Data Platform for the Connection of Cognitive Ports

Title:	Document Version:
D2.3 – Blockchain design specification M09	1.1

Project Number:	Project Acronym:	Project Title:
H2020-871493	DataPorts	A Data Platform for the Cognitive Ports of the Future

Contractual Delivery Date:	Actual Delivery Date:	Deliverable Type*-Security*:
M9 (September 2020)	M9 (September 2020)	R-PU

*Type: P: Prototype; R: Report; D: Demonstrator; O: Other; ORDP: Open Research Data Pilot; E: Ethics.

**Security Class: PU: Public; PP: Restricted to other programme participants (including the Commission); RE: Restricted to a group defined by the consortium (including the Commission); CO: Confidential, only for members of the consortium (including the Commission).

Responsible:	Organisation:	Contributing WP:
Daniel García Latorre	EVR	WP2

Authors (organisation):	
Daniel García Latorre (EVR)	Yuriy Yatsyk (ITI)
Alejandro Aparicio Blasco (EVR)	Konstantinos Votis (CERTH)
Daniel Vilas Perulan (EVR)	Sofia Terzi (CERTH)
Miguel Llop Cabrera (VPF)	Vasilis Siopidis (CERTH)
Pablo Giménez Salazar (VPF)	Fabiana Fournier (IBM)
Elsa Koukouloudi (ThPA)	Inna Skarbovsky (IBM)

Abstract:

This document provides the functional analysis performed to define the blockchain technology that best fits the project necessities, and all the technical characteristics of the necessary components and connections to integrate this technology in the platform

Keywords:

Blockchain, Distributed Ledger Technology

Revision History

Revision	Date	Description	Author (Organisation)
V0.1	18.03.2020	Initial draft	Daniel Garcia Latorre (EVR)
V0.2	06.05.2020	Added content to sections	Daniel Garcia Latorre (EVR)
V0.3	08.07.2020	EVR internal iteration	Alejandro Aparicio (EVR), Daniel Vilas (EVR)
V0.4	01.09.2020	Input from several partners	Alejandro Aparicio (EVR), Daniel Vilas (EVR)
V0.5	08.09.2020	Pre-final version after partners feedback	Alejandro Aparicio (EVR)
V1.0	11.09.2020	Final draft after second round of feedback	Alejandro Aparicio (EVR)
V1.1	25.09.2020	Final version	Daniel Garcia Latorre (EVR)

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement № 871493.

More information available at https://DataPorts-project.eu

Copyright Statement

INDEX

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Blockchain technology will be a core component of the DataPorts platform, taking care of data sharing and governance policies, and helping the pilots to build their use cases around the potential benefits of the technology.

1.1 PURPOSE OF THE DOCUMENT

This document establishes the technical specifications for the development and deployment of the blockchain components of the DataPorts platform. Functional and technical requirements have been analysed for both the platform and the use cases. Furthermore, to introduce the reader to the current blockchain applications ecosystem, a selection of technologies have been explained thoroughly, and a detailed comparison between them will be used to choose the most suitable technology for the project.

The use cases from Thessaloniki Port and Valencia Port for the DataPorts platform are presented, accompanied by a network design proposal that will suit the necessities derived from the analysis and development of the use cases.

1.2 SCOPE OF THE DOCUMENT

Deliverable D2.3 focuses on the functional analysis to be performed to design the blockchain components that will be part of the DataPorts platform. The description of use cases and analysis of available technologies help determine the most suitable blockchain network design as well as the necessary components and connections to integrate.

1.3 STRUCTURE OF THE DOCUMENT

This document is structured as follows:

- Section 2 includes a brief description of blockchain technology and a thorough analysis and comparison of existent blockchain implementations to seek the most suitable technology to use in DataPorts.
- Sections 3 and 4 contain a description of the use cases to be deployed and the requirements that derive from them.
- Section 5 contains the global DataPorts scenarios to be included in the blockchain platform, as well as an explanation as to why blockchain is suitable for such scenarios.
- A design proposal for the blockchain components is presented in Section 6
- Main conclusions and future work to be undertaken are described in Section 7
- Section 8 includes lists for references and acronyms
- The Annex section is comprised of technical data of the use cases provided by the pilots

2 BLOCKCHAIN TECHNOLOGIES

Described by many as the future of digital money, mainstream blockchain technology applications appeared in the early 2010s in the form of cryptocurrency transactions. Assets in a blockchain are represented as a result of cryptographically signed transactions between network participants in the ledger, the final balance of every account being the net sum of incoming transactions minus transactions sent. A cryptographic key (e.g. a very long randomly generated password) is the only way a blockchain user can claim ownership of their assets. This process entails a series of operations that bring multiple benefits to business processes that will be explored in following chapters.

Blockchain is presented in this section as a core part of the DataPorts platform as it is needed for data integrity verification and the introduction of data governance policies. Furthermore, some blockchain concepts are explained in the following subsection for the sake of completeness, and to help the reader understand the decisions taken during the network design process.

2.1 BASIC BLOCKCHAIN CONCEPTS

Blockchain is a distributed ledger similar to a database that forms a permanent, tamper-proof record of transactional data.

A DLT, or blockchain, works as a decentralized database (Figure 1) that is managed by different computers belonging to a peer-to-peer network, or P2P (peer-to-peer). Each of the computers in the distributed network maintains a copy of the ledger, which avoids a Single Point Of Failure (SPOF) because all copies are updated and validated simultaneously.

Figure 1 – Distinction between P2P network models

The transactions that make up the ledger are stored in the form of blocks. Each block stores the information about the transactions made, the time the block was added to the chain, and hash of the previous block (see Figure 2). In this way, the validity of future transactions can be verified by consulting the last state of the address.

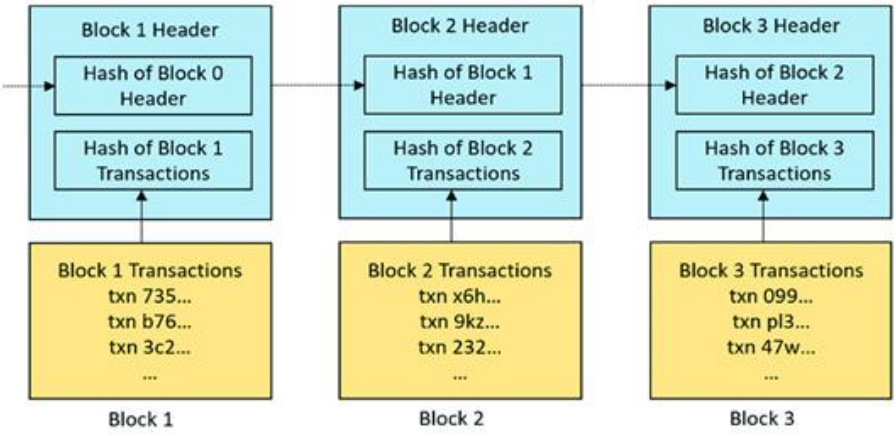


Figure 2 – Data structure of a blockchain

The main components of blockchain technology are:

- **Cryptography:** Set of encryption, signature, and access control techniques from different disciplines of cryptography that allow transactions to only be registered or consulted by who is authorized.

  Hash functions: these are functions that allow you to encrypt any input data, generating a hash, which is encrypted text of a fixed length. It has several properties:

  - Obtaining the original data from the generated hash is practically impossible.
  - The same output hash is always obtained for a specific input.
  - Any modification of the input data generates a completely different hash.
  - These characteristics allow the verification of the integrity of a message and guarantee that it has not been tampered with

  Asymmetric Cryptography: Also known as public key cryptography, it allows secure communication between two parties through the use of a private and a public key.

  The private key is kept secret by the owner of the key pair and the public key will be shared. A message encrypted with the private key can only be decrypted with the public key and vice versa.

  Digital signature: Combination of the two previous methods, consists of applying a hash function on the data set that we want to exchange and encrypting the resulting hash with a private key. A signature is obtained and attached to the data submission.

- **Consensus:** A set of rules accepted by all participants, which translated into executable code, verifies, validates, and distributes the transactions to all parties. These allow the different participants to agree when validating the transactions that are requested on the network and that generate the blocks that make it up.
- **Blocks**: Each package of transactions is grouped into a block, which, once validated by the consensus mechanism, is added after the immediately previous block, thus generating a chain that cannot be manipulated without generating alerts.
- **Smart contracts**: Clauses and conditions translated into code, which embedded in the database can be executed by applying business logic to internal and external information to the blockchain. Smart contracts are deployed within the blockchain network so that their content cannot be modified; thus, they cannot be controlled by a single actor.

At its core, a blockchain is a ledger through which data is added and updated in real-time via consensus of the different nodes running the software in the network.

2.2 MAIN BENEFITS OF BLOCKCHAIN TECHNOLOGY

One definition for blockchain is that of a distributed database copied multiple times among different participants, cryptographically protected, and organized in blocks of mathematically related transactions.

It is a system that allows parties that do not fully trust each other to maintain a consensus on the existence, status and evolution of a number of shared factors, based on trust and consensus is built from a global network of computers that manage a large data structure. It can be open to the participation of anyone who wishes (public blockchain) or limited to only some participants (private blockchain)

Blockchain combines a series of cryptographic algorithms like public-key-cryptography, elliptic-curve (EC) digital signatures and hashing which together guarantee the following key characteristics:

- Decentralized network: There is no central authority and no central data storage, i.e. no single point of trust, vulnerability, or failure.
- Trustlessness: A blockchain does not require trust in any authority or every participant.

- Consensus-based network: A process allows participants to come to an agreement over what (e.g. the validity of a transaction) is true or false.
- Transparent and traceable transactions: all transactions in a blockchain are visible and verifiable to participants with rights.
- Immutable transactions: transactions and blocks added to the blockchain are technically impossible to manipulate or modify.
- Security: assets in the blockchain are cryptographically secured, and due to its decentralized nature, there is no single point of failure being Denial of Service resistant by design.
- Pseudonymous and anonymity: implementations in block chain vary a lot in this respect, most chains allow implementing pseudonyms for id nodes in the network, but anonymity levels tend to be low due to traceability in transactions. The validity of all transactions is available to everyone on the network

Public blockchains are implementations of the distributed ledger where the data inside the blockchain (transactions) is open to the public and everyone can take part as a node while private blockchains also called permissioned blockchains operate inside a previously defined network of participants.

2.3 BLOCKCHAIN TECHNOLOGIES

This section will provide a detailed description of the available blockchain technologies to be considered for the DataPorts platform.

2.3.1 Ethereum

In 2014 the Ethereum proposal [1] by Vitalik Buterin came to light. It was presented as a platform also open as Bitcoin, but for the creation of decentralized applications. The implementation of smart contracts [2] changed the rules of the game in the blockchain ecosystem, since the possibility of executing a logic that generated a deterministic and immutable result appeared. This change meant an expansion of possible blockchain applications, opening a creative path for programmers that goes further than cryptocurrency exchanges.

The most relevant aspects are explained at a high level below.

2.3.1.1 Nodes

In this case, a node is a device that communicates with the Ethereum network. As it is a public network, as in Bitcoin, the use of this network does not require a large economic investment in its infrastructure, it is only necessary to pay for the use.

The classification of node types in Ethereum is similar to that of Bitcoin:

- Full nodes: Responsible for verifying the blocks that are broadcast on the network to ensure that the blocks and transactions comply with the rules of the network. The full nodes keep the network online and provide full auditability of the ledger.
- Light nodes: They do not carry out verifications or maintain the total state of the network, they depend on full nodes to obtain information.

It should be noted that the data is not kept encrypted, with the implications that this entails from a point of data privacy.

On the other hand, since each Full node is a complete replica, they all contain the same information. The high redundancy of the data provides high availability of the data.

2.3.1.2 Network

Ethereum from a communications point of view uses a P2P network based on the Kademlia network[1]. The nodes contain a table with network addresses and a timestamp signed by the node. In this way, Denial of Service (DoS) attacks are mitigated, since false nodes need to make the effort to generate addresses that correspond to a key pair. It should be noted that the transactions involve a cost for the use of infrastructure provided by third parties, being necessary to have the network cryptocurrency (Ether). Any node can generate transactions and depending on the commission that you leave to the miners, it will be processed sooner or later.

Ethereum does not implement privacy on the network, everyone can see all transactions, it is also necessary to have Ethers (network cryptocurrency) to be able to pay for transactions.

Crashes of nodes or network partitions do not affect the Ethereum network as long as the set of honest nodes in the main partition is at least 51% just like in Bitcoin.

2.3.1.3 Status

The state in Ethereum is broken down into two parts:

- Transactions, represented by state transition functions (string data)
- The result when executing said functions (status data)

This separation allows you to start a node without downloading all the transaction history; it is enough to have the state of the accounts synchronized.

