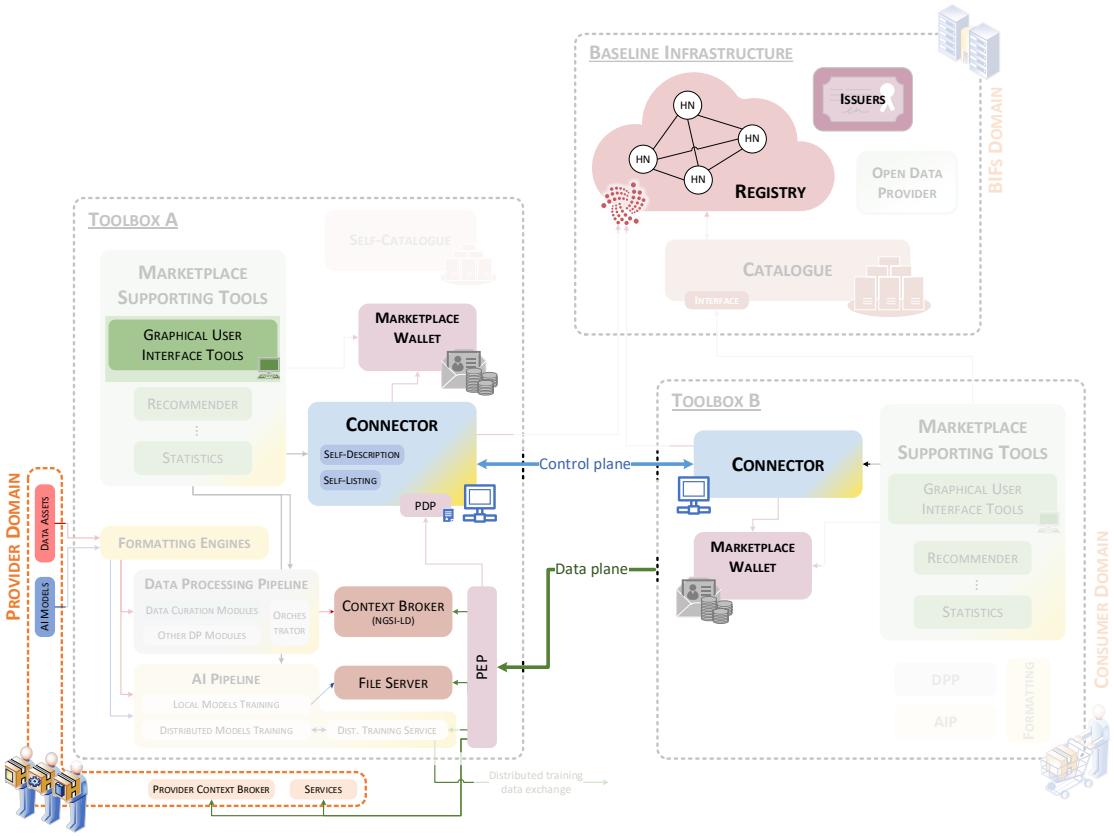


Figure 20 Detail of components involved in asset exchange scenario

3.4.2 Step-by-step definition and data flows

The step-by-step definition of the scenario is as follows:

- A consumer decides to acquire a previously discovered offering (e.g. clicks on the respective button in the corresponding GUI Tool).
- Consumer's connector (ConnB), using the participant identity credentials, queries the provider's connector for the offeringits counterpart for a specific offering.

- Provider's connector (ConnA) answers with the offering details, including information such as cost and access restriction policies.
- ConnA and ConnB negotiate the offering and, once a mutual consensus is reached, proceed to sign an agreement.
- Connectors interact with a smart contract on the Registry (L2 ISC). This results in an agreement, which is represented by a Fungible Token (FT) owned by the consumer.
- Access policies negotiated during step 4 are included as claims or assertions on a VC. This is the basis for further authorization together with the FT ownership (ABAC model), which will be further developed in future iterations.
- ConnB receives the needed information to access all assets described in the offering and redirects it to the consumer backend for further usage. Besides, credentials are stored on the Marketplace Wallet.
- Consumer requests the assets from its own domain. During the first iteration the ConnA is always used as proxy into the consumer domain. This could change in subsequent versions.
- Access policies are enforced on the PEP (ConnA) based on the credentials + VC assertions. Interactions with Registry and Issuers/Validators are needed to verify all presented claims.
- Upon successful validation, asset is provided to the consumer from the back-end NGSI-LD Context Broker (or access is denied).
- Consumer proceed with downloading the asset.

3.4.3 Specifications of involved components

As depicted in Figure 19, the main components involved are the following ones:

- **SEDIMARK Connector:** The SEDIMARK Connector is the main actor of the asset exchange scenario within the SEDIMARK marketplace. It plays a crucial role in managing offerings exposed by providers, facilitating secure and efficient exchange of the underlying assets between participants. It reuses several concepts coming from GAIA-X and IDSA, and leverages Distributed Ledger Technology for providing trustworthiness to participant identities as well as to their exposed offerings. SEDIMARK Connector builds on top of the Eclipse Dataspace Components (EDC), a framework that supports sovereign, inter-organizational data sharing. The EDC implements the IDS Dataspace Protocol (DSP) and relevant protocols associated with GAIA-X. By leveraging the robust and extensible framework of the EDC, the SEDIMARK Connector contributes to the overall goal of creating a secure and decentralized data marketplace.
- **Registry:** The Registry is a critical component in the architecture that leverages a set of IOTA Hornet and Wasp nodes to support Distributed Ledger Technology (DLT) and smart contract functionalities. On the one hand, IOTA Tangle is a Distributed Ledger Technology (DLT) that utilizes a direct acyclic graph (DAG) structure for the ledger. The Tangle network comprises Hornet nodes that use peer-to-peer communication to propagate messages and transactions. On the other hand, Wasp is a node software developed by the IOTA Foundation to run the IOTA Smart Contract Protocol (ISCP) on top of the IOTA Tangle. Smart contracts extend the functionality of DLT from storing transactions to

"performing computations". Thus, it is possible to run code reflecting contractual arrangements between parties that are resilient, tamper-resistant, and autonomous. In a nutshell, this component is key for enabling trustworthiness and accountability to the asset exchange phase, providing a decentralized and secure method for recording transactions and ensuring transparency and trust within the system.

- **Issuers / Verifiers:** Issuers and verifiers are fundamental components in a self-sovereign identity ecosystem. In particular, in this scenario Issuers are responsible for signing the VC after an agreement is reached between the provider and the consumer. Then, during the asset exchange stage, verifiers support the validation of the presented VC.

Besides the aforementioned components, which form the core of this scenario, there are other auxiliary components which have been used to test it. In particular, all exchanged assets are stored by the provider on an NGSI-LD based context broker. Besides, a specific marketplace wallet has not yet been implemented during the initial iteration, so credentials are currently kept in memory.

3.4.4 Integration specification

The integration of the components from this scenario is carried out following a service-oriented approach. Interactions between participants' Connectors follows the Dataspace protocol. However, the state machine is slightly modified to expand it and provide additional functionalities relying on SEDIMARK Trust Layer in order to enhance the trustworthiness of the operation. In this sense, SEDIMARK Connector interacts with the Registry through the DLT Enabler interface once an agreement over an offering has been reached off-chain.

Moreover, this scenario interacts with the following ones:

- Participants onboarding scenario, as DID are needed to establish a trust layer through secure connections
- Offering Discovery scenario, which is part of the offering lifecycle, is a precondition, as the required offering ID is a mandatory input parameter
- Marketplace GUIs: Participants will be able to use the connector via the Marketplace GUIs to publish an asset, contract an offering and download contracted offerings.
- The availability of the "asset retrieval endpoints" defined during the offering registration is a must, as otherwise, the asset procurement stage will fail when trying to retrieve them. This means the scenario might interact with the results derived from the Data Quality Improvement scenario and the results or services provided by AI-related scenarios.

3.4.5 Results

The implementation of this proof-of-concept scenario is instrumental in demonstrating the potential and effectiveness of the envisioned solution for asset exchange. Even though this is only the first iteration, the scenario showcases a platform where participants can negotiate access to various offers. This negotiation process is not only secure, ensuring the protection of participant data and transactions, but also decentralized, promoting transparency and trust among the participants.

The deployment of this scenario has allowed us to test and validate the key components of our architecture in a practical context, which is not very different from a real-world one. Furthermore, the results of this proof of concept have provided valuable insights into the

practical implementation of our architectural design. They have allowed us to identify areas of improvement and potential challenges that need to be addressed as we continue to develop and refine our system.

Figure 21 provides a visual showcase of the SEDIMARK connector in action during the scenario testing. However, it's important to note that the majority of this scenario involves automated interactions that occur behind the scenes. All these processes are not easily represented in a visual format. The completion of these flows essentially results in the acquisition of the data asset by the consumer, a process that, despite its complexity, is streamlined and efficient within the SEDIMARK system.

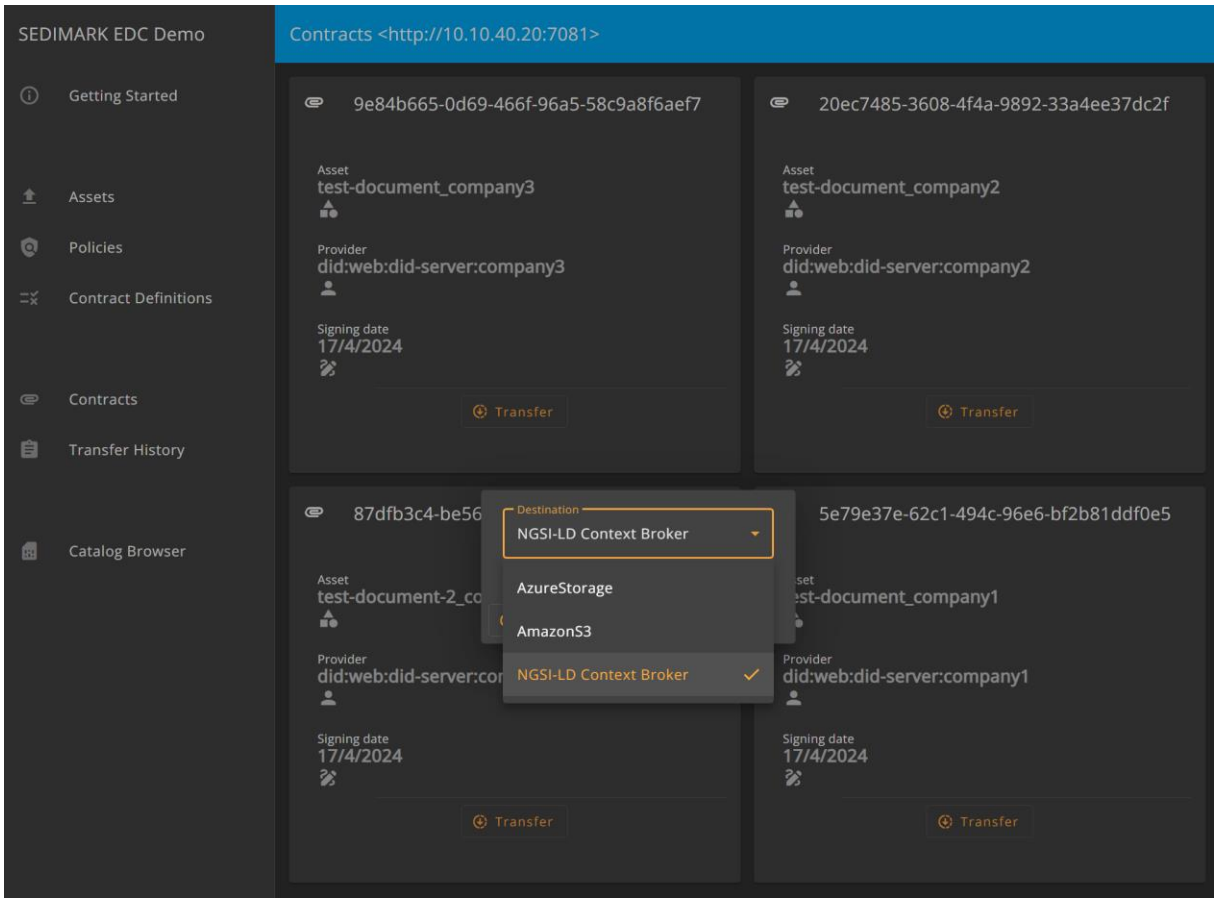


Figure 21 SEDIMARK connector during scenario execution

3.4.6 Guidelines for deployment and execution

In this scenario, all components are designed as microservices, a design approach that structures an application as a collection of loosely coupled services. This design methodology allows for the development of highly maintainable and scalable applications, which are organized around business capabilities. Each of these microservices is encapsulated within its own container, providing an isolated environment in which they can run, irrespective of the host system.

