OpenDSU for Supply Chain Use Cases

Author: Sînică Alboaie, PhD, Axiologic Research
Purpose: A short guide for OpenDSU Architectures with focus on Supply Chain Use Cases
Visibility: Public
Date: May 2024
Version: 1.1

Abstract

This report delves into the use cases of supply chain management, highlighting the roles and interactions of various stakeholders within a blockchain-integrated framework. Key stakeholders, including suppliers, manufacturers, distributors, retailers, and consumers, are analysed in the context of their contributions to the supply chain. The focus is defining business logic through DSUs (Data Sharing Units) notarized on distributed ledgers, facilitating traceability, quality control, and efficient process management. The report also outlines essential architectural components such as nodes, digital wallets, and APIs, which ensure secure data access and integrity. Various DSU categories, sharing patterns, privacy and security considerations are discussed to provide a comprehensive understanding of optimising supply chain processes using blockchain technologies.

1. Supply Chain Use Cases Stakeholder Mapping

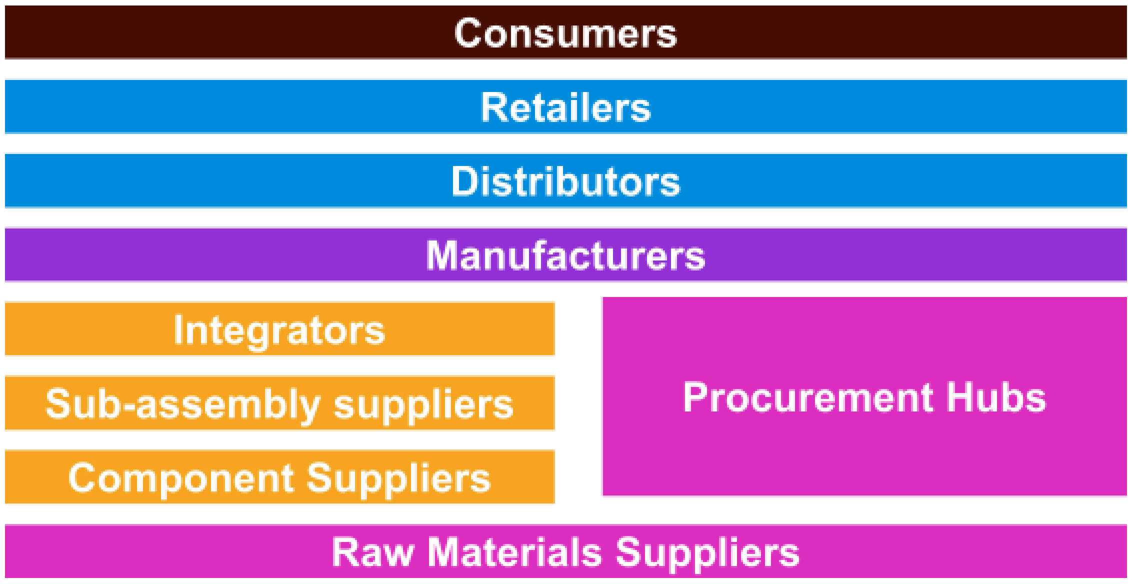

Diagram 1: Supply Chain Use Case Stakeholders

In a supply chain use case, the stakeholders can be abstracted as follows: suppliers, manufacturers, distributors, retailers, and consumers. Suppliers can be divided into several categories: raw material suppliers, component suppliers, and sub-assembly suppliers. Additionally, integrators and procurement hubs play essential roles.

Raw material suppliers provide the basic materials needed for production. Procurement hubs gather and verify these materials and components before supplying them to component suppliers and sub-assembly suppliers, ensuring quality and traceability. Component suppliers deliver individual parts assembled into final products, while sub-assembly suppliers offer partially assembled products that manufacturers will complete. Integrators coordinate and streamline these processes to ensure seamless production.

Manufacturers transform raw materials, components, and sub-assemblies into finished products. Distributors handle logistics and warehousing, moving the products from manufacturers to retailers. Retailers sell the final products to consumers, who are the end-users. At the end of their lifecycle, products may be disassembled into components, some of which are destroyed and others reused.

This report does not intend to delve deeply into supply chain specifics or the business logistics associated with each entity. Instead, we must clearly define how business-relevant entities model their business logic for each use case by updating (writing) in DSU (Data Sharing Unit) notarised in distributed ledgers. Like relational database modelling, this describes data models that outline the different types of DSUs that may appear in a concrete solution. This report introduces some concepts aiming to get a simple language for describing the categories of DSU, their data-sharing properties and recommended implementation details.

