SEcure Decentralised Intelligent Data MARKetplace

D4.1 Decentralized Infrastructure and Access Management - First version

Document Identification	
Contractual delivery date:	31/12/2023
Actual delivery date:	31/12/2023
Responsible beneficiary:	LINKS
Contributing beneficiaries:	LINKS, UC, SURREY, NUID UCD
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):**	
Document reference:	SEDIMARK_D4.1	**Total number of pages:**	35

List of Contributors	
Name	**Partner**
Alberto Carelli Luca Giorgino Andrea Vesco	LINKS
Pablo Sotres Juan Ramón Santana Luis Sánchez Jorge Lanza	UC
Tarek Elsaleh	SURREY
Elias Tragos Honghui Du	NUID UCD

Document History			
Version	**Date**	**Change editors**	**Change**
0.1	21/07/2023	Luca Giorgino, Alberto Carelli, Andrea Vesco, (LINKS)	First version of document structure (Table of Content)
0.2	01/12/2023	Luca Giorgino, Alberto Carelli, Andrea Vesco, (LINKS)	Contributions and finalisations of the main technical sections of the document: Interactions, SSI, Authentication and Authorization, Tokenization
0.3	05/12/2023	Pablo Sotres, Juan Ramón Santana, Luis Sánchez, Jorge Lanza (UC)	Document review and contribution to Section 3.3, 4.1, 5.4.

Document History			
Version	**Date**	**Change editors**	**Change**
0.4	07/12/2023	Tarek Elsaleh	Contribution to Section 3.4, 4.2, 4.4
0.41	11/12/2023	NUID UCD	Contribution to Section 4.2
0.5	11/12/2023	Alberto Carelli	Document ready for internal review
0.6	20/12/2023	Alberto Carelli	Implementation of internal review comments
0.7	22/12/2022	Alberto Carelli	Changes requested by Quality Review
0.8	22/12/2023	ATOS	Quality Review Form
1.0	22/12/2023	ATOS	FINAL VERSION TO BE SUBMITTED

Quality Control		
Role	**Who (Partner short name)**	**Approval date**
Reviewer 2	Maroua Bahri, Nikolaos Georgantas (INRIA)	20/12/2023
Reviewer 1	Grigorios Koutantos (WINGS)	18/12/2023
Quality manager	María Guadalupe Rodríguez (ATOS)	22/12/2023
Project Coordinator	Arturo Medela (ATOS)	22/12/2023

Table of Contents

List of Figures

List of Acronyms

Abbreviation / acronym	Description
API	Application Programming Interface
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
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
NFTSC	NFT Smart Contract
NFT	Non-Fungible Token
ODRL	Open Digital Rights Language
P2P	Peer-to-Peer
PDP	Policy Decision Point

Abbreviation / acronym	Description
PEP	Policy Enforcement Point
RDF	Resource Description Framework
REST	REpresentational State Transfer
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.

This is the first version of the decentralised infrastructure and access management mechanisms. This deliverable presents a comprehensive overview of the decentralized infrastructure and access management mechanisms implemented in the SEDIMARK Marketplace. As the landscape of data exchange evolves, the decentralization approach ensures increased security, transparency, and user-centric control both over data assets and user identity information. Data providers of the SEDIMARK Marketplace can also provide additional types of assets, related to their data. Examples are Machine Learning (ML) Models, data processing pipelines and tools.

Operating within the principles of decentralization, this project addresses the growing need for secure and transparent data exchange in a globalized digital economy. 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. Utilizing a distributed network, the infrastructure eliminates single points of failure, enhancing reliability and ensuring the continued availability of assets to be exchanged. The decentralized infrastructure supports standardized protocols for data exchange, enabling collaboration and data sharing across various platforms and participants.

This deliverable is the first capstone in the SEDIMARK project, realizing the underlying infrastructure and mechanism that allow the fulfilment of the functionalities defined for the Marketplace. An updated version of this deliverable will be provided in the next Deliverable SEDIMARK_D4.2 in July 2025.

1 Introduction

1.1 Purpose of the document

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

The main goal is to describe and analyse the underlying architecture that enables the decentralisation of the SEDIMARK Marketplace. The aim is to provide a clear understanding of the design principles, functionalities, and benefits associated 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 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 interactions that take place in a marketplace and how those are mapped in the current SEDIMARK architecture.

Section 3 presents the decentralised infrastructure laying the foundation of the Marketplace.

Section 4 describes how the components are realized 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 explores the diverse interactions that take place in the data 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.

The SEDIMARK Marketplace provides a decentralized environment for the exchange of data and other assets between different parties.

2.1 Background: Decentralization - Logic Layers

The decentralized marketplace leverages the IOTA DLT (Distributed Ledger Technology) infrastructure. From an architectural point of view, it can be partitioned into two interacting layers: Layer 1 (L1) which is the IOTA ledger and Tangle and Layer 2 (L2) which is the IOTA Smart Contract Framework (ISC). The information in these layers is distributed across a network of nodes. 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 implements the core business logic of the Marketplace, i.e., enables the discovery and trading of assets. The layers are detailed in Section 3.

2.2 Entities and Interactions in the Marketplace

From a high-level point of view, the SEDIMARK Marketplace is composed of different entities interacting with each other.

In the Marketplace, a consumer refers to an entity or individual that purchases and utilizes data for various purposes. Consumers in a data marketplace are typically business companies, organizations, researchers, or individuals looking to access and leverage data to gain insights, make informed decisions, or support their specific needs. Consumers usually look for the quality and reliability of the data they acquire. Thereby it is important that the data assessment pipeline measures the accuracy, completeness, and timeliness of the data to ensure that it meets the Consumer standards.

A Provider is instead an entity that supplies data, access to data, or Machine Learning (ML) models to consumers within the marketplace. Data providers play a crucial role in the ecosystem by contributing valuable assets that meet the diverse needs of consumers. It can provide a variety of data types, including structured and unstructured data. This may include datasets related to demographics, financial transactions, weather patterns, social media activity, and more. In SEDIMARK, the data providers will provide at least datasets for the specific Use Cases (UCs) described in SEDIMARK_D2.1 [15]. In addition to selling raw data, other Providers may also offer the possibility to access to pre-trained ML models or other analytical tools. This diversification allows an extended Marketplace.

The Issuer is a crucial entity of the SEDIMARK Marketplace. In particular, in the context of SSI, it is an entity that issues verifiable credentials or claims to individuals, either Providers or Consumers. These verifiable credentials contain information about an individual, such as personal attributes, achievements, or any other data that the individual may want to share. The issuer digitally signs these credentials, providing cryptographic proof of their authenticity. The

Issuer allows an external entity to become part of the users of the SEDIMARK Marketplace. The user, according to the principles of the SSI model, has control over their own identity information.

A global catalogue is a logical structure which correspond to a central repository that gives access to all the information about the available datasets and services offered within the marketplace. Such information concerns the type of data, its source, format, and any relevant metadata. Similar information is related to the services offered. Catalogues are considered as global since they span the entire decentralized network but can be implemented following a distributed paradigm. The main objective of the catalogue is to allow the users of the SEDIMARK Marketplace to search and to discover datasets matching their specific needs.

The previous entities are shown in Figure 1. The picture also shows the interactions taking place on the ledger, as well as those off ledger.