As in Bitcoin, tree-type structures are used, but a slight variation is contemplated using the so-called Modified Patricia Trie. Trie is a tree-like data structure that allows information retrieval (reTRIEval), it represents an acyclic deterministic finite automaton that serves to store a set of strings. To overcome storage inefficiency problems, complexity is added to the data structure. In this way, the improvements introduced with respect to a binary tree are:

- Faster key search
- Less space required to store large number of small keys
- Better performance for searching the longest prefix

That said, a full node stores transactions, state transitions, and resulting state for all block heights in a local data store. This includes all historical data, even data that is no longer valid. This way customers can check the status of the blockchain without having to recalculate everything from scratch. The chain data is necessary to guarantee the cryptographic chain of custody. Old state data (pruning) can be discarded as it is implicit data and can be calculated. The states are assembled in a state tree linked to the account and the blocks. The nodes use LevelDB[2] (Database Key-Value Pairs) to store the state. The advantages of state management in Ethereum over Bitcoin are:

- Simplicity: It is a more intuitive model (benefit for developers of complex smart contracts).
- Efficiency: Each transaction only needs validation that the shipping account has sufficient balance to pay.

A disadvantage of the Account / Balance model is exposure to double-spending attacks. To avoid this, Ethereum makes each account have a nonce that is increased with each transaction, so as not to send it more than once.

[1] http://xlattice.sourceforge.net/components/protocol/kademlia/specs.html

[2] https://github.com/google/leveldb

From the perspective of ensuring immutability, each block header within the Ethereum blockchain contains a hash of the Modified Patricia Trie representing the transaction set and a hash of the previous block header. Combining this cryptographic chain with the consensus mechanism ensures the immutability of the data, since a modification of a data involves recalculating all the hashes of the chain and it is a very high computational and temporal cost. This makes it, as in Bitcoin, more profitable to use that computational potential to earn rewards for new mined blocks.

In the Ethereum network there is no purpose either, so the proposals for new blocks appear, and forks may occur in the chain, where the block that is part of the longest chain will finally prevail. Therefore, there is a possibility that the transactions in a block may no longer belong to the main branch of the blockchain. To solve this problem, a purpose mechanism has been developed called Casper the Friendly Finality Gadget (Vitalik Buterin 2018), it is a purpose system planned for 2019, its purpose is to ensure that the blocks cannot be reversed by making main branch changes. Currently, in the Ethereum network, it cannot be ensured that a transaction is completely firm, until the block that contains it is 25 more valid blocks.

In the case of this blockchain network, the state changes are made by the Ethereum virtual machine in English Ethereum Virtual Machine (EVM). It is an execution environment based on the stack architecture. It processes user-generated data to cause changes in the state of the Ethereum network. It consists of:

- Virtual ROM (Immutable EVM Code)
- Key-value storage (Persistent storage)

When executing a contract, it gets a new instance of memory for each call. So, it is impossible to access memory areas between different smart contracts or their executions. Therefore, all the information that needs to be shared must be specified in the call parameters. The consequences of using a virtual machine with stack architecture:

- Shorter instructions
- Management and number of registers in each architecture does not depend / Central Processing Unit (CPU)
- Slower processing

The restrictions posed by this model are as follows:

- All communication steps have to be paid to prevent DoS attacks. Contracts can communicate with each other only by transmitting vectors of arbitrary size (there is no state comparison between them). The execution of smart contracts is isolated.
- EVM does not have a hierarchical flow control structure, which makes analysis difficult. This involves a number of difficulties when auditing:
- Do functions return what they should?
- Does the compiler create a bytecode according to the source code?
- Compilers, optimizers, and complex helpers

This last restriction is relevant because it implies a greater difficulty for the developer when implementing Smart Contracts.

2.3.1.4 Consensus

In the implementation of Ethereum the PoW (Proof of Work) consensus mechanism is used, specifically Ethash. This mechanism is used because, as has been said, it serves to protect the service from abusive use (for example: spam or DoS attacks), something relevant in an open network.

As mentioned, the key to its operation is that the clients of the service carry out some type of work that involves a computational cost or time and is easily verified if it satisfies certain requirements. As in Bitcoin,

the difficulty of mining to keep the network secure increases and this implementation is also energy inefficient. The main difference from Hashcash is that Ethash is a Memory Hard algorithm (Lerner 2014), this implies that large amounts of data are used to produce the proof of work to cause cache failures and thus limit the process to memory accesses. This makes the mining algorithm resistant to ASIC (Application-specific integrated circuit) mining.

Block generation time on the Ethereum network is around 12 seconds. This time was established taking into account the size of the block and the reward mechanism.

This implementation does not have a solution to Selfish Mining Attack (Ittay Eyal 2014) and there is also the possibility that users who pursue a legal mining strategy but not honestly use bribery through Smart Contracts (Patrick McCorry 2018) [3].

2.3.1.5 Smart Contracts

As mentioned, it is in Ethereum where Smart Contracts are successfully implemented for the first time. These are software programs that collect the terms of a contract between the parties and are stored in a blockchain network. These programs are executed when a series of conditions specified in the contract itself are met. In this way, the Smart Contract within the blockchain is equivalent to an instance of an object in the paradigm of Object-Oriented Programming.

The deployment of the Smart Contract in Ethereum is carried out through a transaction with which the code is uploaded to the network. Through its use, the integration of blockchain with business logic is facilitated, since they allow to automate payments, create DAPs (open source applications that communicate with a Smart Contract of a public network), create Tokens, manage DAOs (autonomous decentralized organizations). The language most used for the development of Smart Contracts in Ethereum is Solidity.

2.3.2 Quorum

Quorum[3] is a licensed blockchain platform developed by JPMorgan Chase [4]. It is based on the official implementation of Ethereum. The development was oriented to business environments where groups of known participants operate exchanging private transactions.

Currently, it can be highlighted that it is used by Alastria[4], a multisectoral consortium that works to adapt the Quorum platform to the needs of the Spanish market and regulation.

The most relevant aspects are detailed below.

2.3.2.1 Nodes

The implementation of the nodes is based on the Ethereum client, taking advantage of all the work behind this community. For new functionalities they are implemented in a new module.

There are two implementations: Constellation implemented in Haskell[5] and Tessera implemented in Java. Both modules are made up of two submodules: Node used to implement the Transaction Manager and Enclave.

- Transaction Manager. It is the module responsible for managing private transactions. Stores and provides access, exchanges the encrypted data with the other sub-modules of the other participants. It is a restful and stateless element.

[3] https://consensys.net/quorum/

[4] https://alastria.io/

[5] https://www.haskell.org/

- Enclave. It is a module isolated from the other components that works together with Transaction Manager, it is in charge of cryptographic functionality, also of generating symmetric keys for data encryption.

As it is a permitted network, an infrastructure is needed to host the nodes, this infrastructure in turn requires management and maintenance. All this expense is passed on to the owner or owners of the network.

2.3.2.2 Network

From a communications point of view, the Ethereum P2P network has been modified to accept connections exclusively from authorized nodes. Transaction creation has been modified to allow data to be replaced by hashes in order to preserve your privacy. The price of GAS has also been removed, so there are no costs when making transactions.

Transactions are divided into two types; On the one hand, there are so-called public transactions that are visible to all members who have access to the Quorum network. The creation of these transactions is done just like in the Ethereum network, all the participants execute the same code and make the same changes in the state. On the other hand, there are the private transactions, they are visible exclusively by the participants whose keys are specified in the parameters of the transaction. For the creation of a private transaction, first the data of the transaction is encrypted, and the hash is calculated to put it as the content of the transaction. Members who have permissions can retrieve the transaction data by establishing a direct connection, otherwise they will only see the hash.

In this network, the owners must provide mechanisms so that they send the queries of the end users to the servers of the permitted network since they cannot be exposed for security reasons. Another responsibility for homeowners is to control node drops and network partitions.

2.3.2.3 Status

The generation and validation of blocks have been modified to replace a single data structure, capable of validating the current state in two parts. The public part of the state is visible to all users on the network; everyone must reach consensus and apply the same status changes.

But the private state of each node is different, due to the fact that the validation logic to manage the transactions is not the same, the private transactions are not executed in all the nodes, for this reason, it is not possible to reach a consensus with this part. of the data.

As in Ethereum, the immutability of the data is ensured by the cryptographic chain and the consensus algorithm. In addition, there is a purpose in this network, since the generation of new blocks is carried out with communication consensus algorithms, for which no forks can occur in the blockchain because all the nodes are aware of the data they incorporate. Knowing that a transaction made cannot be reversed is an important factor at the business level.

2.3.2.4 Consensus

Algorithms based on voting are used to reach consensus. Make consensus through mechanisms other than PoA[6], such as Raft[7] or Istanbul BFT[8].

- Raft to achieve a high rate of block generation, purpose, and creation of blocks on demand.
- Istanbul BFT, Byzantine fault tolerance consensus, inspired by PBFT, for transaction purposes.

[6] https://github.com/ethereum/EIPs/issues/225

[7] https://docs.goquorum.consensys.net/en/latest/Concepts/Consensus/Raft/

[8] https://docs.goquorum.consensys.net/en/latest/Concepts/Consensus/IBFT/

In the allowed networks, block generation time is configurable; factors such as the reduced number of nodes compared to a public network and the consensus mechanism allow it. When using Raft as a consensus mechanism, the default block generation period is 50ms.

Permissioned networks do not have the same vulnerabilities as public networks, access is controlled, only identified members of the network can participate in the consensus algorithm

2.3.2.5 Smart Contracts

Quorum uses Solidity to create Smart Contracts in the same way as in Ethereum. These can be both public and private, that is, visible to all network participants or visible only to a set of participants. The only thing that is necessary for the Smart Contract to be private is to establish in the transaction a list of participants with access.

2.3.3 Hyperledger

Hyperledger was launched in 2015, it is an initiative for the development of DLT technologies, created to advance industry-related blockchain technology. The development is collaborative and is organized by the Linux Foundation, which includes leaders in finance, banking, the Internet of things, supply chains, manufacturing, and technology.

Currently, Hyperledger contains projects of two types: blockchain platforms and tools for their management. Among these, Hyperledger Fabric [5] stands out as a DLT platform with permissions that thanks to its modularity offers versatility for a wide set of use cases at an industrial level.

The most relevant aspects are detailed below.

2.3.3.1 Nodes

They are the main component of the network, they can be created, started, stopped, reconfigured, and deleted; Unlike a traditional blockchain node, they must have an administrator for proper management. The nodes are provided by the organizations belonging to the consortium that operates the deployed Hyperledger system; therefore, it is necessary to make an investment in infrastructure.

The nodes are identified with digital certificates and belong as stated to an organization. The nodes can be of three types:

- Client - Node that hosts a client application on the network. This application proposes transactions by invoking the functions implemented in smart contracts displayed in the ledger.
- Peer - They can store more than one ledger (participating in several channels), they can contain smart contracts (called chaincode in Hyperledger terminology) for status modification. The roles that peers can play are as follows:
    - Committing peer: It refers to all the nodes of a channel, they receive transaction blocks and validate them before incorporating them into their copy of ledger.
    - Endorsing peer: Nodes that have one or more chaincodes installed and clients that use them are called.
    - Leader peer: If the organization has multiple nodes in a channel, the leader node is the one that is in charge of distributing the transactions of the channel to the other members.
    - Anchor peer: When a node has the need to communicate with another node in another organization, it can use one of the channel's anchor peers that are configured for that organization. (It is an optional role).
- Orderer - Node that participates in the Ordering Service - service in charge of generating the blocks for each channel. A special type of node is the Orderer, they are in charge of ordering the

transactions, grouping them into blocks and spreading them. It is a modular service that can support different consensus protocols.

Depending on the number of nodes contributed, a higher level of replication and fault tolerance will be achieved. In fact, from a fault tolerance point of view it must be borne in mind that on one hand the Ordering Service nodes go - because the same Ordering Service is used by all channels - and on the other the Peers and Clients of each channel.

2.3.3.2 Network

It is the infrastructure that allows interaction between client applications with chaincodes and ledgers. Normally the network is made up of a set of organizations that come together as a consortium and establish the network's permissions, agreeing on a set of policies that can be modified in the future.

It should be noted that the network in Hyperledger is structured in channels. In fact, a channel is a private sub-network between two or more peers, so that only the organizations enabled to do so can participate in it. Each channel maintains its own blockchain, along with its smart contracts, and only channel participants can view that blockchain and the transactions sent to it. In addition, the consensus on the next block, and the transactions included in it, is done by channel. Therefore, it can be understood that each Channel is equivalent to a different blockchain.

Thus, apart from the nodes mentioned in the previous section, the network contains other elements such as: policies, channels, organizations, membership; to isolate the data as much as possible so that only authorized nodes have access. This allows for great configuration of network governance, while providing adequate levels of data privacy.