2. Architecture Components: Full Nodes, Digital Wallets and APIs

We typically identify several essential components in a supply chain system architecture that incorporate blockchain and OpenDSU off-chain storage to ensure support for data sharing and off-chain computational integrity. Digital Wallets manage keys for employees from large entities or handle all the data for smaller entities, ensuring secure data access and key management. Applications cater to users who do not require direct key management, providing access to public information or operating based on a centralised model for simplified interaction. Standard OpenDSU APIs are crucial for reconstructing and validating DSUs and implementing digital wallets by providing necessary backend services. Custom APIs using OpenDSU are developed to support specific functionalities that enhance the capabilities of digital wallets. Full Nodes underscore the system's backbone, representing the blockchain nodes and other infrastructure elements critical for maintaining data integrity and the reliability of the entire network. This architecture ensures that each component plays a specific role, optimising the effectiveness and security of the supply chain management system.

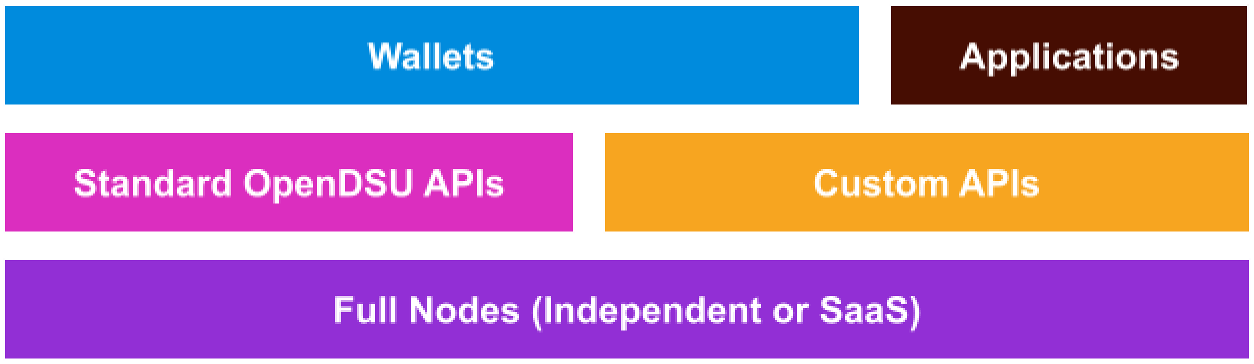

Diagram 2: Typical Components

The following table outlines various actors within the supply chain, categorised by size and operational needs, and provides recommendations for each technological approach. These categories range from large entities, such as manufacturers and distributors, to individual home users, such as consumers. Each category has distinct interaction levels and requirements for blockchain technology, from managing dedicated blockchain nodes to utilising simple consumer apps or digital wallets. This classification helps understand how different actors can optimally engage with blockchain technologies to enhance efficiency, security, and transparency in their respective roles within the supply chain.

Actor Category	Examples	Recommended Approach
Large Entities	Manufacturers, Distributors, Procurement Hubs	Dedicated blockchain nodes or SaaS nodes
Medium Entities	Retailers, Integrators	SaaS nodes
Small Entities	Sub-assembly Suppliers, Component Suppliers, Raw Materials Suppliers	Digital Wallets
Home Users	Consumers	Wallets or simple Consumer Apps

Table 1: Optimal Blockchain Technology Approaches for Different Supply Chain Actors

3. DSU Data Models for Supply Chain Use Cases

Five distinct categories have been identified in the modelling of DSUs to represent the supply chain: Quantity, Product, batch DSUs, Digital Twin, and Process State.

Quantity DSUs track raw materials through various stages, from procurement to production. Product DSUs describe each distinct product or raw material type, detailing specifications and lifecycle information to uphold product integrity. Batch DSUs focus on traceability and quality control by recording details like production and expiration date, quantities in a lot, provenience of components, etc. Digital Twin DSUs act as virtual counterparts of physical entities (e.g. finished goods), offering real-time data akin to NFTs that notarise the status of distinct physical items on the blockchain. Lastly, Process State DSUs capture supply chain processes' current and historical status, providing insights into production stages and operational efficiency, thereby facilitating better process management and optimisation.