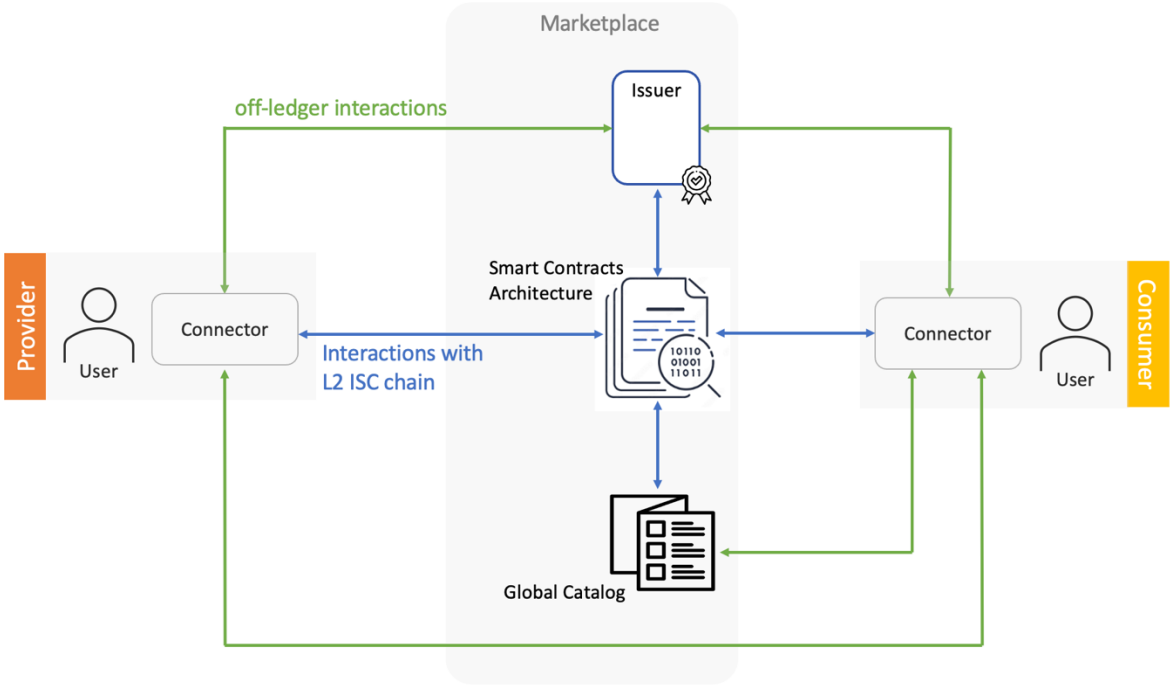

Figure 1 Marketplace - core entities and interactions

All the interactions are mediated by the Connector, which is a module that acts as an interface. The Connector is the component that enables the exchange of data and services between different entities within the SEDIMARK Marketplace. It is deployed on each participant domain and, therefore, is part of a decentralized network of peers.

The off-ledger interactions are instead common direct connections between a couple of Connectors following a P2P (Peer-to-Peer) paradigm. 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 employ the decentralized DLT infrastructure. An accurate description of such infrastructure is detailed in Section 3, while its current physical implementation can be found in SEDIMARK_D3.3 [16]. These kinds of interactions are

communications between entities that directly rely on the distributed ledger. The rationale behind is that this form of communication allows the immutability of the information exchanged and provide an additional layer of trust of the parties involved. Moreover, being the information permanent, allows future auditing and makes the scaffold for the monitoring functionality of the transactions happening into the marketplace. As an example, consider a customer that after browsing the catalogue, wants to buy access to an asset: its Connector mediates the trading by leveraging directly to the Smart Contract Architecture (described in Section 6). It has to be noted that for the sake of simplicity, Figure 1 does not show the on-ledger interactions happening with L1. This layer is used for the storage and retrieval of the related necessary public components of the SSI identity of the involved entities as described in Section 5.1.

The business logic of the marketplace is realized through different actions involving the entities described above. The following list details the steps required by the processes that are iterated among different users to implement the core functionality of the marketplace.

Onboarding of a participant

This operation enables the access to an external user to the SEDIMARK Marketplace. It is a mandatory procedure required to benefit from any of the other functionalities.

- The Connector of a participant generates an Identity according to the SSI paradigm. The process of identity generation involves the creation of a keypair, a DID (Decentralize Identifier) and a DID Document. The corresponding Public Key is stored on L1, while the Private Key is securely stored at the participant side.
- The same Connector initiates the interaction with the Issuer of the SEDIMARK platform to request a Verifiable Credential (VC). On the one hand, it allows a user (Customer) to access the content of the Marketplace, on the other hand, it allows the Provider to publish offers.
- Holding a valid VC guarantees access to the services of the secure and decentralized SEDIMARK marketplace.

The VC is locally stored by the Connector and will be employed later, wrapped into the corresponding Verifiable Presentation (VP). The VP will be, in turn, used to access the services of the Marketplace. The detailed analysis of identity management according to the SSI model is analysed in Section 5.2 of this deliverable, together with examples of DID, VC and VP.

Offering publication

This operation allows a Provider to declare its availability in trading a specific asset, either data or services, by advertising offerings that will be later indexed by catalogues within the platform. This procedure is reserved for a Provider and requires the onboarding procedure.

- The Connector of a Provider is able to interact with the ISC infrastructure (at L2) in order to publish offerings related to its own assets.

This procedure is enabled only for those Connectors who effectively joined the Marketplace. Eventually NFT (Non-Fungible Token) and datatoken are derived from the asset to make it tradable. The details of these operations are described in Section 6.

Offering/Catalogue Browsing

This operation allows a user of the SEDIMARK Marketplace to list and inspect the offering describing the assets advertised in the SEDIMARK Marketplace.

- Catalogues fetch the offering recorded and show them to the participants. The participants are capable of seeing the offerings after their VP is verified. A Participant is able to see only the offerings that it is allowed to access.

Asset Purchase

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. The participant acquires the access to the asset desired in exchange of funds delivered to the Provider. Two options are possible:

1. A Consumer chooses an asset and decides to buy access to it. The choice triggers the interaction with the ISC chain for buying the access to the asset. If the purchase is completed successfully, the participant receives the datatoken related to the specific asset bought in exchange of its funds. The received datatoken represents the Proof of Purchase of the asset directly from the Provider.
2. A Consumer chooses an asset and decides to buy access to it. The choice triggers a P2P interaction between Connectors involved (Provider and Consumer) following the IDS protocol [17]. Once an agreement has been reached and the access has been purchased, the consumer can access the asset directly from the Provider.

3 Core Technologies for Decentralization

The following subsections outline the different technologies that serve as foundation 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. 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, called Unspent Transaction Output (UTXO) ledger, works in accordance with the concept of unspent transaction. UTXO is an unused or leftover cryptocurrency in a transaction. Every crypto transaction consists of an input and an output. Every time a transaction is executed, the input is deleted, and the output is generated. Any output that is left behind and is not spent immediately is an Unspent Transaction Output that can be later spent in a new transaction. The adoption of UTXO was the key condition to push for scalability and throughput. In fact, many parallel writes to the ledger are possible [3].

IOTA Tangle, a permissionless, feeless and miner-less consensus protocol based on DAG. The DAG is called the Tangle [6]. The DAG is replicated among a decentralized network of computers, also called nodes. To issue a transaction, nodes must work to approve other transactions. Therefore, nodes who issue a transaction are contributing to the network's security. It is assumed that the nodes check if the approved transactions are not conflicting. If a node finds that a transaction conflicts with the tangle history, the node will not approve the conflicting transaction. As a transaction receives additional approvals, it is accepted by the system with a higher level of confidence. In other words, it will be difficult to make the system accept a double-spending transaction. It is important to observe that the protocol does not impose any rules for choosing which transactions a node will approve [6]. The equilibrium in the Tangle is based on the fact that if a large number of nodes follow a reference rule, then for any node it is better to stick to the same rule, otherwise its transactions will be never validated. As a result, the IOTA Tangle enables global consensus on the UTXO ledger state without the need to impose a linear order among transactions [3].

3.2 IOTA Smart Contract (ISC) chain

IOTA Smart Contract (ISC) chain is a framework that extends the L1 of the IOTA DLT. ISC introduces multiple programmable ledgers on top as L2. The result is a multi-chain system where all chains anchor its state on the IOTA Ledger at L1. Each chain is a standalone blockchain with equivalent to Ethereum smart contracts functionality [3]. Figure 2 shows the anchoring between the two levels.