The orchestration of these containers is managed using Docker Compose, a tool that allows for the definition and management of multi-container Docker applications. Docker Compose uses a YAML file to specify the services, networks, and volumes of an application, allowing for

the efficient management of containers. This orchestration tool simplifies the deployment process, ensuring consistency across different environments, and allows for the scaling of services when necessary.

These containers are then deployed across different virtual machines. Virtual machines provide an additional layer of abstraction from the physical hardware, allowing for the efficient use of resources and improved scalability and flexibility. This deployment strategy allows for the distribution of resources, improving the reliability and availability of the system.

The SEDIMARK connector, a key component in this scenario, is built using Gradle [16] language. Gradle is a powerful build automation tool that is used to compile, test, and deploy code. It uses a declarative language to describe the project configuration, making it easy to understand and maintain. The use of Gradle in the construction of the SEDIMARK connector ensures a streamlined build process and allows for the efficient management of dependencies.

3.4.7 Software licences

SEDIMARK Connector: Apache License 2.0 / GPLv3 (to be decided)

IOTA Hornet & Wasp: Apache License 2.0

Issuers / Verifiers: TBD

3.5 AI-related scenarios

3.5.1 High-level description

This scenario provides the first integration of the AI components developed within SEDIMARK, with the main goal of realising a distributed training scenario, showing the way the AI modules work together and exchange information and data in a seamless way. At a high level, the scenario revolves around the distributed training of a model aimed at predicting electricity consumption. Leveraging data from the Energy use case from MYT pilot case, we're tasked with preserving confidentiality while extracting valuable insights. Distributed training emerges as the optimal solution, allowing data to remain local and eliminating the need for data exchange. Towards building the common distributed model only model parameters are exchanged and the model training takes place locally. It's important to note that this scenario operates as a standalone AI scenario, exclusively involving components from the AI Enabler. Currently, there's no integration with components from other Functional Enablers. However, this focused effort underscores our commitment to advancing AI capabilities within the SEDIMARK framework, laying the foundation for future collaborations and innovations.

Figure 22 offers a graphical depiction of the fundamental components engaged in this particular scenario. It delineates the interaction and contribution of these modules, as outlined in the architectural system view, towards the system’s comprehensive functionality. This visual aid serves as a roadmap for comprehending the practical execution of the scenario.

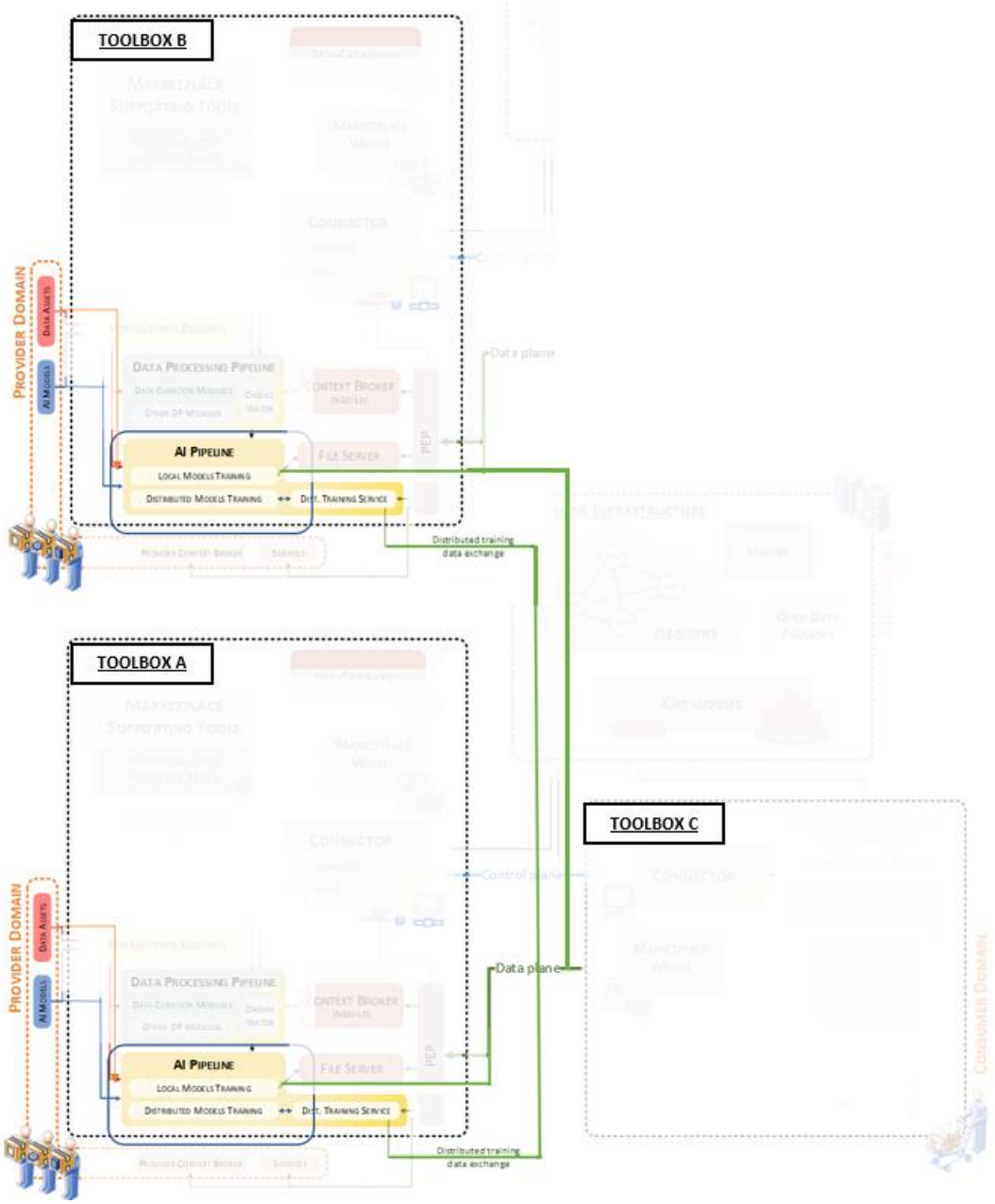


Figure 22 AI related scenario component interaction

3.5.2 Step-by-step definition and data flows

This scenario is a modification of the scenario described in SEDIMARK_D5.1, section 5.2.5.2 and more specifically the model-shared distributed model training scenario. It provides the provider-initiated process for distributed model training, only including through interaction with components from the AI Enabler. Here are the sequential steps of the process:

- Provider possesses some data and aims to construct a model on it.
- Provider initiates the distributed training process and starts building a local model based on model configuration and training parameters.

- Providers local distributed model training module (sha mrock.AI) starts and creates a server listening for connections from other clients.
- Provider directly communicates the model description to interested participants, inviting them to join the training process (the modification here compared to the process described in D5.2 is necessary since there is currently no integration with the Offering Enabler).
- Other interested participants possess locally similar datasets and are ready to engage in the distributed training process.
- Each interested participant initiates their distributed training module, assuming the role of another gossip node. They initialize their model, set up a local server, and establish a connection with the provider's server to facilitate the exchange of weights.
- The training process unfolds through iterative rounds, characterized by local training iterations, weight exchange, and weight aggregation. Participants engage in these cycles until either the model converges, or a predefined maximum number of iterations is reached.

The data flow for the scenario is a modified version of the data flow presented in SEDIMARK_D2.2 and can be seen in the following Figure 23:

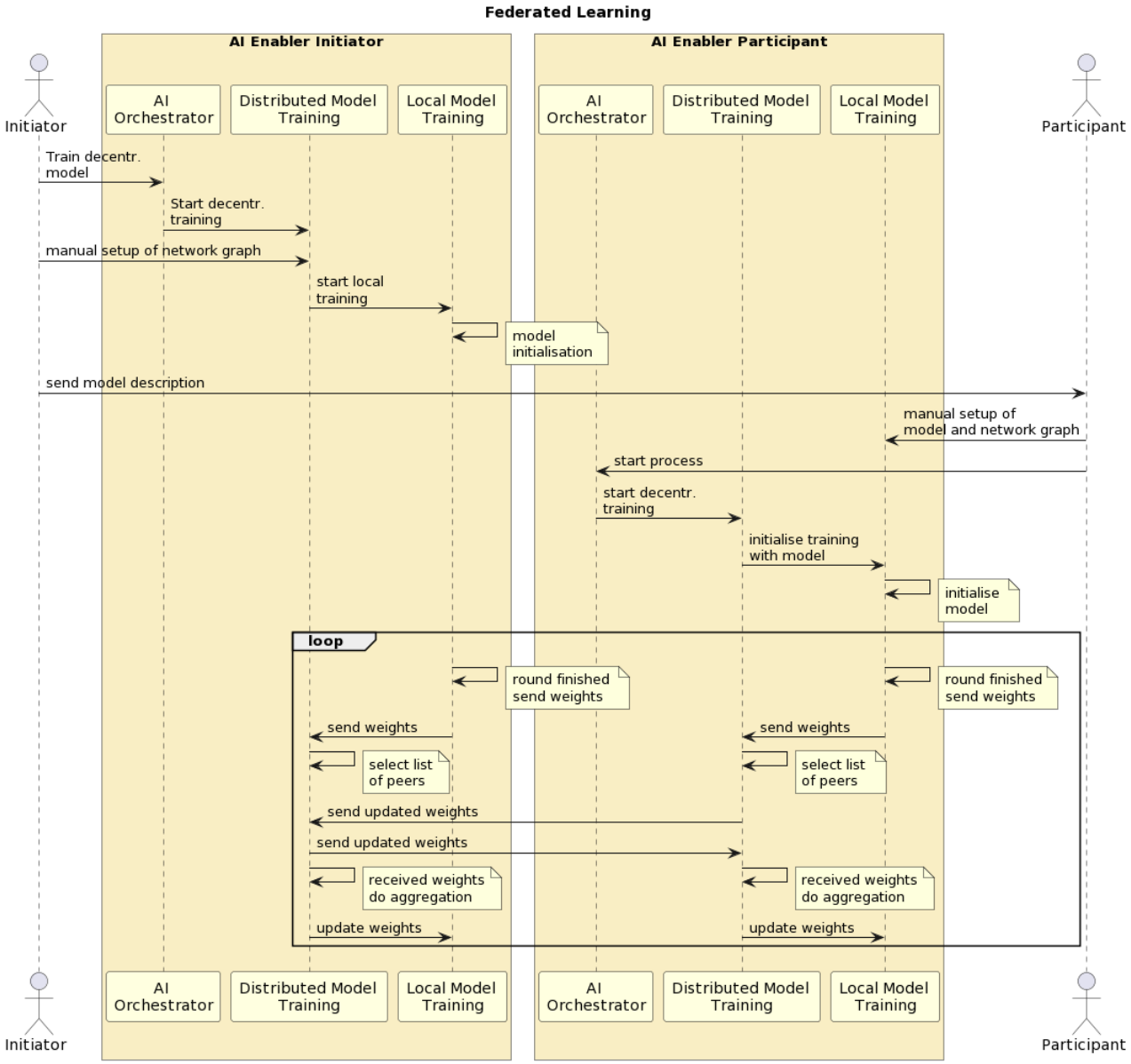


Figure 23 AI-related scenario data flows

3.5.3 Specifications of involved components

- **AI orchestrator:** Serving as the cornerstone of the AI Enabler, the AI Orchestrator assumes responsibility for governing the functionality and cooperation among other components within the ecosystem. Implemented as a Python module, it orchestrates various processes, including model initialization, parameter exchange, and aggregation. This critical component ensures seamless coordination and synchronization of activities, driving efficiency and effectiveness.
- **Local model training:** This is the component that performs the actual training of the AI models based on local data. In the context of the problem of energy consumption prediction, we chose to combine a set of models based on decision trees, namely XGBoost, LightGBM, CatBoost and Random Forest. The training process of our model

consists of the training stage and hyperparameter finetuning stage. We aimed to include the general form of the distribution of actual consumer data. This component facilitates the customization and adaptation of models to specific datasets, enabling localized insights and optimizations tailored to unique requirements.

- **Distributed model training:** This is the component that manages the distributed training of the AI models in SEDIMARK. It is the Shamrock-AI component built-in Python based on the Flower library. Through decentralized training processes, Shamrock-AI empowers stakeholders to leverage diverse datasets while preserving data privacy and security.
- **Results component:** A novel addition to the SEDIMARK architecture, the Results Component addresses the need for comprehensive monitoring and evaluation of model training outcomes. Designed as a trial-specific component, it captures and aggregates results generated during the model training process. Implemented as a Python module, this component saves results in a CSV format, facilitating further analysis and interpretation. By providing insights into model performance and efficacy, the Results Component enhances decision-making and optimization efforts within the AI ecosystem.

3.5.4 Integration specification

The integration involved the interconnection between the three components discussed above. The AI Orchestrator is implemented as a standalone Python module that is part of each distributed node and among others keeps track (at each node) of the network graph and gets the local model from the local training module. Thus, it has currently been implemented as the interface between the local model training module and the distributed training module. The interconnection between the peer distributed modules (of the various participants) is implemented using gRPC, with procedures for registering and requesting model updates. This interface is the one that is used for exchanging the weights of the model between the participants at each distributed training round.

3.5.5 Results

Centralized Approach

1) Decision Tree-XGBoost

The performance of our model is based on the evaluation metrics root mean squared error (RMSE), mean absolute error (MAE) and mean absolute percentage error (MAPE). After computing these errors, we conclude that the best model based on these metrics is the XGBoost Regressor. In the computation of the MSE on the test set we removed the outliers in order to have more intuitive results. The scores are the following:

MAE = 3.6 kWh

MSE = 17.16

To achieve the results, we went through an extensive hyperparameter fine-tuning process. This involved meticulously identifying the most optimal parameters for each model tailored to our specific problem and dataset. The final hyperparameters of the model as well as the weights extracted are the following:

Number of Estimators: 3000

Learning Rate: 0.01

Max Depth: 5

The below Figure 24 shows the comparison between the model's energy consumption predictions and the actual measured values across different instances. This plot indicates a good fit by the model to the ground truth data indicating that the model is generally successful in capturing the trend of the data.

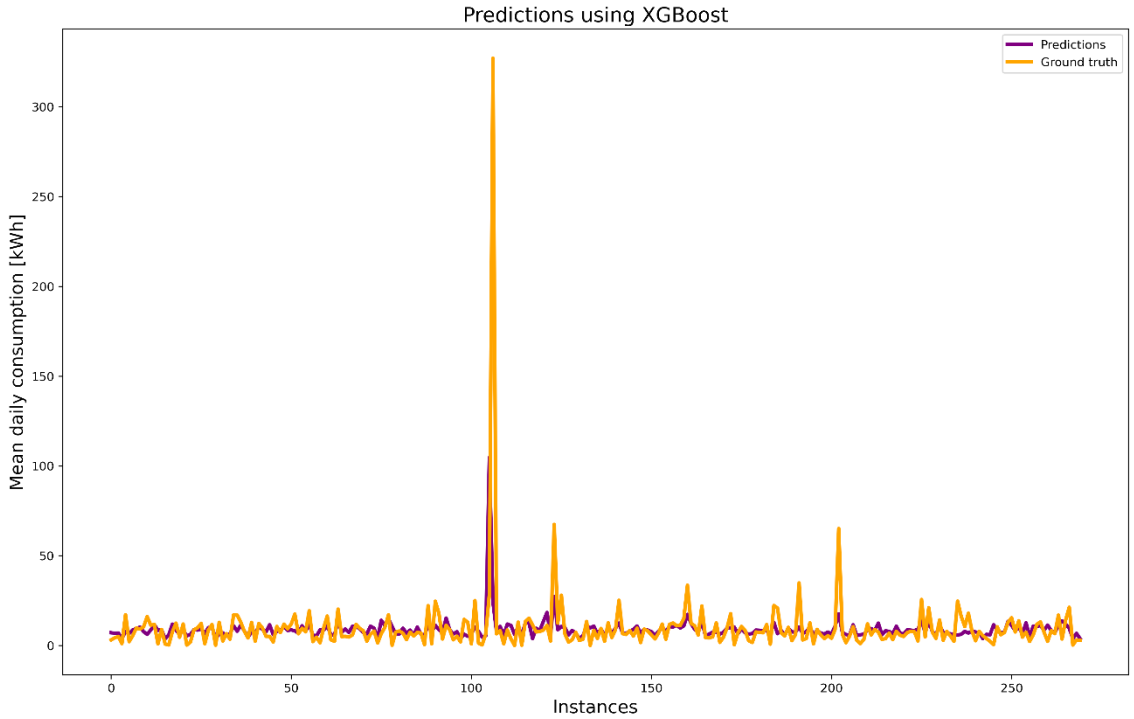


Figure 24 Predictions using XGBoost

The scatter plot below (Figure 25) displays the correlation between the ground truth and the predictions made by the XGBoost model. The dotted red line represents the y=x reference, where a perfect prediction would lie. The spread of points around this line indicates the model's prediction accuracy.

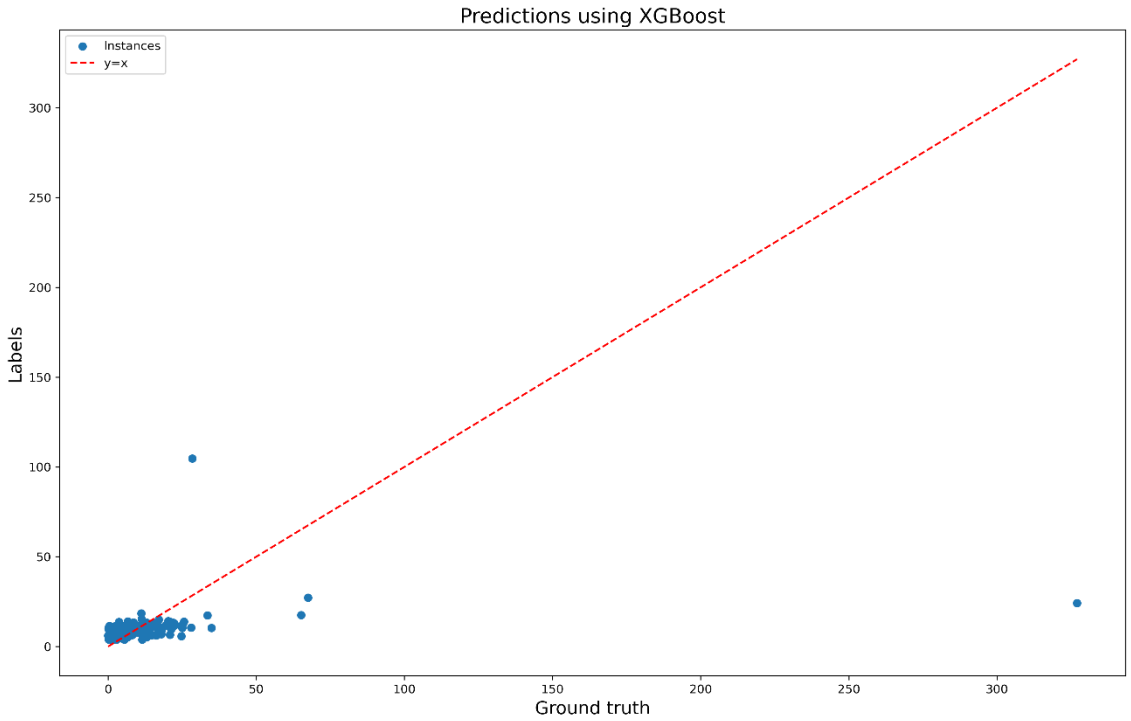


Figure 25 Scatter plot using XGBoost

In conclusion, the line plot demonstrates correspondence between the model's predictions and the actual data, suggesting that the XGBoost model captures the underlying patterns in energy consumption effectively. The scatter plot further supports this, with many data points clustering around y=x.

2) Neural Network

Again, we went through an extensive hyperparameter fine-tuning process. This involved meticulously identifying the optimal parameters for each model tailored to our specific problem and dataset. Fine-tuning was performed during the process of training the models.

After hyperparameter tuning, we concluded that the best model is a model that consists of a series of densely connected layers arranged in Sequential order. More specifically:

- Three Dense layers with 256,128 and 32 neurons respectively.
- The first 3 layers use ReLU activation functions. ReLU (Rectified linear unit) activation function is chosen for its simplicity and efficiency in training, allowing for faster convergence.
- The final layer is a Dense layer with a single neuron and a sigmoid activation function, suitable for regression tasks where the output is a continuous function. The sigmoid function is suitable for regression tasks, in this case, daily energy consumption.
- Two Dropout layers with a rate of 0.1 applied after the first and the second dense layers to prevent overfitting and generalize better in unseen data.
- Adam Optimizer suitable for neural network training due to its adaptive learning rate and momentum.

In the computation of the MSE on the test set we removed the outliers in order to have more intuitive results. The scores are the following:

MAE = 3.08 kWh

MSE= 27.86

This line plot in Figure 26 compares the predicted daily energy consumption against the actual consumption. The lines follow each other closely, suggesting that the model has learned to predict consumption patterns accurately. There are occasional spikes in both prediction and ground truth lines, which are not fully captured by the model. Overall, the neural network's predictions are in good alignment with the actual values indicating its effectiveness.