DSU Category	Description
Quantity DSU Types	Quantity DSUs represent the measurable amounts of raw materials within the supply chain. They track the quantities of these materials at various stages of procurement, storage, and production.
Product DSU Types	Product DSUs encapsulate detailed information about individual products, including specifications, features, and configurations. They provide a comprehensive view of each product’s attributes and lifecycle data.
Batch DSU Types	Batch DSUs manage information about groups of products or materials produced or processed together. They ensure traceability and quality control by tracking batch-specific details like production date, lot number, and expiration date.
Digital Twin DSU Types	Digital Twin DSUs are essentially NFTs (tokens representing blockchain notarization of the state for distinct, non-fungible physical entities) that reflect the real-world status of an actual entity, such as a package, component, or physical product instance. They provide a virtual representation that includes real-time data and status updates.
Process State DSU Types	Process State DSUs capture the current status and historical data of various processes within the supply chain. They provide insights into production stages, workflow progress, and operational efficiency, enabling better process management and optimisation.

Table 2: DSU Categories (from the Supply Chain Use Cases)

These categories form a relatively comprehensive framework for categorising all possible DSU types in supply chain use cases implemented using OpenDSU technologies.

In DSU sharing, three distinct patterns emerge to manage how data updates and ownership are handled within a supply chain. The “Single-Writer” pattern restricts data writing to only one entity at a time, ensuring clarity and consistency in updates. The “Single-Owner” pattern allows for a unique scenario where only one entity can make updates at any given time. Still, the ownership and responsibility for updates can shift between entities, making tracking modifications across different stages possible. Meanwhile, the “Multiple Writers” pattern accommodates updates from several entities, necessitating advanced systems to resolve conflicts and maintain data integrity.

Sharing Pattern	Implementation
Single Writer	This pattern allows only one entity to write or update the DSU, ensuring precise and consistent data management without conflicts from multiple sources.
Single Owner	In this pattern, only one entity can update the DSU's state at any given time, but ownership can change over time. This allows tracking and correlating updates with the responsible entity at each stage.
Multiple Writers	This pattern permits multiple entities to write or update the DSU. It requires robust mechanisms to manage potential conflicts and ensure data integrity, allowing for collaborative data management.

Table 3: DSU Sharing Patterns

Multiple business entities typically are planned to read or write to the DSU's state in all these patterns. The grouping of these entities is part of the DSU data model. Furthermore, if distinct groups of entities require different parts of information or if access to keys allows writing changes, it is advisable to establish separate DSU types. This approach ensures finer granularity and provides long-term access control, facilitating effective data management within the supply chain.

Category	Recommended Sharing Pattern
Quantity DSUs	Single Owner (ownership will change over time to reflect the current custody)
Product DSUs	Single Writer (the manufacturer defines the data in Products)
Batch DSUs	Single Writer (the manufacturer defines the data in Batches)
Digital Twin DSUs	Single Owner (ownership will change over time to reflect the current custody)
Process State DSUs	Multiple Writers

Table 4: DSU Categories and Write Access Sharing Patterns

In the above table, we assign the sharing patterns for each DSU category to provide insight into how to describe each type of DSU as belonging to a specific category and the implicit expectation of the intended sharing pattern from the updates' point of view. From the read-access perspective, this is a parallel concern and is much more flexible.

4. Privacy & Security

In this chapter and its subsections, we will analyse the relevant considerations and factors that influence the privacy and security characteristics of OpenDSU solutions for the supply chain. We will briefly cover essential elements such as data encryption, decentralised access control, centralised access control, data integrity, auditability, anonymisation, compliance, availability, and scalability. Additionally, we will delve deeper into “Cryptographic Keys Generation” and “Cryptographic Keys Rotation” details. The discussion will also summarise the relevant considerations on decentralised access control and asynchronous message-based key sharing, along with recommended implementation strategies for various critical elements. This comprehensive analysis aims to understand how to enhance and secure OpenDSU deployments effectively and clearly.

Decentralised Access Control

A key concept central to understanding OpenDSU is "Decentralised Access Control," which can be described as granular and cryptographic key-based access control. In theory, each DSU possesses a unique control key (seed key) that can be used to sign new versions of the DSU and save these lists of versions in a blockchain. This seed key can also be used to derive a key for symmetric encryption and can be shared separately to grant read access. Sharing the "Seed" key provides write access. This establishes the foundation of a decentralised access control mechanism unique to OpenDSU.