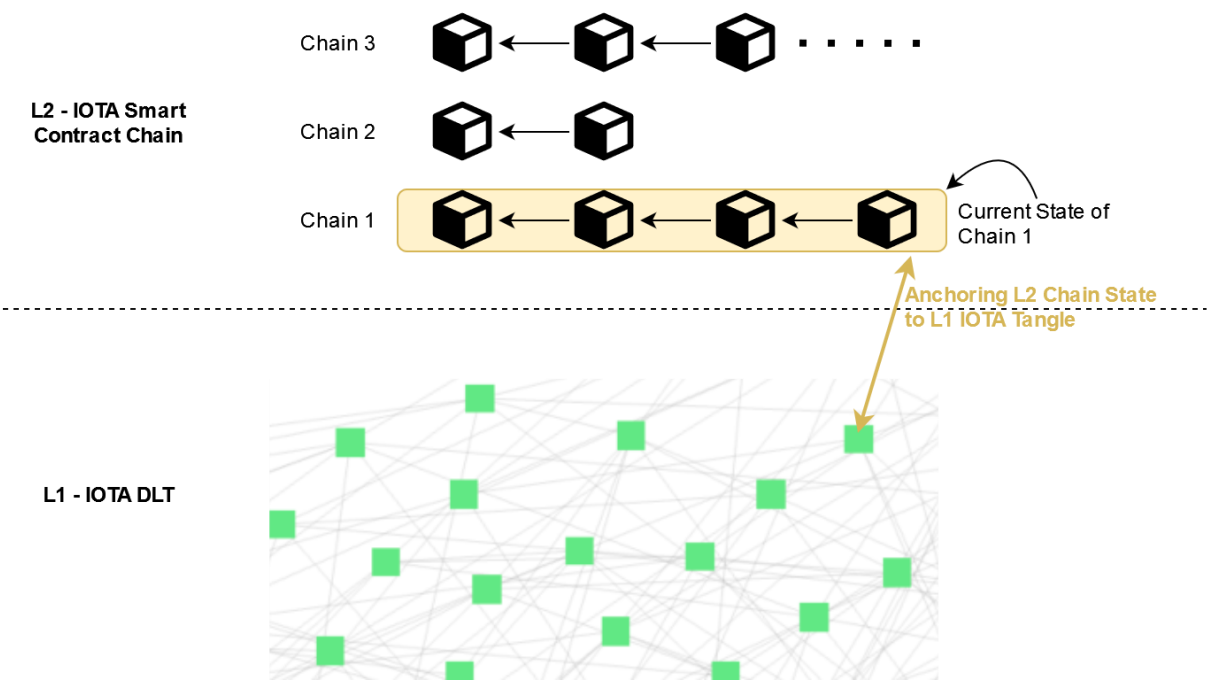

Figure 2 Anchoring between the DLT layers

A validator is a node on the chain that works to secure the ISC chain by validating it via participation in a distributed consensus with other validators. The implementation of the validator node is called WASP. Each ISC chain is run by a set of validator nodes.

The validator nodes form a group called “committee of validators”; which runs a chain by collectively taking control of the associated L1 account [3]. The committee of validators runs a consensus protocol on state updates/blocks. This makes an ISC chain and smart contracts on it a fault-tolerant and distributed computing system, in which [2N/3] + 1 of non-faulty nodes is enough to produce valid state updates [3].

ISC enables the validator nodes of the chain to control the account on L1 through threshold signatures as a proof of consensus among validator nodes. Each validator node in the committee owns its private key share which is used to produce a partial signature of the transaction. To produce a valid threshold signature for the transaction, a quorum of [2N/3] + 1 validator nodes is enough to cooperate. The cooperation of validator nodes and signature aggregation is a part of the consensus process [3]. A distributed key generation (DKG) process runs at the chain deployment stage such that each validator securely generates and keeps its private key share [3].

3.3 Dataspace Protocol

As defined by IDSA (International Data Spaces Association) in [18], the Dataspace Protocol is a set of specifications designed to facilitate interoperable data sharing between entities governed by usage control and based on Web technologies. These specifications define the schemas and protocols required for entities to publish data, negotiate usage agreements, and access data as part of a federation of technical systems termed a dataspace.

This protocol includes the specifications to define the interactions between components and is meant to be the technical specification for the IDS-RAM. The current version of the protocol comprises four main topics specified in different documents, laying the foundation for the main interactions among data space participants.

- Data Space Model and Terminology: it defines the ontologies and taxonomies to enable the interoperability between participants.
- Catalogue Protocol: defines how datasets are published and accessed, providing the necessary common interfaces. In the Data Space Protocol, datasets are represented following the Data Catalogue Vocabulary (DCAT) ontology.
- Contract Negotiation Protocol: defines the set of interactions required to conduct contract negotiations between two participants, ensuring that both participants agree on the terms for access and control rules. Contracts are represented following the Open Digital Rights Language (ODRL) ontology.
- Transfer Process Protocol: defines how the data exchange is carried out once an agreement has been reached between two participants. Differently from the previous specifications from the IDS-G, no protocols are defined for this exchange, but the control of the transfer process.

All of the specifications rely on the use of the Connector as the core component of a data space to define all the interfaces for the different secure interactions. Besides, two planes are defined: the control plane, which is covered by the Data Space Protocol Specification; and the data plane, which is only in charge of the data exchange.

In SEDIMARK, we leverage the IOTA DLT and ISC chain and we combine them with data space Connectors in order to expand and provide additional features for both Catalogue and Contract Negotiation protocols, thereby enhancing the trustworthiness of such 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. RDF triples, which consist of a subject, predicate, and object, form 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 contrast to a centralized triple store, whereby all data management occurs 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 DTSs, SPARQL, allows for federated queries whereby a query can be executed on multiple SPARQL endpoint as well as the local endpoint, and in turn merge all results from different remote sources. It is important to note that this feature needs to be employed in a manner that overcomes through optimisations performance issues related to: network latency, bandwidth consumption, availability and reliability of remote endpoints, restrictions imposed by endpoints, and handling of complex queries.

There are a number of open-source implementations for DTSs, which mainly build upon open-source tools such as Jena TDB, Sesame, and Virtuoso. SEDIMARK will analyse the implementations and will select the most suitable one to implement the DTS, aiming for simplicity of implementation and speed of response in the query federation process.

4 Implementation Perspectives

4.1 Connector

Connectors facilitate secure peer-to-peer information exchange between participants in the Marketplace. They provide a control plane for contracting, ensuring that assets are securely requested and provisioned in the marketplace. Furthermore, they play a role in both the registration and sharing of offerings by hosting both the participant self-description and the offering self-listing.

Every participant in the Marketplace has at least one connector deployed in their own domain. This entity acts as the main entry point into the distributed SEDIMARK Marketplace. More specifically, it contains software components that allow P2P interaction among connectors, following Dataspace Protocol principles, as well as interaction with the IOTA DLT.

Concerning the DLT plane, the IOTA software components 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. It is important to mention that these modules can be loosely coupled with the connector core functionality, so they can be developed independently using different languages and deployed as microservices. SEDIMARK connectors are based on the Eclipse Dataspace Components (EDC) one [19].

As previously stated, connectors facilitate both the registration and sharing of offerings by hosting the participant self-description and the offering self-listing. In this sense, SEDIMARK connectors offer an interface for offering publication. Whenever a provider tries to register a new offering, the connector 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.

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 state machine

defined by the Contract Negotiation Protocol [20]. SEDIMARK connectors builds on top of that state machine in order to interact with the smart contracts on the ISC chain. 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.