In this network, the owners must provide mechanisms so that the end users can interact with the applications deployed on the client nodes, and in this way interact with the different existing channels. As indicated, it is the responsibility of organizations to control the drops of their nodes and network partitions.

2.3.3.3 Status

In Hyperledger Fabric the state is saved per channel and is made up of two main elements, called: World State and Blockchain.

The World State is a distributed database structure that contains the current state of the channel. On the one hand, the advantage of maintaining this database is that it makes it unnecessary to traverse the entire blockchain to calculate a certain value. On the other hand, a database provides efficient methods for storing and consulting information.

Instead, the Blockchain, as its name indicates; represents the channel blockchain. Each block stores a set of transactions, where each block is cryptographically linked to the previous block. In Hyperledger Fabric, unlike World State, all this data is physically saved as a file. The choice of this design is due to the small set of operations that are carried out on the data. Mainly you need to add data to the end of the blockchain and the query is a very rare operation. It is the Blockchain that guarantees the immutability of the data.

State changes in a channel occur when the client applications interact with the chaincode stored in the peers proposing transactions.

Data privacy is achieved in two ways. On the one hand, through the channels themselves. Thus, of all the organizations in a network, only a subset of them belongs to the channel. Each channel has its own ledger, and only the organizations participating in the channel can access its content. In addition, private transactions can be created within the same channel, so that only a subset of the channel's organizations can see it. In this case, the sensitive data does not pass through the Ordering Service but is sent directly to the peers that each organization authorized to see that private information has on the channel. As an alternative, customers can encrypt their data before making the transaction and then establish a mechanism to share the key with the recipient.

2.3.3.4 Consensus

Strictly, Hyperledger does not follow a traditional consensus approach like other Blockchain technologies. There are no message exchanges between network participants to determine the next block in the chain as it is done in Bitcoin or Ethereum. Instead, the Ordering Service comprised of the Orderer nodes is trusted. These nodes are in charge of ordering the transactions and forming the blocks to produce a deterministic state change. Thanks to this, it is achieved that the purpose of the transactions is immediate once a new block is added in the peer of each channel.

The sorting service can be implemented in different ways. From a centralized model for development to a distributed one to tolerate failures in nodes and communication. As it is a modular component, it has the advantage that its implementation can be adapted to particular solutions to tolerate network failures, node dips and/or Byzantine failures. It is possible to have more than one sorting service for the different applications.

Fabric currently offers several consensus protocols. One of them is CFT (crash fault-tolerant) implemented with Kakfa[9] and Zookeeper[10]. At the moment, it does not have any byzantine fault tolerant (malicious behaviour) although they are foreseen.

2.3.3.5 Smart Contracts

Smart Contracts in Hyperlegder are called chaincode. These can be implemented in common technologies like Java, Node.js or Go. The chaincode offers functions to be invoked from client applications through the sending of transactions, to consult the information available in the ledger and / or modify it.

In Hyperledger there are two types of chaincode: user - programmed by the blockchain operators - and system - which provides the logic required by Hyperledger to execute part of the transaction flow.

2.3.4 Corda

Corda[11] is an open source environment developed by the American blockchain company R3 [6]. It is a platform focused on working with financial agreements and business processes in the blockchain, guaranteeing full transaction traceability, high capacity of integration with existing processes, and seamless auditability. Corda is often sold as BaaS (Blockchain as a Service), with R3 providing customer service.

Corda supports smart contract deployment and private transactions, being a prominent figure in the enterprise blockchain ecosystem. Although heavily focused on banking applications, Corda promises to deliver enterprise-grade solutions for production environments in other industries.

2.3.4.1 Nodes

A Corda network is made up of nodes, each of which runs an instance of Corda and one or more CorDapps. Communication between nodes is point-to-point and does not rely on global broadcasts. Each node has a certificate that maps its network identity to a real-world legal identity. The network is permissioned, with access requiring a certificate from the network operator

2.3.4.2 Network

Corda does not use a BroadCast Global network to communicate the performance of a contract, but rather uses a mechanism in which transactions are visible only between participants and shared only with those who have a legitimate reason to see it.

[9] https://kafka.apache.org/

[10] https://zookeeper.apache.org/

[11] https://www.corda.net/

2.3.4.3 Status

The transactions are based on a status object that represents an agreement between two or more parties, borrower (s), lender (s), and a security, within a timestamp. We could say that the transactions in Corda are 1 to 1.

2.3.4.4 Consensus

Corda offers three tools for global consensus distribution:

- Smart contract logic that enables users to transact using pre-agreed rules. It is a part of CorDapps.
- Uniqueness and time stamping services. These are known as notarial pools. Services can order transactions temporarily, which eliminates conflicts.

Flow Framework that simplifies the process of writing complex protocols between distrustful users.

2.4 COMPARATIVE ANALYSIS

The Four previous flavours of blockchain technology have been compared to select the most suitable technology for DataPorts. Contrary to more mainstream public blockchains, these implementations of blockchain technology are well prepared for the enterprise and incorporate most functionalities that are sought after in any production environment.

Features/Technologies	***Corda***	***EEA (Enterprise Ethereum Alliance)***	***Quorum***	***Hyperledger Fabric***
Intent	Corda has been designed as a blockchain for recording and automating legal agreements executions between specific identifiable parties	EEA solutions are intended to conform to an EEA spec and provide different implementations in different domain spaces	Quorum has been developed by JP Morgan aims to provide a permissioned blockchain for the financial services enterprises which supports transactions and contracts privacy	Fabric was designed and developed as modular and extendable framework with generic capabilities which can be used for solutions in various industries, such healthcare, supply chain, manufacturing, etc.
Maturity and product readiness	Mainnet (live network), and Testnet (beta network) available	Depends on adopter and implementation	No Mainnet or Testnet, but smart contracts compatible with Ethereum Mainnet and test nets, relatively small community of contributors	One of the first available and most mature enterprise blockchains. Fabric test network available.
Performance and scalability	If the asset owner changes frequently, each new party will have to verify the state of past transactions. (The non-validation notary checks that the input states to the transaction have not been previously consumed.) Particularly when a state changes hands frequently this will lead to a certain overhead.	Depends on the adopter and the chosen consensus mechanism. The standard does not mandate anything particular regarding performance. Better performance than Mainnet due to implementation of consensus protocols for permissioned networks, such as Raft and iBFT.	PoA consensus allows for higher throughput than public Ethereum	Low latency of finality/confirmation (due to order-execute model allowing for parallel execution), high throughput due to pluggable consensus based on known and relatively small number of ordering nodes

Features/Technologies	*Corda*	*EEA (Enterprise Ethereum Alliance)*	*Quorum*	*Hyperledger Fabric*
Privacy and confidentiality	**Pros**: Separation of ledgers (only the involved/entitles parties get sent the state and the smart contracts), off-chain execution engines for smart contracts. Merkle trees and tear-offs and graph pruning when not all state should be shared. Identities can be masked using randomized public keys. **Cons**: Notary witnesses' transactions. If there is more one notary and they communicate among themselves can be information leakage. No emphasis on complex cryptographic technologies, like zero-knowledge proof and multiparty computation.	**Pros:** Ethereum is a public blockchain, therefore it provides pseudonimity to its users. **Cons**: Privacy groups. On-chain: Restricted private transactions - sent/received by intended subset, Unrestricted private transactions: only decrypted by intended recipients, Privacy group: Privy to private transactions. Permissioning layer on the levels of peer node connectivity/ account/transaction type. ZK enablement on selected use cases by selected implementations	**Pros:** Supports private contracts (deployed to transaction parties only, business logic is confidential) and private transactions - where the hash is stored on chain and private state is communicated off chain using transaction managers. Private state and public state cannot be committed atomically. Private transactions are not suitable for asset-exchange because double spending is possible. POCs for Zero Knowledge security layer (ZKS) and anonymous Zether purely for payments and asset exchanges. Since it is not possible to do exchange of assets using private transactions, a workaround exists in the Quorum public chain where a party can use ZKP to hide the participants and amounts exchanged and that way to implement privacy in public exchange **Cons:** There are no independent channels for transactions between a subset of participants.	**Pros:** Data isolation using channels, private data collections support for private data and ZKProof technologies. **Cons:** Private data collections can be difficult to manage and code into smart contracts.
Use case fitness	Fits best for bilateral-multilateral agreements. Not good if there should be global state to be shared among all participants of the network. Token SDK.	Mostly not generic purpose, use-case driven solutions. Some implementations are well known (like Quorum or Hyperledger Besu), others are small and their value hard to gauge. Each implementation needs to be examined separately.	ERC standards, z-tokens, focused on financial services	Generic purpose blockchain

Features/Technologies	*Corda*	*EEA (Enterprise Ethereum Alliance)*	*Quorum*	*Hyperledger Fabric*
Security and identity	One-time public keys, separation of ledgers. Public keys based on PKI with both individual and organizational identity.	Privacy& Anonymity is mostly achieved through the use of ephemeral keys. Public keys – distributed, and interoperable between Ethereum based chains. Coupled to PKI via proofs.	Simplified security model provides the ability to deploy contracts and submit transactions privately.	Pluggable security modules for membership services. Allows for different organizations and participants in a common network to utilize their own certificate authority, and as a by-product, implement varying cryptographic algorithms for signing/verifying/identity attestation. Public keys based on PKI with native organizational identity used throughout consensus and permisssioning
Extensibility and interoperability, integration support for enterprise systems	Pluggable consensus, integration of oracles part of basic design	Convergence to Ethereum Mainnet, global proven network where results can be globally persisted, and silos merged and interoperate. Different solutions support subsets of consensus protocols: RAFT/iBFT/PoW/PoS	Pluggable consensus: RAFT, IBFT, Clique. Private Quorum networks cannot communicate. No integration support for enterprise systems like Oracles in Corda	Pluggable consensus:(Kafka, RAFT, pBFT), privacy and membership services
Unique features	Peer to peer infrastructure, bilateral transactions. dynamic communication	Strong privacy offering, support for built in tokenization or built in incentivization mechanisms	All parties share global network, private transactions, consensus on public state, no consensus on private state	All parties share global network with ability to create sub-networks within the netwok
License	Apache 2 license, open source	Depends on implementation, main protocol open source - GPL license	LGPL-3.0 license, open source	Apache 2, open source
Transaction processing or smart contract support	The smart contract is references in state and is sent together with the transaction. Signatures are collected by implementing a flow which defines to whom to send the transactions and whose signatures are necessary. The business logic is written as a whole (no separation to client-chaincode). The smart contract basically just validates the state - the business logic	EVM	Based on Ethereum smart contracts and Solidity, which is not Turing complete. Cannot commit atomically private states and public states	Smart contracts in Turing complete languages (Go, Java, JavaScript)

Features/Technologies	*Corda*	*EEA (Enterprise Ethereum Alliance)*	*Quorum*	*Hyperledger Fabric*
	runs off chain (parties agree on value, not the process arriving at the value). Smart contracts support legal prose in addition to code.			
General purpose/industry focused (general purpose smart code language etc)	Designed for highly regulated financial and banking services environment. Smart contract language is general purpose: Smart contracts can be written in any language which complies to java machine code. (Java/Kotlin)	Industry focused solutions with a few more general-purpose approaches such as Quorum. The development is done for EVM+various languages at the application level (depends on the adopter)	General purpose, however, mostly suited for financial services (some things are hard to implement using just the constructs in Solidity)	General purpose.
Permissioned/permissionless	Permissioned	Both	Permissioned	Permissioned
Governance model	Corda network foundation, non-profit	Enterprise Ethereum Alliance defining a set of specs. Depends on the adopter - have a full range from purely privately maintained network all the way up to one which is spin off-connected-compliant and connectable to Mainnet.	Codebase - JPM Morgan primary gatekeepers with open source contributions from the community. Deployed networks - up to consortia behind those networks	Codebase by Linux foundation hosting umbrella of Hyperledger projects, deployed networks -up to consortia behind those networks
Cloud offerings	AWS/Azure	IaaS offerings (like Kaleido), deployments on AWS/Azure	Kaleido, Microsoft Azure Blockchain Service	IBM cloud, AWS, Azure, Google, and Oracle
Ease of use and operation / development environment and documentation	Basic examples easy to run, however not a lot of explanations provided as to why certain commands need to be invoked. To understand deeper needs to dive in. Easy to orchestrate complex business logic thanks to the flows	The largest blockchain development community with lots of documentation, tools, forums	Ability to leverage community and a broad set of tools from Ethereum	Extensive documentation, VStudio support for IBP development, strong community

