SEcure Decentralised Intelligent Data MARKetplace

D4.2 Decentralized Infrastructure and Access Management - Final version

Document Identification	
Contractual delivery date:	31/07/2025
Actual delivery date:	30/09/2025
Responsible beneficiary:	LINKS
Contributing beneficiaries:	LINKS, UC, SURREY
Dissemination level:	PU
Version:	1.0
Status:	Final

Keywords:
Decentralisation, Identity, SSI, Authentication, Tokenisation, Catalogue, Connectors

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. and is also partly funded by UK Research and Innovation (UKRI) under the UK government's Horizon Europe funding guarantee. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission or UKRI.

Document Information

Document Identification			
Related WP	WP4	**Related Deliverables(s):**	SEDIMARK_D4.1
Document reference:	SEDIMARK_D4.2	**Total number of pages:**	54

List of Contributors	
Name	**Partner**
Alberto Carelli Michele Festa	LINKS
Pablo Sotres Juan Ramón Santana Víctor González Jorge Lanza Luis Sánchez	UC
Tarek Elsaleh Sneha Hanumanthaiah, Anuroop Keshav	SURREY

Document History			
Version	**Date**	**Change editors**	**Change**
0.1	14/03/2025	LINKS	First version of document (Table of Content)
0.2	04/06/2025	LINKS	Initial contributions in Sect. 2
0.3	12/06/2025	LINKS	Update Sect. 3, 4, 5
0.4	18/06/2025	UC	Contribution in Sect. 3.3
0.5	25/06/2025	SURREY	Contribution in Sect. 3.4
0.6	15/07/2025	LINKS	Update Sect. 6
0.7	28/08/2025	SURREY	Update Sect. 3, 4
0.8	03/09/2025	UC	Update Sect. 2, 4, 5

Document History			
0.92	16/09/2025	LINKS	Version ready for Technical Reviews
0.95	30/09/2025	LINKS	Version ready for Quality Review
0.97	30/09/2025	ATOS	Quality Review Form
1.0	30/09/2025	ATOS	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	**Who (Partner short name)**	**Approval date**
Reviewer 1	Nikolaos Babis (MYT)	30/09/2025
Reviewer 2	Thomas Bousselin (EGM)	19/09/2025
Quality manager	María Guadalupe Rodríguez (ATOS)	30/09/2025
Project Coordinator	Miguel Ángel Esbrí (ATOS)	30/09/2025

Table of Contents

Document Information 2
Table of Contents 4
List of Figures 6
List of Acronyms 7
Executive Summary 9
1 Introduction 10
1.1 Purpose of the document 10
1.2 Structure of the document 10
2 Interactions in a Secure and Decentralized Marketplace 11
2.1 Foundations: Decentralization in the Marketplace 11
2.2 Entities and Interactions in the Marketplace 11
3 Core Technologies for Decentralization 21
3.1 IOTA DLT 21
3.2 IOTA Smart Contract (ISC) chain 22
3.3 Dataspace Protocol 23
3.4 Distributed Triple Stores Protocol 23
3.4.1 Decentralised Offering Management 24
3.4.2 Decentralised Query Resolution 28
3.4.3 Node Dynamics and Resilience 28
4 Implementation Perspectives 30
4.1 Connector 30
4.2 Catalogue 32
4.3 Issuer 35
4.4 Verifier 36
4.5 DLT-Booth 37
4.6 Main Libraries 38
5 Digital Identity and Access Management 40
5.1 Background: Self-Sovereign Identity (SSI) 40
5.2 Identity Data Models – Final Version 41
5.2.1 DID Document 41
5.2.2 Verifiable Credentials 42
5.3 Authentication 45
5.4 Authorization and Access Policies 46

6 Tokenization 49
6.1 Types of Tokens and Standards 49
6.2 Smart Contracts Overview and Token Creation 50
6.3 Token usage and available operations 51
7 Conclusions 52
8 References 53

List of Figures

Figure 1: Marketplace - core entities and interactions 12
Figure 2: Architecture of onboarding stage 15
Figure 3: Architecture of offering lifecycle in the marketplace 16
Figure 4: Overview of catalogue architecture in the marketplace 17
Figure 5: Architecture of offering negotiation, agreement tokenization and asset exchange 19
Figure 6: Anchoring between the DLT layers 22
Figure 7: Interaction Sequence for Centralised Mode 25
Figure 8: Interaction Sequence for Decentralised Mode 26
Figure 9: Triple Store Distribution Mechanism 27
Figure 10: Decentralised Query Resolution 28
Figure 11: Contract Negotiation Protocol [19] (a) and Transfer Process Protocol [20] (b) FSMs 31
Figure 12: Detail of the IDS transfer data plane interactions between connectors ... 32
Figure 13: Centralised Catalogue Deployment 33
Figure 14: Decentralised Catalogue Deployment using BIF-sharded Local Catalogues 34
Figure 15: Decentralised Deployment using Federated Local Catalogues at Provider Domains 34
Figure 16: Decentralised Deployment using Local Catalogues at Provider Domains 35
Figure 17: Example of DID Document 42
Figure 18: VC data model and VP data model 42
Figure 19: Application-layer Holder authentication process. 45
Figure 20: Example of SEDIMARK offering ODRL policy 48
Figure 21: Digital asset tokenization 49

List of Acronyms

Abbreviation / acronym	Description
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
DCAT	Data Catalogue Vocabulary
DT	Datatoken
DTSC	Datatoken Smart Contract
DID	Decentralize Identifier
Dx.y	Deliverable number y belonging to WP x
DAG	Directed Acyclic Graph
DKG	Distributed Key Generation
DLT	Distributed Ledger Technology
DTS	Distributed Triple Store
EC	European Commission
EDC	Eclipse Dataspace Components
EVM	Ethereum VM
FRESC	Fixed-Rate Exchange Smart Contract
FT	Fungible Token
GSP	Graph Store Protocol
IDSA	International Data Spaces Association
IPFS	InterPlanetary File System
ISC	IOTA Smart Contract
JWT	JSON Web Token
L1	Layer 1
L2	Layer 2
ML	Machine Learning
MVM	Minimum Viable Marketplace
NFTSC	NFT Smart Contract
NFT	Non-Fungible Token

Abbreviation / acronym	Description
ODRL	Open Digital Rights Language
P2P	Peer-to-Peer
PDP	Policy Decision Point
PEP	Policy Enforcement Point
RDF	Resource Description Framework
REST	REpresentational State Transfer
SC	Smart Contract
SDK	Software Development Kit
SSI	Self-Sovereign Identity
TLS	Transport Layer Security
UCs	Use Cases
UTXO	Unspent Transaction Output
VC	Verifiable Credential
VM	Virtual Machine
VP	Verifiable Presentation
WP	Work Package
W3C	World Wide Web Consortium

Executive Summary

In response to the growing demand for secure and transparent data exchange, the infrastructure of SEDIMARK Marketplace leverages cutting-edge technologies to establish a resilient network.

The decentralization approach ensures increased security, transparency, and user-centric control both over different types of assets and user identity information. The SEDIMARK Marketplace leverages distributed ledger technologies to establish a resilient and scalable infrastructure. The decentralized architecture of the marketplace is built on a robust distributed ledger employed for user identity management, as well as blockchain foundation, fostering tamper-resistant contracts.

This deliverable presents the final version of the Decentralized Infrastructure employed for the SEDIMARK Marketplace and the APIs that enable the functionalities satisfying the requirements and the objectives for the Project. It represents an important capstone for every stakeholder that plans to use the Marketplace for offering assets and consuming data and services. Here, after a careful examination and design process across all the partners of the project, the final version of the SEDIMARK Marketplace is shaped and consolidated. This document stems from the previous capstone realized with Deliverable SEDIMARK_D4.1 [26] and Deliverable SEDIMARK_D3.4 [25] in December 2023. The current deliverable builds upon the previous version, extending its scope and functionality. This document highlights the key differences, changes, and additions made since the last iteration, providing a clear overview of the enhancements and improvements implemented in this version. Additionally, each section of the current deliverable includes context and rationale for the updates, ensuring a comprehensive understanding of the project's evolution.

1 Introduction

1.1 Purpose of the document

This document serves as a comprehensive exploration of the final version of the decentralized infrastructure and access management framework implemented within the SEDIMARK Marketplace.

The main goal is to consolidate, update and describe the underlying architecture that enables the decentralisation in the SEDIMARK Marketplace. The aim is to provide a clear understanding of the design principles, functionalities, and benefits associated with the decentralized infrastructure that leads to an improved trustworthiness of the Marketplace.

Additionally, the aspects related to the access management of users of the platform are considered in the framework of Self-Sovereign Identity (SSI). Moreover, the mechanisms and the components that implement the core business logic of the Marketplace are mapped onto these features.

Finally, this document also aims to provide the details about how the tokenization of the assets exchanged together with Smart Contracts chains enable SEDIMARK as a secure and decentralized marketplace.

1.2 Structure of the document

The document is organized as follows:

Section 1 introduces the objective of the document.

Section 2 presents the updated interactions considered taking place in a marketplace and how those are mapped in the final SEDIMARK architecture.

Section 3 reviews the decentralised infrastructure, i.e., the foundation of the Marketplace, and the decentralized offering management providing the related relevant updates.

Section 4 describes the final version of the realized components involved in the infrastructure according to an implementation point-of-view.

Section 5 focuses on the access management mechanisms that regulate and control Users' capabilities into the marketplace.

Section 6 analyses the tokenization framework that allows the assets to be traded in the marketplace.

Section 7 concludes and summarizes the document.

2 Interactions in a Secure and Decentralized Marketplace

This section defines the core entities and explores the various interactions that take place in the marketplace. In the SEDIMARK Marketplace there are different participants who move the assets of interest: from the providers who generate and sell valuable assets to the consumers who seek valuable insights. The target of the SEDIMARK marketplace is to foster collaborations and facilitate the seamless exchange of data relying onto a secure and decentralized infrastructure.

2.1 Foundations: Decentralization in the Marketplace

The key objectives of the project require a decentralised, intelligent, trustworthy and interoperable data and services marketplace. The SEDIMARK Marketplace provides such decentralized environment for the exchange of data and other assets between different parties.

The core of the SEDIMARK Marketplace is a fully decentralized solution that eliminates any single point of data collection or failure. This approach empowers data providers to retain their data locally, on their own premises, and only share it with the desired consumers. The decentralized architecture of SEDIMARK facilitates peer-to-peer communication among various participating nodes (providers, consumers, etc.), avoiding the need for intermediate nodes. This in turn allows consumers to access providers' assets directly.

The Marketplace's architecture utilizes IOTA Distributed Ledger Technology (DLT), which ensures seamless application interactions, protects users from intermediary value extraction, and efficiently manages high transaction volumes. Additionally, IOTA supports parallel transaction processing and, combined with tokenization and Layer 2 Ethereum VM-compatible (EVM) chains, fosters new types of interoperable digital economies. Unlike traditional blockchains, IOTA employs a particular data structure that permits interactions at any time, enabling operations to occur simultaneously rather than relying on linear blocks processed at fixed intervals.

From an architectural point-of-view, the employed technology can be partitioned in two interacting layers: Layer 1 (L1) – IOTA Tangle and Layer 2 (L2) – IOTA Smart Contract Framework (ISC). The information in these layers is distributed across a network of nodes.

Together, these layers compose the logic of the decentralized infrastructure and help enforce access policies of the Marketplace. In particular, L1 is responsible for managing participant identities. These two layers have the role of defining the decentralised infrastructure as well as contribute to enforcing access policies. L1 is employed for the management of the identity of the participants, while L2 is implements the core business logic of the Marketplace, i.e., enables the discovery and trading of assets.

The layers are detailed in Section 3 in the past SEDIMARK_D4.1 deliverable [26]. The actual implementation, together with the final physical architecture is described in SEDIMARK_D3.4 deliverable [25].

2.2 Entities and Interactions in the Marketplace

The SEDIMARK Marketplace is composed of various entities that interact to facilitate the exchange of data and services. The entities and their interactions are crucial to understand the Marketplace functionalities provided to its users. This section provides an overview of the

entities involved, their roles, and the dynamics of their interactions in the final version of the Marketplace, building upon the foundational concepts established in the previous version of the deliverable SEDIMARK_D4.1 [26]. From a high-level perspective, the core entities and their interactions remain consistent with the previous version of this deliverable, ensuring continuity with the functionalities already defined and developed in the first part of the project. The high-level view of the entities and their interactions is reported in Figure 1.

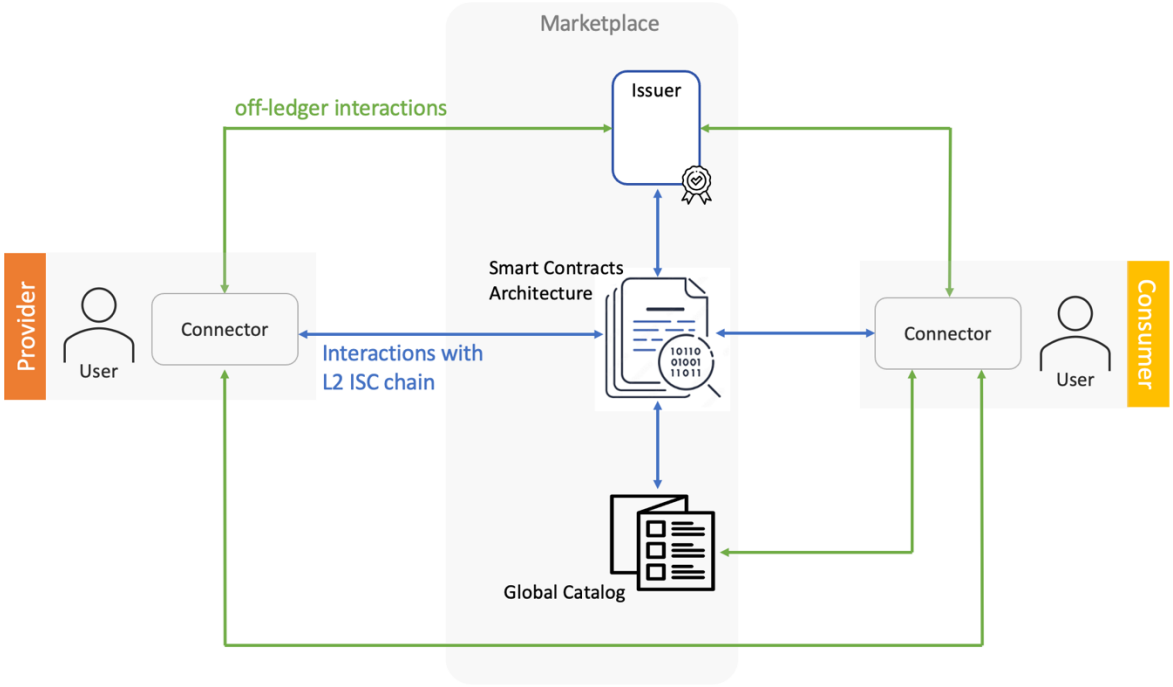

Figure 1: Marketplace - core entities and interactions

The key entities in the SEDIMARK Marketplace are its users (or participants). Users are the main actors in the ecosystem that provide and/or consume services. More specifically, a user can be a Consumer or a Provider.

Consumers are individuals or organizations that purchase and utilize data and services for a variety of own purposes. The main activities of consumers in the Marketplace are:

- Asset Acquisition (Purchase): Consumers look at the offerings, evaluate the ones(s) desired and based on its own subjective criteria decide whether to complete the purchase.
- Asset Utilization (Access): Once assets are acquired, consumers can use them by accessing them. They might also integrate them into workflows, analyses, or applications, leveraging insights to make careful decisions.

Providers are entities that supply data, access to data, or services such as machine learning (ML) models to Consumers within the marketplace. They play a crucial role in the ecosystem complementary to the Consumers. Their main activities are related to the Offerings Publication: Providers can offer a variety of assets, including structured and unstructured data, as well as pre-trained machine learning models and tools. The Consumers create the respective offerings of the assets (e.g., data, ML models, etc.) they are willing to sell.

Also, the Providers are essential to the ecosystem, offering valuable assets that meet the various need of the Consumers. They can supply a range of data types, encompassing both structured and unstructured data. This may include datasets related to demographics, financial transactions, weather patterns, social media activity, and more. It has to be pointed out that in

the context of SEDIMARK, Providers will deliver datasets covering the four specific Use Cases (UCs) outlined in SEDIMARK_D2.1 [14]. Moreover, beyond selling raw data, some providers may also offer access to pre-trained machine learning models or other analytical tools. This diversification enhances the variety of the SEDIMARK Marketplace's offerings.

The issuer is another crucial entity within the SEDIMARK Marketplace, particularly in the context of Self-Sovereign Identity (SSI). The main activity is the provisioning of Verifiable Credentials (VCs) to the external users who wants to join the Marketplace. It can be considered as the main point of entrance to the Marketplace. More details are reported in Section 4.3.

After completing the onboarding operation with the Issuer, the users can employ the Catalogue. The Catalogue is another key entity of the Marketplace. It allows the users to browse existing offerings in the SEDIMARK Marketplace. Additional details on the operations enabled by the catalogue are given in Section 4.2.

The different operations needed for the communications among the entities in the Marketplace are enabled through a special component, named Connector. It is a software tool to enable secured peer-to-peer information exchange between Participants. It relies on the Registry (i.e., the DLT) for trust purposes through a sub-component named DLT-Booth. The DLT-Booth is deployed on each participant domain and, therefore, is part of a decentralized network of peers. Further details related to the Connector are reported in Section 4.1, while the newly added component DLT-Booth is considered in Section 4.5.