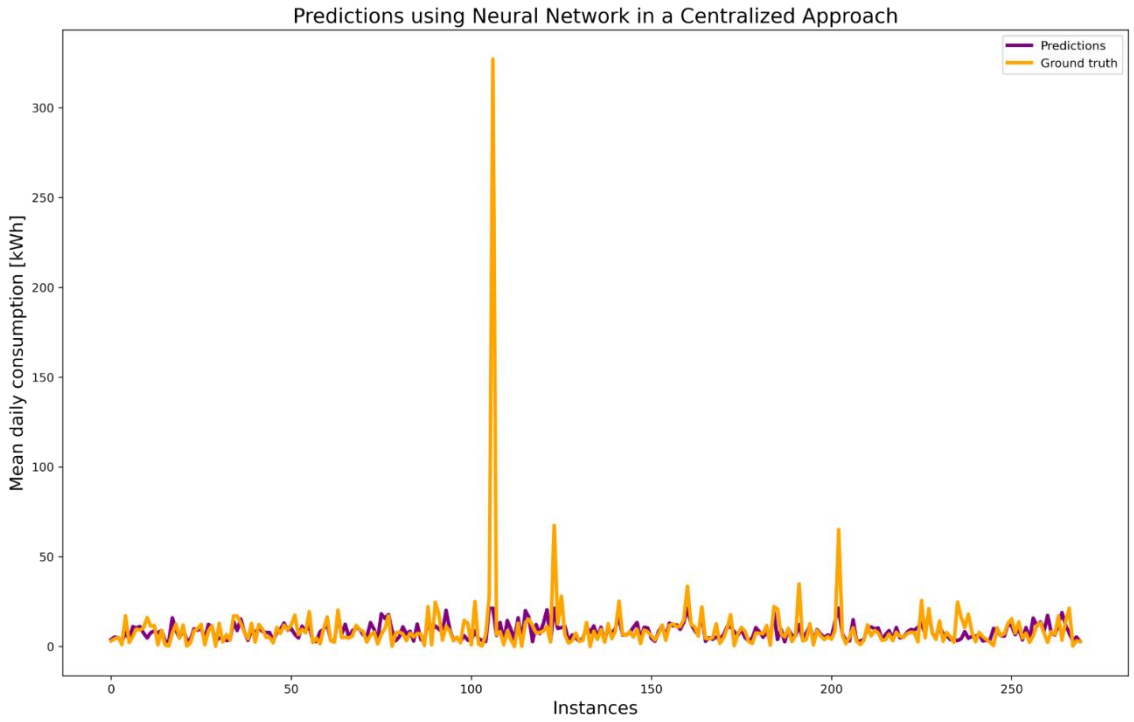


Figure 26 Predictions using Neural Network

The scatter plot in Figure 27 depicts individual predictions plotted against the actual values. Most points cluster near the line y=x in the lower range, indicating high accuracy for lower values.

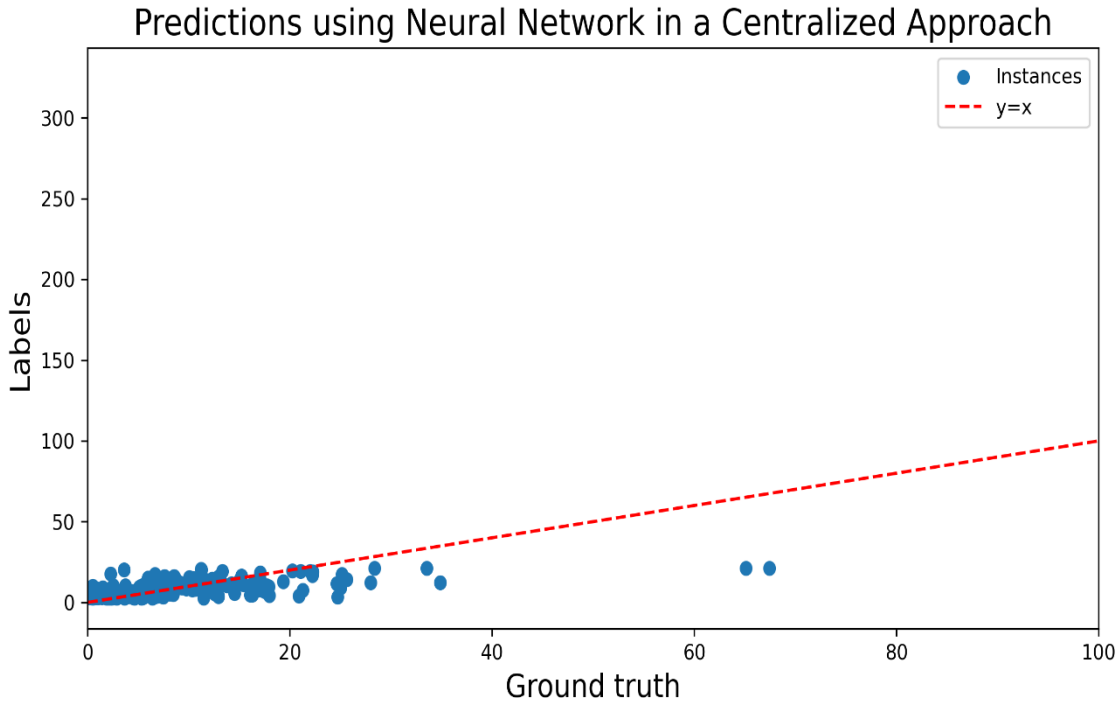


Figure 27 Scatter plot using Neural Network

Insights:

The XGBoost model outperforms the neural network significantly in all metrics. The neural network shows a promising approach with its ability to learn complex patterns, the XGBoost model's balance of accuracy, interpretability and robustness against outliers make it a more effective choice for predicting energy consumption within this dataset. The analysis emphasizes the importance of selecting the appropriate modeling approach based on the specific characteristics and challenges of the dataset

Decentralized Approach

In a decentralized machine learning framework, such as federated learning, data is distributed across multiple nodes or devices, and instead of centralizing the data, the model is trained locally on each node. Federated learning, particularly with neural networks, offers the flexibility and scalability necessary to handle complex data patterns effectively, potentially boosting model performance and applicability across varied scenarios.

In our setup, we have federated learning with two machines. Each machine acts as a server and client at the same time. Each machine performs local training for one epoch. Following the completion of local training, each machine initiates bidirectional communication to exchange model weights with its counterparts. After each local epoch, the weights in both nodes are aggregated by averaging the individual weights. Afterwards, each machine possesses an identical set of updated weights. The cycle of training, weight transmission and aggregation is repeated for 100 iterations.

Results:

The performance of the model is assessed on each machine's (Node 0/1) validation subset and it is measured by both Mean Absolute Error (MAE) and Mean Absolute Squared Error

(MSE). Given that these subsets are of equal size, the overall performance is determined by averaging the individual metrics.

In Figure 28, the MSE loss shows a sharp initial decline in the first few iterations, indicating the model's quick learning from data. After stabilization, the MSE loss for both nodes decreases, indicating converging performance. The model closely tracks each other on the dataset.

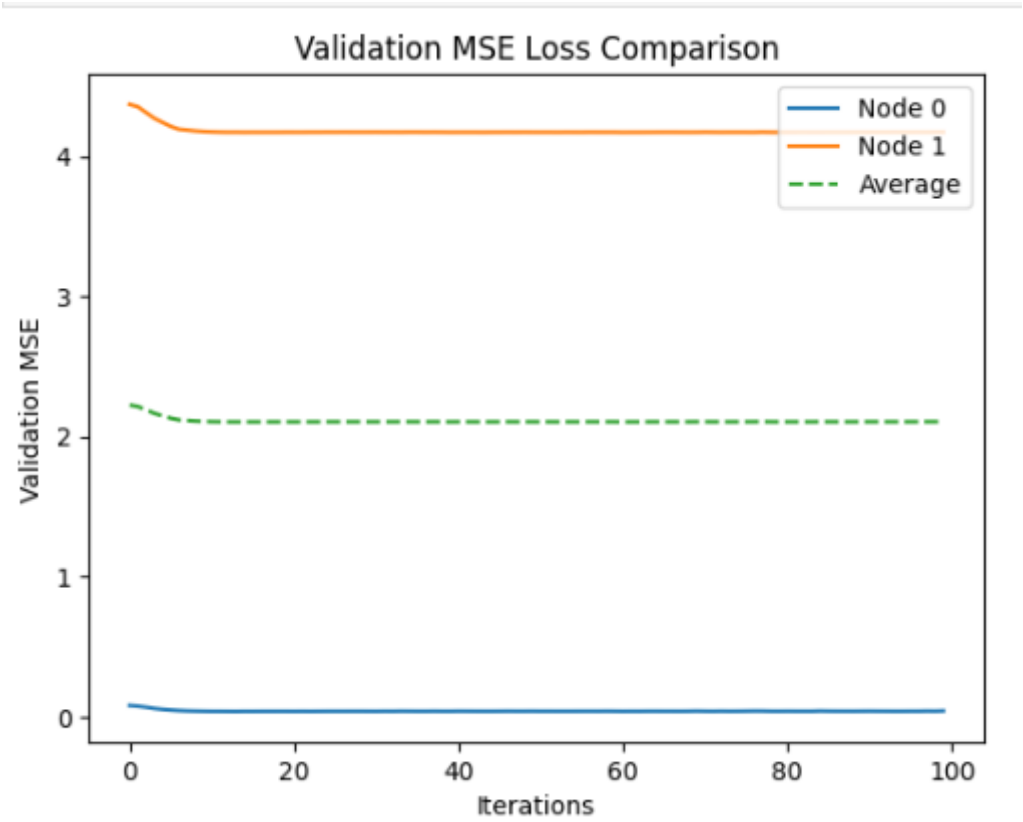


Figure 28 MSE validation loss over iterations for the decentralized machine learning model trained on Node 0 and Node 1. Average loss is calculated for the global model.

Figure 29 shows a rapid learning phase with a sharp decline in error for both nodes. As iterations progress, MAE curves flatten, indicating convergence. Both nodes perform similarly across iterations, with lines aligned. The average MAE remains consistent, confirming similar performance levels.