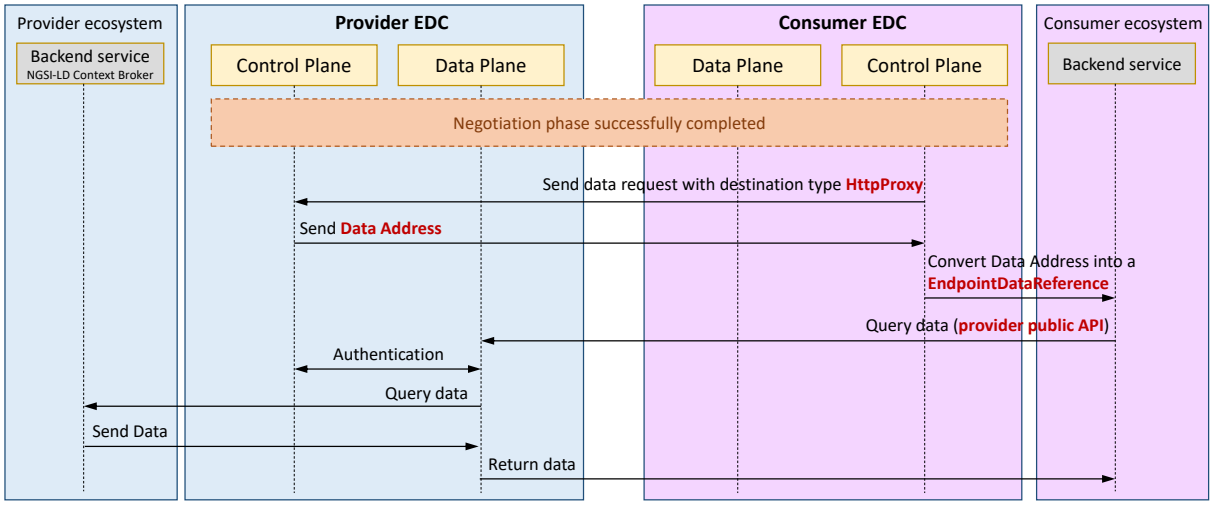

Figure 3 Detail of the IDS transfer data plane interactions between connectors

Finally, connectors are also involved in the asset transfer process. Figure 3 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). This will be the case during SEDIMARK's first iteration. However, this can change in future iterations as multiple PEPs could coexist in the provider domain depending on the nature of the offered assets.

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 may adopt several configurations based on how it is constructed and populated, whether in a centralised or decentralised manner. The Catalogue can operate on two different levels, Global (operating as part of the SEDIMARK baseline infrastructure) and Local (operating as part of the SEDIMARK Participant's toolbox).

The Global Catalogue acts as a main point of contact for onboarded Participants to search and discovery Offerings through a unified query interface. The Global Catalogue relies on the Registry to construct the Catalogue based on the retrieval of verified Offering registrations originating from Self-Listings published via Connectors of onboarded Providers.

The Local Catalogue is a variant of the Catalogue that is located at a Participant's node and behind it's corresponding Connector. By default, it provides the means to search for Offerings that are available from the hosting Participant, primarily Providers, using the same querying interface and operations exposed by the Global Catalogue. Moreover, it can also serve as a

searchable catalogue for Offerings belonging to other Participants in the Marketplace. In addition, it contributes to faster global searches in decentralised implementations of the Catalogue.

Global Catalogues can employ a number of deployment scenarios. They can rely on Self-Listings or Local Catalogues for retrieving information relating to Offerings, which can either reside on the Global Catalogue, or can be polled for on-demand. How the Catalogue is constructed depends on the Marketplace's capabilities in relation to the provision of Baseline Infrastructure which can support the dedicated hosting of the Global Catalogue. Considering that SEDIMARK aims to be a fully decentralised Marketplace, the final target is to implement a decentralised Catalogue, without the need to rely upon a central server for users to be able to discover offerings. However, decentralising the discovery process is not an easy task, so an implementation of the Catalogue will start from a centralised Catalogue, ensuring that the rest of the functionalities for discovery and sharing of offerings are executed properly. Then, the Catalogue implementation will move gradually towards becoming more server-less and decentralised, putting more emphasis on functionalities on the Participants' Local Catalogue. The different deployment scenarios that are being considered within SEDIMARK are illustrated below.

Scenario A: Centralised Catalogue (BIF)

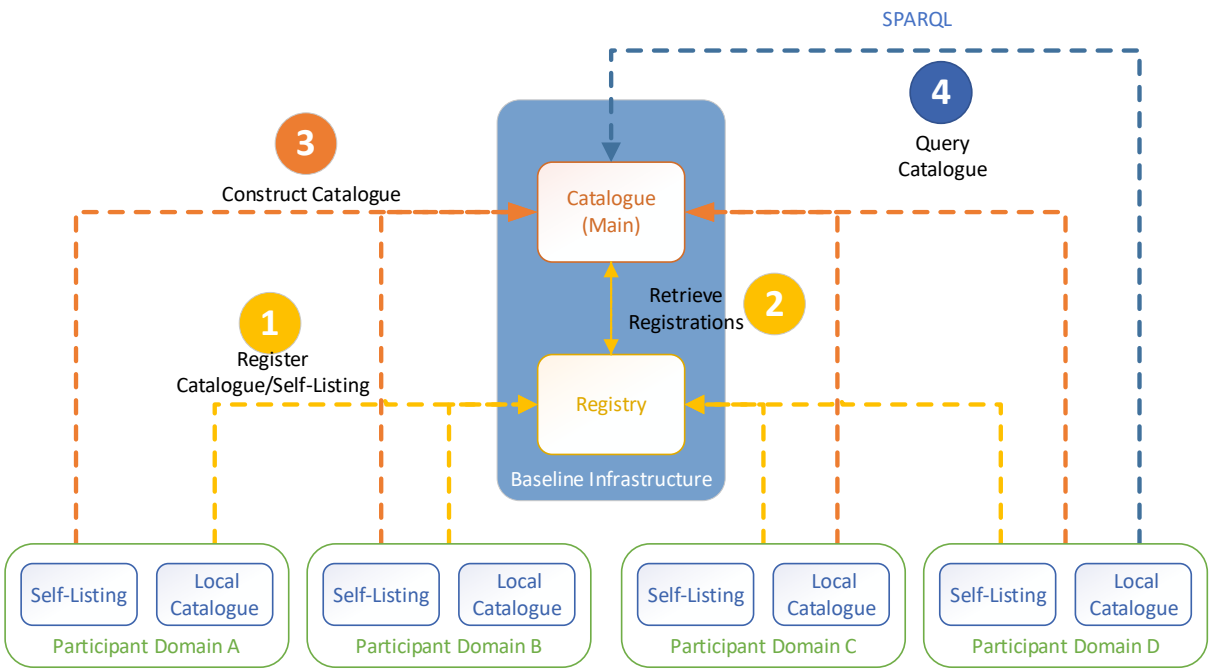

Figure 4 Centralised Catalogue Deployment

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 constructed centrally. Finally, Participants query Global Catalogue for Offerings of interest.

Scenario B: Decentralised Catalogue (BIF-Sharded)

The process of constructing the Catalogue follows the same steps (1-3 and 5) as in Scenario 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 acts as an index for all Offerings within the Marketplace and also Local Catalogues for each Provider.