The various entities interact with each other employing different communication channels. In this deliverable, the on-ledger communication and off-ledger communication are distinguished for the scope the deliverable and to provide a clear separation of the functionalities related to the underlying decentralized infrastructure. In any case, every communication channel set up for the sake of the interaction is protected through security mechanisms that guarantee the confidentiality and authenticity (i.e., resorting to HTTPS) of information exchanged.

All the interactions are mediated by the Connector, which acts as an interface at the Participant side to access the SEDIMARK domain.

The off-ledger interactions are instead common direct connections between a couple of Connectors following a P2P (Peer-to-Peer) paradigm. The communications do not rely onto the DLT.–The off-ledger interactions are employed, for instance, to establish the initial connection with the Issuer of the SEDIMARK Marketplace for a user who does not yet own a credentials allowing him to interact with the SEDIMARK ecosystem. On the other hand, a SEDIMARK user can establish a direct connection with a Catalogue, relying onto an off-ledger interaction, to easily browse the offerings available on the Marketplace.

The on-ledger interactions, also shown in Figure 1, employ both L1 and L2 of the decentralized DLT infrastructure. The summary of the background functionalities of the infrastructure from logical point of view is reported Section 3, while the final physical instance is reported in SEDIMARK_D3.4 deliverable [25]. These kinds of interactions are communications between entities that directly rely on the distributed ledger. Therefore, the communications on this means allows the immutability of the information exchanged and leverages an additional layer of trust of the parties involved. Moreover, the information are permanently stored, enabling monitoring and transparency of the transactions as well as future auditing capabilities. Referring to Figure 1, consider as example a Consumer who after reviewing the Catalogue desire to purchase an asset. The Connector facilitates this operation interfacing with the (IOTA) Smart Contract architecture (outlined in Section 3.2 and Section 6). For the sake of clarity, Figure 1 omits the on-ledger interactions happening with L1, which is mostly employed for

storage and retrieval of public elements associated with the Self-Sovereign Identity (SSI) of the involved entities. Further details are reported in Section 5.

The business logic of the marketplace is realized through different actions involving the entities described above. Such actions reflect the series of fundamental procedures and events that the SEDIMARK platform supports through the decentralized infrastructure and APIs provided in this deliverable. The operational framework is structured into five core stages:

- Participant Onboarding, which manages identity creation and credential issuance according to the SSI paradigm;
- Offering Tokenization, which enables the registration and publication of data offerings within the Marketplace;
- Offering Discovery, which allows participants to browse, identify, and evaluate available data assets prior to initiating negotiations;
- Offering Negotiation and Agreement Tokenization, which supports secure and verifiable contract establishment between participants; and
- Asset Exchange, which ensures compliant, auditable, and traceable delivery of data assets.

The following list details the steps required by the processes that are iterated among different users to implement the core functionalities of the marketplace.

Participant Onboarding

The onboarding procedure is a prerequisite for any external entity wishing to interact with the SEDIMARK Marketplace. It serves as the gateway to all subsequent functionalities and ensures that participants are securely and uniquely identified within the ecosystem. The entities involved in this procedure are illustrated in Figure 2.

In the final implementation of the Marketplace, the onboarding process is initiated by the participant through the DLT-Booth component embedded in its Connector. This module is responsible for generating a digital identity in line with the Self-Sovereign Identity (SSI) framework. The identity creation process involves generating a cryptographic keypair, a Decentralized Identifier (DID), and a corresponding DID Document. The public key is published on Layer 1 (L1), while the private key is securely stored on the participant's side using a newly integrated key management feature within the DLT-Booth.

Once the DID is registered in the Verifiable Data Registry, the DLT-Booth proceeds to request a Verifiable Credential (VC) from a trusted Issuer within the SEDIMARK platform. To validate ownership of the DID, the Issuer issues a cryptographic challenge, which the participant must sign using its private key. A successful signature confirms control over the identity and authorizes the Issuer to issue the VC.

The Issuer then uses its own SSI credentials to interact with the Identity Smart Contract (SC), registering the participant's VC as active. Upon completion, the VC is returned to the participant and stored locally within its domain. The DLT-Booth also maintains a secure database of VCs and manages private keys using a dedicated key management library.

Holding a valid VC enables the participant to access the secure and decentralized services of the SEDIMARK Marketplace. For Customers, the VC grants access to marketplace content, while for Providers, it implicitly authorizes the publication of offerings. When interacting with other components, the VC is encapsulated within a Verifiable Presentation (VP), which serves as proof of identity and authorization. Verifiers that trust the Issuer will accept the VP and rely on its embedded claims for access control decisions.

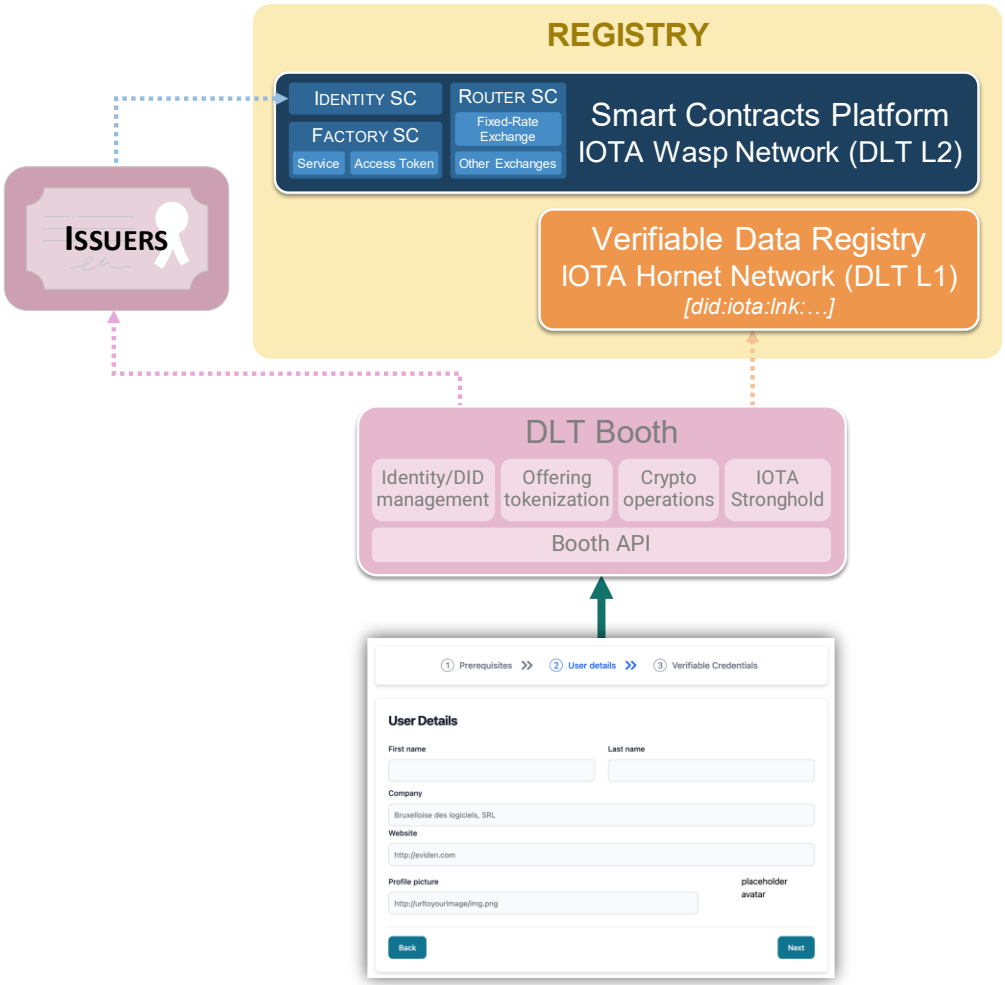

Figure 2: Architecture of onboarding stage

To ensure trust and interoperability, the Issuer's public key must be publicly accessible. This is achieved by hosting the Issuer at a predefined web endpoint, where secure TLS communication ensures confidentiality and integrity during identity-related operations.

A comprehensive analysis of identity management under the SSI model, along with updated examples of DID, VC, and VP, is provided in Section 5.2 of this deliverable.

Offering Tokenization

This operation allows a Provider to declare its willingness to trade a specific asset, either data or services, by advertising offerings that will later be indexed by catalogues within the SEDIMARK platform. This procedure is exclusive to Providers and requires prior completion of the onboarding process.

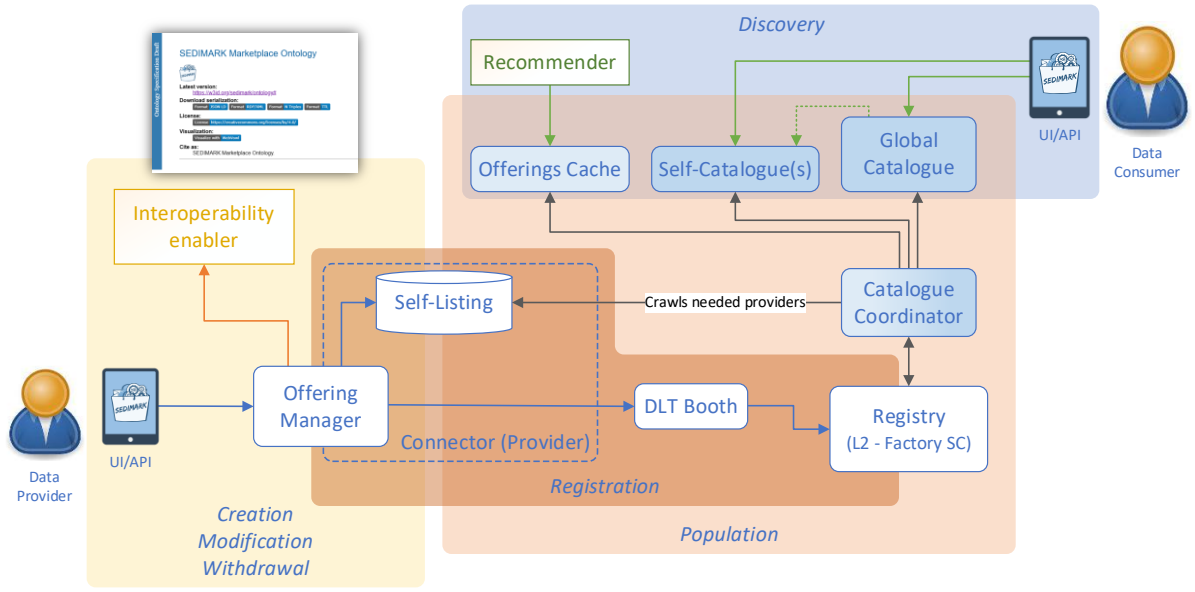

Figure 3: Architecture of offering lifecycle in the marketplace

In the current version of the Marketplace, the DLT-Booth component within the Provider's toolbox is responsible for interacting with the ISC infrastructure at Layer 2 (L2) to publish offerings related to its own assets. Once a participant is onboarded, it can proceed to configure and publish data offerings. As illustrated in Figure 3, this process begins with the local setup of the offering via the provider-facing interface, where metadata, access policies, and technical endpoints are defined. The Offering Manager, integrated within the participant's Connector, orchestrates the tokenization process. It communicates with the DLT-Booth to initiate interaction with the Registry's Factory Smart Contract, triggering the conversion of the offering into a verifiable digital asset anchored on the distributed ledger.

Within the participant's domain, the Offering Manager oversees the entire lifecycle of the offering from creation to publication. On-chain, the Factory SC ensures that the token and its metadata are securely stored and registered, guaranteeing integrity, traceability, and non-repudiation. Once registered, the offering becomes discoverable via the Registry and technically accessible through the Self-Listing interface, governed by the access conditions defined by the Provider.

Tokenization is achieved by minting a non-fungible token (NFT) via the ServiceBase SC (see Figure 4), compliant with the ERC-721 standard. This ensures each token is uniquely identifiable and interoperable across platforms supporting ERC-721. The token is owned by the Provider's identity and includes a reference to the offering description document, accessible through the Self-Listing interface. It also embeds a cryptographic hash of the document, allowing any participant to verify its integrity and authenticity. This mechanism prevents tampering and reinforces trust and transparency. The offering is also converted into a set of datatokens, one of which will be returned to Consumers upon asset purchase. Both the NFT and datatokens are derived from the original asset to make it tradable within the Marketplace. The technical details of the tokenization operations are further elaborated in Section 6 of this deliverable.

Offering Discovery

The offering discovery stage is enabled through an extensible catalogue architecture, as depicted in Figure 3. This stage plays a crucial role in allowing participants to explore and identify relevant offerings advertised in the SEDIMARK Marketplace before initiating any

negotiation process. Each catalogue operates independently and exposes offerings through solution-specific search endpoints. These catalogues are curated by their respective coordinators, who populate them using the offerings registered in the authoritative Registry as a reference point.

Figure 4 showcases how the catalogue architecture works and, as an example, how the centralized catalogue coordinator interacts with the Factory SC and the different participant self-listing to populate a triple-store database that can later be used by the GUI to discover available offerings.

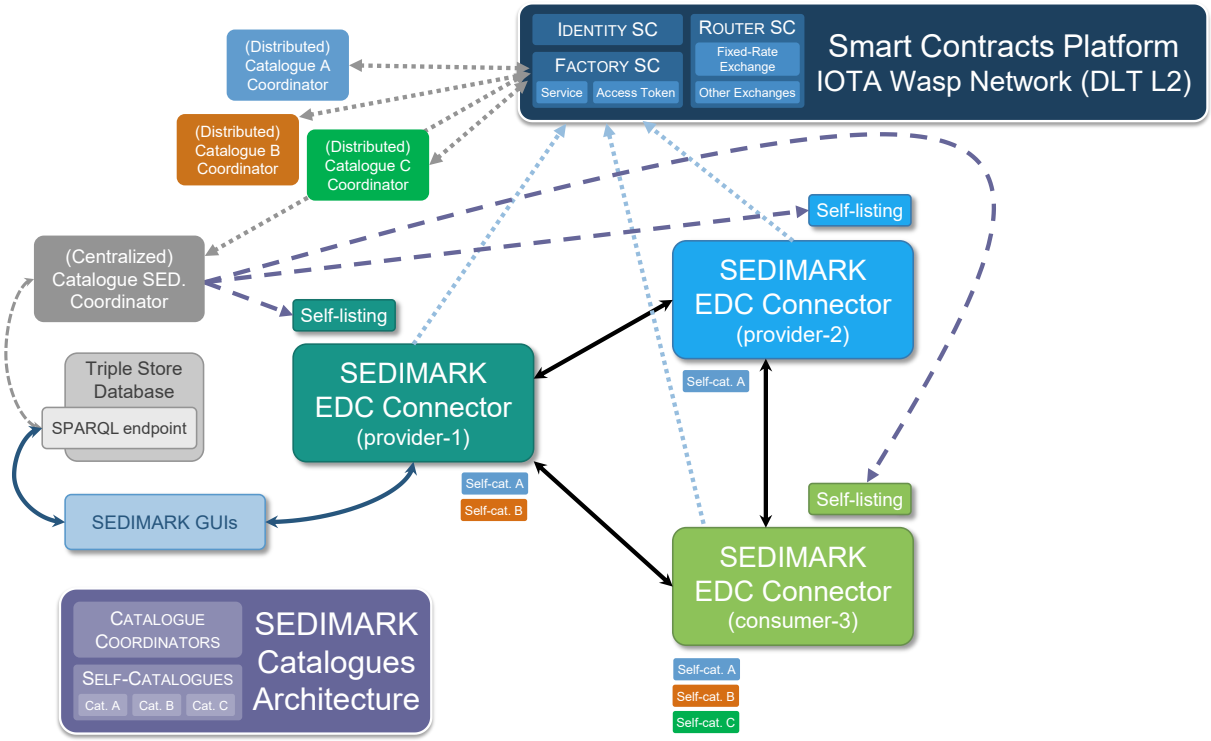

Figure 4: Overview of catalogue architecture in the marketplace

While catalogues may differ in their internal design and indexing strategies, this diversity allows for tailored optimizations that suit specific use cases or domains. The responsibility for deploying, maintaining, and operating each catalogue lies entirely with its owner, who contributes a value-added discovery service to the broader Marketplace ecosystem. It is important to highlight that catalogues are not considered core components of the Marketplace itself. By externalizing their implementation, the architecture remains open and flexible, encouraging third-party stakeholders to innovate and integrate specialized discovery mechanisms.

Despite their external nature, catalogues maintain alignment with the Marketplace's trust model by referencing the Registry for offering validation. This ensures that all offerings presented through catalogue interfaces are consistent, verifiable, and anchored to the authoritative source of truth, preserving the integrity and reliability of the discovery process.

A more detailed analysis of the offering discovery mechanisms is provided in Sections 3.4 and 4.2 of this deliverable.

Offering negotiation and agreement tokenization

The operation realizes the buy-and-sell functionality of the Marketplace. This procedure is twofold and represents an indirect interaction between the Consumer and the Provider, mediated with on-ledger interactions happening on the Layer 2.

Once an offering has been identified, the Consumer may initiate a negotiation to request access under specific terms. This process is conducted between the DataSpace Connectors of the involved parties, following a peer-to-peer communication model based on the DSP protocol. The negotiation logic is governed by a finite state machine (FSM), as defined in the DSP Contract Negotiation specification, which structures the exchange of contract proposals, counter-offers, and confirmations (see Figure 11a in Section 4.1). As illustrated in Figure 3, the negotiation architecture includes the Connectors, the DLT-Booth, and the relevant Smart Contracts (SCs). The architecture supporting this functionality, illustrated in Figure 5, includes the Connector components, the DLT-Booth, and the relevant Smart Contracts (SCs).