In contrast to centralised access control systems that require a central service to grant access, possessing the right key enables read or write access. This approach aligns more closely with the decentralised nature of blockchain solutions. Introducing centralised access control could be cumbersome and may diminish security and privacy rather than enhance it. For instance, one can easily envision data accessible directly from wallets rather than through custom Web APIs. Therefore, relying on centralised access control may not always be a good approach. Decentralised Access Control offers a sound strategy for Security and Privacy by Design.

Keys Sharing Using Asynchronous Messages

Depending on the cryptographic Key Generation policies, different entities can share DSUs by sending asynchronous messages that contain read or write keys (KeySSIs in OpenDSU terminology). In this context, OpenDSU supports message queues based on W3C DIDs, a straightforward mechanism integrated with BDNS (Blockchain Domain Naming System). This system avoids the need for intermediate nodes to transmit encrypted messages between the various entities in the system, whether they are wallets controlled by users, companies or organisations in general (server-side components).

Keys Sharing By Scanning QRCodes or 2D Matrices

KeySSIs that allow access to read data from DSUs or to create new versions in more extreme cases can be shared by scanning QR codes or 2D matrices printed on packages or equipment. Different policies related to “Cryptographic Keys Generation and Keys Rotation” and the pre-sharing of gold keys can help secure these sharing processes.

Privacy & Security Models

Aspect	Explanation
Data Encryption	By default, all DSU data is encrypted at rest and in transit to prevent unauthorised access and breaches.
Decentralised Access Control	By default, each DSU has a unique cryptographically generated key (called a seed key). The decision regarding **keys generation** and **keys rotation policies** will affect this aspect.
Centralised Access Control	Centralised access Control for APIs can be implemented, which applies to APIs (OpenDSU standard APIs or custom-created). Identification and authorisation can be implemented using standard protocols, such as OAuth, and custom authorisation schemas.
Data Integrity	The DSUs are intended to be anchored in a distributed ledger. The actual ledger will affect the properties of the Data Integrity aspect.
Auditability	Custom audit logs must be constructed for each wallet or each API.
Anonymisation	By default, OpenDSU does not offer much support for anonymisation; when required, it must be implemented using the appropriate tools and techniques.
Compliance	OpenDSu offers a rich set of tools and customisation opportunities to ensure adherence to relevant laws, regulations, and industry standards (e.g., GDPR, HIPAA) to ensure that the DSU's privacy and security practices meet required legal and ethical standards.
High Availability	Mechanisms for backup, redundancy, and recovery mechanisms are enabled by the sophisticated keys management proposed by the OpenDSU.
Scalability	Blockchain technology and the decision regarding key management could affect scalability. On the performance side, the encryption mechanisms enabled by default on the DSU have an inevitable effect but can be compensated using regular scalability techniques. The nature of the anchoring and bricking mechanism enables reasonable levels of scalability.

Table 7: Sharing Patterns and Keys Generation and Keys Rotation

Cryptographic Keys Generation

From the perspective of “keys generation” methods for generating seed keys used to control the writing to DSUs (the anchoring), we have identified four methods: “Random”, “Constant Derivation”, “Path Derivation”, and “Pre-Shared Path Derivation”.

In the “Random Keys Generation” case, such a DSU is controlled by a random 256-bit key. This approach offers the highest security but can present challenges regarding “Keys Management” costs and the risk of losing control over blockchain anchors.

The "Path Derivation" method is based on the idea that keys can be derived from a "golden key." Depending on the current business object, keys are deterministically derived for each business object, typically using an HMAC function that takes the initial secret and a value like a natural key, such as an identification code or a serial number.

"Pre-Shared Path Derivation" refers to situations where a "golden key" is pre-shared between two business partners. Each partner independently applies the path derivation algorithm.

"Constant Derivation" is used when a golden key is not feasible, such as when a barcode or QR code is printed on a product, and the DSU information must be readable by anyone with the physical package or product. A "Const DSU" is created in this case, which can have only one version. It contains a key that allows reading a "mutable DSU" that holds the actual information. This technique enables read-sharing while keeping the key that allows writing to the DSU private to the DSU's initial creator.

Strategy	Description
Random	A DSU is controlled by a randomly generated 256-bit key.
Path Derivation	Keys are derived from a "golden key" using a deterministic method
Pre-Shared Path Derivation	A "golden key" is pre-shared between two business partners (or a group)
Const Derivation	A "Const DSU" contains a key for reading a "Mutable DSU" with the actual information.