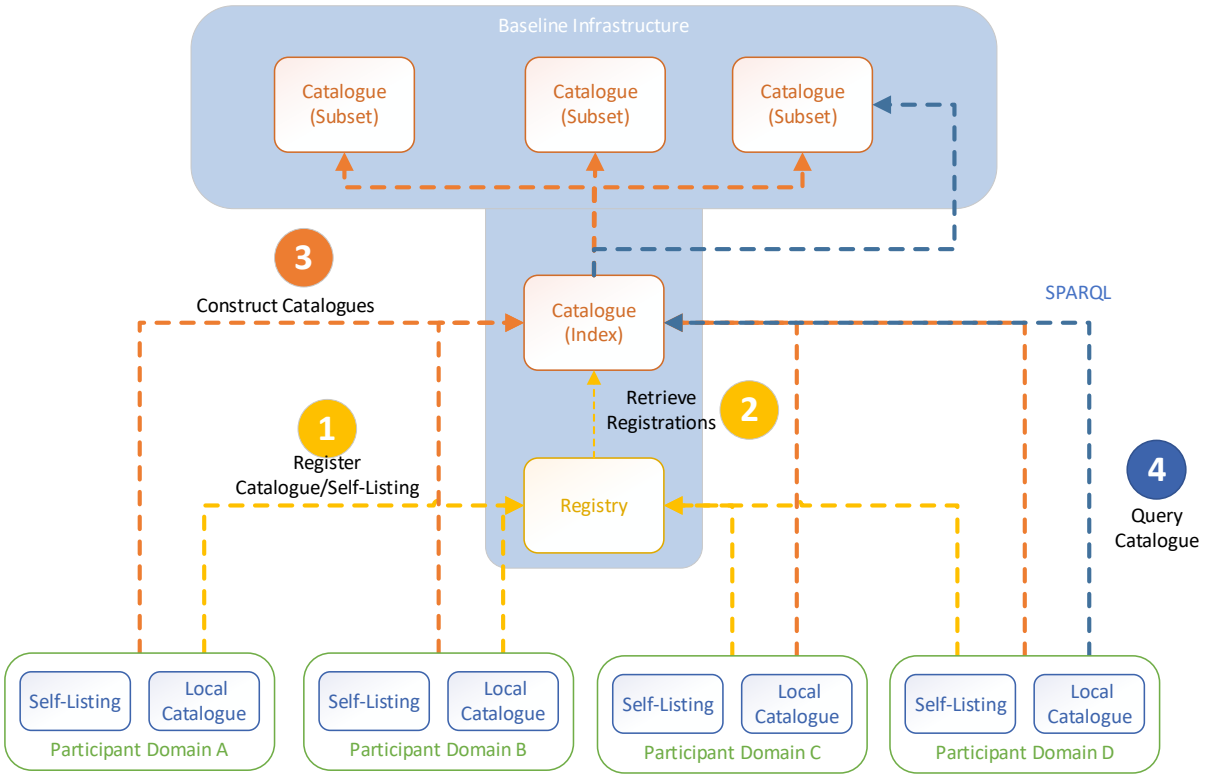

Figure 5 Decentralised Catalogue using sharding approach with BIF

Scenario C: Decentralised Catalogue (Provider-Federated)

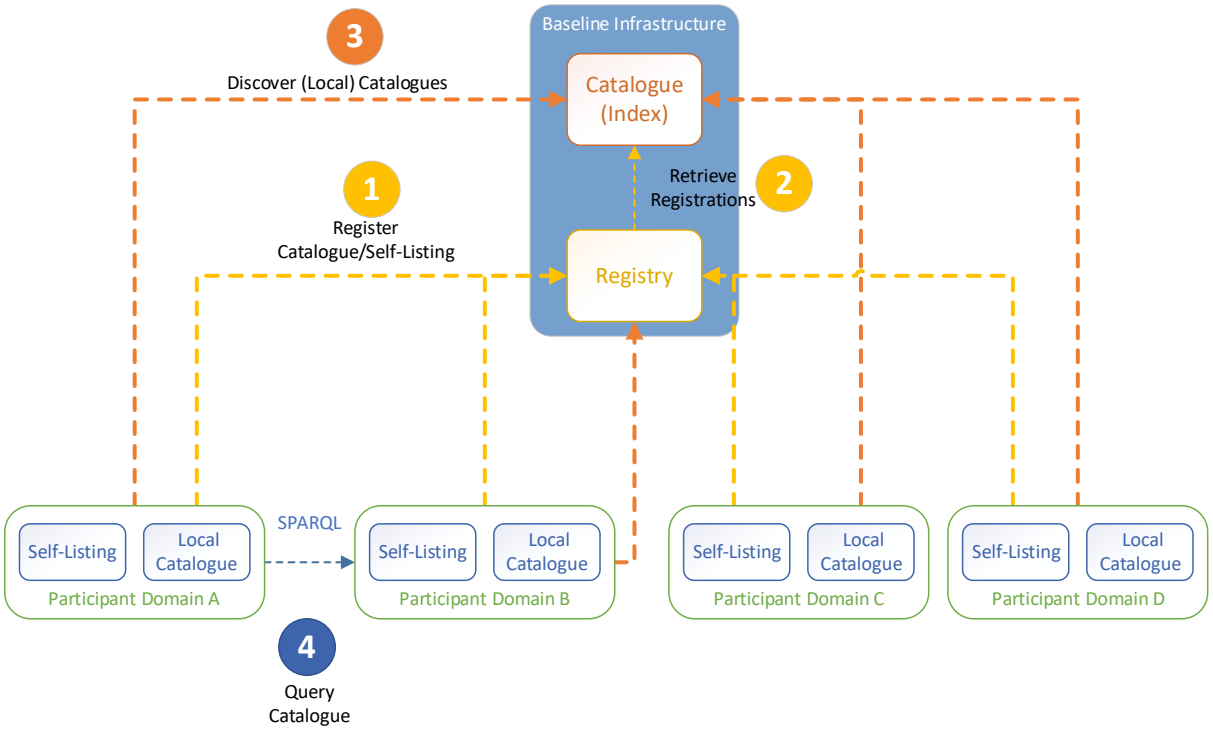

Figure 6 Decentralised approach based on Federated Catalogue at Provider Domains

Here, the process follows as in Scenario B, except that the Global Catalogue acts only as an index for relaying queries from Participants to relevant Local Catalogues of Participants or providing relevant Local Catalogues for direct querying. It has to be noted here that each Participants' Local Catalogue only stored information about their own Offerings. This can be really helpful in scenarios where a Participant wants to keep an Offering *private* and hide its existence, only revealing it to other Participants that have been authorised to access it.

Scenario D: Decentralised Catalogue (Provider-Sharded)

Finally, this scenario involves the Global Catalogue relying on a subset of Provider Local Catalogues to hold a subset of all the Offerings Descriptions made available on the Marketplace. This can be seen similar to Scenario B, where in Step 3, the sharding and distribution of the Catalogue entries are not stored into nodes of the baseline infrastructure, but in selected Participants.