The outcome of a successful negotiation is formalized through agreement tokenization, implemented via a dual-token model. As already explained, each offering is represented by a non-fungible token (NFT), which guarantees its uniqueness and immutability. In parallel, a set of fungible Data Tokens (DTs) is generated to manage access rights. These DTs are minted by a dedicated DataToken Smart Contract (DTSC), compliant with the ERC-20 standard, and linked to the offering's NFT. The Provider receives a predefined supply of DTs, which serve as verifiable proofs of purchase and are required to access the asset. To enable trading, the Provider deposits the DTs into a Fixed-Rate Exchange Smart Contract (FRESC), part of the Router SC. This contract acts as the default decentralized exchange, although additional exchange mechanisms may coexist. Each DT is associated with a unique exchange instance, where the Provider sets the price in native tokens. The instance maintains essential metadata, (such as token address, ownership, and pricing) and supports efficient token swaps and transaction traceability.

While the NFT ensures the uniqueness and immutability of the offering, the associated DTs provide a flexible and fungible mechanism for access control. The combination of both token types enables a clear separation between offering representation and access rights, while ensuring that each purchase is recorded on-chain and can be independently verified. This dual-token model strengthens the trust, auditability, and operability of the Marketplace.

During the execution of the DSP state machine, once the negotiation reaches the *agreed* state, the Consumer's Connector interacts with the Router SC via the DLT-Booth to finalize the agreement and complete its tokenization. The Consumer acquires the DT by exchanging native tokens through the corresponding FRESC instance. The DT address is then included in the DSP's *ContractAgreementVerificationMessage*, allowing the Provider to verify token ownership. If verification succeeds, the FSM transitions to the *finalized* state. If verification fails or inconsistencies are detected, the FSM moves to the terminated state, and the DT acquisition may be rolled back accordingly.

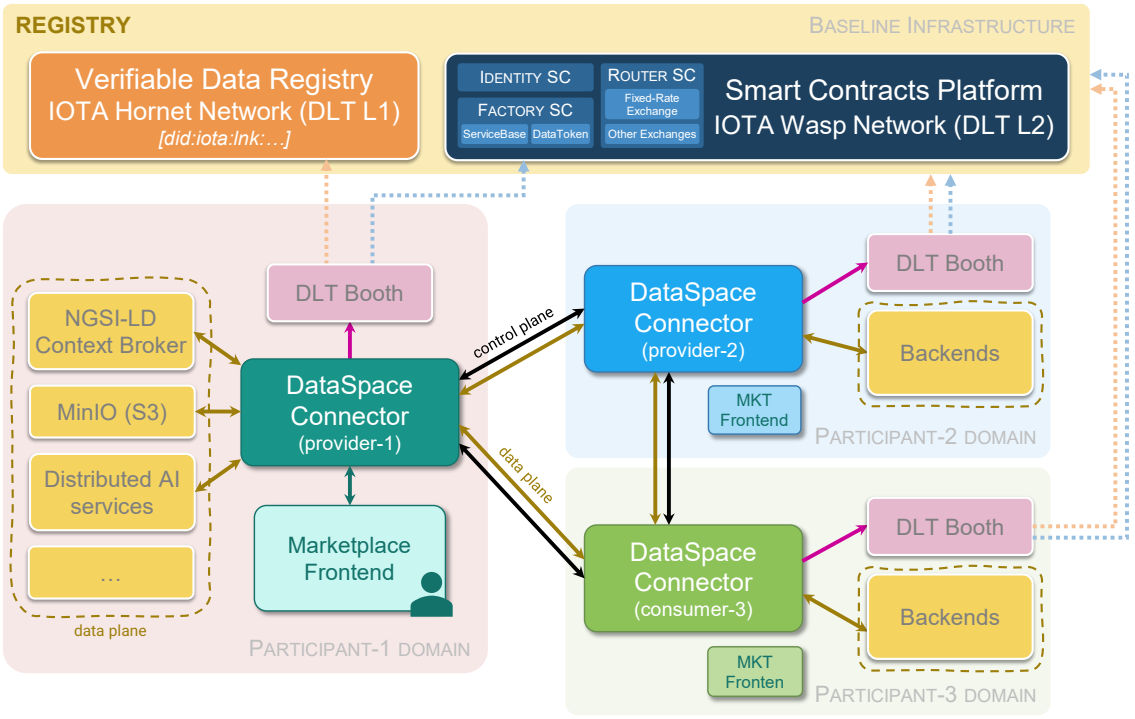

Figure 5: Architecture of offering negotiation, agreement tokenization and asset exchange

As mentioned, the DT acquired through this process acts as an on-chain, verifiable proof of purchase and is required to unlock access to the asset. This mechanism ensures that only Consumers who have successfully negotiated and paid for access can retrieve the asset. Additionally, it enables third-party Verifiers to confirm the existence and integrity of the agreement without accessing its full content, preserving confidentiality while ensuring accountability.

A more detailed explanation of the tokenization mechanisms is provided in Section 6.

Asset Exchange

Once the contract negotiation process has successfully concluded and a formal agreement is in place, the Consumer may initiate the asset exchange phase. This stage is also governed by the Dataspace Protocol, which defines the message flows and verification mechanisms required to securely transfer the agreed digital asset. While the core logic of the exchange is protocol-driven, its concrete implementation depends on the underlying data plane technology. Therefore, this section focuses on the generic security features integrated into the proposed framework to ensure robustness and compliance.

To begin the DSP Transfer Process FSM (see Figure 11b in Section 4.1), the Consumer issues a TransferRequestMessage that includes the identifier of the established agreement along with a Verifiable Presentation (VP). This VP encapsulates the credential issued during the onboarding phase and indirectly confirms the Consumer's right to access the asset. This entitlement is inferred from the existence of a valid Data Token (DT) linked to the NFT representing the offering and held by the Consumer.

The credential itself does not contain explicit proof of DT ownership. Instead, ownership is verified through the cryptographic relationship between the VP and the Consumer's Decentralized Identifier (DID). The Verifier component resolves the DID and confirms that the VP was issued by its legitimate holder, thereby validating DT possession without exposing

sensitive token data. The DID also embeds the Consumer's public EVM address, enabling on-chain verification of token ownership.

Access to the asset is not solely determined by DT possession. It is also subject to policy-based controls defined at the offering level. These controls are expressed using ODRL policies, which specify permissions, constraints, and obligations associated with asset usage. The evaluation of these policies is performed by Policy Decision Points (PDPs), with the Verifier acting as a specialized PDP optimized for on-chain validation. The PDPs assess the VP and its embedded claims (cryptographically signed during onboarding stage) against the defined policy rules. These may include conditions such as the Consumer's role, intended use, or time-based restrictions. DT ownership itself is also modelled as an ODRL constraint, allowing it to be seamlessly integrated into the broader access control logic. More details regarding available ODRL policies are discussed in Section 5.4.

Once the PDP returns a positive authorization decision, a Policy Enforcement Point (PEP) ensures that access is granted in accordance with the evaluated policy. If authorization is successful, the Transfer Process FSM transitions from the "requested" to the "started" state. Otherwise, it moves to the "terminated" state, and the exchange is aborted.

The actual delivery of the asset is performed through a secure data transfer mechanism that complies with DSP specifications. This mechanism typically relies on a pre-established data plane between Connectors, tailored to the specific technology stack in use. Within this framework, DataSpace Connectors act as PEPs, enforcing access decisions and ensuring that the asset is transmitted strictly under the agreed conditions. To protect confidentiality and integrity, the data transfer process may incorporate additional safeguards such as encryption, integrity checks, and secure session protocols.

3 Core Technologies for Decentralization

The following subsections outline the background of the different technologies that serve as foundation from the logic point-of-view for the decentralized infrastructure of the SEDIMARK Marketplace.

The initial two subsections describe the working principles of IOTA DLT and IOTA Smart Contract Chain, which together form the SEDIMARK Registry. In this final version of the technical deliverable for the data marketplace, the usage of IOTA's decentralized infrastructure is confirmed. It introduces several key improvements while maintaining the existing architecture based on the Tangle and Smart Contract Framework of the previous version of the deliverable. The primary enhancements focus on the physical infrastructure, consolidating multiple Docker Compose files into a single, configurable file that allows for the seamless activation of various services and ease of deployment. Additionally, an extra server has been integrated into the decentralized infrastructure, provided by a project partner, to further enhance performance and scalability. Additional information related to the final version of the implementation supporting the decentralized infrastructure are reported in SEDIMARK_D3.4 [25].

After that, basic principles of the Dataspace Protocol and Distributed Triple Stores, which enable Connectors and the SEDIMARK Catalogue, are also presented.

3.1 IOTA DLT

The IOTA DLT overcomes known scalability bottlenecks in its distributed ledger by using a Directed Acyclic Graph (DAG) [6]. The DAG structure is used in both the ledger and the consensus layers. Therefore, the IOTA DLT can be conceptually divided into two components as defined in [3]:

IOTA Ledger: is the ledger technology that is based on the principle of unspent transactions, named UTXO (Unspent Transaction Output). In this context, an unspent transaction output refers to cryptocurrency that remains unused or is left over after a transaction has been completed. Any output that is left and is not spent immediately is an Unspent Transaction Output that can be later spent (i.e., used as an input) in a future transaction. The adoption of UTXO model is the key condition for enhancing scalability and throughput within IOTA. It makes possible to have many parallel writes onto the ledger [3], increasing the system's efficiency. In fact, enabling numerous transactions to be processed simultaneously, the UTXO ledger can supports a high volume of transactions without the bottlenecks typically associated with traditional blockchain architectures. Additionally, since each output must be referenced in future transactions, it creates a clear and traceable history of all transactions, which helps prevent issues such as double-spending. This design choice not only optimizes transaction processing but also promote a robust framework for the secure and decentralized data marketplace, ensuring that users can confidently perform transactions with transparency.

IOTA Tangle: the Tangle is a permissionless and feeless consensus protocol based on a Directed Acyclic Graph (DAG). This DAG structure itself is called Tangle [6]. This data structure is replicated across a decentralized network of computers, known as nodes. When a node initiates a transaction, it must also validate and approve previous transactions, thereby enhancing the overall security of the network. Each node is responsible for ensuring that the transactions it approves do not conflict with the existing history in the Tangle. If a node detects a conflict with the Tangle's history, it will reject the conflicting transaction.

As a transaction accumulates more approvals from other nodes, its acceptance within the system increases, leading to a higher level of confidence in its validity. This mechanism makes it increasingly difficult for double-spending transactions to be accepted.

Consequently, the IOTA Tangle facilitates a global consensus on the state of the UTXO ledger without necessitating a linear ordering of transactions. This approach enhances the scalability and also allows for a high throughput, making it an appropriate infrastructure for a decentralized data marketplace.

3.2 IOTA Smart Contract (ISC) chain

The IOTA Smart Contract (ISC) chain serves as an additional framework that enhances the Layer 1 capabilities of the IOTA DLT. It introduces multiple programmable ledgers as Layer 2, creating a multi-chain ecosystem where each chain anchors its state to the IOTA Ledger at L1. This architecture allows each ISC chain to function as an independent blockchain, offering functionality similar to that of Ethereum smart contracts. As shown in Figure 6, the anchoring mechanism between the two layers ensures the integration of L1 and L2 layers and state consistency between the chain and the Tangle.

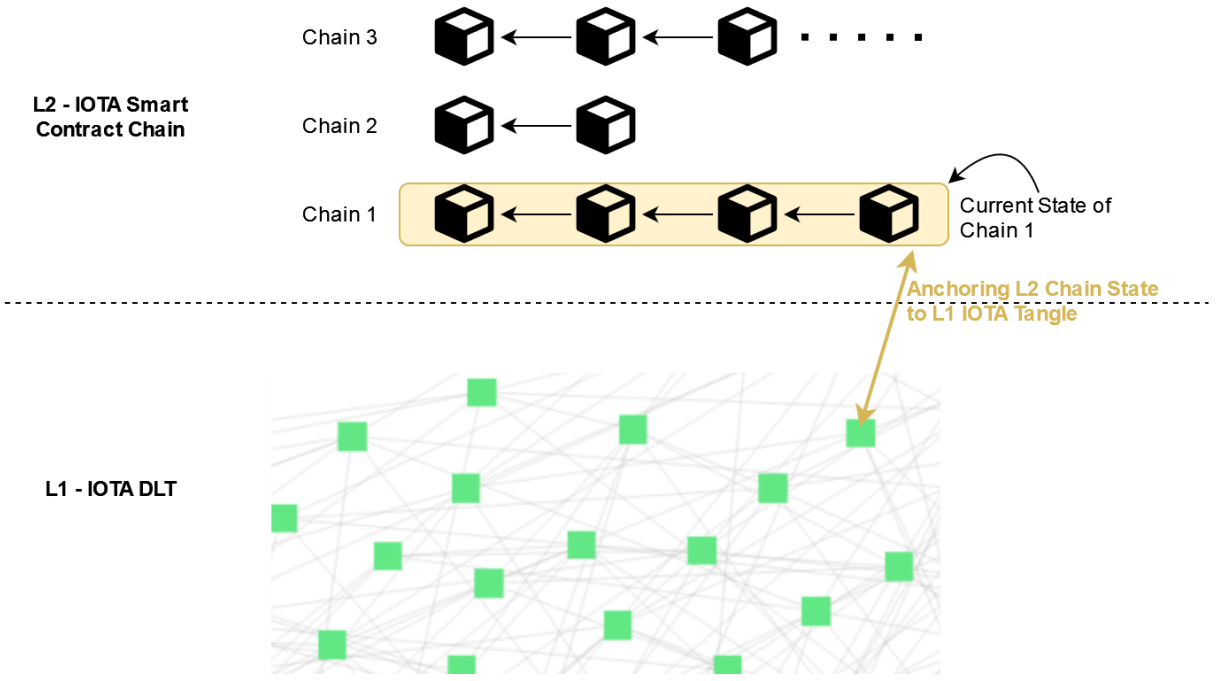

Figure 6: Anchoring between the DLT layers

Validator nodes play a crucial role in securing the ISC chain by participating in a distributed consensus process alongside other validators. The implementation of these validator nodes is facilitated through a software application named IOTA Wasp (IOTA's WebAssembly Smart Contracts Platform). Each ISC chain operates under the “governance” of a group of validator nodes, collectively referred to as the "Committee of Validators". This committee is responsible for managing the associated L1 account and executing a consensus protocol for state updates and block validations [3].

The design of the ISC chain ensures fault tolerance and distributed computing capabilities, requiring only [2N/3] + 1 non-faulty validators to achieve valid state updates. This resilience is

a key feature of the ISC framework. Validator nodes employ threshold signatures to control the L1 account, providing proof of consensus among the committee members [3].

Each validator holds a part of the private key, which is used to generate a partial signature for transactions. To create a valid threshold signature, a minimum set validator node must collaborate (at least of [2N/3] + 1, where N is the number of nodes), with the process of cooperation and signature aggregation embedded into the consensus mechanism. Such key is generated during the deployment of the chain, through a Distributed Key Generation (DKG) process, allowing each validator to securely generate and retain its private key share. The DKG mechanism together with the committee of validators ensure that the ISC chain operates with a high level of security and integrity [3].

3.3 Dataspace Protocol

According to the International Data Spaces Association (IDSA) [17], the Dataspace Protocol outlines a collection of specifications aimed at enabling interoperable data sharing between entities, all governed by usage control principles and standard Web technologies. These specifications define how data is published, usage agreements are negotiated, and data is accessed within a federation of interconnected systems known as a dataspace.

The protocol serves as the technical blueprint for the International Data Spaces Reference Architecture Model (IDS-RAM), detailing how components interact within the system. Its current version addresses four main thematic areas, each described in separate documents, which together establish the core interactions among participants in a dataspace:

- Data Space Model and Terminology: Establishes common ontologies and taxonomies to ensure interoperability between different participants.
- Catalogue Protocol: Describes the mechanisms for publishing and accessing datasets using shared interfaces. It adopts the Data Catalogue Vocabulary (DCAT) ontology for dataset representation.
- Contract Negotiation Protocol: Specifies the procedures for negotiating agreements between participants, ensuring mutual consent on data usage terms, modelled using the Open Digital Rights Language (ODRL) ontology.
- Transfer Process Protocol: Details the process of exchanging data post-agreement. Unlike earlier IDS-G specifications, it doesn't define specific transfer protocols but rather focuses on managing the transfer process itself.

At the heart of these interactions is the Connector, a central component that provides secure interfaces across all protocols. The architecture also distinguishes between two operational planes: the control plane, governed by the Dataspace Protocol Specification, and the data plane, which handles the actual exchange of data.

Within the SEDIMARK project, we integrate IOTA's Distributed Ledger Technology (DLT) and ISC chain with data space Connectors to enhance the functionality of the Catalogue and Contract Negotiation protocols. This integration adds trust and transparency to these operations.

3.4 Distributed Triple Stores Protocol

A Distributed Triple Store (DTS) is a means of storing and retrieving triple statements modelled using RDF (Resource Description Framework) in a distributed manner. The RDF model is based on the concept of triple statements, which consist of a subject, predicate, and object,

and forms the basic data entities in the Semantic Web, enabling the representation of information in a machine-readable manner. This also enables the linked data paradigm, whereby DTSs are linked through shared ontological concepts, towards co-creating knowledge graphs that provide more context relating to a particular entity.

In a centralized triple store, all data management operations occur in a single instance, which typically introduces resource and performance bottlenecks as the triple count significantly increases. To overcome this problem, distributed triple stores employ a decentralized architecture where triples are maintained by individual stakeholders across different locations. This approach potentially allows for more efficient storage and retrieval of triples.

The main query language for the DTS, is the SPARQL Protocol and RDF Query Language (SPARQL). It allows for federated queries whereby a query can be executed on multiple SPARQL endpoints as well as the local endpoint, and in turn merge all results from different remote sources. It is important to note that this feature is employed in a manner whereby the DTS can overcome performance issues related to:

- Network latency
- Bandwidth consumption
- Dynamic availability and varying reliability of remote endpoints
- Restrictions imposed by endpoints
- Handling of complex queries

A set of optimisation techniques need to be applied to address these issues and handle the recovery of the Catalogue as soon as possible when a remote node(s) become unavailable or unreliable.