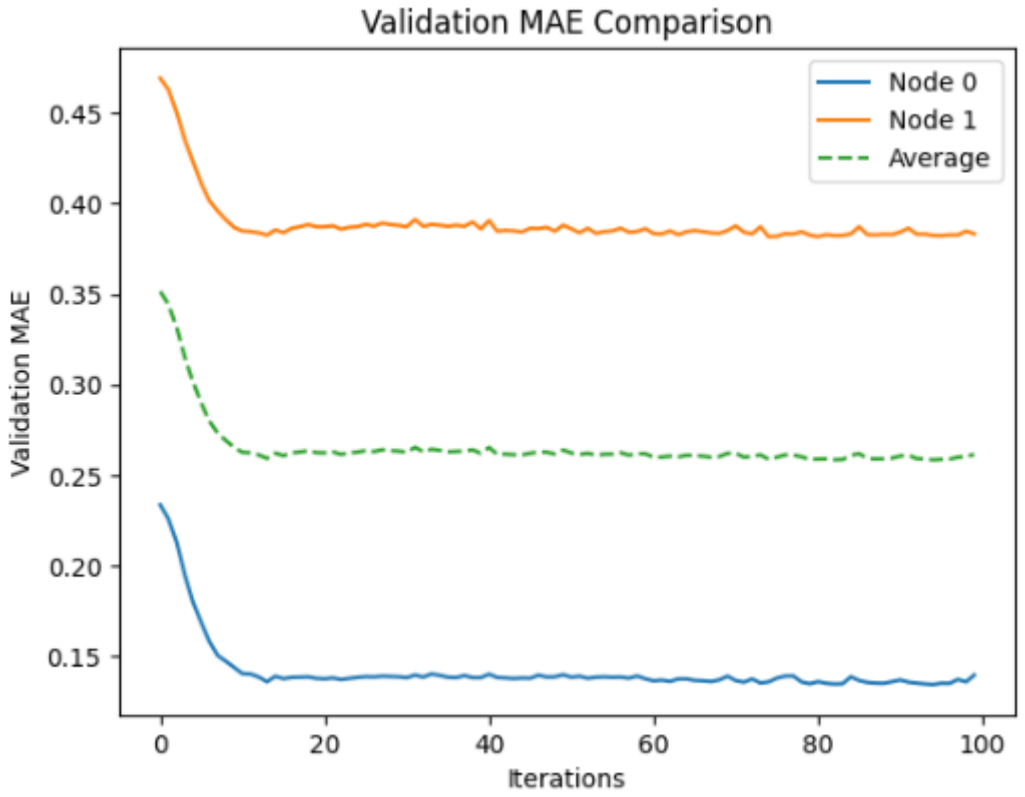


Figure 29 Validation MAE over iterations for the decentralized machine learning model trained on Node 0 and Node 1. Average loss is calculated for the global model.

The consistency between two nodes and the average suggests that both nodes contribute similarly to the federated learning process. The lowest MSE for the global model was recorded at 2.104 in the 60th iteration. This is the global model's highest accuracy point.

Predictions:

By monitoring the average validation MSE across iterations, we identify the iteration with the smallest value. In this iteration, the global model reaches the best performance and we use these weights to compute the predictions plots. The test set utilized for the plots, is acquired by combining the tests sets of the individual nodes.

In the computation of the MSE on the test set we removed the outliers in order to have more intuitive results.The scores are the following:

MAE = 2.95 kwh

MSE =26.26

The line plot in Figure 30 shows that the model accurately predicts daily energy consumption, closely tracking the ground truth. However, it underperforms in higher consumption ranges, indicating limitations in handling higher variance or less frequent data patterns.

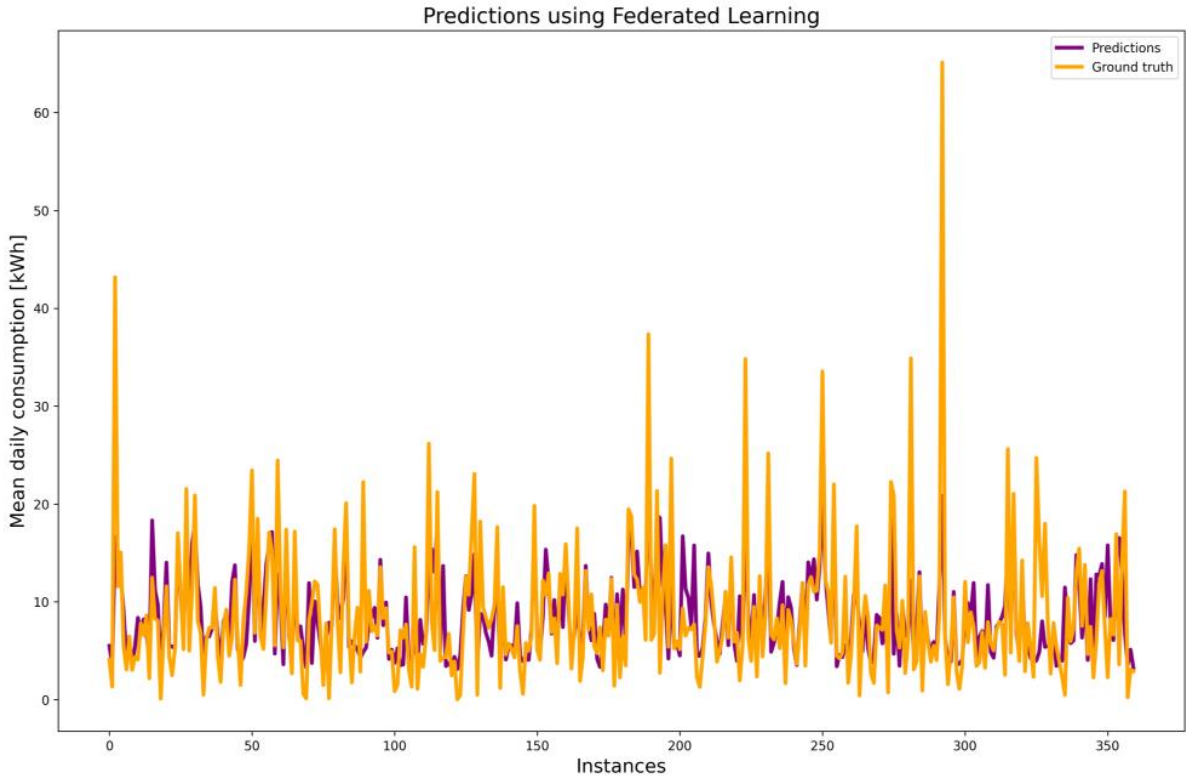


Figure 30 Predictions using Federated Learning on test set

The scatter plot in Figure 31 shows that the model's predictions are most accurate for lower energy consumption values, clustering near the line of y=x. However, there is notable deviation at higher values, suggesting the model may struggle with predicting higher energy consumption ranges.

The plots show that federated learning models on both nodes can predict energy consumption with reasonable accuracy, but are more reliable for average or common patterns. They can generalize across datasets but face challenges modeling the full range of consumption behaviors, especially for extreme cases.

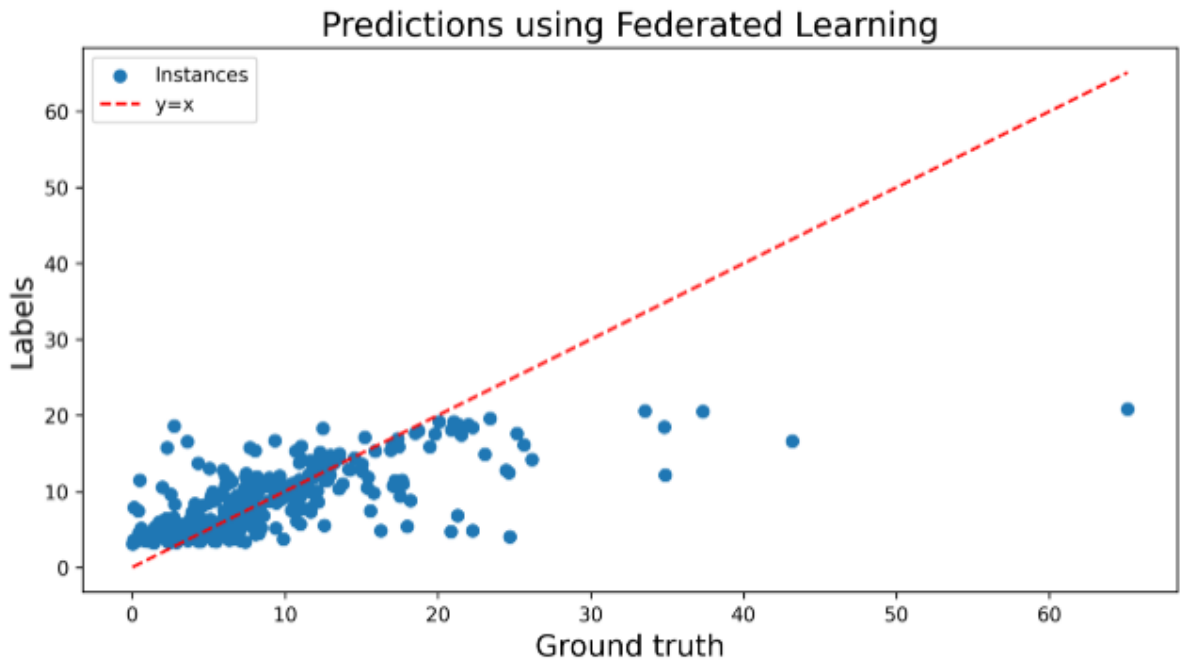


Figure 31 Scatter plot using Federated Learning on test set

The communication cost in federated learning is calculated using the formula:

$$Comm.cost = Size * Freq.of\ Comm * (No.machines - 1)$$

Where the size of the model affects the amount of data needed to be transmitted. The frequency of communication refers to how often model updates are exchanged between machines and the total number of machines involved in federated learning. In our experiment, with two machines and a model size of 44,481 parameters, the communication cost is 17.375 MB. This represents the data transfer overhead between the two machines over 100 iterations. The speed of the network between the machines is crucial, as bigger capacity networks can handle larger communication costs. This streamlined communication is essential for maintaining the learning process momentum and allowing real-time improvements, contributing to the practicality and scalability of our approach.

Centralized vs Decentralized Approach:

We evaluate the models based on accuracy, convergence and communication cost.

Centralized Neural Network vs Decentralized Model

Model accuracy:

The centralized neural network exhibits discrepancies in training and validation loss, particularly with high energy consumption. The decentralized model shows significant accuracy, while the federated learning approach maintains similar accuracy levels compared to centralized training.

Convergence:

For the centralized neural network, the convergence appeared to be stable, with validation and training losses staying close, aside from a late spike that may indicate a specific issue. The decentralized model showed a fast initial decrease that indicates the models on both nodes were converging effectively over the iterations. The plots suggested a similar performance.

Communication cost:

The centralized approach typically does not involve communication costs as seen in the decentralized systems since all computations are done within a single server. In the decentralized case, the total cost is small due to the small number of machines in the experiment and the small model used.

The centralized neural network offers simplicity and scalability, while the decentralized federated model offers robust accuracy, convergence, localized data processing, privacy, and security benefits. Both models have their limitations, making them suitable for scenarios where data cannot be centralized.

XGBoost vs Decentralized Model

Model accuracy:

XGBoost excels on tabular data, handling missing values with low error rates and accurate predictions, particularly for lower energy consumption values. The decentralized neural network aligns with actual data, while federated learning maintains a notable level of accuracy.

Convergence:

XGBoost's convergence pattern and decentralized neural network's rapid loss reduction suggest effective learning and convergence, similar to centralized models, promoting faster convergence and model optimization.

Communication cost:

XGBoost has no communication cost since the model training and predictions are contained within a single system. In the decentralized case, the total cost is small due to the small number of machines in the experiment and the small model used.

The XGBoost model is known for its accuracy and interpretability in a centralized context, while the decentralized neural network model offers similar accuracy and convergence efficiency with the added benefits of data privacy and security. Both models have their specific advantages, with the choice based on application requirements, computational resources, and dataset characteristics. The decentralized model demonstrates federated learning's ability to handle complex datasets while maintaining user privacy and data integrity

3.5.6 Guidelines for deployment and execution

The following guidelines should be followed for the distributed gossip-learning-based model training:

Installation of the shamrock software: this requires the installation of Anaconda, the creation of a conda virtual environment using Python 3.10 and the installation of the required libraries for running the code (i.e. flower, pandas, etc.).