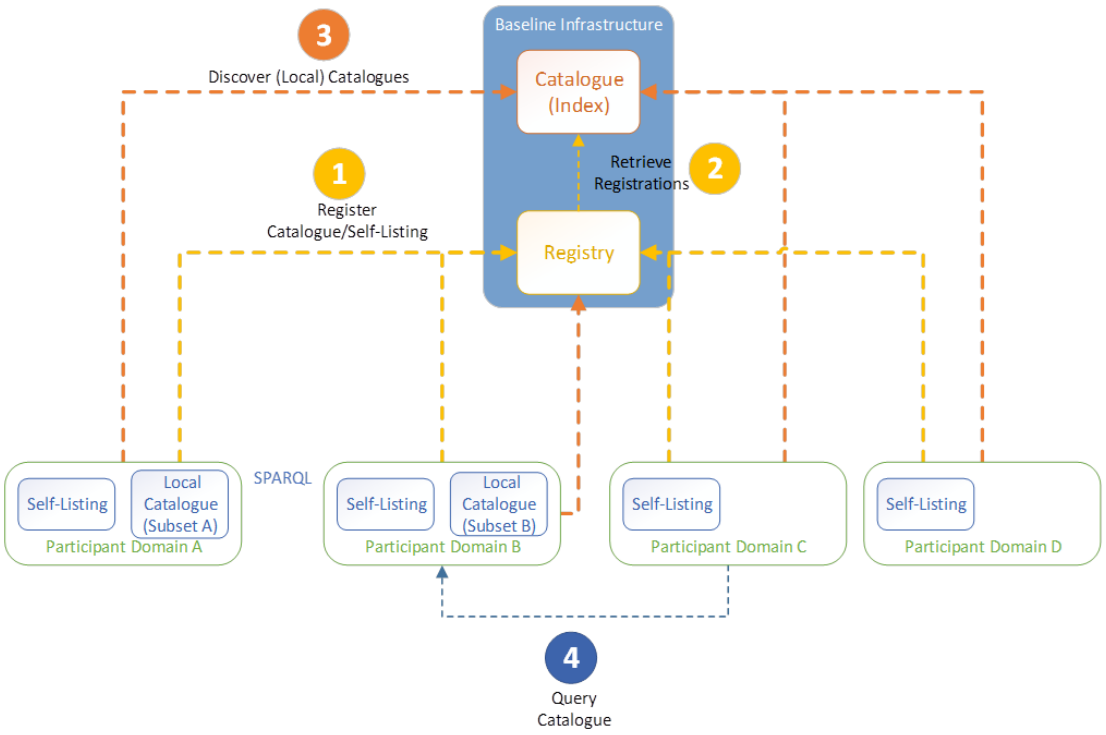

Figure 7 Decentralised Catalogue based on Select Providers hosting a Subset of Descriptions

4.3 Issuer

The Issuer of the SEDIMARK marketplace is responsible for issuing Verifiable Credentials to the users. It works as described in Section 5.1 following the Self-Sovereign Identity paradigm to govern the onboarding process of external users into the SEDIMARK Marketplace. The SEDIMARK Issuer has its own identity; the key information for the overall marketplace is its public key. By design, the public key of the Issuer is always well known by all entities in the marketplace. This prior knowledge provides the participants the capability to verify the identity of other entities for authentication and authorization purposes, as described in Section 5.3.

4.4 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.

IPFS API: is a library that provides an API wrapper for interacting with the IPFS (InterPlanetary File System) network. IPFS is a distributed protocol designed for decentralized file storage and sharing. It allows Rust developers to communicate with an IPFS node, issue commands, and interact with the IPFS network programmatically. It abstracts the details of the IPFS API, making it easier for Rust applications to integrate with IPFS. The Rust library for connecting to the IPFS HTTP API uses Hyper/Actix as the backend [13].

Ethers.js [14] library aims to be a complete and compact library for interacting with the Ethereum Blockchain and its ecosystem. It was originally designed for use with ethers.io and has since expanded into a more general-purpose library. The library provides an interface that simplifies the interaction with Ethereum smart contracts. It allows developers to deploy smart contracts, send transactions to contracts, and call contract methods. It also provides tools for creating and managing Ethereum transactions. Developers can construct transactions, estimate gas costs, and sign transactions using cryptographic keys.

EDC Connector [19]: 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 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 overall SSI stack, depicted in Figure 8, enables a new model for trusted digital interactions in decentralized systems.

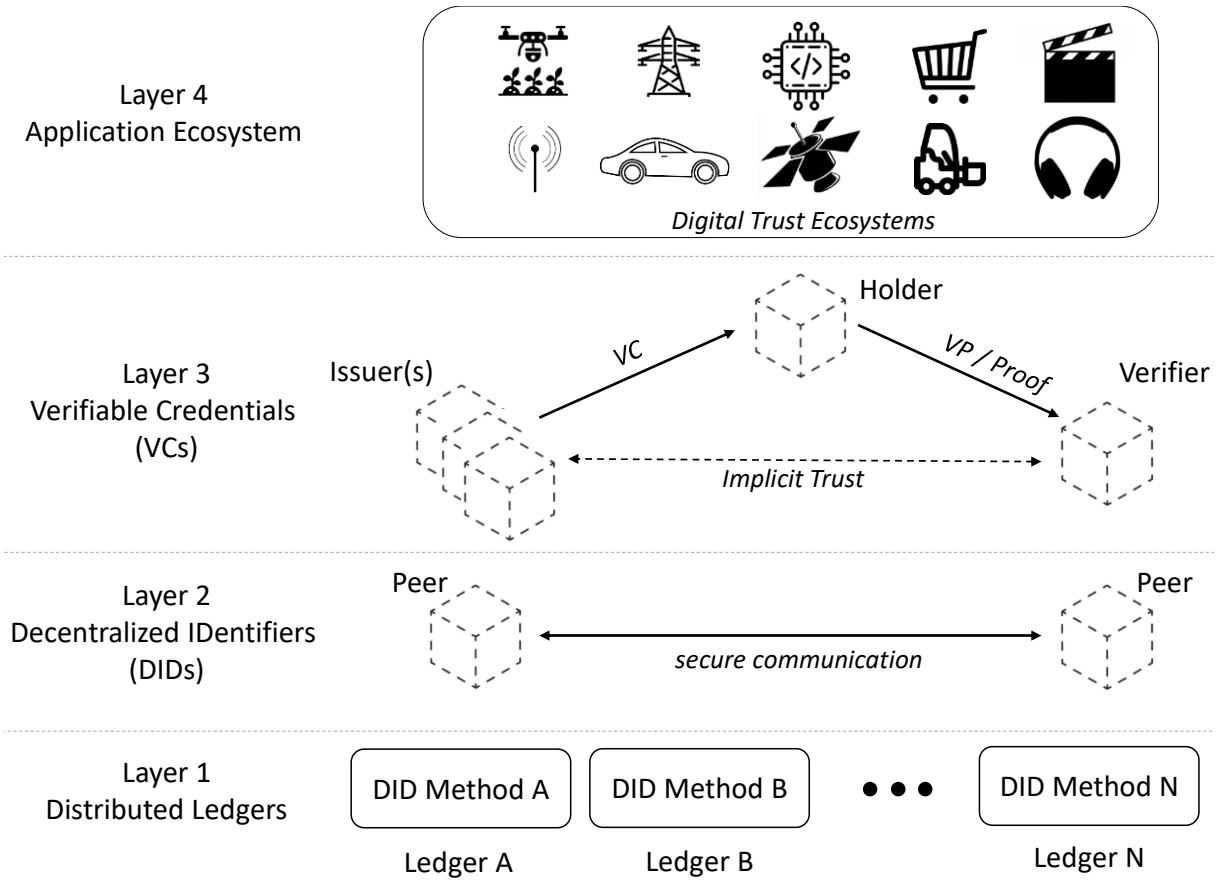

Figure 8 SSI stack

The Layer 1 is implemented by means of any Distributed Ledger Technology (DLT) acting 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.

Thus, DIDs associate a node with a DID Document to enable trustable interactions with it. 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**, that is, generate an identity key pair (sk, pk) for authentication purposes, the corresponding DID Document containing the public key of the pair (pk) and store the DID Document in the distributed ledger and make it retrievable by the DID,
- **Resolve a DID**, that is, retrieve the DID Document from the ledger pointed to by the DID,
- **Update a DID**, that is, generate a new key pair (sk', pk') 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**, that is, provide immutable evidence on the ledger that a DID has been revoked by the owner.

The DID Method implementation is ledger-specific and it makes the upper layers independent of the DLT of choice.

The Layer 2 makes use of DIDs and DID Documents to establish a cryptographic trust between two nodes. In principle, both nodes prove the ownership of their private key (sk) bound to the public key (pk) in their DID Document that is stored on the distributed ledger.

While the Layer 2 leverages DID technology (i.e. the security foundation of the SSI stack) to begin the authentication procedure, the Layer 3 finalizes it and deals with authorization for services and resources with Verifiable Credentials (VCs) [9].