For implementation, a number of tools can be adopted for building that DTSs, such as the open-source Jena TDB (Triple Database Benchmark) by Apache Jena, Sesame, and Virtuoso. For SEDIMARK, Jena TDB is used as the database technology for the triple store, and is accessed through a customised version of the Jena Fuseki server which provides the standard interfaces for CRUD (Create, Read, Update, Delete) via the REST-based Graph Store Protocol (GSP) and graph querying via the REST-based SPARQL endpoint.

A DTS will mainly be used to store instances of a particular information entity. In the case of the Catalogue, the Offering Description is the entity, which is based on the SEDIMARK Information Model. As Offering Descriptions are published to the Catalogue, each Offering is represented as a separate named graph. It is done in this manner to facilitate the complete removal of Offering Descriptions once they have expired or become void.

3.4.1 Decentralised Offering Management

To enable the distribution of Offerings among different nodes, a coordination module called the Catalogue Coordinator is required to retrieve Offering Descriptions from all Participants Self-Listing and then distribute the Offering Descriptions.

The Coordinator is a core component of the decentralised marketplace infrastructure that orchestrates the discovery, retrieval, validation, storage, and serving of offerings provided by participants. It interacts with the DLT Booth to fetch offerings from the underlying Distributed Ledger Technology (DLT) and manages a catalogue of offerings that can operate in either a centralised or decentralised mode.

The Coordinator also functions as a server responding to consumer search queries, effectively acting as a mediator between participants and the global or distributed catalogue.

Key responsibilities of the Coordinator include:

1. Polling the DLT for updated offerings.
2. Fetching offering metadata and full descriptions.
3. Validating and storing offerings.
4. Maintaining a local or distributed catalogue.
5. Serving consumer queries.
6. Monitoring node health and redistributing data as needed.

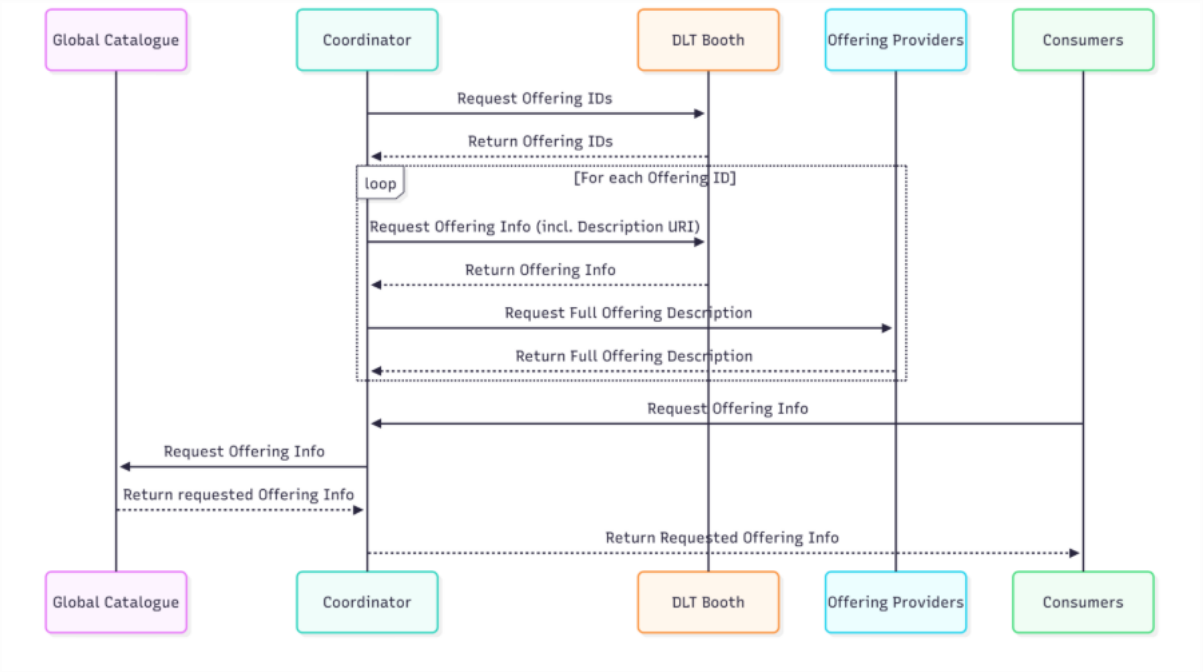

Figure 7: Interaction Sequence for Centralised Mode

In centralised mode (Figure 7), all offerings fetched from the DLT are stored locally in a single Global Catalogue. For Query resolution, Consumer queries are resolved directly from the Global Catalogue. The advantage of this mode is Simpler implementation, easier monitoring. The limitations are that it can be a single point of failure, and a generally lower resilience.

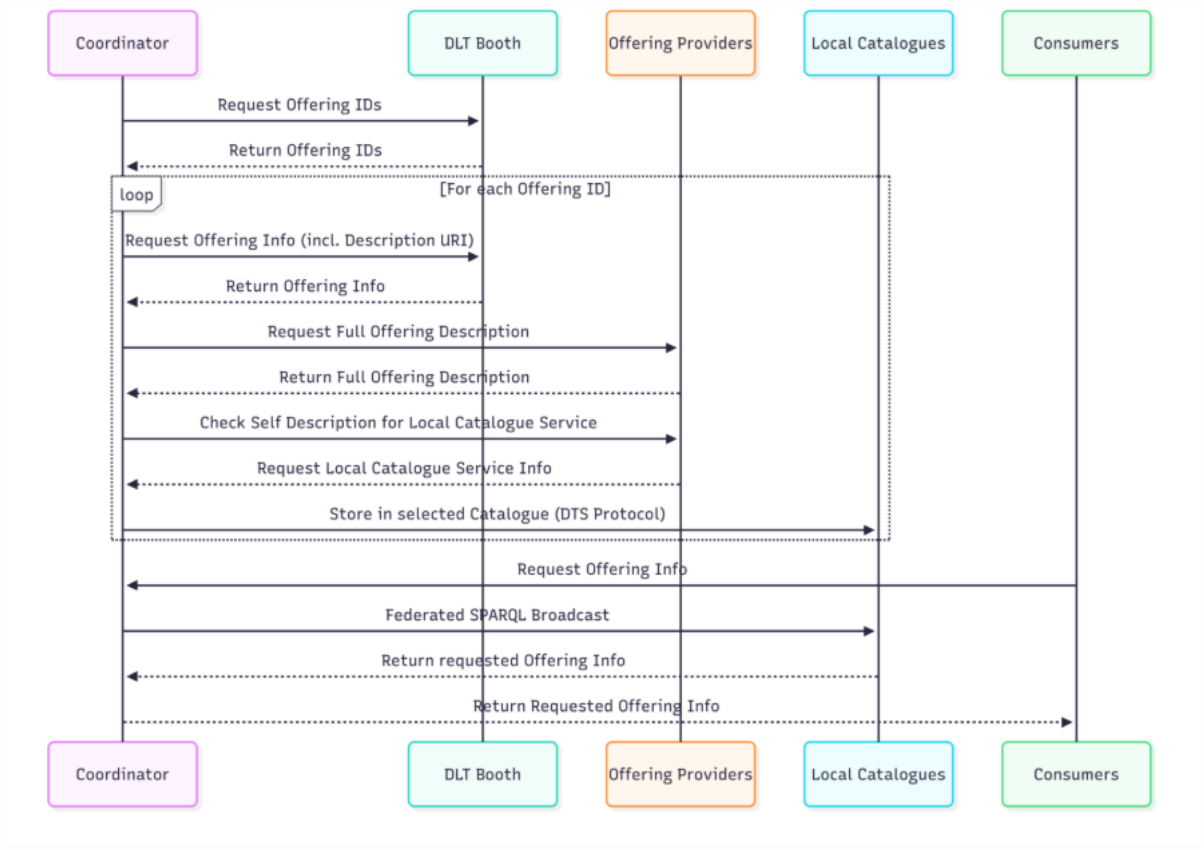

Figure 8: Interaction Sequence for Decentralised Mode

In decentralised mode (Figure 8), Offerings are distributed among nodes that provide a catalogue service. The Provider Self-Description endpoint is used to identify nodes offering catalogue functionality. The Coordinator maintains node list in RedisDB. For query resolution, Consumer queries are resolved across the distributed nodes. The Advantages of this mode is High availability, resiliency and improved scalability. The limitations would be a more complex implementation and the needs for a node health monitoring.

The method applied to distribute the Offering Description graphs data across multiple nodes efficiently is based on consistent hashing. It supports scalable storage, dynamic node management, and flexible querying while minimizing redistribution.

Figure 9 illustrates the distribution mechanism for the triple store handled by the Coordinator, which undergoes the following steps:

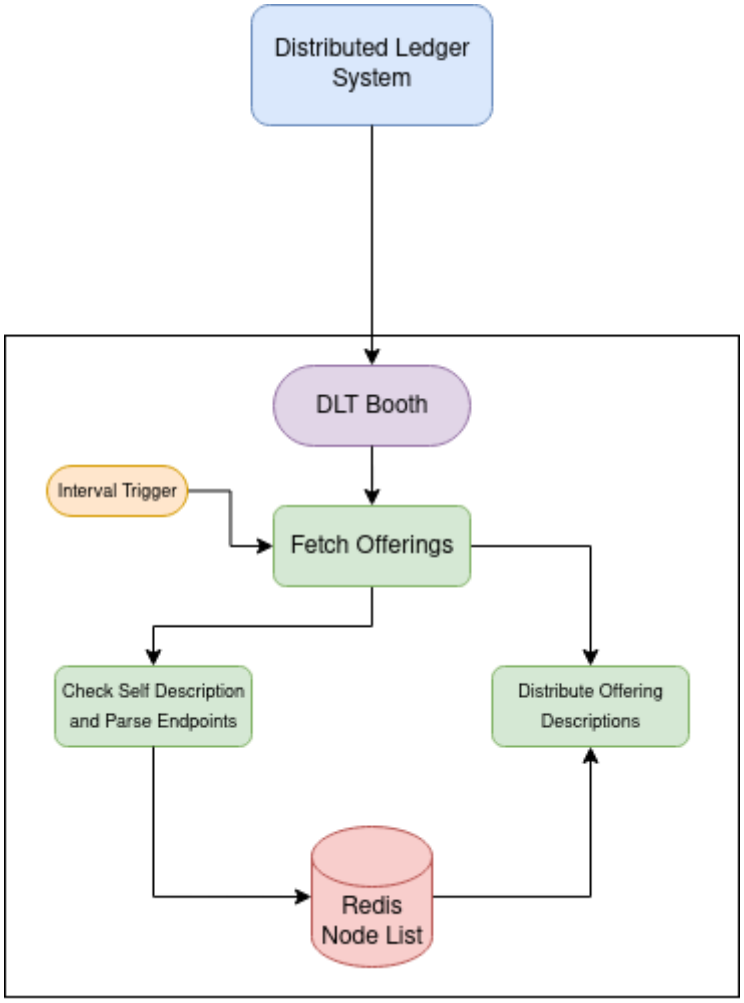

Figure 9: Triple Store Distribution Mechanism

1. The Coordinator triggers a timely function call to fetch the offering IDs from the DLT using DLT-Booth.
2. The list of offering IDs is compared with the cached ones to determine new offerings registered with the DLT.
3. The Coordinator then fetches offering metadata (including the offering description URI) for each new offering from the DLT.
4. The flow bifurcates into two processes here. Process 4a describes about maintaining a list of nodes for offering distribution in decentralised manner; Process 4b explains the distribution of newly retrieves offerings.
   a) The domain address from the new offerings metadata (using OfferingURI) is extracted to identify if the entity hosts a catalogue service using its Self-Description endpoint. If it does, the catalogue endpoint is then appended to the Redis DB consisting of a list of nodes.
   b) The full offering descriptions are retrieved from the participants' Self Listing endpoint (OfferingURI) and posted to selected catalogues retrieved from the Redis DB. The selection criterion implemented is Consistent Hashing—a technique that ensures the data is distributed equally and minimal data redistribution takes place in case of node failures.

3.4.2 Decentralised Query Resolution

In a decentralised setting, a query is sent to the endpoint hosting the Global Catalogue. The queries are processed based on their type, either offering-specific or theme-based properties. It will then modify the query into a federated query, which will then allow the SPARQL Query resolver to target other DTS nodes that have the relevant Offerings stored. Figure 10 illustrates this process of query resolution, which can be summarized in three steps:

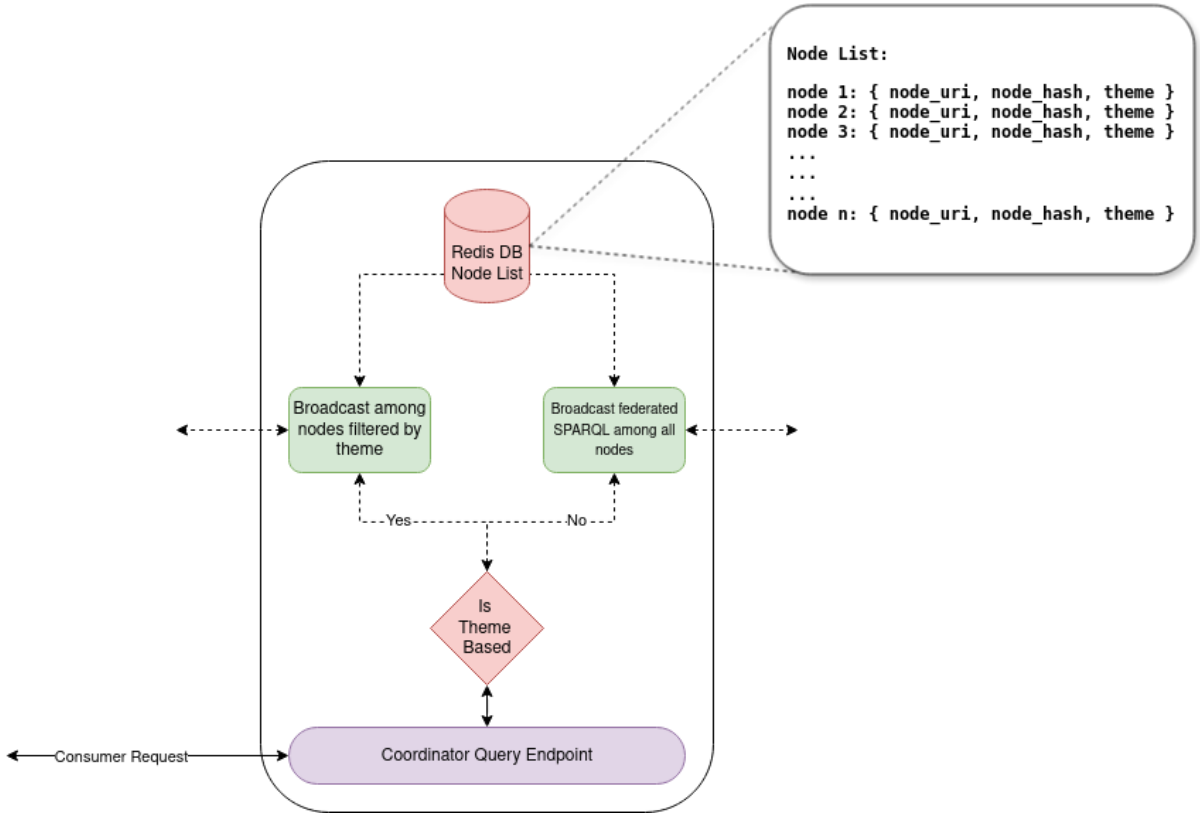

Figure 10: Decentralised Query Resolution

1. **Query Based on Offering ID**: The Coordinator determines the node the relevant offering is hosted on using consistent hashing. The SPARQL query is simply forwarded to the node and the offering description is retrieved.
2. **Query Based on Themes**: If the consumer query references a general theme, the nodes corresponding to the theme are filtered using reverse indexing sets in Redis. The query is then broadcasted, and all the relevant offering descriptions are retrieved and returned to the consumer.
3. **Fallback Query Execution**: If the query does not match any offering ID or theme, it is broadcast to all nodes. The query is adjusted accordingly to include each node in the execution plan.

3.4.3 Node Dynamics and Resilience

The Coordinator monitors the health of distributed nodes. If a node becomes unavailable, Offerings stored on that node are redistributed to active nodes. This ensures Catalogue availability and consistency. It applies the consistent hashing method specified in section 3.4.1 to minimise disruption and maintain data distribution.

The advantage of Consistent Hashing is that when a node is added or removed, consistent hashing ensures that only a small subset of the data needs to be redistributed. Most of the data remains unaffected, significantly reducing operational overhead.

For addressing node failure, a Redis assisted redistribution approach is adopted whereby Redis stores a backup of the node-to-service offering mappings. Upon node failure, this mapping helps reassign only the affected offerings to new nodes without scanning the entire dataset. This approach ensures a fault-tolerant, scalable RDF storage and querying system, capable of handling dynamic infrastructure changes with minimal disruption.

4 Implementation Perspectives

In this section the implementation aspects of the developments in the context of the decentralized infrastructure are presented. This section presents for each of the developed modules an overview from the technical point-of-view of the inner workings and the functionalities provided. The aim is to highlight the key components that have been developed in the course of the project, highlighting their role in the SEDIMARK Marketplace and how they contribute to the overall objective of the secure and decentralized data marketplace.

4.1 Connector

Connectors facilitate secure peer-to-peer information exchange between participants in the Marketplace. In this sense, the processes of offering tokenization, contract negotiation, and asset exchange are primarily orchestrated by Connectors, which act as the participant's main interface to the data space environment. SEDIMARK Connectors are based on the Eclipse Dataspace Components (EDC) one [18].