Features/Technologies	*Corda*	*EEA (Enterprise Ethereum Alliance)*	*Quorum*	*Hyperledger Fabric*
Summary	**Pros**: flexible and easily orchestrated transaction processing with the help of flows. Good for use cases requiring bi-lateral transaction executions between parties. Granularity of control of state visibility is on transaction basis. **Cons**: relatively weak confidentiality and privacy assurances, less scalable, not very good fit for global state sharing among all participants in the network, less fits when need to assure the process at arriving at consensus value (on-chain business logic)	**Pros**: Convergence to Ethereum mainnet (interoperability with Ethereum based solutions), strong privacy offerings **Cons**: practically impossible to optimize/deviate from the main protocol; except for implementations like Quorum or Besu (which will be covered separately) it is a set of particular solutions for particular problems not necessarily fitting our purpose	**Pros**: performance wise much better than public Ethereum, support for private transactions **Cons**: double spending possible when using private transactions, small community of contributors and not clear what JPMC commitment to the project, most fitted for financial services and tokenization use cases	**Pros**: performance, generic purpose blockchain, suitable for variety of use cases where global state should be shared among network, support for different privacy mechanisms. **Cons**: less suitable for bilateral peer-to-peer contracts

Table 1 – Comparative Analysis

A summary is provided in the following table:

	Corda	*Ethereum (EEA)*	*Quorum*	*Hyperledger Fabric*
Developer	R3	Vitalik Buterin	JPMorgan Chase	Linux Foundation
Consensus	Notary nodes	Proof of Work (PoW)	Raft-based | Istanbul BFT	Kafka | Solo
Operation mode / Network Access	Permissioned	Public	Permissioned	Permissioned
Smart Contracts	Smart Contract	EVM Smart Contracts	EVM Smart Contracts	Chaincodes
Language used in Smart Contracts	Kotlin	Solidity EVM	Solidity EVM	Go, Node.js
Privacy mechanisms	Yes. All the transactions are private	NO	Yes	Yes
Security	Access with digital certificates	Open	Access with digital certificates	Access with digital certificates
State	Transaction chain	Blockchain and Database (key / value)	Blockchain and Database (key / value)	Blockchain and Database (key / value)
Administration / Governance	Nodes, networks, policy, permissions, infrastructure	Node owner only	Nodes, networks, policy, permissions, infrastructure	Nodes, networks, policy, permissions, infrastructure
Transaction		Agreed format, they are validated within the block.	Agreed format, they are validated within the block.	Deploy and invoke functions to activate chaincodes.

	Corda	*Ethereum (EEA)*	*Quorum*	*Hyperledger Fabric*
Transaction cost	Free	GAS + Mining commission	Free	Free
Max. Transaction throughput	600TPS	~ 25TPS	100TPS	3500TPS *
Confirmation time / block	Immediately	~12s	Immediately	Immediately
Fault Tolerance		High	Depends on the number of nodes	Depends on the number of nodes
Attack vulnerability	Controlled environment	Susceptible to anyone exploring vulnerabilities	Controlled environment	Controlled environment
Cryptocurrency	No	Ether	No	No

Table 2 – Blockchain technologies comparison summary

2.5 CONCLUSIONS OF THE TECHNOLOGY ANALYSIS

After careful review of the selected technologies, the focus has been put on the set of capabilities that the different solutions bring to the table. The differences between special functionalities of the platforms and their implementation must reflect the different nature and use cases that they try to address. It is the objective of this analysis to select the most suitable blockchain technology for the DataPorts platform.

First of all, a strong requirement for DataPorts is the need for privacy between participants for certain transactions. This removes the possibility of using a public blockchain like Ethereum to deploy the DataPorts applications since plaintext public transactions will not be suitable for future use cases and services that will be offered in DataPorts. Moreover, the fact that transaction fees can be very volatile because of the instability of the USD/ETH pair makes another strong case against the use of public blockchains. Administrative credentials for the machines that host the nodes may also be required, which is not possible in a network of volunteer nodes, further discarding the candidacy of any public-based solution.

Both Corda and Hyperledger Fabric, as well as Quorum, are technologies mature enough and readily available to be used in the project. However, given that development of the Quorum software has been stalling recently, the future of the product is uncertain. Considering the importance of reliability and continuity of service of the DataPorts platform, the roadmap of the technologies must ensure that future upgrades, bug fixing, and new functionalities are included in the finished platform and future deployments and services.

Analysing possible use case fitness, Corda is more aimed at bilateral contracts and the financial services industry. On the other hand, Hyperledger Fabric is a tried-and-tested solution for supply chain operations and end-to-end asset traceability, making it a more suitable candidate in this regard for a project like DataPorts.

Even though every technology is capable of handling the expected transaction flow of the platform, the modularity and detailed design of a Fabric network makes it a more suitable solution with regards to platform scalability and performance. The possibility of including multiple channels and distinct ordering services makes the case for Hyperledger Fabric and its fine tuning for tailored applications.

Channels in Fabric networks also mean that information in the ledgers is compartmentalized and only those with permission will be able to execute transactions and read and/or write data into the ledgers. Security and privacy are of the utmost importance in a platform that will be used by many companies, some of which may be competitors.

Credential issuance and distribution is solved in Hyperledger Fabric by means of Certificate Authorities, owned by the different organizations that are created in the network and participate in the channels. Since DataPorts will be used by many different participants from several companies, independent certificate emission and derived keys for each organization are very useful characteristics of the technology that directly affect DataPorts' value proposition.

Every technology can be deployed in cloud infrastructure from the main providers like Google, Microsoft, or Amazon. Therefore, cloud platform compatibility is not an issue for any solution analysed.

It is because of these many reasons that **the analysis concludes that Hyperledger Fabric is the technology of choice for the DataPorts project**. Given that the platform will be used for years to come and the development will last for a considerable period, we recommend the use of Long-Term Support versions like Hyperledger Fabric instead of the most current stable versions in order to ensure maximum hardware compatibility and software stability. At the time of writing, the most recent LTS version of Hyperledger Fabric is 2.2, released July 20th, 2020.

3 VALENCIA PORT USE CASES

VPF will introduce two use cases for data sharing, one for on-chain data sharing and one for off-chain data sharing. The use case of on-chain data sharing will be the use of blockchain to share data regarding the Verified Gross Mass of containers (VGM Use Case). The use case of off-chain data sharing will be the sharing of the consignment note and the control document between the shipper, consignee, road haulier and public authorities in electronic form according to Spanish regulations (Consignment Note Use Case).

3.1 VERIFIED GROSS MASS OF CONTAINERS USE CASE

3.1.1 Ambition

In 2014, the International Maritime Organization (IMO) amended (MSC.1Circ.1475) the Convention on the Safety of Life at Sea (SOLAS) to require, as a condition of loading a full container on a ship for export, that the container has a verified the gross weight. This requirement became mandatory worldwide on 1st July 2016. The shipper became the responsible for obtaining the verified gross weight of a full container and communicating it to the shipping company.

The regulation prescribes two methods by which the loader can obtain the verified gross weight of a full container:

- Method 1: once the packaging and sealing of a container has been completed, the loader can weigh, or arrange for a third party to weigh the full container.
- Method 2: the shipper or, by shipper's order, a third party may weigh all packages and cargo items, including the weight of pallets, stowage wood and other packaging and fastening materials packed in the container, and add the tare weight of the container that appears on the door of the container to the sum of the individual weights of the contents of the container. (subject to national regulation, currently in draft)
- With respect to both method 1 and 2, the weighing equipment used must meet the precision standards and requirements of the State in which the equipment is being used.

Weight estimation is not allowed.

The full text of the regulation is available on the Internet through the following link[12].

There are also some guides published by the IMO regarding the verified gross weight of a container transporting goods[13], which can be consulted on the Internet through the following link[14].

To attend this requirement, ValenciaPort Foundation created in 2016 conPESO. conPESO is a marketplace platform that facilitates the compliance of SOLAS regulations on weighing containers for the port logistics community. The platform offers users an effective solution to allow containers to arrive at the port with verified gross weight, reducing last minute incidents or delays at container terminals or the appearance of congestion situations. Furthermore, conPESO offers a fast and automated method so that the verified gross weight reaches the shipping line and the terminal and allows the port to be more competitive[15]. When conPESO was created, the use of blockchain was not ready yet and this service was created without the

[12.] http://www.worldshipping.org/industryissues/safety/SOLAS_CHAPTER_VI_Regulation_2_Paragraphs_4-6.pdf

[13.] MSC.1/Circ.1475 Annex. Guidelines regarding the verified gross mass of a container carrying cargo. International Maritime Organization.

[14.] http://www.worldshipping.org/industry-issues/safety/MSC_1-Circ_1475__Guidelines_Re_garding_The_Verified_Gross_Mass_Of_A_Container_Carrying_Carrying_Carrying_Carrying-.pdf

[15.] htttps://conpeso.com

characteristics of consensus, provenance, immutability, and finality that a DLT provides. The use of DataPorts on-chain data sharing via a blockchain DLT is very well suited for the next generation of the service for requesting VGM and distributing the associated data to relevant parties. This use case is also introducing requirements for the registration of the weight of trucks and semi-trailers, as well as the management of payment mechanisms for the generation and provision of VGM data (i.e. via tokenization). It will need to consider the integration with on-line credit card payment systems (currently a virtual ATM terminal from BBVA is used) and the regulations for VAT payment of the services.

3.1.2 Value proposition

The VGM data service will allow sharing data between shippers and their representatives, conPESO allows:

- Know the scale network and the offer of services and prices available.
- Select the most convenient scales along the path of the container from the place of loading to the port.
- Enter the weighing request with all the necessary data to issue the verified gross weight of the container.
- Automatically send the verified gross weight to the port, and to the shipping company that transports your container once the weighing has been completed without requiring any management on your part.
- Manage all verified gross weights regardless of the method used to obtain them.
- Be directly connected in real time with the port system.
- Simplify the payment of weighing services through an on-line recharge system similar to the one used in pre-paid mobile phones.
- Avoid having to empty the truck before or after weighing to calculate the verified gross weight of the container.
- Lower operating and management costs of the weighing operation and the transmission of information.
- Minimize errors, the weighing result is entered only once automatically.
- Weighing result query.

For scale operators conPESO allows to:

- Offer their services to the largest network of clients in his environment.
- Increase the income statement.
- Guarantee the client's payment through electronic payment mechanisms.
- Eliminate any manual documentation management for weighing, minimizing operating costs.
- Get the scale to serve a greater number of trucks.
- Connect the scale to the port.

For carriers conPESO allows to:

- Use the scale that represents the least deviation on the container route.
- Reduce the time and paperwork necessary to weigh the truck.
- Avoid having to weigh a truck twice to obtain the verified gross weight of the container.
- Decrease the costs in the transport operation.
- Avoid paying the weighing service when running on behalf of the shipper.

- No additional cost for the use of conPESO.

3.1.3 Data sources

The data sources involved in the use case are the following:

- User
- Company
- AccountingEntry
- Billing
- Scale
- Vehicle
- VGM

More detailed information can be found in Annex A

3.2 CONSIGNMENT NOTE USE CASE

3.2.1 Ambition

Currently, although the data for transporting goods to and from the port is digitalized with the PCS using structured data formats, this data is not recognized yet by the authorities as a "legal" consignment note nor a an official transport control documentation for the controls by the authorities during the road movements. Since 2012 a regulation was established for the "legal" requirements for an electronic consignment note, where a digital signature of the shipper and consignee was required, as well as a proof of delivery of the goods at destination. For the control document, it was required to use only the paper form until 2019 when it was approved the Royal Decree 70/2019 of 15th February, which incorporated the legal possibility that both the consignment note and the control document may have electronic formats. In the Official State Gazette of February 22, 2020, the Resolution of the General Directorate of Land Transport has been published, which establishes the characteristics that must be met by administrative control documents in electronic support required in transport by road.

The objective of this use case is to take advantage of the information contained in the Single Transport Document (DUT) of ValenciaportPCS, so that it can accompany the merchandise electronically, not requiring its paper support. In this sense, the current Road Transport Service of ValenciaportPCS allows the agents involved in carrying out the road transportation of goods to generate and manage transport, release and acceptance orders necessary to carry out this transportation within the port facilities, as well as the notification of the pickup and delivery of containers in the terminals and/or container depots. To achieve this goal, the structured data (in XML or JSON formats) and PDF/A linked document for the consignment note will be generated in ValenciaportPCS and the DataPorts off-chain data sharing via a blockchain DLT will take care of registering the digital signatures of the shipper and road-haulier, the proof of goods delivery and the digital hash of the control document to ensure the consensus, provenance, immutability and finality of these legal documents and data, acting as a digital notary of these documentation.

3.2.2 Value Proposition

With this use case, the consignment note notary service will allow to use advanced data sharing techniques using off-chain DLT solutions to ensure the integrity, ownership, rights and delivery of goods between shippers, road hauliers, consignees and transport authorities.

For shippers, the consignment note use case will allow to:

- Prepare and submit the consignment note and control document in PDF/A format.