A VC is an unforgeable, secure, and machine verifiable digital credential that contains further characteristics of the digital identity of a node than its key pair (sk, pk), the DID and the related DID Document.

The combination of the key pair (sk, pk), 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. Moreover, a node can enrich its identity with multiple VCs issued by different Issuers.

The Layer 3 works in accordance with the Triangle-of-Trust depicted in Figure 8.

Three different roles coexist:

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 (e.g *context, ID, type, Issuer of the VC, issuance* and *expiration dates*) and most importantly, the DID and the claims about the identity of the node in the *credentialSubject* field.

The Issuer signs the VC to make it an unforgeable and verifiable digital document. The 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 sk. Issuers are also responsible for VCs revocation for cryptographic integrity and for status change purposes [9].

On top of these three layers, it is possible to build any ecosystem of trustable interactions among nodes.

5.2 Identity Data Models

The following subsections present the DID and VC data model used in SEDIMARK.

5.2.1 DID Document

This is 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:rms:0x5df7a084466eac8e45d27a5e9ba56185d70224c069869cea12343f5122c32aee",
"verificationMethod":[{
        "id":"did:iota:rms:0x5df7a084466eac8e45d27a5e9ba56185d70224c069869cea12343f5122c32aee#Sv1P9l4SZHcYcoIfi4fjxgYAs4BCxQ_QVOKWXXAsyUo",
        "controller":"did:iota:rms:0x5df7a084466eac8e45d27a5e9ba56185d70224c069869cea12343f5122c32aee ",
        "type":"JsonWebKey",
        "publicKeyJwk":{
                "kty":"OKP",
                "alg":"EdDSA",
                "kid":" Sv1P9l4SZHcYcoIfi4fjxgYAs4BCxQ_QVOKWXXAsyUo",
                "crv":"Ed25519",
                "x":"2MJGpQwoDl_kTpRVdpCf4oJw2ZkFStrPEppkNRKzV9s"
        }
}] }
```


5.2.2 Verifiable Credentials

Figure 9 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.

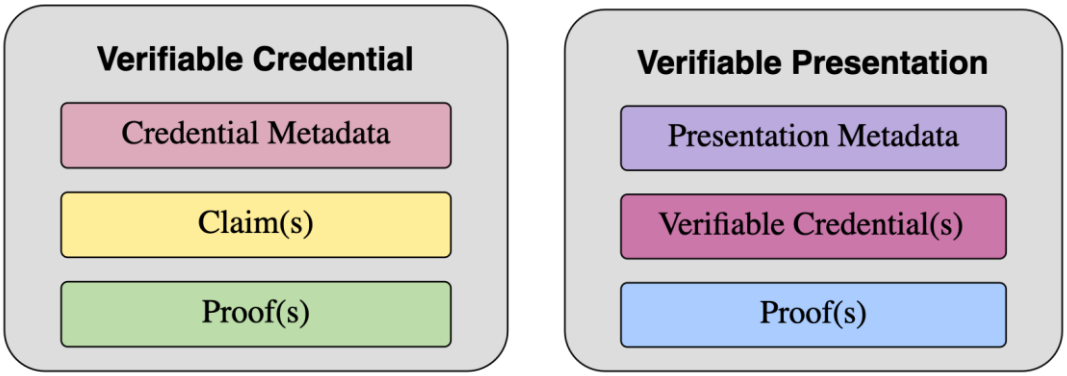

Figure 9 VC data model and VP data model

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


```
{
"@context": "https://www.w3.org/2018/credentials/v1",
"id": "https://sedimark.com/credentials/3732",
"type": [
        "VerifiableCredential",
        "MarketplaceCredential"
 ],
"credentialSubject": {
        "id":"did:iota:rms:0x5df7a084466eac8e45d27a5e9ba56185d70224c069869cea12343f5122c32aee",
        "name": "John",
        "surname": "Doe",
        "userOf": "SEDIMARK marketplace"
},
"issuer":"did:iota:rms:0x9de0315413b3961a62ac5f514b92e600ae55b7347e0f5b617d0c639b16bb645d",
"issuanceDate": "2023-11-22T17:13:19Z",
"expirationDate": "2024-11-22T17:13:19Z",
"proof": {… The Issuer's Signature …}
}
```


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


```
---------- JWT header ----------
{
        "kid":"did:iota:rms:0x5df7a084466eac8e45d27a5e9ba56185d70224c069869cea12343f5122c32aee#Sv1P9l4SZHcYcoIfi4fjxgYAs4BCxQ_QVOKWXXAsyUo",
        "typ": "JWT",
        "nonce": "475a7984-1bb5-4c4c-a56f-822bccd46440",
        "alg": "EdDSA"
}
---------- JWT payload ----------
{
        "exp":1700674385,
        "iss":"did:iota:rms:0x5df7a084466eac8e45d27a5e9ba56185d70224c069869cea12343f5122c32aee",
        "nbf":1700673785,
        "vp":{
                "@context": "https://www.w3.org/2018/credentials/v1",
```