Each participant operates at least one Connector within their own domain, serving as the primary gateway to the distributed SEDIMARK marketplace. The Connector enables secure, policy-compliant peer-to-peer interactions and asset transfers, functioning as a control plane for contract execution and ensuring that asset access requests are handled securely and in accordance with agreed terms. It supports Dataspace Protocol-based communication between Connectors and integrates with the DLT-Booth to delegate cryptographic operations and interact with the underlying smart contract infrastructure. Concerning the DLT plane, the IOTA software components available through the DLT-Booth enable interactions with both L1 and L2 to issue data and value transactions, handling SSI-based identity and interact with the smart contracts in a seamless manner.

Beyond its role in negotiation and exchange, the Connector also incorporates internal components that manage the lifecycle of data offerings. Among these, the Offering Manager plays a central role in coordinating the creation, registration, and tokenization of offerings. It exposes a Self-Listing interface through which other participants can access the published offerings of a given Provider. This component interacts with the DLT-Booth API to carry out tokenization and to register offerings in the Registry. It also acts as a bridge between the participant-facing configuration interface of the Marketplace Frontend and the internal mechanisms of the Connector.

The Offering Manager validates the offering format using the tools provided by the Interoperability Enabler based on the Marketplace Information Model. Upon correct validation, offering is locally stored and appended to the connector's offering Self-listing, making it individually addressable through the connector via a secured REST API. After that, the Connector interacts with the smart contracts to register the offerings in the form of NFT addressed in Section 6. The NFT contains a pointer to the full offering in the Self-listing for performance reasons. This way, offerings are registered in the Registry and access to their full content is distributed across the different provider's connectors. Anyone can use the Registry to retrieve the existing offerings and query/crawl the corresponding offering self-listings in a trustworthy manner. In parallel, the connector executes various procedures adhering to IDS Catalogue protocol principles. This enables interoperability with other connectors, allowing for the exchange of offerings and asset descriptions in an IDS context.

Through this integration, the Offering Manager ensures that assets are properly registered within the data space and establishes the necessary bindings between the Connector and the internal data sources, as previously illustrated in Figure 5. This tight coupling guarantees that offerings are not only discoverable via the Registry but also technically accessible through the Self-Listing interface, while the actual data assets remain accessible through the Connector itself, all under the governance policies defined by the Provider. As a result, the offering lifecycle is fully aligned with the decentralised, interoperable principles that underpin the architecture of the proposed framework.

In addition, connectors are essential entities in achieving the capabilities of the data space enabler. They fulfil the mechanisms to ensure the secured access to all the assets described by an offering, bridging the gap between providers and consumers by enabling P2P access to a wide range of assets.

In this sense, and in an ideal marketplace, participants should be able to agree on access controls, pricing models, and licensing terms to protect their interests and ensure data privacy. Connectors lay the foundation for such functionalities by following IDS Contract Negotiation Protocol principles. Therefore, whenever a Consumer decides to acquire a previously discovered offering, its connector queries its counterpart for the specific offering. All these interactions are secured using the participant identity credentials. Provider's connector answers with the offering details, including details such as cost and access restriction policies. After that, a negotiation procedure is established and, once a mutual consensus is reached, both connectors proceed to sign an agreement. All these steps are part of a finite state machine defined by the Contract Negotiation Protocol and included, for reference, in Figure 11a.

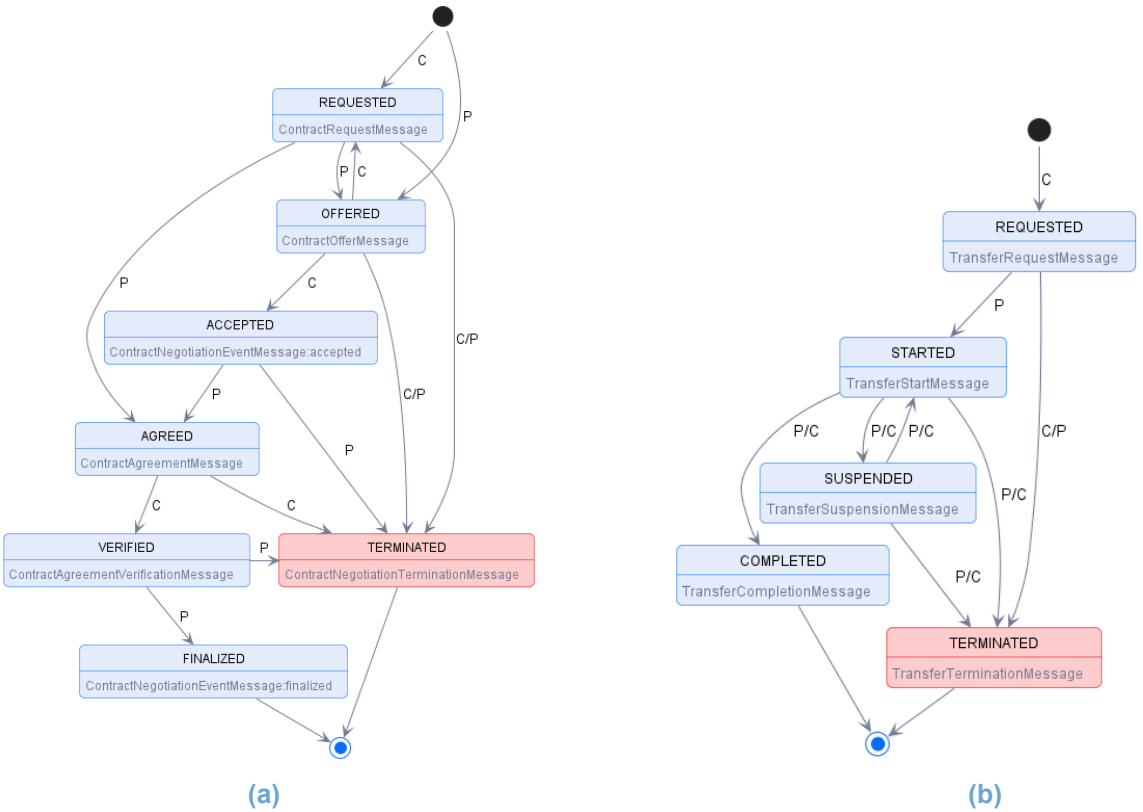

Figure 11: Contract Negotiation Protocol [19] (a) and Transfer Process Protocol [20] (b) FSMs

SEDIMARK connectors builds on top of that state machine in order to interact with the smart contracts on the ISC chain, as explained in Section 2.2. The final result is an agreement,

represented by a Fungible Token (FT) owned by the consumer, which is the basis for the subsequent secure exchange of assets between participants.

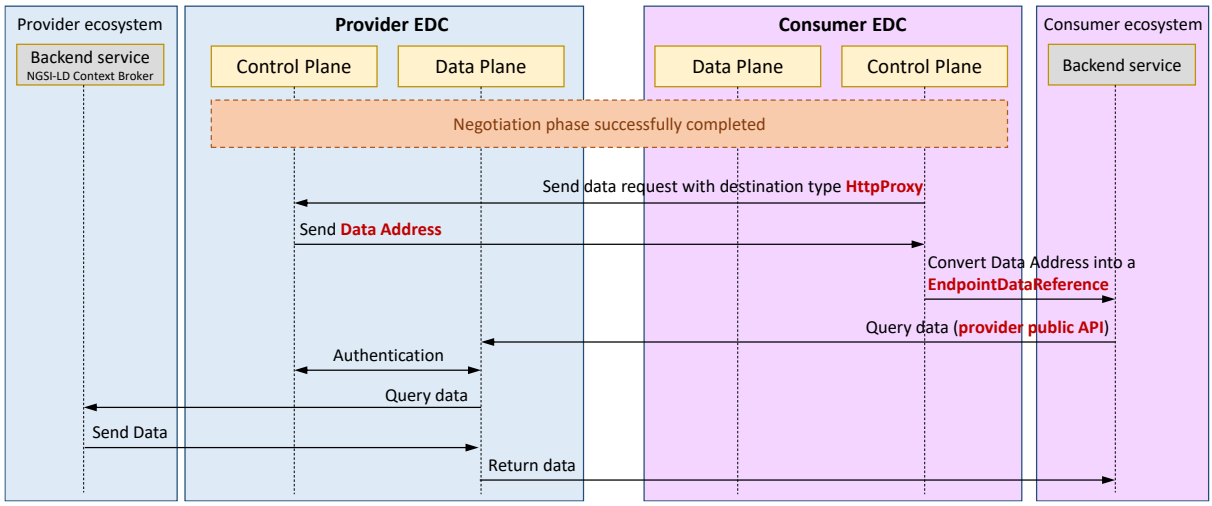

Figure 12: Detail of the IDS transfer data plane interactions between connectors

Finally, Connectors also participate in the asset transfer phase, which is governed by the IDS Transfer Control Protocol. The corresponding state machine that defines its operational flow is illustrated in Figure 11b. Figure 12 illustrates the interactions that occur between two connectors when accessing data stored in a context broker on the provider domain. After triggering the transfer procedure following the principles of the IDS Transfer Process Protocol, consumers request assets from their own domain using the information received from the control plane as well as credentials acquired during the offering negotiation phase. In the figure, the provider's connector acts as a proxy into the provider ecosystem, authenticating and authorizing it. From a security standpoint, this implies that it acts as both a Policy Decision Point (PDP) and Policy Enforcement Point (PEP). More details on the authorization process as well as the access policies are discussed in Section 5.4.

4.2 Catalogue

The Catalogue is a system entity that serves as the main means for Participants to search and discover Offerings made available in the marketplace. It adopts a DTS approach as described in Section 3.4 for the storage and retrieval of information in relation to Marketplace Participants and Offerings.

The Catalogue is packaged as part of the SEDIMARK Participant toolbox, and can operate in a centralised or decentralised manner. In the centralised configuration, the Catalogue is hosted within one Participant domain. Whereas in the decentralised configuration, the Catalogue is hosted among multiple Participant domains.

The Catalogue has two forms, Global and Local. The Global Catalogue acts as a main point of contact for all onboarded Participants to search and discover Offerings through a unified query interface based on the SPARQL language. The Global Catalogue relies on the Registry to populate the Catalogue based on the retrieval of verified Offering Descriptions originating from their respective Self-Listings at the Provider's domain. A Local Catalogue assists a Global Catalogue with storing a subset of all the Offerings registered at the Registry, and is employed when the Catalogue is configured as decentralised. Although all Catalogue queries would normally go through the Global Catalogue, the Local Catalogue can also be queried if its

endpoint is known. Via the DTS protocol, a Global Catalogue will federate the query to the Local Catalogues to resolve, compile and then return the compiled result to the Participant.

As a requirement for a Minimal Viable Marketplace (MVM) operation, and as part of the Baseline Infrastructure, a Global Catalogue must be provisioned by a Participant undertaking the role of a Marketplace Operator, which operates in a centralised manner. A Local Catalogue hosted on another Participant domain can be employed in the case where the Global Catalogue requires the distribution of Offering Descriptions, when configures as decentralised.

Based on the above, the Catalogue will operate mainly in either of several modes when started, which are Centralised, Decentralised using sharding among Provider nodes, and Decentralised using sharding among all nodes. This will be managed by the Catalogue Coordinator as specified in Section 3.4.1.

The different deployment modes that enabled within SEDIMARK are illustrated below.

Mode A: Centralised Catalogue

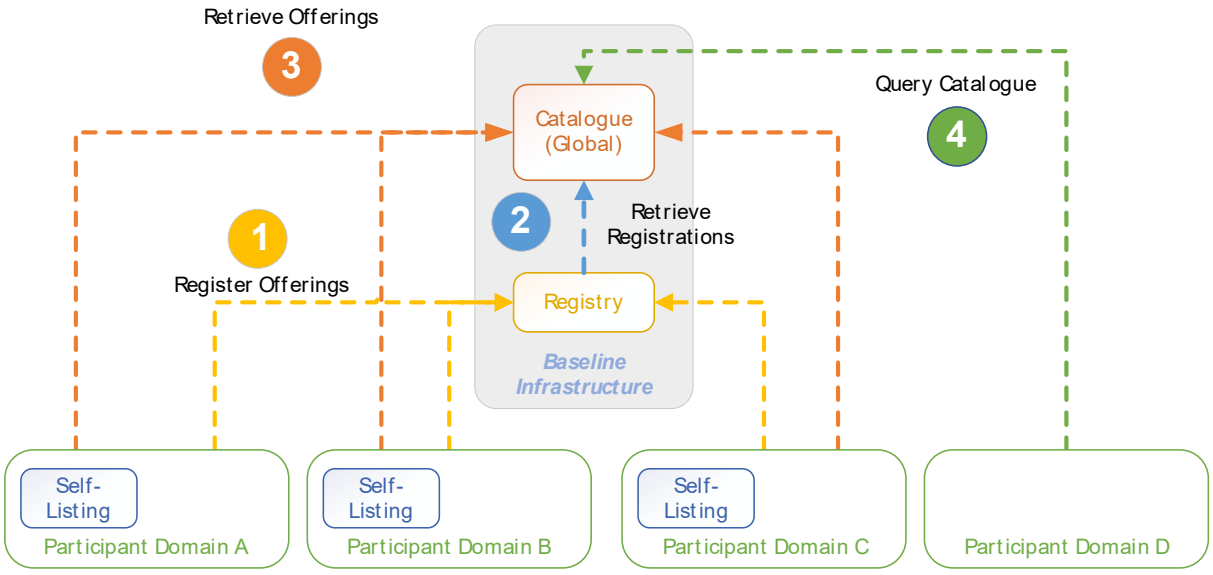

Figure 13: Centralised Catalogue Deployment

First, Self-Listings from Participants with Offerings (i.e. Providers) are registered with the DLT-based Registry. The Global Catalogue then retrieves all verified Offering entries from the Registry and then all descriptions of Offering entries from their corresponding Self-Listing. The Catalogue is then populated centrally. Finally, Participants query Global Catalogue for Offerings of interest.

Mode B: Decentralised Catalogue (BIF-Sharded)

The process of constructing the Catalogue follows the same steps (1-3 and 5) as in Mode A. In step 3, the entries of the Global Catalogue are then sharded into a number of subsets which are distributed either into other nodes of the baseline infrastructure. This helps to speed up the query responses in cases where the size of the entries becomes too large. In step 4, the Catalogue relies on DTSs distributed among other nodes within the Baseline Infrastructure, whereby each node holds a subset of Offerings from all Providers. The Global Catalogue here federates the query from a Participant to the Local Catalogues within the Baseline Infrastructure.

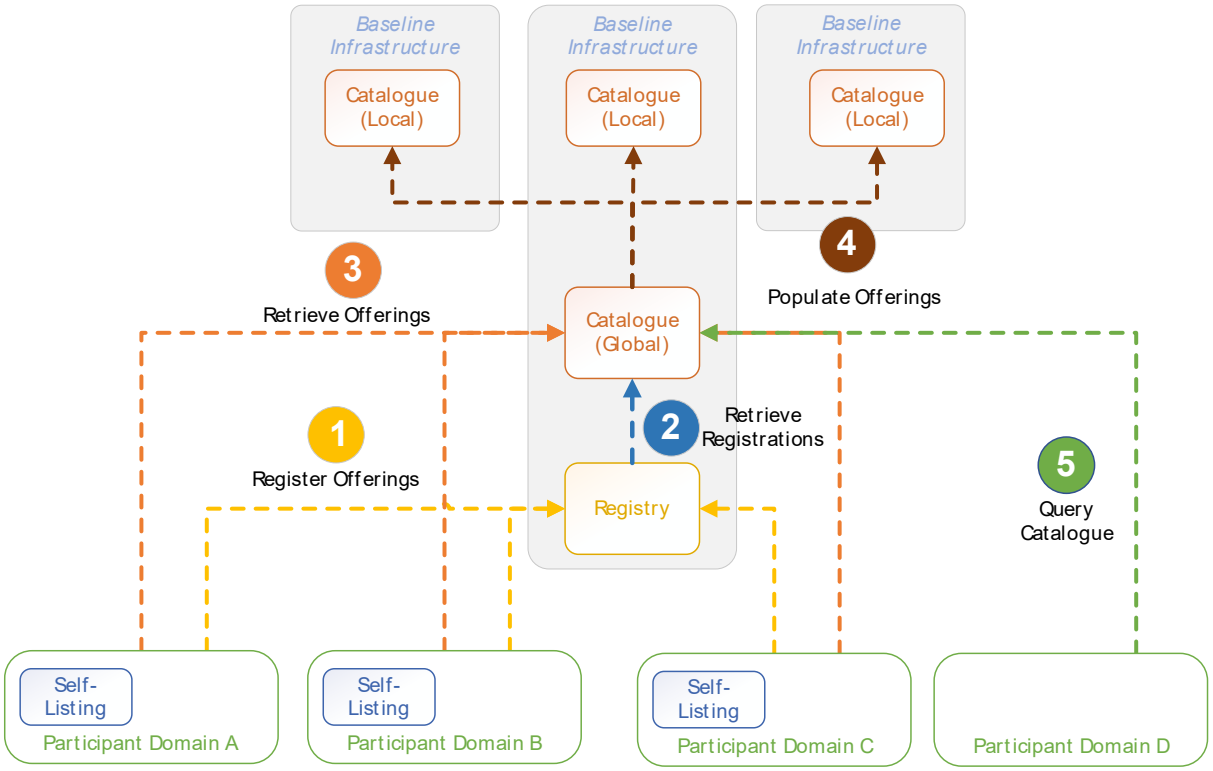

Figure 14: Decentralised Catalogue Deployment using BIF-sharded Local Catalogues

Mode C: Decentralised Catalogue (Provider-Federated)