Testing of the current installation using dummy data to identify potential issues/conflicts or missing libraries. This requires the execution of a trial Python file that does local training (i.e. ./run_local.sh) creating two different shamrock nodes as different processes on the same machine. The output should show a decrease in loss/mae, along with evidence of aggregation occurring at the Node output. No indication of aggregation would suggest a lack of communication between nodes, indicating a problem.

There is a dedicated Python file where the data input and data preprocessing methods are implemented/executed. The code requires the data to be inputted to shamrock in the form of ((X_train, Y_train), (X_test, Y_test)).

Executing the scenario on multiple machines using a LAN network requires currently a manual setup of the network graph, so at each machine, the user should set the local IP address and port and the target IP addresses/ports of the remote participants. Firewalls should be enabled to allow connection to those specified ports.

```
#remote machine IP address and port
 peers = ["ip_worker2:8083"]

 #local machine host and port
 addr  = "ip_worker1:8082"
```

Then on each worker-machine, the code is run by setting Torch as the Keras backend and executing the distributed training main Python file

Results get written to a.csv file in the results directory.
iteration is the number of iterations the node has run locally.

number_of_coms is the number of communication updates the node has received from the other node.

3.5.7 Software licences

Shamrock-AI: currently the module has a proprietary license, just for use within the SEDIMARK project and not allowing any other uses. It is being considered to be open-source in the next period, and NUID UCD is considering the various options for open-source libraries.

Local model training: currently the module developed by WINGS has a proprietary license, just for use within the SEDIMARK project and does not allow any other uses.

3.6 GUIs

3.6.1 High-level description

The Graphical User Interfaces (GUIs) scenario implements the marketplace of the SEDIMARK ecosystem, consisting of a web application enabling participants to go through most of the functionalities offered by SEDIMARK:

- Onboarding the ecosystem
- Authenticating and managing their accounts
- Browsing the catalogue of offerings
- Consuming offerings from other participants
- Receiving personalised recommendations about interesting offerings that are available for purchase
- Publishing an offering
- Managing ongoing contracts, as well as published or consumed offerings

- Access SEDIMARK orchestrators for AI and data processing

In this sense, this scenario, as depicted in Figure 32, cements together the participant onboarding (3.1), offering lifecycle (3.3) and asset data exchange (3.4) scenarios into a user-friendly application so participants can focus on business and data sharing without having to consider the implementation details of the platform.

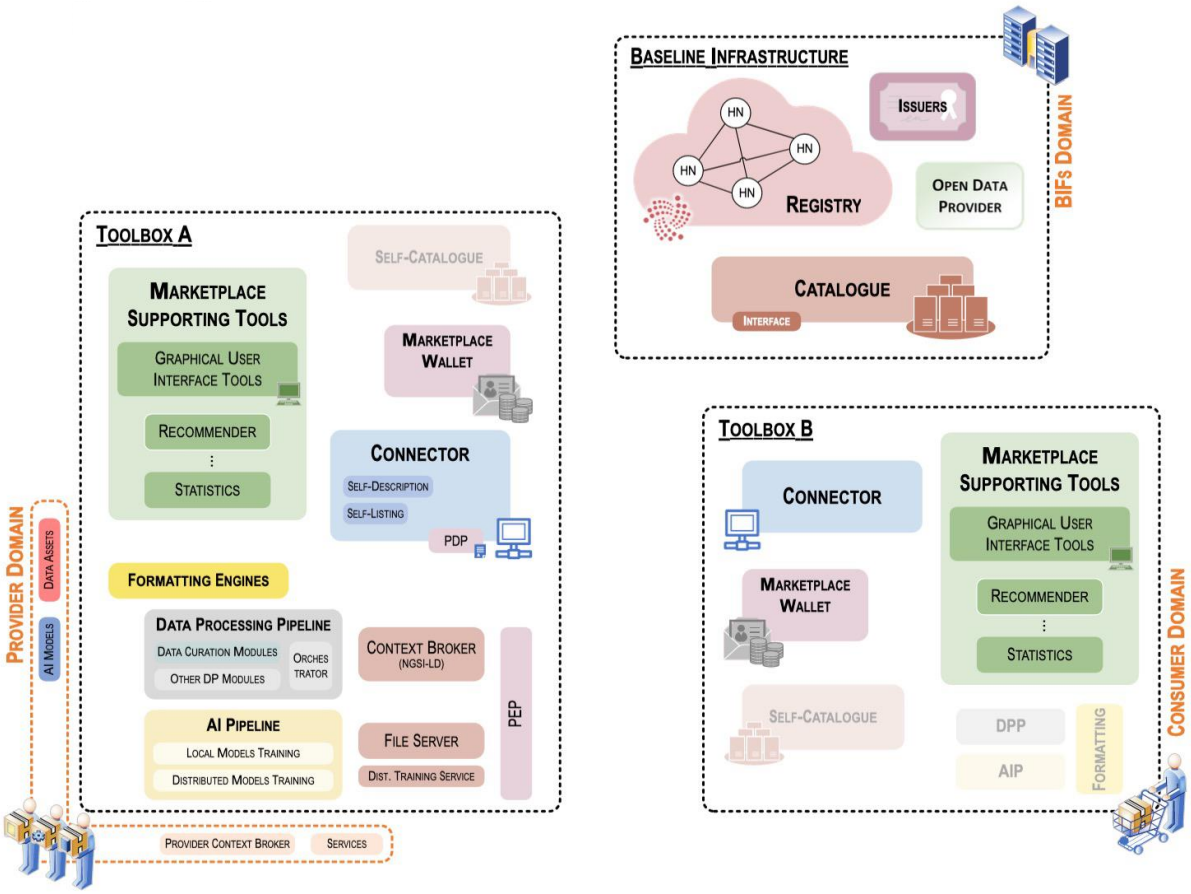


Figure 32 GUIs scenario components

3.6.2 Step-by-step definition and data flows

3.6.2.1 Onboarding

This scenario consists of the integration in the frontend of the participant onboarding procedure described in section 3.1. During this process, a new participant:

- Generates its Decentralized Identifier (DID).
- Receives her/his Verifiable Credential (VC), to be stored in her/his wallet.
- Creates an account and agrees to SEDIMARK terms of use (to be defined).

3.6.2.2 Authenticating and account management

This scenario ensures that participants can use the marketplace to:

- Log in to their account, enabling them to publish/consume offerings.

- Manage/edit their account data.

3.6.2.3 Browsing the catalogue

This scenario makes sure the marketplace offers an interface to browse the offerings catalogue, with the following features:

- Non-authenticated users can browse the catalogue, but can only see public offerings.
- Authenticated participants can see all offerings they are allowed to access.
- Sorting and filtering catalogue queries, based on offerings' provider, name, description, keyword, price, etc.
- Getting recommendations based on the users' interests and marketplace usage (previous queries, clicked offerings, etc.)

3.6.2.4 Consuming offerings

This scenario enables participants to select an offering from the catalogue and start negotiations with its provider to eventually purchase access to its corresponding dataset or service.

3.6.2.5 Publishing offerings

This scenario ensures the marketplace provides all the tools for users to easily create an offering wrapping a dataset or a service, and publish it to the catalogue.

3.6.2.6 Managing contracts, published and consumed offerings

This scenario consists of providing a dashboard in the marketplace frontend, where participants can monitor, sort and filter all the contracts they are involved in, either as a provider or as a consumer.

3.6.2.7 Accessing AI and data processing orchestrators

This scenario aims at providing a convenient way to access SEDIMARK's added value tools such as data cleaning/processing tools and AI pipelines directly from the marketplace frontend.

3.6.3 Specifications of involved components

The marketplace GUIs consist of a single containerized frontend web application, implemented using Javascript and React through NextJS 13 framework. The latter supports server-side rendering (SSR), which allows to fetch and process data on the server-side, and ship readily rendered HTML pages to the client's browser. This is beneficial to optimize the user experience in the marketplace since the federated catalogue of offerings needs to be fetched and built prior to be displayed to the user's browser.

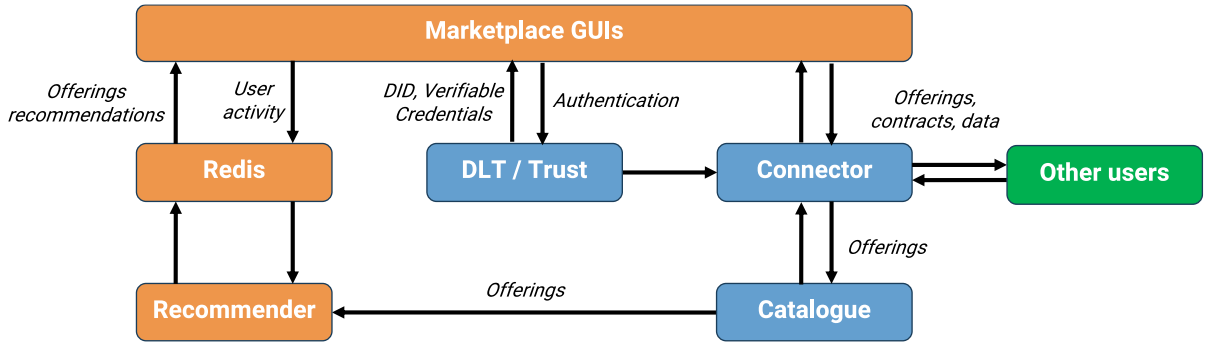


Figure 33 Marketplace architecture

To enable participants to get personalized recommendations of offerings when browsing the catalogue, the marketplace features two additional components:

- The recommender system: a containerized service outputting recommendations of offerings, to be displayed on the catalogue browsing page, depending on the participant ' s activity in the marketplace (previous catalogue queries, selected offerings, etc.)
- A message queue (Redis [17]): acting as a buffer between the marketplace and the recommender system. The former publishes the user's browsing activity in a queue, so the latter can use this information to tailor its offering recommendations and publish them in another queue, where the marketplace can get them.

The marketplace components are illustrated in orange in Figure 33. Figure 33

3.6.4 Integration specification

As the gateway to access most of SEDIMARK's functionalities, the marketplace interacts, directly or indirectly, with a large set of components of SEDIMARK. This large number of dependencies can make its integration particularly challenging. More specifically, the marketplace needs:

- DLT enabler / trust components: for the participant onboarding and authentication.
- Connector / asset data exchange:
  - To fetch the list of offerings from the catalogue for prospective consumers.
  - To enable providers to publish new offerings or update existing ones in the catalogue.
  - To manage contract negotiations.

To transfer data between providers and consumers.

- Catalogue and offering lifecycle services:

For the recommender to identify potential offerings to recommend.

- AI and data processing orchestrators:

To enable users to process their data, consumed or to be provided, directly via the marketplace GUIs.

These components will be integrated into the marketplace as they get operational. For the first version of SEDIMARK, a proof-of-concept version of the marketplace will be built using a minimal working version of these components. In the meanwhile, the marketplace is being

implemented by mocking the missing services, following closely the designed mock interfaces, which have been described in deliverable SEDIMARK_D4.5.

The microservice architecture of SEDIMARK, described in detail in deliverable SEDIMARK_D2.2, will facilitate the integration of all backend components with the marketplace GUIs, separating each component into a subset of containerized applications.

3.6.5 Results

The first version of the SEDIMARK marketplace GUIs will cover the following use cases:

- Onboarding new participants by creating a DID, issuing a VC and creating an associated account.
- Browsing the catalogue of offerings with no access restriction policies, i.e. all participants in the ecosystem can see offerings from all other participants.
- Getting offering recommendations based on user behaviour.
- Publishing and consuming free dataset offerings (not services such as computing or storage).
- Monitoring of past and present transactions via a dashboard.