- Complete or modify the consignment note and control document automatically when new data becomes available through smart contracts that establish the procedures, origin and rules for these actions.
- Register the consignment note and control document in a repository connected to an off chain DLT.
- Ensure the integrity of the consignment note and control document using the blockchain DLT capabilities.
- Provide a digital signature to the consignment note using blockchain DLT capabilities and the use of digital identities.
- Check the haulier digital signature and the proof of delivery using the capabilities of blockchain DLT for the proof of origin and finality.
- Provide an audit trail of the documents, signatures, additions and modifications of the consignment note and the control document.

For road hauliers, the consignment note use case will allow to:

- Prepare and submit the consignment note and control document in PDF/A format.
- Complete or modify the consignment note and control document automatically when new data becomes available through smart contracts that establish the procedures, origin and rules for these actions.
- Register the consignment note and control document in a repository connected to an off chain DLT.
- Provide the control document to the authorities whenever required using the regulations established to this end.
- Ensure the integrity of the consignment note and control document using the blockchain DLT capabilities.
- Provide a digital signature to the consignment note using blockchain DLT capabilities and the use of digital identities.
- Check the shipper digital signature and the proof of delivery using the capabilities of blockchain DLT for the proof of origin and finality.
- Provide an audit trail of the documents, signatures, additions and modifications of the consignment note and the control document.
- Speed up the billing of the services carried out.

For transport authorities, the consignment note use case will allow to:

- Verify the integrity of the control document.
- Check the audit trail of the consignment note and control document.

For consignees, the consignment note use case will allow to provide the proof of effective delivery of the goods and its status.

For developers and owners of port systems (i.e. port community systems), the consignment note use case will allow to:

- Evaluate the capacities of the DataPorts platform
- Assess the use of DataPorts platform to introduce new port services that require an audit and notarization function.

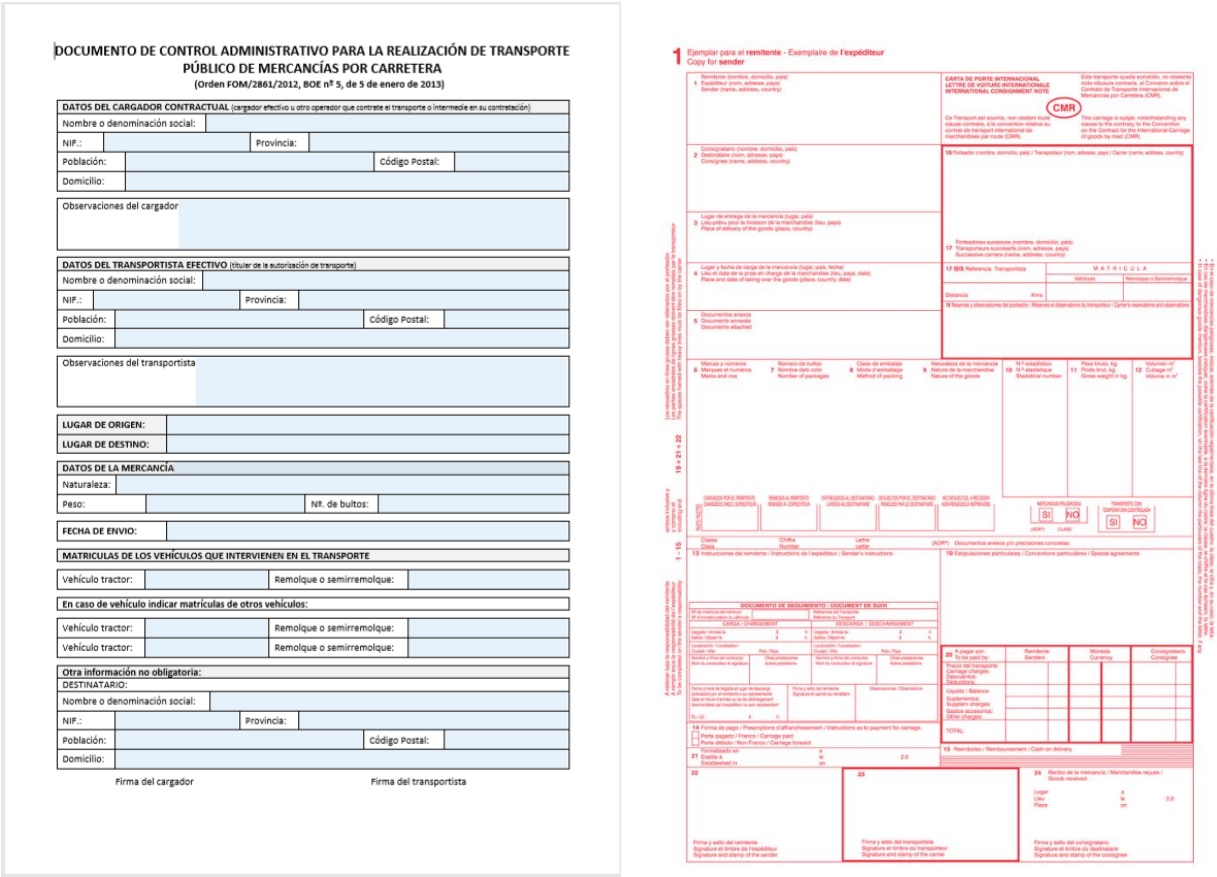
Assess the benefits, difficulties and costs associated with the use of DLT technologies with off-chain data storage.

3.2.3 Data Sources

The data sources involved in the use case are the following:

- User (User personal data should not be stored on-chain)
- Company
- ConsignmentNone PDF/A format
- controlDocument PDF/A format
- Documents (documents data should not be stored on-chain)
- TransportDocument

More detailed information can be found in Annex A



DOCUMENTO DE CONTROL ADMINISTRATIVO PARA LA REALIZACIÓN DE TRANSPORTE PÚBLICO DE MERCANCÍAS POR CARRETERA

(Orden FOM/2861/2012, BOE nº 5, de 5 de enero de 2013)

DATOS DEL CARGADOR CONTRACTUAL (cargador efectivo u otro operador que contrate el transporte o intermedie en su contratación)

Nombre o denominación social:

NIF.: Provincia:

Población: Código Postal:

Domicilio:

Observaciones del cargador

DATOS DEL TRANSPORTISTA EFECTIVO (titular de la autorización de transporte)

Nombre o denominación social:

NIF.: Provincia:

Población: Código Postal:

Domicilio:

Observaciones del transportista

LUGAR DE ORIGEN:

LUGAR DE DESTINO:

DATOS DE LA MERCANCÍA

Naturaleza:

Peso: Nº. de bultos:

FECHA DE ENVIO:

MATRICULAS DE LOS VEHÍCULOS QUE INTERVIENEN EN EL TRANSPORTE

Vehículo tractor: Remolque o semirremolque:

En caso de vehículo indicar matrículas de otros vehículos:

Vehículo tractor: Remolque o semirremolque:

Vehículo tractor: Remolque o semirremolque:

Otra información no obligatoria:

DESTINATARIO:

Nombre o denominación social:

NIF.: Provincia:

Población: Código Postal:

Domicilio:

Firma del cargador Firma del transportista

1 Ejemplar para el **remitente** - Exemplaire de l'**expéditeur**
Copy for **sender**

CARTA DE PORTE INTERNACIONAL
LETTRE DE VOITURE INTERNATIONALE
INTERNATIONAL CONSIGNMENT NOTE

CMR

Figure 3 – Physical documents involved in the Consignment Note Use Case

4 THESSALONIKI PORT USE CASES

ThPA will introduce two use cases in their blockchain network: Data driven applications for strategic and real time decisions, and Mobility improvement for passengers, visitors, and professionals of the Port.

4.1 DATA DRIVEN APPLICATIONS FOR STRATEGIC AND REAL TIME DECISIONS

4.1.1 Ambition

Ports are the nodes in the global network providing a key link between sea and land and the connection with the hinterland. The port community or the port ecosystem consists of a variety of stakeholders like port authorities, terminal operators, shipping companies and agents, rail operators, logistics service providers, trucking companies and inspection authorities like Customs, Hygienic services etc. In the last years, the port ecosystem stakeholders have developed several solutions with focus on individual business needs. As the transport industry becomes more demanding in terms of increased efficiency and lowered costs, sharing and timely provision of accurate information between the business network is more critical than ever. The type of data exchanged and shared must support the operational and decision-making needs. The trend for Digital Ports asks for the exploitation of the capabilities of available systems with new techniques to better organize and co-organise, operate, and co-operate towards management of supply chains.

Port Description & business figures

This use case finds its applicability in the container terminal of Thessaloniki Port Authority. The container terminal lays in the western part of pier 6. Its dimensional characteristics with 550 m. length and 340 m. width accommodate ships with a draught of 12 m. Being part of the Free Zone, it covers a surface area of 254,000 m2 with an on-site storage capacity of 5,000 TEUs in ground slots. It is equipped with state-of-the-art technologies and modern container handling equipment. Four cranes are used for container loading-unloading services (2 post panama), 1 transtainer for loading –unloading trains and mainly straddle carriers as well as tractors, front lifts, trailers, and forklifts for storing and moving containers. With a total throughput of ~ 425.000 TEUs (2018) and increasing trends, modernization and expansion plans, the container terminal of ThPA dynamically strengthens its position in the area of Eastern Mediterranean.

Equipment

With the term «equipment» the systems to be used in this DataPorts use case are presented. More specifically:

- **Container Terminal Total Management System:** It is the Terminal Operating System (TOS) of the container terminal. It consists of the following modules:
    - **Document submission** that supports the submission of all documents (in XML format) for the import, export and transhipment activities.
    - **Customer service** that provides feedback to the port clients regarding the operational and administrative status of their containers.
    - **Entry/Exit control** that provides the authorization to entry/exit the container terminal.
    - **Loading/Unloading control** that controls and monitors the loading/unloading activities of ships and trains.
    - **Yard planning** that offers effective yard utilization.
    - **Yard inventory control** that ensures accuracy in recording the yard status.
    - **Geographical Information System** that monitors in real time all terminal operations and presents them in a user-friendly environment.
    - **Resource management** that organizes and executes the container handling while supports better use of terminal equipment.

- **Administration t**hat covers several administrative activities for the delivery and pick up of the container, while keeping logistics warehousing records etc.
- **Invoicing** that facilitates the electronic issuing of invoices.

- **Truck Appointment system**: It is the system used by authorized users (mainly truckers and forwarders) to book appointments for the delivery and/or pick up of containers.
- **Vessel calls**: It is the system used to declare the expected arrivals of container vessels, including dates (ETA), container size and type, type of operation (loading/unloading) etc. (it applies to conventional cargo vessels as well).
- **Gate Control System**: It is the system used to control access in the Free Zone of ThPA. RFIDs tags are used to check the trucks/vehicles, OCR for container id, truck plates and physical condition of the containers.

4.1.2 Value Proposition

The scope of this use case is to support ThPA ecosystem stakeholders to better organize and manage the pickup and delivery of containers, with emphasis to land gate access, by utilizing the data available by several sources and the capabilities of DataPorts platform. It will be achieved with the fulfilment of the following objectives:

- Support and facilitate ThPA for proactive and reactive actions regarding the operation of Gate Control System
- Improve the operational performance of the port supply chain and consequently of the whole supply chain
- Increase visibility of operations for stakeholders involved
- Improve the environmental burden caused by trucks traffic

The Use Case is depicted diagrammatically in the following figure:

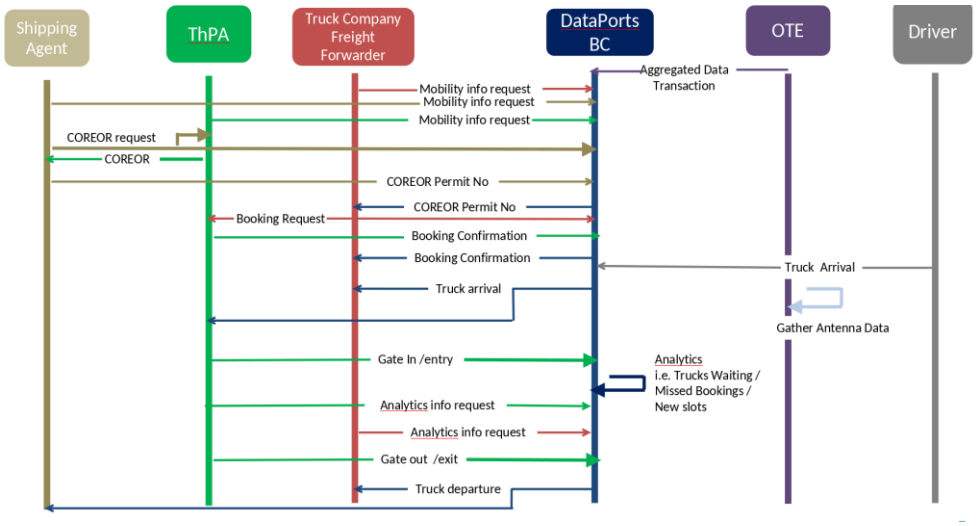


Figure 4 – Information flow in the ThPA use case

This use case directly involves several actors from ThPA ecosystem namely:

- Shipping agent that will be responsible for the electronic submission of the COntainer RElease ORder (COREOR) request and will receive information about the truck departure from the port. The collaboration between the shipping agent and the truck companies and/or freight forwarders regarding the COREOR permit number (absolutely necessary for pick up the containers) will be done through block chain.