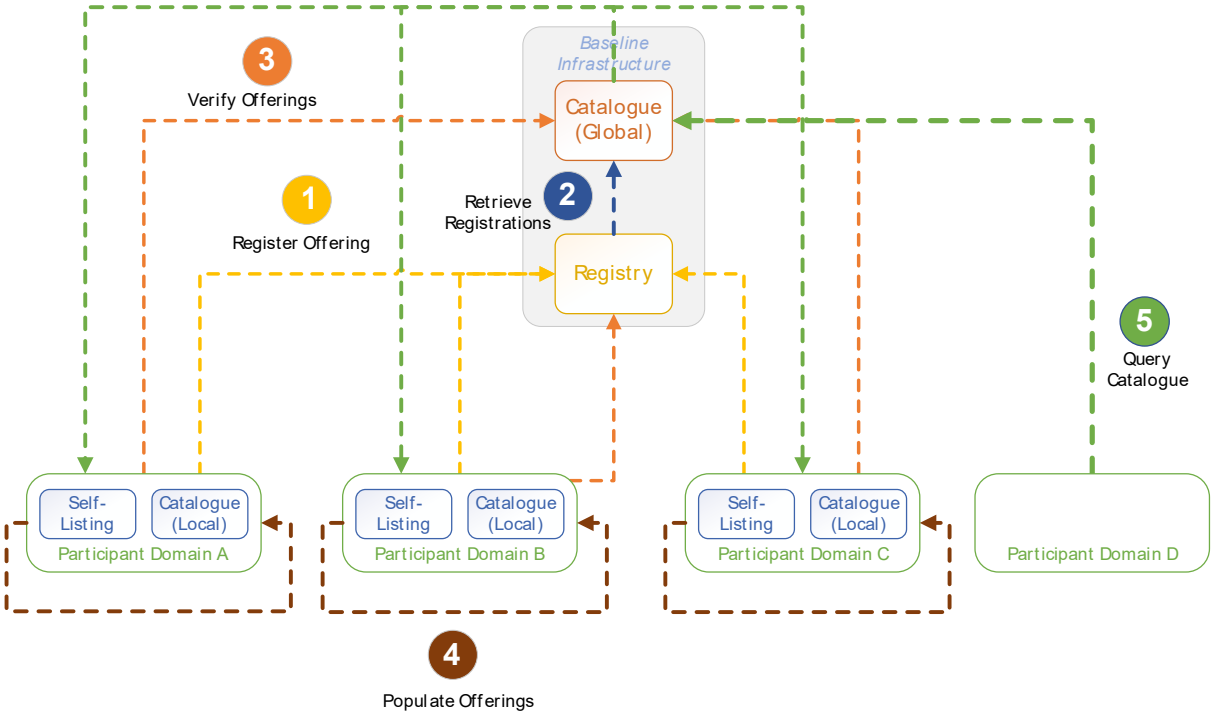

Figure 15: Decentralised Deployment using Federated Local Catalogues at Provider Domains

Here, the process follows as in Mode B, except that the Global Catalogue relays queries from Participants to relevant Local Catalogues hosted on Participant domains. Additionally in this mode, each Participants' Local Catalogue only stores information about their own Offerings. The Global Catalogue can also verify the Offerings being made available, before it makes use of the Local Catalogue. This involves comparing the Offering Descriptions in the Self-Listing, with the one's in the Local Catalogue.

Mode D: Decentralised Catalogue (Provider-Sharded)

Finally, this mode involves the Global Catalogue relying on Local Catalogues hosted at Provider Participant domains, where each holds a subset of all the Offerings Descriptions made available on the Marketplace. This is done so that each Provider shares in the load for storing Offering Descriptions, based on the DTS Distribution protocol specified in Section 3.4.1. This can be seen similar to Mode B, where in Step 3, the sharding and distribution of the Catalogue entries are not stored into the Local Catalogues of the Baseline Infrastructure, but in Local Catalogues hosted at the Providers.

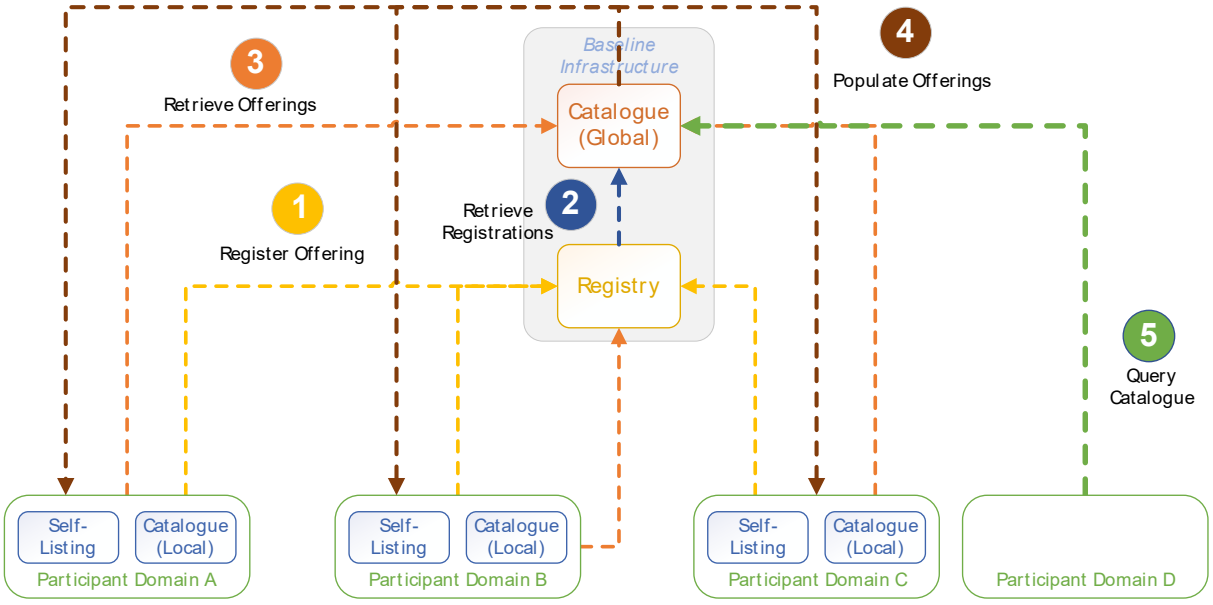

Figure 16: Decentralised Deployment using Local Catalogues at Provider Domains

4.3 Issuer

The Issuer of the SEDIMARK marketplace is the component responsible for issuing digital credentials to the various users. It is the main point of entrance for the users to the Marketplace, in fact it enables the onboarding procedure, which allows an entity to become part of the SEDIMARK Marketplace.

The issued credentials adhere to the principles of the Self-Sovereign Identity paradigm. In fact, the users retain full ownership of their identity. The primary functionality of the Issuer is to release Verifiable Credentials to the external users of the Marketplace. Additional details and examples related to the SSI paradigm are reported in Section 5.1.

The Issuer is a trusted component that provides assurance regarding the identity of other entities within the Marketplace. The credentials issued serve as proof of identity and authorization, enabling the secure interactions (i.e., data exchanges) in the SEDIMARK Marketplace.

The issuer registers a new participant and associates a credential that asserts the validity of the participant in the marketplace. Verifiers that trust the Issuer will accept those credentials in the authentication and authorization process of a participant.

The Issuer, together with the other entities in the SSI paradigm, facilitates the interoperability and collaboration among different Consumers and Providers resorting to the use of standardized credentials.

Through the DLT-Booth component, a user can create its own identity by interacting with the Issuer. The Issuer, in turn, generates a new Verifiable Credential for the user. The status of the credential is tracked in a Smart Contract controlled by the Issuer itself. Finally, the issuer returns the Verifiable Credentials to the user who has requested it. The SEDIMARK Issuer has its own identity; the key information for the overall marketplace is its public key. By design, the Issuer is reachable to a known predefined web address: https://issuer.stardust.linksfoundation.com/prod/ exposing reachable endpoints for the APIs.

From the technical perspective, the Issuer is a software service running as an HTTPS server. The secure TLS connection provides confidentiality and integrity to the communication with clients during the identity management phases.

The application has been developed in the course of the project in Rust language. The most updated and final version of the SEDIMARK Issuer source code can be found in the repository of the SEDIMARK project at https://github.com/Sedimark/sedimark-issuer/ .

In order to gather the required DID Document for the onboarding, the Issuer needs to be configured according to parameter related to the DLT infrastructure. The repository also provide the necessary examples.

First of all, the issuer needs to manage its own self sovereign identity, so it implements a module for self-sovereign identity management. It can interact with the DLT and control its own DID. In addition, the Issuer also interacts with the ISC platform to complete the registration process.

The Issuer exposes HTTPS endpoints for participants registration. When a client connects to the Issuer, it requests a nonce to solve a cryptographic challenge. This mechanism provides a proof of possession of client's DID and prevents replay attacks. Next, the client (DLT-Booth) completes the registration request providing some data (e.g., username) and the proof of control of the DID to the Issuer. If the DID is authenticated, the Issuer responds with a valid Verifiable Credential. From now on, the registered DID is a participant of the SEDIMARK Marketplace.

4.4 Verifier

In the SSI framework, the Verifier plays a crucial role in establishing trust within the digital identity of the entities. The Verifier is responsible for validating the claims made by a user through VCs and Verifiable Presentations (VPs). The Verifier ensures that the information presented by the user is accurate and trustworthy triggering its own verification mechanisms.

In the SEDIMARK Marketplace, the Verifier intervention is necessary when a Consumer is requesting the access to the asset. Obviously, the Producer has the interest to verify the user is legitimate and has correctly purchased the asset. In particular, the Verifier has two primary responsibilities:

Verification of Verifiable Presentations: the Verifier needs to validate the VPs submitted by Consumers. This process involves checking the authenticity of the user's claims to ensure that

they are indeed a registered participant in the marketplace. In this way, the Verifier ensures that only SEDIMARK-authenticated users can access the assets.

Verification of Proof of Purchase: additionally, the SEDIMARK Verifier is responsible for validating the Proof of Purchase associated with datatokens. This process involves accessing the user address and checking the balance of the datatokens related to the offering of the asset. In case the balance is positive, the user has effectively completed the purchase operation and therefore should gain access to the Asset.

Additional Policies: the Verifier has been designed to be flexible. Customized access policies can be easily defined and validated to per perform additional verifications before granting the access. Additional details on such policies can be found in Section 5.4.

This components is an http server that executes verification procedures for the Verifier in the SSI model. It is designed to be expanded and customizable for any use case. The Verifier is implemented as a service application and it is written in TypeScript. It can be instantiated anywhere since it relies on the decentralized infrastructure. Nevertheless, the design envisioned for SEDIMARK is to run on the Provider premises. This architecture allows for enhanced security and privacy, as sensitive verification processes occur locally (i.e., at the Provider) rather than relying on external servers.

The Verifier exposes two primary API endpoints:

- Verify VP: verifies the authenticity of the claims made in the VP against the corresponding VC (employing L1).
- Verify Proof-of-Purchase (PoP): It checks the user's ownership of the datatoken (resorting to appropriate SCs in L2).

Nevertheless, the Verifier can also be easily extended to support additional verification procedures to enforce the desired policies.

4.5 DLT-Booth

The DLT Booth is a crucial interface within the SEDIMARK Toolbox, designed to facilitate interactions with DLT for SSI-related operations and the Ethereum Virtual Machine (EVM) for Smart Contract transactions. This component integrated into the Connector component and it is specifically tailored to meet the technical needs of the SEDIMARK Marketplace.

It is an HTTP server, written using the Rust language. The DLT Booth is packaged as a Docker image, allowing for ease of deployment and scalability, and requires a connection to a PostgreSQL database for managing identity-related data and transaction records.

The primary functionalities of DLT Booth encompass both SSI and EVM operations. For SSI, it enables the creation and publication of Decentralized Identifier (DID) documents on the configured DLT, facilitating the establishment of unique identities. Users can also request the issuance and revocation of VCs from the designated Issuer, which is essential for onboarding to the SEDIMARK Marketplace. Additionally, it allows for the creation of VPs, enabling users to embed multiple VCs into a single, shareable document, therefore enhancing privacy and control over personal data (i.e., their identity).

On the EVM side, DLT Booth supports EIP191 [27] formatted message signatures, ensuring compatibility with Ethereum-based applications. It also allows users to execute transactions on the L2 of the decentralized infrastructure, facilitating interactions with smart contracts and the various marketplace operations tied to ISC framework.

To deploy the DLT-Booth, their users (either developer or participant to the Marketplace) must ensure that their Docker environment is properly set up and that the PostgreSQL database is configured and accessible. Once the HTTP server is started, the DLT-Booth can process incoming requests for both SSI and EVM operations at its endpoints.

The detailed list of APIs available, which are a fundamental part of this deliverable, are defined and detailed in the corresponding documentation for DLT-Booth. They are available both as OpenAPI specification and as an example ready to be executed by means of specialized software (e.g., Postman). The technical documentation containing the APIs' endpoints developed is available here: https://github.com/Sedimark/dlt-booth/blob/main/api/dlt_booth.yaml]. However, the functionalities provided through these APIs are as mentioned above and pertain to interactions with the DLT (L1) and ISC (L2) from an architectural infrastructure perspective, as well as onboarding, offering management, and purchasing from a functional point-of-view.

In summary, DLT-Booth serves as a core building block for the SEDIMARK Connector, providing essential functionalities for managing identity through SSI by interfacing with the DLT and enabling smart contract interactions by interfacing with ISC. Its design prioritizes performance, security, and ease of integration, enhancing its capabilities within the SEDIMARK domain.

4.6 Main Libraries

This section describes the main software libraries and components used to implement the marketplace mechanisms.

IOTA SDK [10]: The IOTA SDK (Software Development Kit) provides developers facilities to interact with IOTA DLT by providing account abstractions and clients to interact with node APIs. This is a Rust-based library that provides a convenient and efficient way to interact with nodes in the Shimmer and IOTA networks running the Stardust protocol. It consists of two main modules: *client* and *wallet*. The client module offers low-level functions that allow to have fine-grained control over the interactions with the distributed nodes. The module provides access to the underlying API endpoints and enables advanced operations such as custom message construction and direct communication with the network. The wallet module provides high-level functions for managing accounts, generating addresses, creating transactions, and interacting with the network. It offers a simple interface for developers to build applications network.

IOTA Identity [11] is a framework developed by the IOTA Foundation to provide a decentralized and self-sovereign identity solution. IOTA Identity is a decentralized identity (DID) method, identified along with the associated library. Identity library is a Rust implementation of SSI, using IOTA DLT at its core. It implements the World Wide Web Consortium (W3C) Decentralized Identifiers and W3C Verifiable Credentials specifications. This library can be used to create, resolve and authenticate digital identities and to create verifiable credentials and presentations in order to share information in a verifiable manner and establish trust in the digital world. The IOTA Identity framework implements the most common standards and patterns for Decentralized Identity. It is designed to work for Identity for People, Organizations, Things, and Objects acting as a unifying-layer of trust between everyone and everything. It supports the secure storage of cryptographic keys, which can be integrated into a key management system. IOTA Identity is written in Rust and has strong guarantees of memory safety and process integrity while maintaining exceptional performance.

actix-web [12] is a powerful, high-performance web framework for building web applications in the Rust programming language. It is part of the Actix ecosystem, which includes other

libraries for actor-based concurrency or database interactions. It is designed around asynchronous programming, taking advantage of Rust's asynchronous capabilities to handle a large number of concurrent connections efficiently. The library provides middleware support: middlewares can handle tasks like authentication and logging.

EDC Connector [18]: The EDC Connector is a component of the Eclipse Dataspace Components (EDC) project, an open-source project governed by the Eclipse Foundation. The project offers a full suite of components for implementing data spaces that comply with IDSA requirements on IDS protocol, with its reference architecture model rules and agreements as well as with the IDSA certification scheme. The EDC Connector is a Java- based software component designed for sovereign; inter-organizational data exchange based on IDS. The connector framework defines modules for performing data query, data exchange, policy enforcement, monitoring and auditing. SEDIMARK Connector is built taking the EDC Connector as a starting point.

koa [21] / Express [22]: Koa is an HTTP middleware framework for node.js, designed to enhance the writing experience of web applications and APIs. Koa's middleware stack flows in a stack-like manner, allowing it to perform actions downstream before filtering and modifying the upstream response. On the other hand, Express is a minimal and flexible Node.js web application framework that provides a robust set of features for web and mobile applications. Both frameworks serve a similar purpose but have subtle differences which make them more suitable for specific purposes.

Traefik Proxy [23]: Traefik is an open-source Edge Router that simplifies service publication in a microservices context. It is a dynamic, robust, and versatile reverse proxy and load balancer that has been designed with modern, distributed, and microservices architectures in mind. It receives requests on behalf of the system and identifies which components are responsible for handling them. Besides, it is natively compliant with most container and cluster technologies, such as Docker and Kubernetes.

Apache Jena [24]: an open-source Java programming framework for building applications using Semantic Web and Linked Data paradigms, through the provision of tools and libraries for storing models based on RDF and OWL graphs, and in turn provides a SPARQL query engine, ARQ2.

5 Digital Identity and Access Management

5.1 Background: Self-Sovereign Identity (SSI)

The Self-Sovereign Identity (SSI) [2] is a decentralized digital identity paradigm that gives a node full control over the data it uses to build and to prove its identity. The SSI stack enables a new model for trusted digital interactions in decentralized systems.

The SSI paradigm lays its foundations by means of any Distributed Ledger Technology (DLT) which acts as the Root-of-Trust (RoT) for identity public data. In fact, DLTs are distributed and immutable means of storage by design [5] .