```
            "type": "VerifiablePresentation",
            "verifiableCredential": [
        "eyJraWQiOiJkaWQ6aW90YTpybXM6MHg1MTc4NGM4YmQzMzU5YWRiNGNmMDIyM
        DJiNTcxMDM2ODFmNWQ4N2I5MzI2YWUxMzYwNjE5ZDhiNDJjYzdmMGZjIzBTdFQ0e
        C1tN1RuZGJyS2FTVjFWeThtcWtsYmhLc19pbEFCLU1yNUJZdGciLCJ0eXAiOiJKV1QiL
        CJhbGciOiJFZERTQSJ9.eyJpc3MiOiJkaWQ6aW90YTpybXM6MHg1MTc4NGM4YmQzM
        zU5YWRiNGNmMDIyMDJiNTcxMDM2ODFmNWQ4N2I5MzI2YWUxMzYwNjE5ZDhiNDJj
        YzdmMGZjIiwibmJmIjoxNzAwNjczNzg1LCJqdGkiOiJodHRwczovL3NlZGltYXJrLmNvbS9j
        cmVkZW50aWFscy8zNzMyIiwic3ViIjoiZGlkOmlvdGE6cm1zOjB4NWRmN2EwODQ0NjZl
        YWM4ZTQ1ZDI3YTVlOWJhNTYxODVkNzAyMjRjMDY5ODY5Y2VhMTIzNDNmNTEyMm
        MzMmFlZSIsInZjIjp7IkBjb250ZXh0IjoiaHR0cHM6Ly93d3cudzMub3JnLzIwMTgvY3JlZGV
        udGlhbHMvdjEiLCJ0eXBlIjpbIlZlcmlmaWFibGVDcmVkZW50aWFsIiwiTWFya2V0cGxhY2
        VDcmVkZW50aWFsIl0sImNyZWRlbnRpYWxTdWJqZWN0Ijp7Im5hbWUiOiJKb2huIiwic3
        VybmFtZSI6IkRvZSIsInVzZXJPZiI6IlNFRElNQVJLIG1hcmtldHBsYWNlIn19fQ.xM_ZLg6Z
        JAGFmmUQTv9gVKxg67DTpeJOcRCyCqISPSRkb0aTaR573sMVc_mPzNfGGdfmyedoz
        cF2dRIIP1r5Cg"
            ]
        }
}
```


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

… The Holder's signature…

5.3 Authentication and Authorization

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.

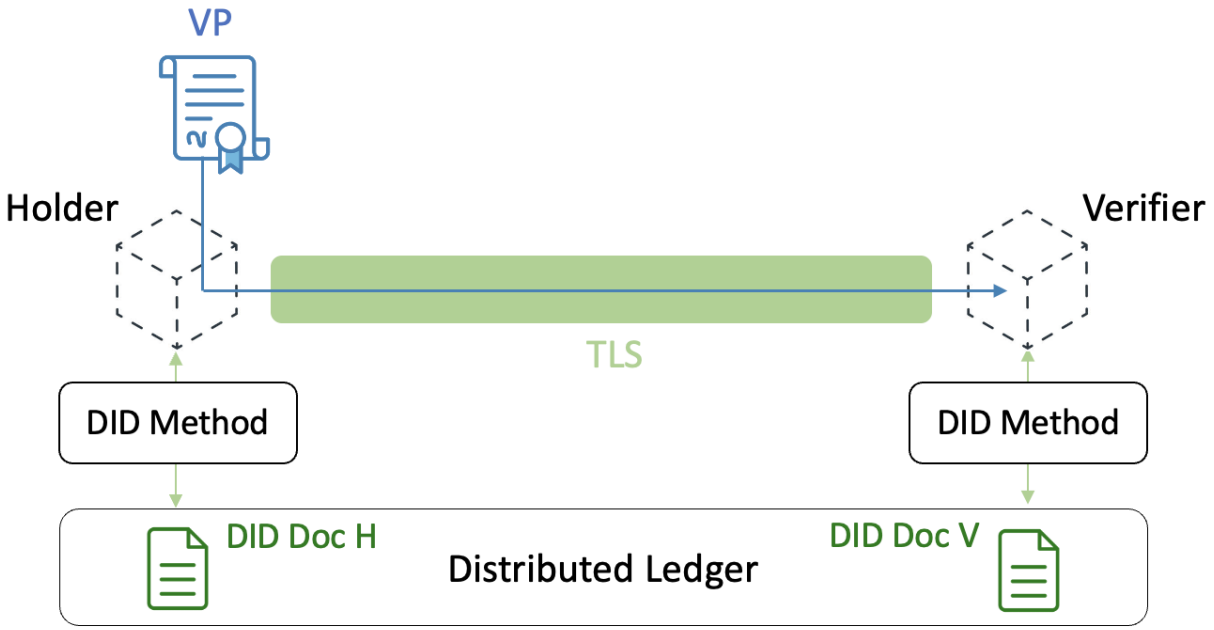

Figure 10 Application-layer Holder authentication process.

Figure 10 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 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 Access Policies

As previously mentioned, in an ideal marketplace, participants should be able to come to a consensus regarding asset access controls, pricing models, and licensing terms to safeguard their interests and ensure data privacy. In practice, once a deal is reached, the technical implications of agreeing on pricing and licensing terms come to an end. Obviously, the price has to be paid for the actual transaction to happen and accepting a license implies a legal binding. However, defining access control policies for particular assets has technical implications, as providers need both to be able to define them within the offering and enforce them when consumer try to retrieve the underlying assets.

Within SEDIMARK, and in line with IDS, we will use the Open Digital Rights Language (ODRL) for defining access policies. However, as this is a complex topic, we have opted to implement an "all or nothing" approach for the initial SEDIMARK iteration. Therefore, no access policies will be enforced, allowing consumer to access assets without additional restrictions from the moment they acquire the corresponding offering, as long as they own the datatoken representing the agreement.

All the work carried out on access policy definition and enforcement will be described in SEDIMARK_D4.2.

6 Tokenization

The SEDIMARK Marketplace uses the tokenisation process to make assets discoverable and tradable. 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. Figure 11 shows the tokenization model based on the adoption of two different types of tokens.

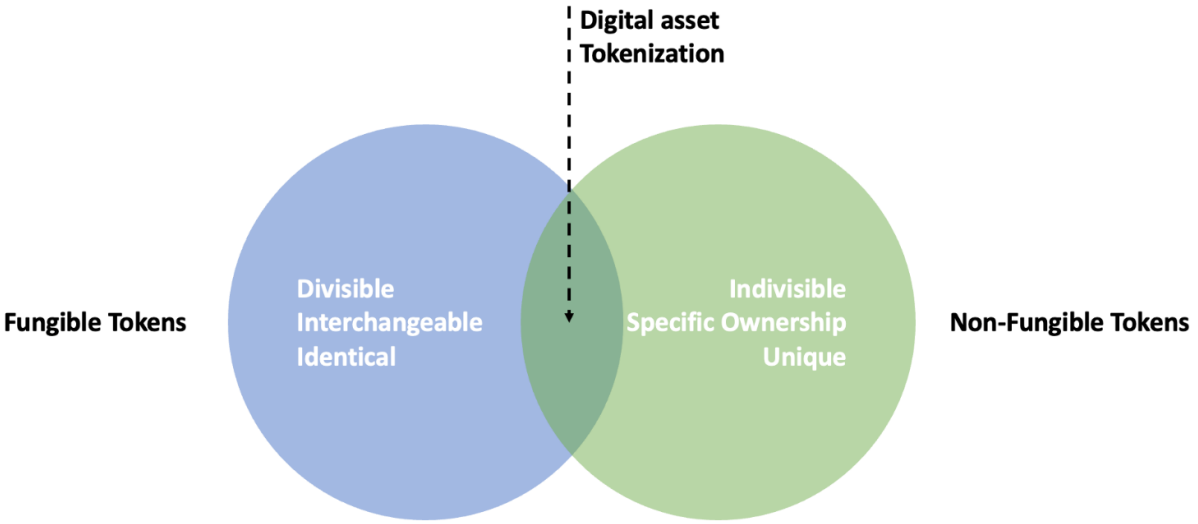

Figure 11 Digital asset tokenization

Non-Fungible Tokens (NFT): a NFT is a type of digital token that represents ownership or proof of authenticity of a unique item (e.g., artwork, collectible, music, video, virtual real estate, other digital creation). A NFT cannot be copied, substituted, or subdivided. In the SEDIMARK marketplace, a NFT represents an Offering of an asset made by a Provider; it implicitly represents the asset.

Datatokens (DTs): fungible tokens, meaning they can be divided into smaller units and are interchangeable with one another. In SEDIMARK marketplace, an asset is described by a NFT and a certain number of Datatokens are minted and made purchasable. Any Provider gives access to an asset against proof of purchase of the related Datatoken.

The SEDIMARK marketplace uses a set of Smart Contracts (SCs) to rule tokenising of assets, making offers and buying access to assets. Smart Contracts are software applications that operate on the decentralized network of validators who execute and validate the same code described in Section 3.2. One notable characteristic of such applications is their deterministic execution, ensuring that running the same code on multiple validators results in identical outcomes. This is possible through virtual machine (VM) abstraction. The smart contract applications are coded using a common language compatible with such VM.

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:

- **NFT Smart Contract (NFTSC):** ERC-721 compliant contract [1]; the NFTSC is responsible for minting an NFT representing a specific Offering of the Owner. 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 at a price determined by the Owner. Moreover, upon a Datatoken transferal, 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. Provider after making an offer add the minted Datatokens to the FRESC to enable trading of the asset at a fixed price. The price of each asset is set by the Provider, and it is expressed in native tokens. 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.

Possession of a Datatoken associated with a specific NFT (i.e., an Offering) is evidence of the purchase of access to the associated Assets.

7 Conclusions

This document is the first version of the decentralised Infrastructure and access management mechanisms. This document analysed the decentralised infrastructure and the mechanism for implementing the access control of the users of the SEDIMARK Platform.

The decentralized infrastructure and access management presented in this document lay the foundation 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.

This framework 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 with the complete set of access management policies, will be presented in the next Deliverable SEDIMARK_D4.2 in July 2025, after extending and integrating the technical works deriving from the co-joint progress and development of the other WPs.

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 v2.0. W3C Working Draft. https://www.w3.org/TR/vc-data-model-2.0/ , 2023.

[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] Ethers.js. https://github.com/ethers-io/ethers.js

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

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

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

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

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

[20] Contract state machine. https://eclipse-edc.github.io/docs/#/submodule/Connector/docs/developer/contracts?id=state-machine

[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/.