Table 5: Key Generation Strategies

Cryptographic Keys Rotation

In the context of OpenDSU, key rotation is a crucial security practice for managing access and protecting data integrity over time. Given the dynamic nature of digital environments, it becomes essential to understand and implement effective key rotation strategies.

Three main strategies are analysed for the "keys rotation" within OpenDSU: "Never", "Ownership Change", and "Destroy & Recreate". Each strategy has specific applications and implications depending on the lifecycle and sensitivity of the data involved.

The Never strategy suggests that the key is never rotated. This can be a viable alternative, especially for DSUs representing objects with a short and ephemeral life. In such cases, key rotation may be unnecessary because the data becomes obsolete quickly, or the DSU is decommissioned before any security risk escalates. However, for long-term security, it's essential to consider rotating the golden keys used for path derivation, which ensures that new instances are safeguarded even if the base key remains the same.

The Ownership Change strategy involves rotating the key whenever there is a change in ownership. This method maintains the continuity of the DSU's identity but allows for the deletion of old bricks encrypted with outdated keys. This approach is advantageous in environments where DSUs frequently change hands, ensuring that previous owners cannot access the new encrypted data.

Finally, the "Destroy & Recreate" strategy is the most drastic. It involves deleting the old DSU entirely and creating a completely new DSU. While the old anchor may remain, all metadata in the system is updated, effectively resetting the security parameters. This method is suited for situations where a complete renewal of the DSU is necessary, such as when transitioning to entirely new operational parameters or when a significant security breach has been identified.

Keys Rotation	Description
Never	The key is never rotated because the relevant life period of the DSU is short enough.
Ownership Change	The key is rotated whenever there is a change in ownership. Alternatively, enterprise policies regarding key rotation can force a change in ownership. There is continuity in DSU identity, but old bricks encrypted with the old keys will be deleted or marked as not accessible anymore.
Destroy & Recreate	The old DSU has been deleted, and a new DSU has been completely recreated. Only the old Anchor remains, but all the off-chain data from the system is encrypted again with the new keys.

Table 6: Possible Key Rotation Pattern (for Seed Keys)

Overall, selecting the correct key rotation strategy in OpenDSU depends on balancing security needs with operational efficiency. Each method offers different levels of protection and implications for data management, making it crucial to align the chosen strategy with specific business requirements and the nature of the data handled.

Recommended Implementation Strategies

The following table provides detailed implementation suggestions for various architectural elements commonly used in supply chain systems. These elements include Single-Writer DSUs, Single-Owner DSUs, Multiple-Writer DSUs, Digital Wallets, Key Management, Web APIs, and Blockchain (Ledgers). Each entry outlines recommended practices and strategies to optimise these components' performance, security, and efficiency within a blockchain-integrated supply chain framework.

Element	Recommended Implementation
Single Writer DSU	Normal DSUs are made for this pattern. We expect that most DSU types conform to this pattern
Single Owner DSU	DSUs with a change in ownership at the anchor and cryptographic control keys.
Multiple Writers DSU	SVDs (Self-Validating Data) saved in DSUs could be an excellent approach to ensuring computational integrity. A small group of entities will get access to the write keys and be able to add transactions to the SVD.
Digital Wallets	Two techniques can be used to implement digital wallets: VersionLess DSU, which offers a type of DSU without versioning, and SSApps (Self Sovereign Applications), which are based on packaging the wallet code and data in normal DSUs. Each approach has advantages and disadvantages.
Keys Management	An " Enclave " abstraction offers a generic interface for using keys for DSU anchoring and DSU encryption. It also supports W3C DID methods that use DSUs to secure the DID Documents.
Web API	An OpenDSU solution can use custom APIs created for various wallets and applications. A pragmatic approach could be to use the recommended Key Management based on OpenDSU Enclaves.
Blockchain (Ledgers)	OpenDSU is agnostic regarding the blockchain technology or the distributed ledger that can be used. Any ledger can be used to implement anchoring if it supports some sort of smart contract to implement anchoring.

Table 7: Recommended Implementation Methods

Firstly, from the above table, we have to observe that flexibility is a crucial aspect, with each element offering multiple methods and strategies for implementation tailored to specific use cases and requirements. For example, digital wallets can be implemented using VersionLess DSUs or SSApps, each presenting unique advantages. Secondly, ensuring data security and integrity is paramount, as demonstrated by having various methods of implementing “Enclaves” while still keeping the code portable between different environments like web browsers, mobile applications, and cloud APIs. Lastly, the OpenDSU emphasise customisation and integration, allowing for high adaptability. OpenDSU's support for various blockchain technologies and the ability to develop custom APIs ensure that the system can effectively meet diverse operational contexts and business models.