A Decentralized IDentifier (DID) [8] is the new type of globally unique identifier designed to verify a node. The DID is a Uniform Resource Identifier (URI) of the following form:

did : method-name : method-specific-id

where **method-name** is the name of the DID Method used to interact with the DLT and **method-specific-id** is the pointer to the DID Document stored on the DLT.

As example, a DID for the SEDIMARK Marketplace is the following:

did:iota:lnk:0xc6092b44cd422fbfcda4eb86304428fbd4cc718fe4a1c3c92d8157e6588205c6

The DID Method is the software implementation used by a node to interact with the DLT. In accordance with W3C recommendation [8]. A DID Method must provide the primitives to:

- **Create a DID**: generate an identity keypair for authentication purposes, the corresponding DID Document containing the public key and store the DID Document in the DLT,
- **Resolve a DID**: retrieve the DID Document from the ledger pointed to by the DID,
- **Update a DID**: generate a new keypair and store a new DID Document at the same or at a new DID if the node requires changing the DID, and
- **Revoke a DID**: provide immutable evidence on the ledger that a DID has been revoked by the owner.

The DID Method implementation is in general ledger-specific and it makes the upper layers independent of the DLT of choice. In the final version of SEDIMARK, the DLT employed is IOTA, while the DLT Method is implemented with their IOTA Identity Library.

Another important element in SSI are the Verifiable Credentials (VCs) [9]. VCs build upon the DID foundation by providing secure, machine-verifiable digital credentials that convey additional attributes about a node's identity. Together, DIDs and VCs create a robust mechanism for authentication and authorization, enhancing trust in digital interactions.

The combination of the keypair, the DID, the corresponding DID Document and at least one VC forms the digital identity in the SSI framework.

This composition of the digital identity reflects the decentralized nature of SSI. There is no authority that provides all the components of the identity to a node, and no authority is able to revoke completely the identity of a node.

In SSI three roles are always mandatory and form the Triangle-of-Trust. The roles are:

1. **Holder** is the node that possesses one or more VCs and that generates a Verifiable Presentation (VP) to request a service or a resource from a Verifier;
2. **Issuer** is the node that asserts claims about a subject, creates a VC from these claims, and issues the VC to the Holders.

3. **Verifier** is the node that receives a VP from the Holder and verifies the two signatures made by the Issuer on the VC and by the Holder on the VP before granting or denying access to a service or a resource based on the claims.

The VC contains the metadata to describe properties of the credential, the DID and the claims about the identity of the node.

The Issuer signs the VC to make it an unforgeable and verifiable digital document. A Holder requests access to services and resources from the Verifier by presenting a VP. A VP is built as an envelope of the VC. The VC is issued by an Issuer and a signature is made by the Holder with his private key. Issuer and Verifier rely on an implicit trust (i.e., they trust each other).

Based on the following considerations, it is possible to build any ecosystem of trustable interactions among nodes. Additional explanation on the adopted model are available in SEDIMARK_D4.1 deliverable [26].

5.2 Identity Data Models – Final Version

The following subsections present the final version of the DID Document and the updated VC data model, both employed in the final version of SEDIMARK.

5.2.1 DID Document

Figure 17 shows the JSON format of the DID Document, please refer to [8] for a detailed description of all fields. Note that, the DID Document contains the DID of the subject, and its EdDSA public key.

```
{
    "id": "did:iota:lnk:0xbb6fcb4c81cbfb68f9503928a45869eafa354e98ab19586fd88c6c4017a61d48",
    "verificationMethod": [
        {
            "id":
"did:iota:lnk:0xbb6fcb4c81cbfb68f9503928a45869eafa354e98ab19586fd88c6c4017a61d48#We_trPiWFEFl9umAYFceWTUQXXUbiGYZ
IJXXPG95dQw",
            "controller": "did:iota:lnk:0xbb6fcb4c81cbfb68f9503928a45869eafa354e98ab19586fd88c6c4017a61d48",
            "type": "JsonWebKey2020",
            "publicKeyJwk": {
                "kty": "OKP",
                "alg": "EdDSA",
                "kid": "We_trPiWFEFl9umAYFceWTUQXXUbiGYZIJXXPG95dQw",
                "crv": "Ed25519",
                "x": "rUjUKC-laOrbuV-FePFXYjY5BPSR--45fGhdEvLDYSg"
            }
        },
        {
            "id": "did:iota:lnk:0xbb6fcb4c81cbfb68f9503928a45869eafa354e98ab19586fd88c6c4017a61d48#ethAddress",
            "controller": "did:iota:lnk:0xbb6fcb4c81cbfb68f9503928a45869eafa354e98ab19586fd88c6c4017a61d48",
            "type": "EcdsaSecp256k1RecoveryMethod2020",
            "blockchainAccountId": "eip155:1:0xc36271b4787b4ff54d4ac00c57d2d554ef991f81"
        }
    ],
    "service": [
        {
            "id": "did:iota:lnk:0xbb6fcb4c81cbfb68f9503928a45869eafa354e98ab19586fd88c6c4017a61d48#profile",
            "type": "ServiceType",
            "serviceEndpoint": "https://profile.sedimark.example.com/profile"
        },
        {
            "id": "did:iota:lnk:0xbb6fcb4c81cbfb68f9503928a45869eafa354e98ab19586fd88c6c4017a61d48#connector",
            "type": "ServiceType",
            "serviceEndpoint": "https://connector.sedimark.example.com/api/dsp"
        },
        {
            "id": "did:iota:lnk:0xbb6fcb4c81cbfb68f9503928a45869eafa354e98ab19586fd88c6c4017a61d48#self-listing",
            "type": "ServiceType",
            "serviceEndpoint": "https://offering-manager.sedimark.example.com/offerings"
        }
    ]
}
```


Figure 17: Example of DID Document

5.2.2 Verifiable Credentials

Figure 18 shows the generic VC and VP data models recommended by W3C [9]. The VC contains the Credential Metadata, the Claim(s) about the identity of the subject, and the Proof in the form of a signature of the Issuer. The VP is a wrap of the VC where the Proof coincides with the signature of the Holder of the VC.

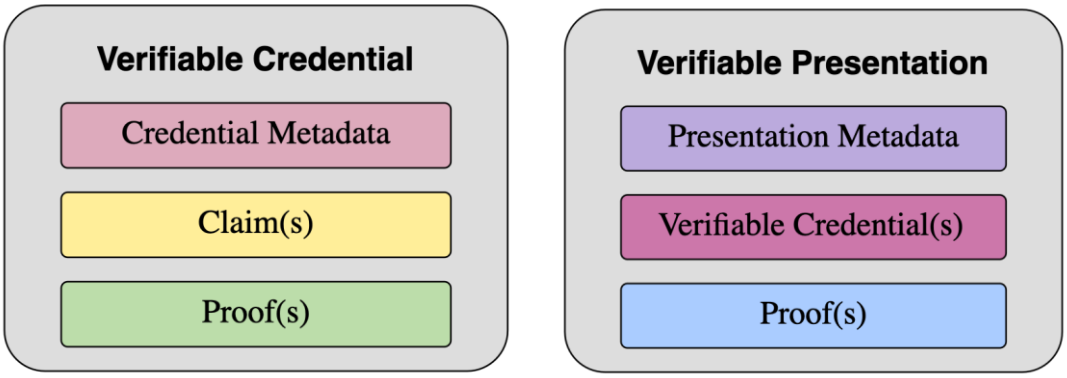

Figure 18: VC data model and VP data model

This is the final version of SEDIMARK specific JSON format VC, according to [9] (refer to it for the description of all fields). Note that, the *CredentialSubject* field contains the DID and claim(s) that describe the identity of the subject.

---------- JWT header ----------


```
{
  "kid":
"did:iota:lnk:0x2ab8359cab2468ba2f4217703503a9bde86c4cf85ee57eb6093ed5c2072c87a7
#lY2duRVYU6SHPBI4kxVtP25809k4e4HcSCVQi8y3LXA",
  "typ": "JWT",
  "alg": "EdDSA"
}
```


---------- JWT payload ----------


```
{
  "exp": 1784719076,
  "iss":
"did:iota:lnk:0x2ab8359cab2468ba2f4217703503a9bde86c4cf85ee57eb6093ed5c2072c87a7
",
  "nbf": 1753096676,
  "jti": "https://issuer.stardust.linksfoundation.com/api/credentials/2",
  "sub":
"did:iota:lnk:0x8ec941e225087c452b9548a032b1e16de71b6775f435c38b1ab485d8f8cdcdbc
",
  "vc": {
    "@context": [
      "https://www.w3.org/2018/credentials/v1",
      {
        "schema": "http://schema.org/"
      }
    ],
    "type": [
      "VerifiableCredential",
      "MarketplaceCredential"
    ],
    "credentialSubject": {
      "schema:alternateName": "someUsername",
      "schema:memberOf": "SEDIMARK marketplace"
    }
  }
```

}

---------- JWT proof ----------

<The Issuer's signature>

This is the final version of SEDIMARK VP JWT (JSON Web Token) encoded, please refer to [9] for the description of all fields.

---------- JWT header ----------

{
"kid":
"did:iota:lnk:0x8ec941e225087c452b9548a032b1e16de71b6775f435c38b1ab485d8f8cdcdbc#ScoOeQJCAirz3X5BViNhsvWYEAYEg-5qQ8_m90KX2oc",
"typ": "JWT",
"nonce": "e186f26f-4852-43fd-a08d-039b35d0ecff",
"alg": "EdDSA"
}

---------- JWT payload ----------

{
"iss":
"did:iota:lnk:0x8ec941e225087c452b9548a032b1e16de71b6775f435c38b1ab485d8f8cdcdbc",
"nbf": 1753951212,
"vp": {
"@context": "https://www.w3.org/2018/credentials/v1",
"type": "VerifiablePresentation",
"verifiableCredential": [

"eyJraWQiOiJkaWQ6aW90YTpsbms6MHgyYWI4MzU5Y2FiMjQ2OGJhMmY0MjE3NzAzNTAzYTliZGU4NmM0Y2Y4NWVlNTdlYjYwOTNlZDVjMjA3MmM4N2E3I2xZMmR1UlZZVTZTSFBCSTRreFZ0UDI1ODA5azRlNEhjU0NWUWk4eTNMWEEiLCJ0eXAiOiJKV1QiLCJhbGciOiJFZERTQSJ9.eyJleHAiOjE3ODQ3MTkwNzYsImlzcyI6ImRpZDppb3RhOmxuazoweDJhYjgzNTljYWIyNDY4YmEyZjQyMTc3MDM1MDNhOWJkZTg2YzRjZjg1ZWU1N2ViNjA5M2VkNWMyMDcyYzg3YTciLCJuYmYiOjE3NTMwOTY2NzYsImp0aSI6Imh0dHBzOi8vaXNzdWVyLnN0YXJkdXN0LmxpbmtzZm91bmRhdGlvbi5jb20vYXBpL2NyZWRlbnRpYWxzLzIiLCJzdWIiOiJkaWQ6aW90YTpsbms6MHg4ZWM5NDFlMjI1MDg3YzQ1MmI5NTQ4YTAzMmIxZTE2ZGU3MWI2Nzc1ZjQzNWMzOGIxYWI0ODVkOGY4Y2RjZGJjIiwidmMiOnsiQGNvbnRleHQiOlsiaHR0cHM6Ly93d3cudzMub3JnLzIwMTgvY3JlZGVudGlhbHMvdjEiLHsic2NoZW1hIjoiaHR0cDovL3NjaGVtYS5vcmcvIn1dLCJ0eXBlIjpbIlZlcmlmaWFibGVDcmVkZW50aWFsIiwiTWFya2V0cGxhY2VDcmVkZW50aWFsIl0sImNyZWRlbnRpYWxTdWJqZWN0Ijp7InNjaGVtYTphbHRlcm5hdGVOYW1lIjoic29tZVVzZXJuYW1lIiwic2NoZW1hOm1lbWJlck9mIjoiU0VESU1BUksgbWFya2V0cGxhY2UifX19.vlwBEVQfZxj7JbgZFRvNiFJHSQCWpftLGEao5LsIbQ4_OF5Zqclq60rhrlwJr8YgyA5TX32hRqDDOLHU8UF-Dg"
]

},
"walletSignature":
"0x78aa9044b6832d666a86a6725bb73584044f6a6ad842a4732cea87b5427331ec1946b485ead34f8173a47582fa3832be1768a155f719cba83b27302841be3af61b"
}

---------- JWT proof ----------

<The Holder's signature>

5.3 Authentication

This section presents the Authentication procedure at any entity of the SEDIMARK Marketplace acting as a Verifier. Then the section introduces the Authorization procedure that is then addressed in detail in Section 5.4.

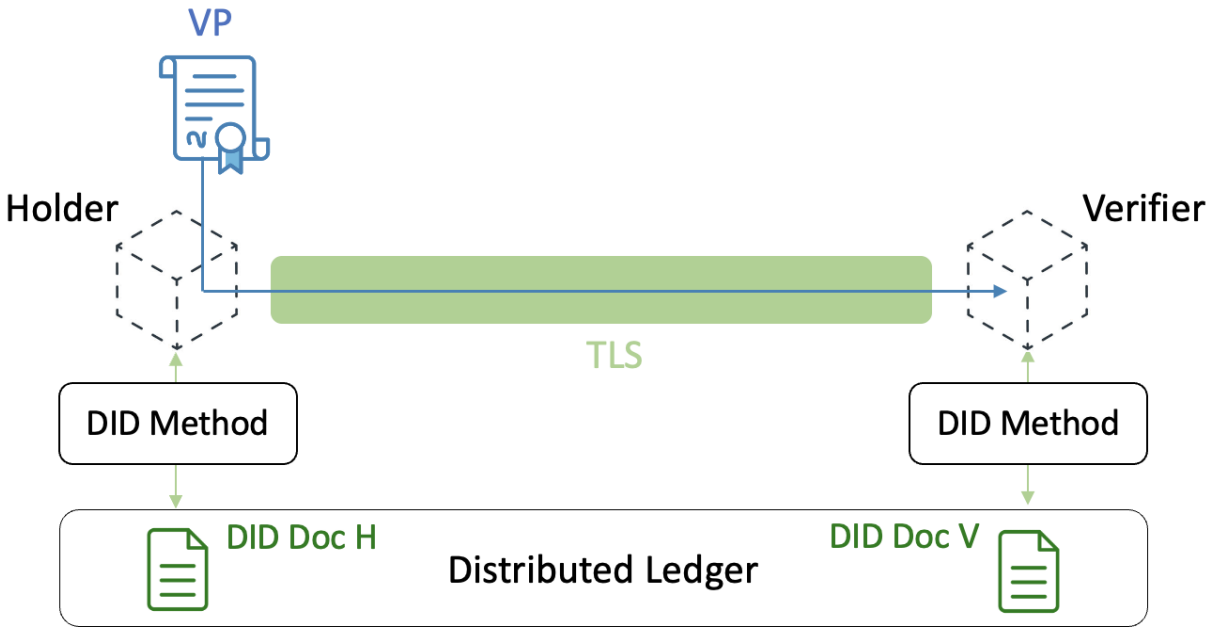

Figure 19: Application-layer Holder authentication process.

Figure 19 shows the authentication process as it takes place in the SEDIMARK Marketplace. The VP is the key component used by the Holder to authenticate itself to a Verifier (e.g., a user wishing to access the global catalogue). Once a secure channel (Transport Layer Security TLS [7]) is established with the Verifier, the Holder proceeds with the application-layer authentication process by presenting his VP. The main advantage resides into such mechanism, where the authentication takes place without any further interactions with the issuers (e.g., no connection overhead). This enhancement is due to the concept of "transitive trust" employing the VC/CP.

The authentication comprises all common verification steps envisioned by the SSI model as specified by W3C in [9]. The verification process comprises these steps:

1. Holder contacts the Verifier and receives a nonce from the Verifier, the nonce acts as a challenge for counteracting replay attacks;
2. Holder prepares and presents the VP to the Verifier;

3. The verifier verifies the VP:
   a) Verifier resolves the holder's DID to retrieve the Holder public key pk_Holder;
   b) Verifier verifies the signature on the VP with pk_Holder;
   c) Verifier checks that the VP adheres to the recommended data model;
   d) Verifier validate the credential in the presentation:
      i) Verifier resolves the issuer's DID to retrieve the Issuer public key pk_Issuer (the Issuer is trusted),
      ii) Verifier verifies the Issuer's Signature on the VC with pk_Issuer,
      iii) Verifier checks that the VC adheres to the recommended data model and the validity of all VC metadata;
4. If all verifications are successful, the Verifier authenticates the Holder, otherwise it closes the connection with the Holder.

Finally, the Verifier checks the values in the *credentialSubject* field of the VC to proceed with the authorization, i.e. the process of determining his access rights.

5.4 Authorization and Access Policies

Access control within the SEDIMARK Marketplace is governed by a policy-based authorization framework that ensures secure, compliant, and flexible enforcement of asset access conditions. These policies are defined by Providers and evaluated during both the negotiation and exchange phases, allowing fine-grained control over who can access what, under which conditions.

Policies are expressed using the Open Digital Rights Language (ODRL), a W3C standard designed to represent permissions, prohibitions, and obligations associated with digital assets. In SEDIMARK, ODRL policies are attached to offerings and interpreted by Policy Decision Points (PDPs), with the Verifier component acting as a specialized PDP optimized for on-chain validation. Once a policy is positively evaluated, a Policy Enforcement Point (PEP), typically the SEDIMARK Connector, ensures that access is granted in accordance with the defined rules.