- ThPA that will respond to its clients through blockchain for the confirmations. Additionally, it will provide several types of status, regarding the truck entry/exit/departure etc. The block chain component is used.
- Truck company & freight forwarders will use blockchain to communicate the bookings and their confirmation to ThPA and receive information from the shipping agents.
- OTE will provide aggregated mobility data so the ThPA ecosystem to receive mobility patterns and data.
- Driver with its mobile application will provide the arrival to the gates of ThPA

Several types of analytics will be defined regarding number of trucks waiting, missed bookings, assignment of new slots.

4.1.3 User stories

The use case addresses the users' needs of ThPA port ecosystem. Some of the users will be direct beneficiaries while some others indirect. However, all will have the opportunity to collaborate in a "smart port" environment. The figure below presents the main stakeholders of the supply chain.

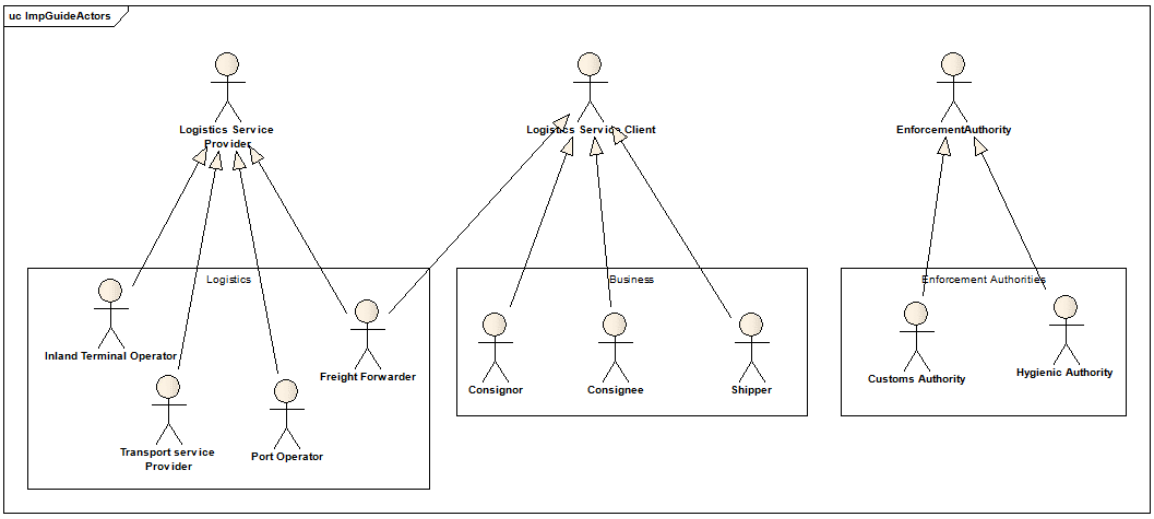


Figure 5 – Stakeholders of the supply chain

Thessaloniki Port Authority:

- I want to have processed historical data for long term decision making and planning
- I want to know the pattern that the containers of a specific company/ship leave the terminal so that I can better plan and manage my yard, my personnel (shifts) and equipment (quantities, days, months)
- I want to know the pattern of containers delivery/pick up (quantities, days, months) from a specific company/ship to better plan and manage my yard, my personnel (shifts) and equipment
- I want to have a model to predict queues creation in the land gate so that I can optimize its operation and take proactive measures
- I want to have real time information about the progress of the appointment system so that I can take reactive measures if needed
- I want to know the companies that keep the appointment rules so that I can create a client-based marketing approach

- I want to reduce the environmental impacts of the truck traffic in order to serve the social policy of my company
- I want to have an easy to use visualization tool in order to facilitate my decisions

Trucking company:

- I want to know the peak days and hours at the gates of container terminal of ThPA so I can avoid delays and better organize my transport
- I want to know in real time the operational status of the gates in order to control my fleet/trucks
- I want to know when my truck has arrived/left the terminal so that I can inform my clients

4.1.4 Data sources

The data sources involved in the use case are the following:

- ThPA systems:
    - Container Terminal Total Management System: It is the Terminal Operating System (TOS) of the container terminal.
    - Truck Appointment system: It is the system used by authorized users (mainly truckers and forwarders) to book appointments for the delivery and/or pick up of containers.
    - Vessel calls: It is the system used to declare the expected arrivals of container vessels, including dates (ETA) , container size and type, type of operation (loading/unloading) etc. (it applies to conventional cargo vessels as well).
    - Gate Control System: It is the system used to control access in the Free Zone of ThPA. RFIDs tags are used to check the trucks/vehicles, OCR for container id, truck plates and physical condition of the containers.
- OTE: Mobility information.
- Shipping agents' internal systems. Details will be provided in the detailed analysis of the use case to take place in WP5. The system responsible for the electronic submission of documents (COREOR request) will be analysed.
- Freight forwarders' internal systems. Details will be provided in the detailed analysis of the use case to take place in WP5. The system responsible for the electronic submission of documents (COREOR request) will be analysed.
- Tucking companies' internal systems. Details will be provided in the detailed analysis of the use case to take place in WP5. The system responsible for bookings in TAS will be analysed.
- Drivers: mobile applications

More detailed information on the datasets involved can be found in Annex A.

4.2 IMPROVE MOBILITY OF PASSENGERS, VISITORS AND PROFESSIONALS OF THE PORT

4.2.1 Ambition

In this Use Case we envisage the exploitation of people mobility data in order ThPA or any shipping port in the future, to offer passenger related mobility (traffic) as a service (e.g. analytics) to customers. (Internal and/or External to the port existing ecosystem)

More specifically, shared data will assist the Port Authorities to become Smart by offering cognitive services and analytics to their business customers, in order to optimize the passengers' and visitors' approaching routes. Such services may improve, the embarkation and disembarkation processes of the passenger ships,

and potentially a Port Authority will be able to the number of the servicing ships, aiming to have a positive economic effect. Similarly, the shipping companies, which will use data services in the future will have the opportunity to minimize delays and eventually, have cost reductions. Port Authorities can gain a competitive advantage towards the digital transformation. In addition, besides the economic benefits for the ThPA, various stakeholders can benefit from the data sharing offerings, in order to improve their services and also have an economic growth.

4.2.2 User Stories

As a Shipping Ports' Stakeholder of DataPorts internal or external (Public authorities, Municipalities, Shipping Companies, Transportation Authorities, Cultural and Trade Associations, etc.) I would like to have access to services (i.e. passengers' mobility patterns) in order gain access to such insights improve my field of operations and have a competitive advantage. This goal may be achieved when I can gain access to valuable data-driven services.

For the above-mentioned reasons, scenarios A & B are proposed in order for shipping ports' stakeholders to have access to passengers' mobility patterns, as a service offered by ThPA.

Proposed Scenario (A)

Data /Service Consumer (ThPA) is going to offer passenger mobility analytics to its customers (mentioned above). Such service is useful for the port's stakeholders but also useful for external users that in the future will be willing to "pay" for such a service. Hence, Smart Contracts may be applied between The Data Provider, the Data Consumer and the Service Provider that may offer the analytics as a service for the consumers.

Therefore, various conditions may be also applied within the Smart Contract. Such conditions/requirements are considered the pricing scheme that is agreed between the involved parties, whether it is a monthly or annual subscription for the Data Consumer with variations on data volume, and /or on the access of certain information as described below. Permission for Data Consumer to access specific data may be given if there are not any payment overdues, and the service may be accessed on specific periods. The Data Consumers may have access to the mobility pattern of the passengers within certain areas around the port such as the Central Macedonia Region, as well as, on specific data fields as those described below. The Data Provider is obliged to provide the pre-agreed dataset in specific time periods (e.g. at the first day of the next month). The dataset has to be accurate and updated. The Service Provider should provide the mobility pattern analytics only if financial obligations by involved parties are fulfilled. Moreover, the service will not be permitted to given to any Data Provider's competitor and data provided may not altered at any time. Additional permission requirements will be given during the use case implementation.

In this scenario, OTE will have the role of the Data Provider, ThPA will be the Data Consumer and CERTH will be the Service Provider.

ThPA will have access at this analytics service from CERTH. For that OTE subscribers' mobility data are needed in order to identify mobility patterns of the passengers. This dataset may be fuzzed with additional open datasets (if any available – under investigation).

Describing the flow:

- ThPA requests a service (Analytics e.g. people's mobility patterns in Central Macedonia Region in order to offer it to shipping companies, or any other customers that fulfills Smarts Contracts' rule, for better customer experience but also to be used internally as the operational optimization). The service will be based on analytics from CERTH and data provided by OTE (# of distinct users, voice in, voice out, SMS in, SMS out, data up, data down i.e. full dataset fields) .
- ThPA as a Data Consumer may offer / "sell" (give access) in the future of that analytics service to shipping companies, municipality, and transportation authorities.

- Permissions through Smart Contracts shall be checked (if ThPA is allowed to have access to this service (analytics based on specific fields) and under what conditions based on agreement between ThPA, CERTH and OTE)
- If Ok then
    - OTE sends data to CERTH or can send them periodically (on a Monthly basis- described on Smart Contract)
    - CERTH creates the service and sends or gives access to ThPA
    - All involved parties should fulfill the Smart Contract's terms.
- Otherwise the service request is rejected
- Transactions are recorded and monitored by all parties at any time.
- Violations should be checked through queries in order to validate or not terms.

Proposed Scenario (B)

There can be a similar case where ThPA cannot have full access to data that come from OTE. Can only have access in the numbers of users that move from other locations to Thessaloniki Port area (in coming traffic). Similar process may be followed.

ThPA is going to offer passenger mobility analytics to its customers. Such service is useful for the port's stakeholders but also useful for external users that are willing to "pay" for such a service.

This scenario is differentiated from (A). ThPA is not on Data Consumer but also a Data Provider (Ships' schedules). CERTH as a Service Provider will need to have access to datasets by OTE and ThPA. Dataset may also be fuzzed with additional open datasets (weather, traffic, etc.).

Describing the flow:

- ThPA requests a service (Analytics e.g. people's mobility patterns in Thessaloniki area in order to offer it to shipping companies for better customer experience but also to be used internally). The service will be based on analytics from CERTH and data provided by OTE (# of distinct people moving in out of the port area) and ThPA (ships' schedules).
- All Permissions requirements are checked as in scenario A, based on Smart Contract between ThPA, CERTH and OTE)
- If Ok then
    - OTE and ThPA send the necessary datasets on demand or can follow scenario A
    - CERTH creates the service and based on agreement gives access to ThPA. A variation: in the future, service may be offered to 3rd parties as well for a fee.
- Otherwise the service request is rejected
- Transactions are recorded and monitored by all parties at any time.
- Violations should be checked through queries in order to validate or not terms.

4.2.3 Data Sources

Prior to any usage, subscribers' mobility dataset provided by OTE have the subscribers' consent. In addition, data are anonymized and are in accordance to the GDPR and other European regulations. More specifically, the data provided by OTE that present the subscribers' mobility pattern are anonymous, based on the analysis that takes place internally per field, as well as by applying the rule of "5" (no more than 5 users from one location area. Therefore, they are not considered as personal data and for non-personal data it is not necessary to examine the legal basis (e.g. contractual obligation, legal obligation, legal interest, consent)."

Stakeholders in the business network are considered the CERTH, ThPA, as well as any partner that is willing to use the available dataset and bundle it with a service.

The datasets may also be available to external stakeholders that by using DataPorts data-driven platform will be onboard on the ThPA ecosystem and benefit by using them. The dataset will be available for the external stakeholders upon their acceptance of certain conditions (e.g. no re-selling, offering them to OTE's competitors, altering the dataset, etc.) (Smart Contracts approach)

In the context of the OTE data sharing, it is required a control mechanism to be orchestrated in order to handle how the data are shared, under which Contractual Clauses and regulations. Moreover, upon request for sharing from third party, OTE requires to enforce privacy guidelines and maintain governance over the data even after the sharing is completed. Therefore, in case that the data are used in an unlawful manner, it will be identified.

Given the fact that the data sources will be maintained outside the platform and off-chain, OTE requires a gateway which will enable the sharing of data. This gateway will facilitate access from the platform into the OTE's private data repository. In order to enable access to the third party into the repository throughout the platform, OTE requires an API which will provide all the necessary information to ensure that the client has accept the terms and probably has paid the fee in order to acquire the datasets. The data repository will be a SFTP server or potentially an HDFS.