The final version of the SEDIMARK platform will extend these features to cover:

- The publishing and consuming of offerings for services.
- Policies to rule offerings access when browsing the catalogue or when consuming an offering.
- Getting statistics about SEDIMARK platform usage.
- Integrating with the AI and data processing toolboxes.

A more detailed description of the marketplace GUIs can be found in deliverable SEDIMARK_D4.5.

3.6.6 Guidelines for deployment and execution

The marketplace GUIs, as well as its recommender system, are all containerized microservices. Each of these services will have its associated docker images, and instructions will be provided to either run them locally, using docker-compose or deploy them in a Kubernetes cluster, via a set of provided deployment manifests.

3.6.7 Software licences

The marketplace frontend will be licensed under the Apache 2.0 license.

The recommender system is currently under a proprietary license. UCD is considering the various open-source license options and will release the software as open-source in the next phase of the project.

3.7 Open data enabler

3.7.1 High-level description

This scenario populates the SEDIMARK catalogue with offerings granting access to participants to open data portals. These offerings won't feature any access restriction policy, and their licensing will allow for any data usage, with the sole demand of respecting attributions if relevant. This scenario therefore plays a key role to:

- Populating the catalogue: to accelerate and foster data exchange in SEDIMARK without solely relying on the activity of the first participants of the platform.
- Help participants get familiar with the SEDIMARK ecosystem: by ensuring the catalogue they can access in the marketplace contains offerings they can purchase for free, allowing them to test offering consumption, monitoring and data transfer before actually interacting with other participants.

Provide support for the documentation of the SEDIMARK platform: the open data enabler offerings can be used to create documentation materials such as tutorials showcasing how to publish or consume offerings and monitor contracts.

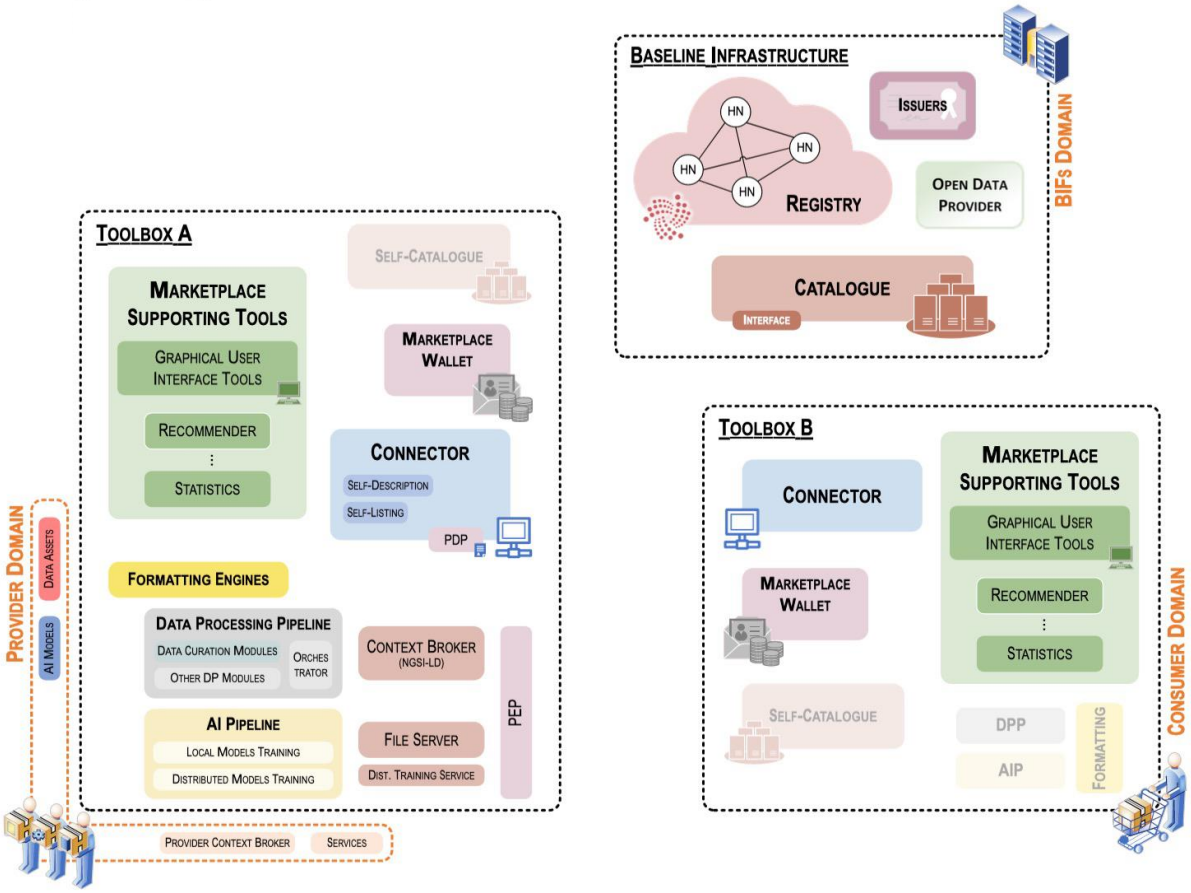


Figure 34 Open data scenario components

3.7.2 Step-by-step definition and data flows

Strictly speaking, the open data enabler is not a software component. It consists of a participant in the SEDIMARK ecosystem, who acts purely as an offering provider. Consequently, its integration is done by:

- Creating a SEDIMARK participant: having its own identity within the ecosystem, hosted in its own premises.
- Provisioning this participant with a minimal version of the SEDIMARK platform: since its sole role is to provide offerings corresponding to open data portals, it only needs the core data space components of SEDIMARK.
- Selecting open datasets to provide in the catalogue. The latter will be drawn from open data portals such as Kaggle or CKAN. For the final version of the platform, additional offerings, providing full access to these APIs, may be provided.
- Provisioning the participants with databases or file servers to host the datasets to be provided.
- Testing the dataset offerings. We will verify that the datasets provided by the open data enabler are visible to any participant in the ecosystem and that their data can be downloaded to other participants' premises.

For the final version of SEDIMARK, these steps will be extended by adding more provided offerings, as well as documentation materials for participants to consume them, but also for SEDIMARK's contributors to know how to add open data offerings to the open data enabler.

3.7.3 Specifications of involved components

As mentioned in the previous section, the open data enabler is a minimalistic participant in the SEDIMARK ecosystem, acting solely as an offering provider. It therefore needs all SEDIMARK core components: DLT enabler / trust framework, asset data exchange / connector and catalogue & offering lifecycle. Toolboxes such as AI and data processing are not required. The marketplace frontend is also just a convenience, not a requirement, since offerings and contracts can be managed directly from the connector API.

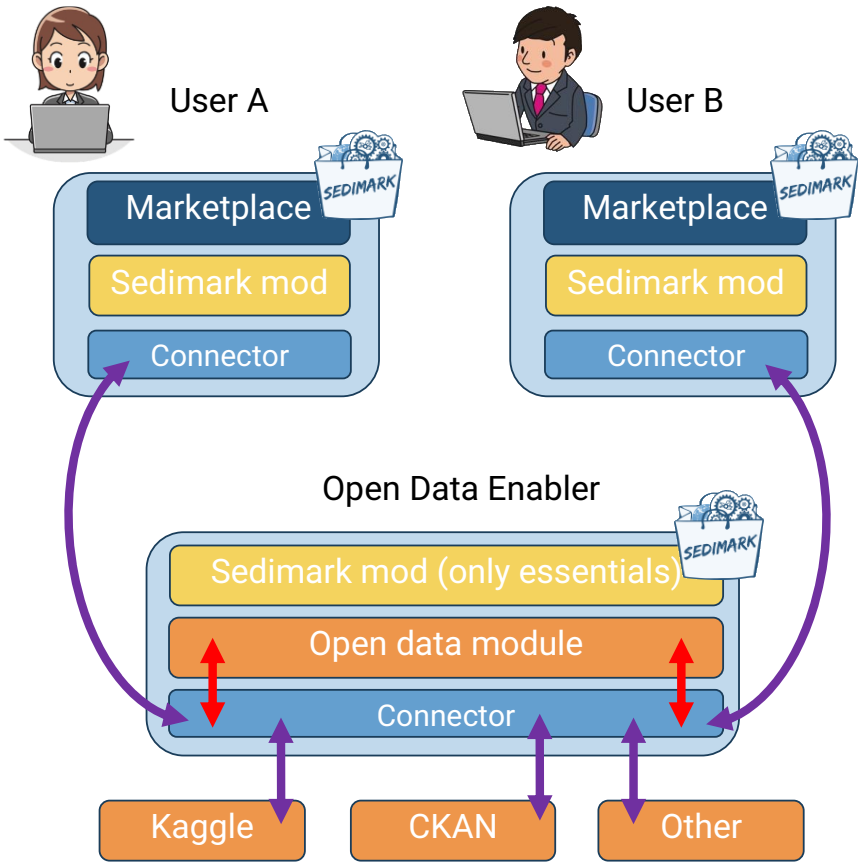


Figure 35 Open data enabler working principle

Figure 35 illustrates how the open data enabler works as a participant. The *Open Data module* consists of a group of microservices needed to expose the open data offerings. For the first version of SEDIMARK, this module will only contain databases or a file server, to expose dataset offerings. However, the final version of the platform, may contain custom software components to enable exposing a full open data API.

3.7.4 Integration specification

The integration of the open data enabler requires all core data space components of SEDIMARK to be operational and deployable as a set of microservices. Full integration involves the following milestones:

- Establishment of an infrastructure to host the core components of SEDIMARK corresponding to the open data enabler participant.
- Creation of offerings self-descriptions compliant with SEDIMARK ontology, representing the datasets to be provided.
- Deployment of open data module components necessary to host the offerings' data (databases, file server, etc.).
- Deployment of a federated catalogue instance in the open data enabler premises.
- Registration of the open data enabler as a participant in the SEDIMARK ecosystem.
- Deployment of a connector instance in the open data enabler premises.

- Testing the provision and consumption of an offering.
- Adding more dataset offerings.

3.7.5 Results

At this stage of the project, a dedicated Kubernetes cluster of two nodes have been created to host the open data enabler as a participant (corresponding to milestone 1 in the previous section). The creation of offerings self-description is ongoing, and will be finalized once the first version of the SEDIMARK ontology and catalogue is released.

3.7.6 Guidelines for deployment and execution

Deployment instructions will be provided as SEDIMARK core components' first version is released. These instructions will be centralized in one GitHub repository in the SEDIMARK organization, and will consist of a detailed description of:

- The components required for the open data enabler to run.
- The necessary configuration for the open data enabler to operate as a participant.
- The current list of offerings provided by the open data enabler.
- How to create and publish a new offering on behalf of the open data enabler.

3.7.7 Software licences

As a participant in the SEDIMARK ecosystem, the Open Data enabler relies on the SEDIMARK core components and therefore does not require any specific licensing.

4 Cross-check of high-priority requirements

The scope of this section is to provide a clear view of the functional and non-functional requirements of SEDIMARK architecture in relation to the seven independent PoC scenarios. For this purpose, two tables are provided (Table 2, Table 3) correlating all the high-priority requirements (H-REQ) that were promised to be fulfilled in the first version, with the PoCs. The aim is to monitor the status of fulfillment and in which scenario they are addressed.