Each policy includes a set of permissions, which may contain multiple constraints and optionally one or more duties. In the current implementation, three types of constraints and one duty are supported:

- Temporal Constraint: This condition restricts access based on time, using ODRL operators such as lt, gt, lteq, or gteq. It is stateless and can be evaluated locally by the Provider's Connector.
- Credential-Based Constraint: This constraint relies on claims embedded in the SEDIMARK Verifiable Credential (VC) issued during onboarding. Typical claims include *memberOf* and *alternateName*. While *alternateName* has limited practical relevance, *memberOf* can be used to demonstrate potential cross-marketplace interoperability. In practice, this constraint is optional and may be omitted depending on the Provider's preferences.
- DataToken Ownership Constraint: This mandatory condition verifies that the Consumer holds a valid DataToken (DT) linked to the offering's NFT. Ownership is checked on-chain during the asset access phase. Although this may seem redundant with the duty described

below, it serves a distinct purpose: confirming that the Consumer still possesses the DT at the time of access, even if it was acquired earlier during negotiation.

In addition to constraints, the policy may include a duty, which defines an obligation that must be fulfilled before access is granted. In SEDIMARK, this duty requires the Consumer to purchase a DataToken during the negotiation phase. In other words, the FSM governing the DSP Contract Negotiation will not reach the FINALIZED state unless this duty is fulfilled. The duty also includes a refinement specifying the price of the DT in native tokens, currently fixed at 1 native token. While static in the current implementation, this refinement anticipates future support for dynamic pricing, allowing Providers to define variable or market-driven prices for asset access.

The verification of this duty is performed during the negotiation phase and involves the following steps:

1. The Holder connects to the Verifier, its identity is authenticated as described in Section 5.3. It also sends the address of the offering to be verified.
2. The Verifier verify the purchase on the ISC platform. The offering smart contract contains a function that verifies the proof of purchase. The smart contract controls that the holder is a valid user of the SEDIMARK Marketplace and that the balance of the corresponding datatokens is greater than 0, i.e., the Consumer has at least purchased the offering corresponding to the desired asset.

An example of a complete ODRL policy, including all three constraint types and the duty, is shown in Figure 20. This example illustrates how temporal restrictions, credential-based conditions, and token ownership requirements can be combined with a purchasing obligation to enforce secure and accountable access to a given SEDIMARK offering.

```
{
    "@id": "https://uc.sedimark.eu/offerings/dummy-offering-id/offeringContract/dummy-offeringContract-id",
    "@type": "sedimark:OfferingContract",
    "odrl:profile": "https://sedimark.eu/odrl/sedi-profile",
    "odrl:uid": "https://uc.sedimark.eu/offerings/dummy-offering-id/offeringContract/dummy-offeringContract-id",
    "odrl:permission": [{
      "odrl:target": "https://uc.sedimark.eu/offerings/dummy-offering-id",
      "odrl:assigner": "did:iota:lnk:0xf053682e4724ba221e2f49dd0adabba135cd4ccb08d492440e163482064b617a",
      "odrl:action": "odrl:use",
      "odrl:constraint": [
        {
          "odrl:leftOperand": "odrl:dateTime",
          "odrl:operator": {
            "@id" : "odrl:lteq"
          },
          "odrl:rightOperand": {"@value": "2025-12-31T00:00:00Z", "@type": "xsd:dateTime"}
        },
        {
          "odrl:leftOperand": "odrl:dateTime",
          "odrl:operator": {
            "@id" : "odrl:gteq"
          },
          "odrl:rightOperand": {"@value": "2025-08-31T00:00:00Z", "@type": "xsd:dateTime"}
        },
        {
          "odrl:leftOperand" : "sedi:claimMemberOf",
          "odrl:operator"    : {
            "@id" : "odrl:eq"
          },
          "odrl:rightOperand": "SEDIMARK marketplace"
        },
        {
          "odrl:leftOperand" : "sedi:dataTokenOwnership",
          "odrl:operator"    : {
            "@id" : "odrl:eq"
          },
          "odrl:rightOperand": "true"
        }
      ],
      "odrl:duty": [
        {
          "odrl:action": [
            {
              "odrl:action" : "sedi:purchaseDataToken",
              "odrl:refinement": [
                {
                  "odrl:leftOperand": "odrl:payAmount",
                  "odrl:operator": {
                    "@id" : "odrl:eq"
                  },
                  "odrl:rightOperand": {"@value": "1", "@type": "xsd:decimal"},
                  "odrl:unit": "sedi:nativeToken"
                }
              ]
            }
          ],
          "odrl:constraint": [
            {
              "odrl:leftOperand": "odrl:event",
              "odrl:operator": {
            "@id" : "odrl:lt"
          },
              "odrl:rightOperand": "sedi:dspContractAgreementFinalized"
            }
          ]
        }
      ]
    }],
    "odrl:prohibition": [],
    "odrl:obligation": []
  }
```


Figure 20: Example of SEDIMARK offering ODRL policy

6 Tokenization

The SEDIMARK Marketplace uses the tokenisation process to make assets discoverable and tradable. The tokenization framework acts as a backbone for trading into the marketplace: any owner can tokenise an asset and make it discoverable for purchase by other Consumers.

The tokenisation is defined as the process of representing the ownership of real-world assets as digital tokens on the DLT. The SEDIMARK Marketplace uses a set of Smart Contracts (SCs) to rule tokenising of assets, making offers and buying access to assets. SCs are software applications that operate on the decentralized network of validators who execute and validate the same code, as described in Section 3.2.

In the SEDIMARK Marketplace the object of the purchase is the offering, which represents a specific asset owned by a Provider which put it up for sale. Such offering is unique. The trading of the asset is regulated with the appropriate SCs. With the SCs infrastructure is possible to trace the existing offerings from the Providers and the respective purchases made by the Consumers. The result of the negotiation between Provider and Consumer include the transfer of tokens from the seller to the purchaser.

6.1 Types of Tokens and Standards

In the final version of the SEDIMARK Marketplace, two types of tokens have been employed for realizing the trading operations.

Figure 21 shows the tokenization model based on the adoption of two different types of tokens.

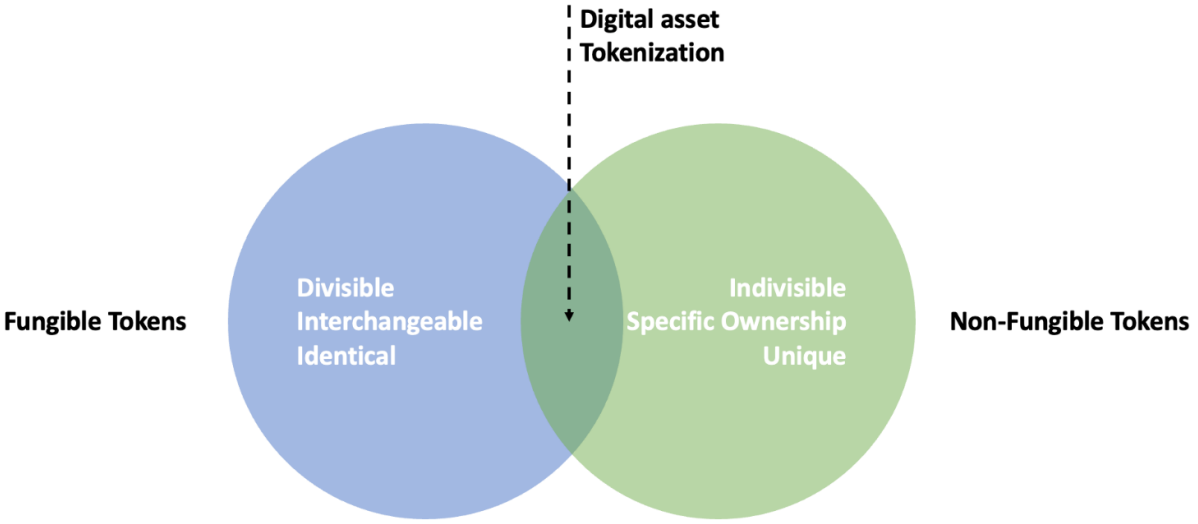

Figure 21: Digital asset tokenization

Non-Fungible Tokens (NFTs)

The purchase operation in the SEDIMARK Marketplace allows a user to obtain the access to the asset. The property of the asset bought remains to the owner, namely the Provider. A representation of the offering on the ISC-Chain of the asset is actually being bought, after its conversion into NFT.

The tokens are minted with a specific SC, named ServiceBase SC, which is an ERC-721 compliant contract [1]. The ServiceBase SC is responsible for minting an NFT representing the specific offering of its owner. ERC-721 is a token standard on the Ethereum blockchain that

defines a set of rules for creating NFT. Each ERC-721 token has a unique identifier, ensuring that no two tokens are the same. Additionally, such standard enforces interoperability, allowing NFTs to be traded across different platforms supporting this standard.

Fungible tokens – "Datatokens" (DTs)

The DT is another SC. It manages the access to an asset and proves the correct purchase of the respective offering.

NFTs and the corresponding DTs related to the asset are two linked concepts. However, the NFT cannot be copied, substituted nor subdivided. Conversely, datatokens are fungible, i.e., they can be used indistinctly for the purchase.

The DTs are used to allow the access to the asset, providing on the ISC-Chain a verifiable proof of purchase.

A DT SC is associated to each NFT SC. The NFT contains the list of the addresses of the minted fungible tokens. When an offering is created, it is also created the corresponding DT SC with a specific supply of tokens. Therefore, a certain number of fungible tokens (one or more) are associated to the specific offering's NFT. A consumer that buys the access to an asset, exchanges a native token to acquire a datatoken. Therefore, the datatoken is a proof of having purchased the asset.

6.2 Smart Contracts Overview and Token Creation

The offering of an asset is unique. The SC contains the offering metadata which are fed from the Offering Manager. The content of the metadata points to the offering description.

SEDIMARK uses Solidity, compatible with Ethereum VM (EVM) and supported by ISC-Chain. Smart Contracts are deployed on the immutable ledger which means that once they are published, nobody can tamper with the code.

The main Smart Contracts deployed on the ISC chain are described in the following:

- **ServiceBase Smart Contract:** ERC-721 compliant contract [1]; the ServiceBase SC is responsible for minting an NFT representing a specific offering of the Provider. Minting the NFT means immutably store it on the ISC chain to make an offer. The contract holds the relevant NFT information together with its metadata.
- **Datatoken Smart Contract (DTSC):** ERC-20 compliant contract [4]; the DTSC is responsible for minting a certain number of ERC-20 tokens (i.e. Datatokens) for the owner of the NFT (i.e. the owner of the associated asset). The DTSC allows the owner to add his datatokens into a Fixed Rate Exchange Smart Contract (FRESC), enabling potential Consumers to purchase access to the asset represented by the NFT. Moreover, upon a datatoken transfer, the DTSC updates the information about the balance of all the wallets involved in such an operation.
- **Fixed-Rate Exchange Smart Contract (FRESC):** is a contract acting as an exchange. Providers, after making an offer, add the minted datatokens to the FRESC to enable trading of the asset at a fixed price. The smart contract can swap Provider's datatoken with Consumer's native tokens. Later, Providers can withdraw the swapped native tokens from the FRESC. A new local instance of the exchange is created within the FRESC every time a new datatoken is added to FRESC. Thus, the FRESC holds many exchange instances, one for each datatoken. Every exchange instance holds essential information required for the exchange process (e.g., datatoken owner, datatoken price, DTSC address). In more detail, the Provider sets the datatoken price when it instantiates the

exchange. Moreover, a unique ID is assigned to each instance; the ID facilitates transactions and token swaps.

6.3 Token usage and available operations

From the usage point-of-view, a user (either provider or consumer) can trigger the following operation on the platform:

- **Creation and publication of the offering.** The creation and the publication of the offering is realized through the interactions with the Offering Manager and the DLT-Booth of the SEDIMARK Toolbox. The tokenisation allows the placement of the immutable SC containing the pointer to the offering information. The offering manager module is employed for managing the offering lifecycle. Also in this case, the DLT-Booth component actually stores the offering onto the DLT interfacing with decentralized infrastructure.
- **Purchasing of the offering.** A consumer has the assurance that the offering advertised is the one desired and it will not change in the future. Therefore, after the purchase, it is possible to demonstrate the agreement on a specific offering with the advertised features. On the other hand, a Provider relying on this mechanism based on a robust decentralized infrastructure, has the guarantee enabled with the Proof-of-Purchase verification.
- **Access to data.** A Consumer buys a specific datatoken related to the desired offering. The ownership of the datatoken enables the Consumer to access to the Provider's asset. In fact, datatokens are strictly related to a specific offering published on the Smart Contract Platform. Therefore, the Provider relies on the offering Smart Contract to verify that the Consumer can access the asset; it can check that the Consumer is a valid user of the SEDIMARK Platform and it purchased a datatoken associated to a specific asset.

In any marketplace, there is value exchange, that is a connection between what is purchased and the currency. In the SEDIMARK Marketplace, the asset exchange between the Provider and the Consumer is realized by giving datatokens in exchange of the access to the asset. In this exchange there is no currency involved. For the sake of simplicity in the SEDIMARK Marketplace it is employed a customized native token. This specific "currency" does not carry any real economic value outside the SEDIMARK Marketplace. The native tokens have been generated with the sole target of supporting the Use Cases activities of the Project. Moreover, having access to the real economic value would imply for the Partners to managing the related financial activities, incurring in a cumbersome overhead. For this reason, in the SEDIMARK Marketplace, the following assumption is made: a single native token corresponds to a single datatoken.

7 Conclusions

This document is the final version of the decentralised infrastructure and access management mechanisms. This document analysed the final version of the APIs of the decentralised infrastructure enabling the various operations in the marketplace and the mechanisms for implementing the access control of the users of the SEDIMARK Platform.

The decentralized infrastructure and access management presented in this document consolidates the foundations for a secure and decentralised marketplace. These features are complemented by the tokenization of the assets together with the chains of smart contracts that orchestrate the secure exchange of assets enabled by the SEDIMARK Marketplace.

The whole architecture is rooted in robust governance mechanisms and user-centric access controls, to define how the data and the services are managed and shared. The interconnected chains of smart contracts constitute an innovation among the complexities of the decentralization technology. The adoption of these technologies and mechanisms results in an improved trustworthiness of the SEDIMARK Marketplace.

The second and final version of the decentralised infrastructure also features the complete set of access management policies, together with the integration of the technical works deriving from the joint development, especially in other WPs – such as WP3 for the AI and data related activities and in WP5 for the final steps related to the integration and deployment.

8 References

[1] William Entriken, Dieter Shirley, Jacob Evans, and Nastassia Sachs. Erc-721. Non-fungible token standard, ethereum improvement proposals, no. 721. https://eips.ethereum.org/EIPS/eip-721 , 2018.

[2] Alex Preukschat and Drummond Reed. Self-sovereign identity. Decentralized digital identity and verifiable credentials. https://www.manning.com/books/self-sovereign-identity , 2021.

[3] Evaldas Drąsutis. IOTA Smart Contracts. https://files.iota.org/papers/ISC_WP_Nov_10_2021.pdf , 2021

[4] Fabian Vogelsteller and Vitalik Buterin. Erc-20. Token standard, ethereum improvement proposals, no. 20. https://eips.ethereum.org/EIPS/eip-20 , 2015.

[5] N. Kannengießer, S. Lins, T. Dehling, and A. Sunyaev, Trade-offs between distributed ledger technology characteristics, ACM Computing Surveys, vol. 53, no. 2, pp. 1–37, 2020.

[6] Serguei Popov. The Tangle. https://assets.ctfassets.net/r1dr6vzfxhev/2t4uxvsIqk0EUau6g2sw0g/45eae33637ca92f85dd9f4a3a218e1ec/iota1_4_3.pdf , 2018.

[7] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. Internet Request for Comments. https://www.rfc-editor.org/info/rfc8446 , 2018.

[8] W3C. Decentralized Identifiers (DIDs) v1.0. W3C Recommendation. https://www.w3.org/TR/did-core/ , 2022.

[9] W3C. Verifiable Credentials Data Model v1.0. W3C Recommendation. https://www.w3.org/TR/vc-data-model-1.0/ , 2019.

[10] IOTA SDK. https://github.com/iotaledger/iota-sdk

[11] IOTA Identity. https://github.com/iotaledger/identity.rs

[12] Actix-web. https://actix.rs/

[13] ipfs-api-backend-actix. https://crates.io/crates/ipfs-api-backend-actix

[14] SEDIMARK, Deliverable 2.1: Use case definition and initial requirement analysis, SEDIMARK, June 2023.

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

[16] International Dataspace Protocol - Version 0.8. https://github.com/International-Data-Spaces-Association/ids-specification/tree/v0.8

[17] IDSA, Dataspace Protocol - Working Draft. https://docs.internationaldataspaces.org/ids-knowledgebase/v/dataspace-protocol/overview/readme

[18] Eclipse Foundation. Eclipse Dataspace Components.https://eclipse-edc.github.io/docs/#/README

[19] Contract Negotiation Protocol state machine. https://github.com/eclipse-dataspace-protocol-base/DataspaceProtocol/blob/main/specifications/negotiation/figures/contract.negotiation.state.machine.png

[20] Transfer Process Protocol state machine. https://github.com/eclipse-dataspace-protocol-

base/DataspaceProtocol/blob/main/specifications/transfer/figures/transfer-process-state-machine.png
[21] Koa. https://github.com/koajs/koa
[22] Express. http://expressjs.com
[23] Traefik Proxy. https://traefik.io/traefik/
[24] Apache Software Foundation, 2021. Apache Jena, Available at: https://jena.apache.org/.
[25] SEDIMARK, Deliverable 3.4: Enabling tools for data interoperability, distributed data storage and training distributed AI models. Final version, SEDIMARK, July 2025.
[26] SEDIMARK, Deliverable 4.1: Decentralized infrastructure and access management. First version, SEDIMARK, December 2023.
[27] Martin Holst Swende and Nick Johnson. Erc-191. Signed data standard, ethereum improvement proposals, no. 191. https://eips.ethereum.org/EIPS/eip-191, 2016.