5 BLOCKCHAIN GLOBAL SCENARIOS

The main use cases to be developed for the whole DataPorts platform revolve around data sharing and data governance. Both concepts will be further explained in the following chapters. However, a clear distinction can be made insofar as the business rules are concerned. Data sharing use cases focus on the mechanisms that take place at the smart contract level that allow certain users to access or consume a certain dataset. On the other hand, data governance has to do with the administration of the network and the data itself that it contains, detailing the process of owning and transferring data, asset creation, data models, information sources, et cetera.

5.1 BLOCKCHAIN FOR SHARED DATA

This scenario encompasses the blockchain registration of certain events associated with actions carried out in the transport chain. The information must have adequate privacy mechanisms to be shared only with the actors involved in the processes, or with whom the information is shared.

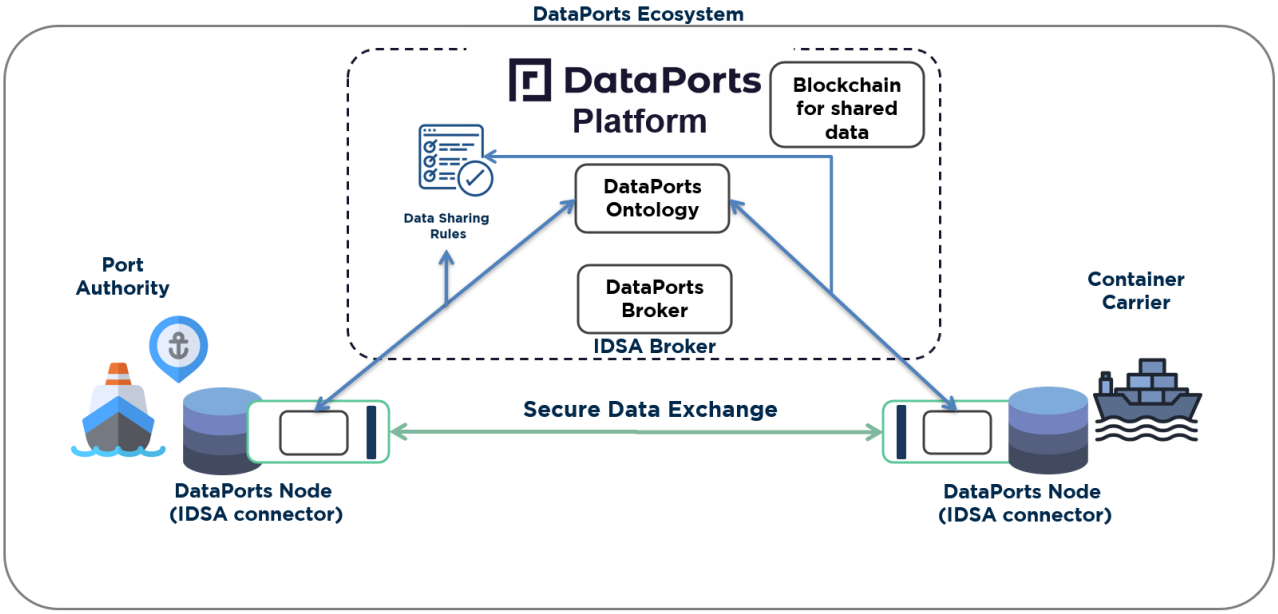


Figure 6 – Data sharing use case description

The events to be registered will be those that are determined to be of interest, such as, for example: entry or exit of a container in the different stages of transport, opening actions, control or verification of the containers at a certain stage.

The consultation operations of the registered events will be carried out on the database of the blockchain ledger. The consumer will launch the information request to the corresponding smart contract that will validate the permission rules, retrieve the events registered in the blockchain and return them to the consumer (Figure 7).

This ensures traceability of the shared data in order to monitor its consumption with security and monetisation purposes.

The use case for shared data in the blockchain provides various platform governance capabilities, interoperability between different platforms, implements the mechanisms by which data can be shared and traded in a secure and reliable way, stores the actions carried out by the participants, obtaining an immutable record of transactions performed in the datasets and consumed by each organisation.

For instance, in the VGM Use Case from ValenciaPort, if a third party wanted to make use of VGM data for analytical purposes, network policies will determine if the external party has access to this type of datasets and if so, an information request will be sent to the blockchain peers. If permitted, the blockchain will issue

a token that will allow the third party to access the data, leaving a trace that will provide proof of request for traceability and monetisation purposes.

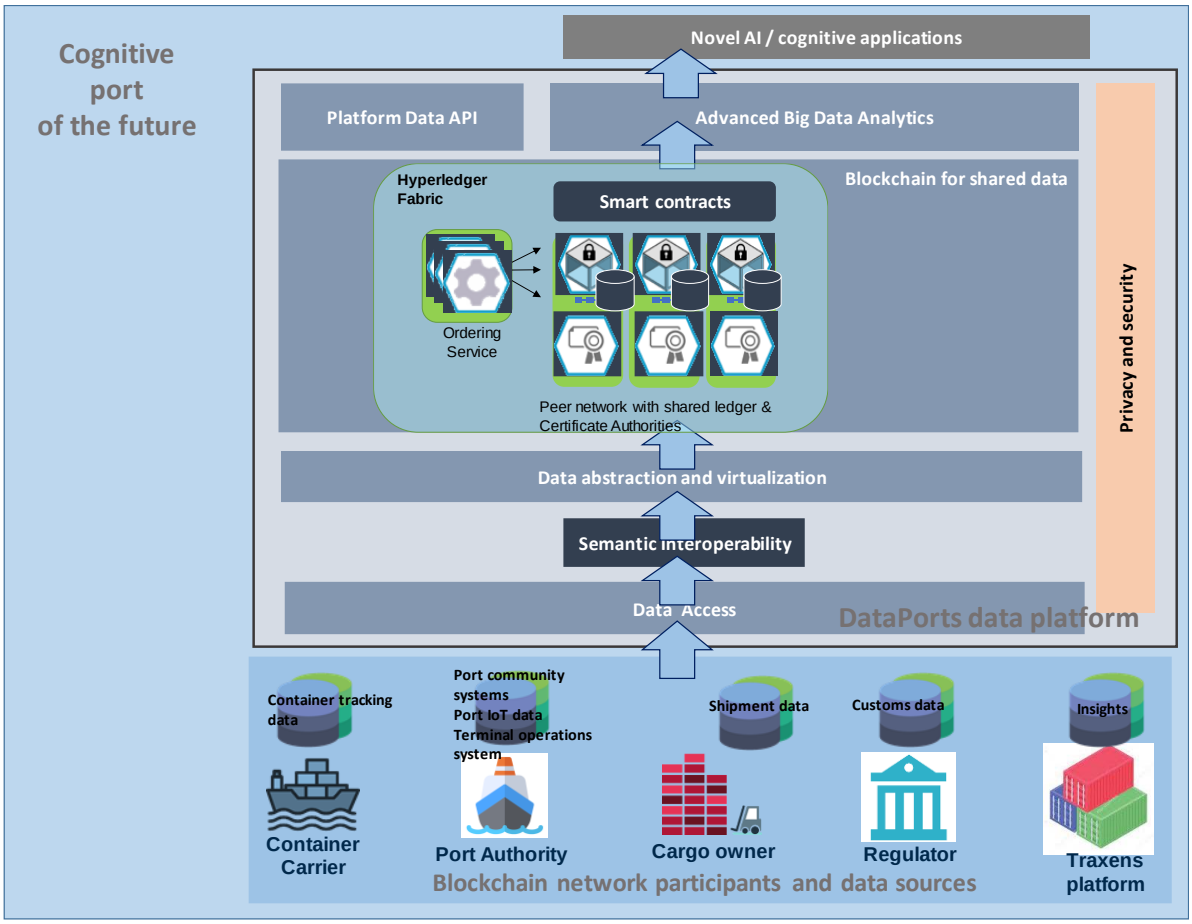


Figure 7 – Platform high level architecture for Data Sharing on the blockchain

5.2 BLOCKCHAIN FOR DATA GOVERNANCE

In a blockchain-based platform, one of the main issues revolves around network governance. By definition, a blockchain network is trustless (i.e. the need for trust in the platform/infrastructure does not exist), but nonetheless an entity must take the role of network administrator in order to create, configure, and deploy, the necessary components of the blockchain network. In this scenario, there are several requirements to be addressed.

- It establishes sharing rules through smart contracts to each dataset. The smart contracts will enable, revoke, or deny the consumption of data sets based on compliance with the privacy policies defined in the smart contracts.
- Data governance should also include special access policies such as requiring special permission from more than one party to access the data.
- End users should be able to define access policies for their datasets in a streamlined way without needing programming experience.

Therefore, the question of how to govern the network is posed. In truly decentralised environments, there is no need for a central authority, as it is the case of public blockchain. However, in an enterprise platform, business rules must be determined by someone.

Usually a network governance body is created within the project with representatives from all stakeholders, which will review the business rules set forth in the smart contracts and accept them.

The governance of the data establishes the rules that will be built upon the smart contracts of the application. They will either allow or deny the consumption of the datasets contained in the DataPorts platform according to privacy policies and business rules.

In order to ensure network-wide enforcement of these policies, there must be a clear definition of who can access what datasets, so the access token can be appropriately issued to the relevant party. As an example, in the Data Strategy Use Case from ThPA, business rules must be determined for shipping agents and truck companies to share their data with DataPorts.

5.3 BLOCKCHAIN CAPABILITIES, BENEFITS AND APPLICATION FOR SHARED DATA AND DATA GOVERNANCE

Blockchain is a fast-growing field that incorporates different technologies into a single package. The advantages can be numerous depending on the use case and the industry involved, and users can leverage several benefits to improve their business models.

It is through careful analysis of the benefits of the technology that a potential user can understand the improvements in security, transparency and efficiency that come packed in an enterprise blockchain solution. This is a summary of the main advantages that one can expect when thinking of deploying a blockchain network for their business:

5.3.1 Decentralisation

Information in a blockchain is replicated multiple times in a network of distributed nodes. Each copy of the ledger is guarded by a node, in charge of verifying transactions and updating the chain with the latest blocks. Keeping information this way ensures that no single point of failure can cause damage to the integrity of the process, preventing data loss due to hardware malfunction or network exploitation attacks like Distributed Denial of Service attacks or ransomware infections.

Another crucial benefit of a distributed ledger is ensuring the validity of the data and the authenticity of the transactions that take place in it. Since blocks are cryptographically signed and linked to the rest of the chain, any given change at any point in the ledger of any node is easily detected due to a change of the block hashes, setting a strong deterrent to malicious actors that seek to change the contents of the blockchain in order to profit themselves.

The consensus protocols in place remove the need for an intermediary of the transaction, giving full power to the owner of the asset to do as they please at any time with their property. DataPorts will be a shared platform among different companies and would benefit from a decentralized platform where data is kept safe and replicated.

5.3.2 Anonymity/Pseudonymity

By design, users are assigned a public key that identifies them in the network. This public key and its private companion are randomly generated, and the real identity of the owner can be kept secret if he or she does not disclose having possession of the private keys linked to an account.

Although there are completely anonymous blockchains, like ZCash[16] or Monero[17], that keep forward secrecy and encrypt all transactions, it is more correct to speak about pseudonymity when talking about identity in a blockchain network. In almost every blockchain, the public key serves as a unique identifier of the user, who keeps this identity for as long as they operate in the network. It works in a similar way to an online forum. Forum users post messages from an account that they previously created associated to a nickname. If they do not disclose their real name in a message, their pseudonym cannot be linked to their actual identity. In

[16] https://z.cash/

[17] https://www.getmonero.org/

the enterprise world, some solutions like Hyperledger Fabric can provide true anonymity to some of their users if configured properly.

Some of the information that will be shared in the platform will need to be kept out of the public eye, incorporating forward secrecy for a subset of transactions in the finished solution, in order to preserve critical information about business transactions between DataPorts members.

5.3.3 Programmable

With the help of smart contracts, blockchain transactions can be programmed according to business events. Some examples of tiresome bureaucracy that can benefit from smart contracts are greenlighting payments, replicating supply chain events that modify the digital assets, transfer of ownership, escrow services, delayed transactions, multi-party agreement over a transaction, etc.

Smart Contracts will help DataPorts by automating some of the manual process that take place in the global shipping and logistics industry.

5.3.4 Security

Since information is distributed in multiple nodes and changes are automatically detected and discarded by the consensus process, a blockchain is considered an extremely secure way of doing business and storing information for several interested parties.

Blockchain nodes are generally deployed in Linux servers that enjoy tried and tested security from hostile takeovers if protected correctly. But the true concept of data security comes from knowing that a malicious actor would need to have simultaneous control of enough nodes to trick the consensus mechanism and tamper with the data written in the ledger. In the Bitcoin blockchain, this effort can be translated to having concurrent control of more than 4000 Linux machines all over the world.