Table 2 Functional requirements of the architecture related to the different PoCs

Identifier	Title	Category	Related PoC scenarios	How does the PoC fulfil the requirement? (fully, partially, at all)
Req-DP-01	SEDIMARK must provide tools to clean the data or flag the bad records without removing them, allowing user to define how to handle bad data	Data processing	PoC2 “Data quality Improvement”	Fully
Req-ML-01	Data need to be cleaned, in a standard common format and of a specific size for proper training of the models	AI-ML	PoC5 “AI-related scenarios”	Partially
Req-RS-01	Recommender models need to be able to compute user profiles from their history	Recommendation	PoC5 “AI-related scenarios”	No
Req-RS-01	Recommender models need to know the available item set and have as many features, metadata and statistics as possible	Recommendation	PoC5 “AI-related scenarios”	Partially

Identifier	Title	Category	Related PoC scenarios	How does the PoC fulfil the requirement? (fully, partially, at all)
Req-RS-03	Recommendation platform must be trained and run in a decentralised way, not leaking user private information	Recommendation	PoC5 “AI-related scenarios”	Partially
Req-EE-01	Data processing modules must be lightweight and energy efficient	Energy efficiency	PoC2 “Data quality Improvement”	Partially
Req-P&D-01	Datasets, data streams and services have to be uniformly described as offerings to support its homogeneous publication and easy discovery	Publication and Discovery	PoC3 “Offering lifecycle”	Partially
Req-P&D-02	Offerings have to be published on a common distributed registry from which they can be listed	Decentralization, Publication	PoC3 “Offering lifecycle”	Partially
Req-P&D-03	Metadata included in offerings must be generic enough to avoid imposing restrictions on the assets data format or the mechanisms used for its provision	Publication and Discovery, Openness	PoC3 “Offering lifecycle”	Partially
Req-P&D-04	Dynamic discovery of datasets from Open Data portals should be enabled	Publication and Discovery, Openness	PoC7 “Open Data enabler”	Partially

Identifier	Title	Category	Related PoC scenarios	How does the PoC fulfil the requirement? (fully, partially, at all)
Req-INT-01	The information model should allow a wide application scope and enable interoperability	Interoperability	PoC2 “Data quality Improvement”	Fully
Req-INT-02	The information model should prescribe aspects of the data and the metadata	Interoperability	PoC2 “Data quality Improvement”	Fully
Req-INT-03	Compliance with the information model should be a (partial) prerequisite for SEDIMARK data	Interoperability	PoC2 “Data quality Improvement”	Partially
Req-INT-04	Data tools should be developed to enforce compliance with the information model	Interoperability	PoC2 “Data quality Improvement”	Partially
Req-INT-05	The information model should prescribe specific metadata fields	Interoperability	PoC2 “Data quality Improvement”	Fully
Req-INT-06	ML Models interoperability	Interoperability	PoC4 “Asset (Data) exchange”	Partially
Req-INT-07	To support use case data loading, processing and enrichment	Interoperability	PoC2 “Data quality Improvement”	Partially

Identifier	Title	Category	Related PoC scenarios	How does the PoC fulfil the requirement? (fully, partially, at all)
Req-SEC-01	Users (Providers and Consumers) must be able to authenticate in the SEDIMARK Trust domain	Security	PoC1 “Participants onboarding”	Partially (e.g., Credential Subject information to be defined/completed)
Req-SEC-02	Each asset must be characterized by an authorization policy decided by the respective Provider. Use of the Asset is dependent on such policies	Security	PoC4 “Asset (Data) exchange”	Partially. Current PoC only covers basic functionality.
			PoC1 “Participants onboarding”	Partially (e.g., Authorization policies to be defined/completed)
Req-SEC-03	Assets origin and integrity must be maintained in the SEDIMARK Trust Domain	Security	PoC4 “Asset (Data) exchange”	Fully
			PoC1 “Participants onboarding”	n.a. - No asset transfer involved in the PoC.
Req-SEC-04	Trust Metadata must be written onto the Distributed Ledger	Security, Decentralization	PoC4 “Asset (Data) exchange”	Partially. Current PoC does not write access policies.
			PoC1 “Participants onboarding”	Fully for Decentralized Identity; Partially otherwise
Req-SEC-05	Assets provisioning must employ a decentralised approach	Decentralization	PoC4 “Asset (Data) exchange”	Fully
			PoC1 “Participants onboarding”	Fully for Decentralized Identity; Partially otherwise

Identifier	Title	Category	Related PoC scenarios	How does the PoC fulfil the requirement? (fully, partially, at all)
Req-SEC-06	Assets must be transferred over secure communication channels (e.g., through TLS)	Security	PoC4 “Asset (Data) exchange”	Fully
			PoC1 “Participants onboarding”	n.a. - No asset transfer involved in the PoC.
Req-STR-01	Datasets must be stored within the provider's domain by default.	Data storage	PoC4 “Asset (Data) exchange”	Fully
Req-STR-02	Offering descriptions (metadata) must be stored in the distributed catalogue provided by SEDIMARK.	Data storage	PoC3 “Offering lifecycle”	Partially
Req-STR-03	Intermediate data within the data processing pipeline needs to be temporarily stored.	Data storage	PoC2 “Data quality Improvement”	Fully
Req-STR-04	Final dataset output of the data processing pipeline needs to be stored in a consumable manner.	Data storage	PoC2 “Data quality Improvement”	Fully
Req-UI-01	UI shall provide login capability	UI, Security	PoC6 “GUIs”	No
			PoC1 “Participants onboarding”	Partially (preliminary login for the PoC itself)

Identifier	Title	Category	Related PoC scenarios	How does the PoC fulfil the requirement? (fully, partially, at all)
Req-UI-02	Users must be able to discover, search/filter offering in the marketplace based on identity-based policies defined by the asset provider	UI, Publication and Discovery	PoC3 "Offering lifecycle"	Partially
			PoC6 "GUIs"	Partially
Req-UI-03	Users have to be able to create, own and manage their own SSI identity	UI, Security	PoC1 "Participants onboarding"	Partially (not completely showing the management of SSI identity)
			PoC6 "GUIs"	Partially
Req-UI-04	Providers should be able to manage their offerings (e.g. adding or removing, setting privacy level, authorization policies, set licence, setting payment policy, ...)	UI	PoC3 "Offering lifecycle"	Partially
			PoC4 "Asset (Data) exchange"	Partially. Current PoC only covers basic functionality
			PoC6 "GUIs"	Partially
Req-UI-05	Consumer shall be able to view Asset information (e.g. Size, Provider, Authorization Policies, Price, etc).	UI	PoC4 "Asset (Data) exchange"	Partially
			PoC6 "GUIs"	Partially

Table 3 Non-functional requirements of the architecture related to the different PoCs

Identifier	Title	Category	Related PoC scenarios	How does the PoC fulfil the requirement? (fully, partially, at all)
Req-NF-01	Decentralisation	Non-functional	PoC5 "AI-related scenarios"	Fully
			PoC4 "Asset (Data) exchange"	Fully
			PoC1 "Participants onboarding"	Fully for Identity; Partially otherwise
Req-NF-02	Security, Privacy, Trust	Non-functional	PoC4 "Asset (Data) exchange"	Partially. Current PoC only covers basic authorization functionality
			PoC1 "Participants onboarding"	Fully for Identity; Partially otherwise
Req-NF-03	Interoperability	Non-functional	PoC2 "Data quality Improvement"	Partially
			PoC4 "Asset (Data) exchange"	Partially. Current PoC only covers basic functionality.
Req-NF-04	Data availability and quality	Non-functional	PoC2 "Data quality Improvement"	Partially
Req-NF-05	Intelligence	Non-functional	PoC5 "AI-related scenarios"	Fully
Req-NF-06	Energy efficiency	Non-functional	PoC5 "AI-related scenarios"	At all
Req-NF-07	Resilience and Reliability	Non-functional	All PoCs	Fully at the current integration phase
Req-NF-08	Scalability	Non-functional	PoC4 "Asset (Data) exchange"	Fully
Req-NF-09	Openness, Extensibility	Non-functional	PoC4 "Asset (Data) exchange"	Partially. While current PoC contributes to this requirement, it does not cover the openness and extensibility of the whole SEDIMARK platform.
Req-NF-10	Usability	Non-functional	Mainly PoC6 "GUIs", all PoCs	Fully at the current integration phase

Identifier	Title	Category	Related PoC scenarios	How does the PoC fulfil the requirement? (fully, partially, at all)
Req-NF-11	Maintainability	Non-functional	All PoCs	Fully at the current integration phase
Req-NF-13	Reusability	Non-functional	PoC4 “Asset (Data) exchange”	Partially. While current PoC contribute to this requirement, it does not cover the reusability of the whole SEDIMARK platform
Req-NF-14	Flexibility	Non-functional	All PoCs	Fully at the current integration phase

5 Conclusions

This deliverable is the outcome of Task T5.1 "Platform continuous integration" and provides details regarding the implementation of the first version of the foreseen platform. The overall system is divided into three phases / versions: The first version (which is going to be delivered in M18-Mar. 2024), the second version (delivered in M27-Dec. 2024), final version (delivered in M36-Sep. 2025).

At the current phase of the project, the consortium has managed to deliver successfully the first integrated release of the SEDIMARK platform. The integration has started early, following the agile methodology to have as early as possible a "minimum viable product (MVP)" that can be tested with limited functionalities and identify early integration issues and problems or missing functionalities.

The supported scenarios will follow a standardized format (high-level description, step-by-step definition, specifications of involved components, integration specification, results, guidelines for deployment and execution, and results). Furthermore, all scenarios will be tested using data from the project's four pilots. Upcoming versions will include incremental work and component sophistication, as well as increased support for the remaining requirements and less or no hard coding. All functionalities will be progressively integrated in the marketplace GUIs.

6 Bibliography

[1] SEDIMARK, Deliverable 5.1: Evaluation methodology, metrics and integration plan, November 2023.

[2] SEDIMARK, Deliverable 4.1: Decentralized Infrastructure and Access Management - First version, December 2023.

[3] SEDIMARK, Deliverable 3.3: Enabling tools for data interoperability, distributed data storage and training distributed AI models. First version, December 2023.

[4] SEDIMARK, Deliverable 4.5: Data sharing platform and incentives - First version, December 2023.

[5] Metamask, [Online]. Available: https://metamask.io/.

[6] JSON Web Tokens, [Online]. Available: https://jwt.io/.

[7] GiHub repository for smart data models, [Online]. Available: https://github.com/smart-data-models/dataModel.DataQuality/tree/master/DataQualityAssessment

[8] Mage.AI, [Online]. Available : https://www.mage.ai/.

[9] Ludwig, [Online]. Available : https://ludwig.ai/latest/.

[10] TensorFlow, [Online]. Available : https://www.tensorflow.org/

[11] PyCaret, [Online]. Available : https://pycaret.org/

[12] Scikit-learn, [Online]. Available : https://scikit-learn.org/stable/

[13] PyOD, [Online]. Available : https://pyod.readthedocs.io/en/latest/

[14] Apace NiFi, [Online]. Available : https://nifi.apache.org/

[15] Stelio Context Broker, [Online]. Available : https://stellio.readthedocs.io/en/latest/

[16] Gradle, [Online]. Available: https://gradle.org/

[17] Redis, [Online]. Available: https://redis.io/