5. Typical Design Steps for Implementing OpenDSU-based systems

The following table outlines the typical design steps and recommended implementations for developing OpenDSU-based systems. These steps involve making critical decisions about DSU modelling, sharing patterns, key sharing, and more. Each step is crucial for ensuring the system's security, availability, and scalability.

Steps (Decisions)	Recommended Implementation
Architecture Elements	Decide what element will be used for each stakeholder (full node, wallet, application). Decide on the main APIs for each type of full node that will be available.
Blockchain Technology	Decide on the Blockchain(s) that will be used. OpenDSU is agnostic and can be used with most blockchains or ledgers. Sometimes, even centralised ledgers (databases or file systems) could be enough for a lean start.
DSU Modeling	For each DSU Category, decide what DSU Types must be created.
Sharing Patterns	Decided for each DSU. Type the sharing groups for reading and write
Keys Sharing **Key Generation**	Decide how the sharing of various keys (seed/write, read) keys will be done (messages, pre-shared golden keys, 2d matrices)
Keys Rotation	Choose what policies apply. Return to the previous step and improve based on the Cryptographic Keys Rotation policies if necessary.
Analise Availability	Design and implement custom methods to ensure high availability.
Analise Scalability	Design and implement custom methods to ensure the required level of performance and scalability.

Table 8: Mandatory steps in designing OpenDSU-based solutions

By following these steps, you can systematically address the critical aspects of implementing secure and efficient OpenDSU-based systems tailored to the specific needs of your supply chain environment.

6. Conclusions

In conclusion, OpenDSU is a specifically tailored framework to enable the integration of blockchain technology into supply chain management. It is made purposely to significantly enhance transparency, security, and efficiency across the network of stakeholders. Businesses can ensure precise and consistent data handling by leveraging DSUs for data modelling and employing robust key management strategies. The proposed architecture components, including nodes, digital wallets, and APIs, are critical in maintaining data integrity and secure access. Different DSU sharing patterns address the varying needs of entities within the supply chain, from single-writer scenarios to multi-writer collaborative environments. The implementation of OpenDSU solutions, with careful attention to privacy and security considerations, offers a scalable and compliant framework for modern supply chains. The report provides actionable insights into optimising supply chain operations through advanced blockchain applications, fostering a more resilient and efficient system.

7. Prerequisite Concepts

The following table enumerates a series of fundamental concepts essential for understanding the implementation and operation of OpenDSU-based systems. These concepts are explained in detail on the OpenDSU website and are referenced throughout this report as assumed knowledge. Familiarity with these terms is crucial for comprehending the discussed architectural elements and their applications within the supply chain framework. The table serves as a quick reference to ensure clarity and continuity for readers.

Concept	Very Short Description
DSU	Data Sharing Units [1]
KeySSI	Key Self Sovereign Identifier [1]
Seed Key	A KeySSI that controls a specific DSU [1]
Write Key	Another name for a Seed Key can update a version of a DSU because it can sign a new version in an anchor. [1]
Read Key	A KeySSI is derived from the seed keys and used for the encryption and decryption of the DSUs. It can be used to create a new version because it cannot derive a private key to sign a new version. [1]
AnchorID	A KeySSI stored in blockchain and identifying an anchor [1] [3]
DID	W3C Decentralised Identifier [1] [2]
BDNS	Blockchain Domain Naming System [1]
Enclave	OpenDSU Enclaves abstract databases storing sensitive data (for example, metadata about DSUs and other data used by APIs or wallets). The physical storage of an enclave can be stored in DSUs or standard databases and associated vaults or KMS for storing golden keys) [1]
Golden Key	The key helps generate other “seed keys” using an HMAC derivation approach (path derivation). Aimed to simplify the number of keys managed by an enterprise, help with backup and recovery and even with “keys sharing” between partners. [1]
NFT	Non-fungible tokens represented as DSUs [3]

Table 9: Prerequisite Concepts used in this report and not explained

8. References

[1] OpenDSU Documentation site, www.opendsu.org
[2] W3C DIDs, https://www.w3.org/TR/did-core/
[3] NFTs in OpenDSU https://www.axiologic.net/downloads/opendsu_and_tokenization.pdf