Even in that hypothetical scenario where more than half the nodes are controlled by attackers and transactions are successfully forged, a record of a great disturbance in the consensus process would be kept, and network administrators could revert the changes by restoring all the ledgers back in time where the network was still not compromised. Such an event has not happened in a public blockchain comprised by a sensible number of nodes, and trust in the security of blockchain platforms is not in doubt.

5.3.5 Immutability

Transactions made in a blockchain are forever written in the ledger, as it is append-only. As mentioned, every data block in the chain is linked to the previous block in the chain, ensuring that the transaction flow is not reversible. This is particularly important in use cases where proof of ownership is very important or when assets change hands very frequently.

5.3.6 Transparency and auditability

In a blockchain, every copy of the ledger is public and identical, so analysing any copy is enough to make sure that a transaction indeed took place. When a block is created, it is also time-stamped, so there is also proof that the transaction took place at an exact moment in time, given that it could not have been made earlier since it would have been included in a previous block. This also means that the origin of any asset can be tracked along the chain. This consensus process also ensure that double spending does not exist, (e.g. consuming twice the same asset, incurring in a duplication of value.)

Critical information that will be placed in the DataPorts platform must be capable of being easily verified and audited by port authorities and customs. Key features of blockchain technology such as transparency and immutability are essential to the DataPorts use cases.

5.3.7 Digital asset creation

Even though blockchain use cases are notoriously related to cryptocurrencies, the assets that live in the

blockchain can represent anything of value. A government for instance could issue identity tokens in a blockchain to citizens that contain their personal information. This information could never be forged, and the risk of identity theft then drops to a minimum.

Smart contracts can also work as insurance policies, containing all the clauses of the policy the moment it was created, and freeing the funds when conditions are met. Property registries, tax returns, health records, diamonds, kilowatt-hour, designation of origin for the food industry, et cetera.

In the DataPorts scenario, assets representing important documents, proof of ownership, bills of lading, et cetera, could be included as assets in the platform, ready to be verified or transferred to other parties.

5.3.8 Efficiency

Blockchain transactions eliminate the need for paperwork in traditional business exchanges, for all records are kept in the ledger, and the business events that trigger transactions can often be incorporated into a smart contract, removing costly trips to deliver just a signed paper form.

The shipping industry has traditionally born the burden of inefficient bureaucracy and large amounts of paperwork. Smart contracts and regular transactions can dramatically improve lead times of processes that involve a great number of manual steps, verification and signing of documents, or proof of payment of an invoice.

5.3.9 Integration

Cutting edge blockchain solutions heavily rely on strong integration with current IT solutions such as Enterprise Resource Planning systems or Customer Relationship Management modules, as well as other blockchains. The events that trigger execution of smart contracts can come from outside the blockchain network, since public-key infrastructure is already deployed in many businesses for encrypted communications and can be used to identify parties in a network even when the credentials themselves were not created by the certificate authority module of the blockchain network.

Other technologies such as Artificial Intelligence that provides inputs to smart contracts, or Internet of Things devices that are members of the network and sign transactions automatically, can also bring new benefits to the value proposition of a blockchain solution, providing better quality of data sources and expanding the traceability chain of the asset lifecycle.

It is important that DataPorts leverage the full potential of blockchain technology by incorporating quality data sources that feed the transaction flow and integrating with existing systems that will also participate at some point of the process flow.

6 HYPERLEDGER FABRIC NETWORK DESIGN

In this chapter, a brief description of the Hyperledger Fabric components to be deployed is included, followed by a design proposal that will provide separation of the use cases for easier development and testing.

6.1 KEY COMPONENTS OF THE HYPERLEDGER FABRIC NETWORK

As determined in the comparative analysis of blockchain technologies, the most appropriate technology for the DataPorts project is the Hyperledger Fabric network. The different necessary components of the Hyperledger fabric network are described below:

Peer. They are nodes contributed by the entities - organizations - participating in the blockchain (registry). These nodes store both the blockchain and a key-value database of the current state of the chain. Hyperledger differentiates between several types of peers: committing peer, endorsing peer, leader peer and anchor peer. Every peer is a committing peer: storing the blockchain and maintaining the World State (see State definition below). Any peer that has a chaincode installed is an endorsing peer. Then each organization can have one or more leader peers. These are responsible for propagating the blocks received from an Orderer (Ordering Service node) to all committing peers in the organization. An anchor peer is a peer used to communicate with peers from other organizations on the Channel, so that the Channel's membership is properly managed.

Channel. It is a private sub-network between two or more peers, so that only the authorized participants can participate in it. Each channel maintains its own blockchain, along with its Smart Contracts, and only channel participants can view that blockchain and the transactions sent to it. In addition, the consensus on the next block, and the transactions included in it, is done by channel. Therefore, it can be understood that each Channel is equivalent to a different blockchain. It is a mechanism that allows for a higher level of isolation between participants if deemed necessary. It should be noted that the same peer can participate in more than one Channel at the same time, keeping both isolated.

Chaincode. It is the name that Smart Contracts receive in Hyperlegder. These can be implemented in common technologies like Java, Node.js or Go. And they can interact with the state of the blockchain through a well-defined REST API. In Hyperledger there are two types of chaincode: user - programmed by the blockchain operators - and system - which provides the logic needed by Hyperledger to execute part of the transaction flow.

Ordering Service. It is the service in charge of building the blocks for each Channel, executing for this the consensus phase of the transaction flow. This service is modular and has different consensus protocols. Said service may be made up of a set of nodes. In fact, to be fault tolerant it is interesting that it is not restricted to a single node.

State. The state in each Channel is made up of its blockchain (registry) and the **World State**, key-value database that maintains the current state of the registry. Thus, each of the peers that belong to a Channel maintains a local copy of both. In addition, the implementation of Hyperledger Fabric (from version 1.0) allows to use two different databases to store the World State, GoLevelDB or CouchDB. The first is a memory-embedded database, while CouchDB uses a client-server model through a REST API over HTTPS.

Private Data Collection. Collection of private data that can only be stored by the peers of the subset of organizations of a Channel authorized to it. It is a mechanism that allows information to be exchanged between an organization subset of a Channel without the need to create another one. The fact that only the peers of the organizations authorized to do so can store the information makes the level of replication of the information stored in them less. The database in which it is stored is usually called SideDB.

Client. It refers to a client application, which is authorized to participate in the blockchain (Channel) and proposes transactions. This implies sending the transaction to one or more endorsing peers based on the endorsement policy established in said Channel. Once it has received enough responses with the appropriate

approval, it sends the transaction along with the endorsements obtained to the Ordering Service, which includes it in a block and propagates it for subsequent validation and confirmation in the registry.

As can be seen in Hyperledger Fabric there are three types of nodes: Client, Orderer and Peer. All of them require a valid certificate to be able to communicate over the network.

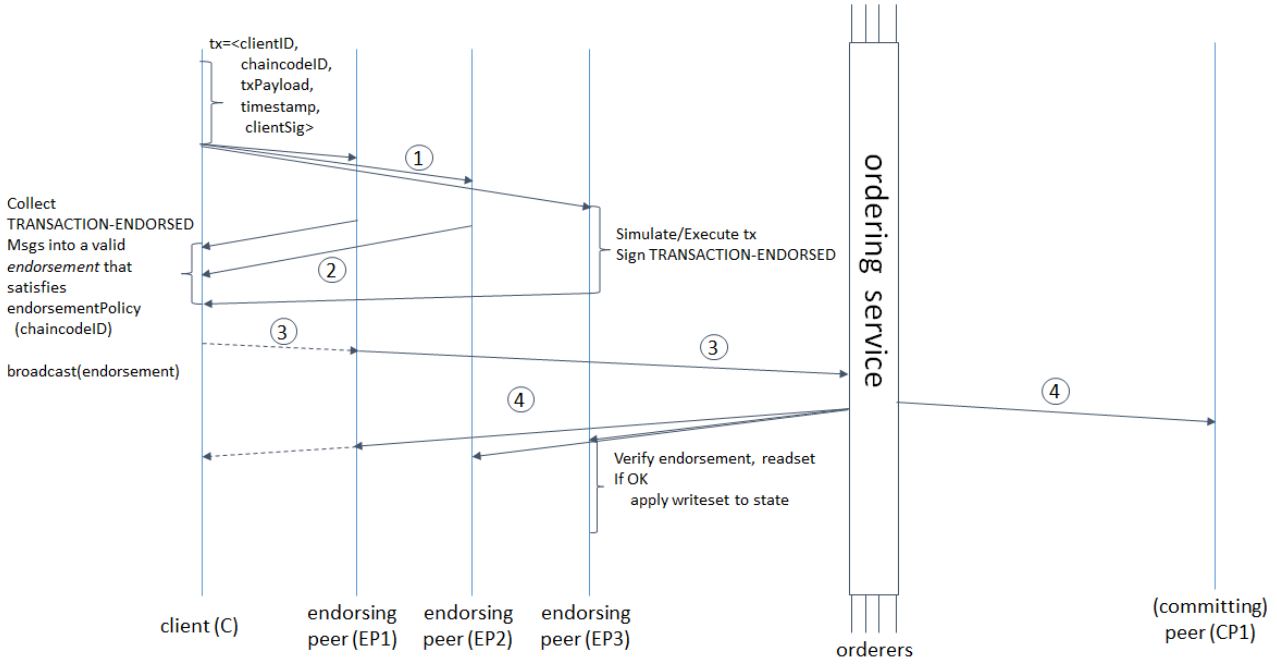


Figure 8 – Communication flow of a transaction in Hyperledger Fabric[18]

Transaction flow

From the elements introduced in the previous section, the flow that follows a transaction of a public nature (which can be seen by all Channel organizations) can be partially derived. This flow – represented in Figure 8 – consists of the following phases.

Endorsement phase. In this phase, the Client application builds a transaction to invoke a function in a chaincode already uploaded to the blockchain. The client signs the transaction with their credentials and sends the transaction – step 1 in Figure 8 – to as many endorsing peers (from different organizations) as the endorsement policy for that chaincode set. Each endorsing peer that receives the transaction verifies that the client that sent it is authorized to do so (on that channel). Once verified and authorized, the peer executes the chaincode function, the planned read and write operations against a copy of the blockchain state, generating the results (response) of the transaction that include: response value, the data set read-set and the modified dataset write-set. After executing, the endorsing peer calls the Endorsement System Chaincode (ESCC) chaincode which signs the response with the identity of the endorsing peer. It then sends the response to the signed transaction to the client application, step 2 in Figure 8. At this point, the client verifies that the response is signed by the endorsing peer. The client collects the responses from the endorsing peers and verifies that the contents of the responses received for said transaction are the same. If it is not, you must forward the transaction to other endorsing peers until you have as many equal responses as established in the approval policy.

Ordering phase. It is in this phase when Hyperledger executes the chosen consensus algorithm. To do this, once the approval phase is complete, the client sends (broadcast) a well-formed transaction message to the Ordering Service, step 3 in Figure 8. This message contains: the read-write sets, the signatures of the endorsing peers who have participated, and the channel identifier. This service, which receives messages

[18] https://hyperledger-fabric.readthedocs.io/en/release-1.4/arch-deep-dive.html

from multiple clients from multiple channels, queues transactions by Channel. Create blocks of transactions by Channel, sign the block with your identity and send the block to all peers on the Channel using a gossip protocol, step 4 in Figure 8.

Validation phase. In this phase, all the peers of a Channel receive the next block to be added for the chain of said Channel, sent from the ordering service through the gossip protocol. Once the block is received, each peer verifies the signature of the Orderer, once the block is authenticated, it is decoded and all the transactions included in it pass first through the validation of the system chaincode Validation System Chaincode (VSCC) and later by the validation of the Multi-Version system Chaincode (MVCC) [7]. The VSCC verifies that the endorsement policy for the chaincode that has executed it has been complied with. If it is not the case, the transaction is marked as invalid. The MVCC checks that the value of the keys read (set) by the transaction in the approval phase are the same as the local state of the record in the peer at the time of confirming the transaction (commit). If the values of both versions of the read-set (endorsement vs. current) do not agree, this implies that a concurrent transaction that has been previously executed (from the same block or previous block) has modified the read-set value so that the transaction is validated at this time should be marked as invalid.

Ledger update phase. This is the last phase for a transaction that has successfully passed all previous phases. The blockchain is updated by adding the new block to its local copy, and updating its state, World State (database with the current value of the key-value pairs) with the write-sets of the transactions marked as valid by the MVCC. Updates to the entire block in the World State are applied atomically.

In the case of sending private information to a subset of the Channel organizations, the workflow presents the following divergences:

Endorsement phase. In this phase, the Client application marks the transaction as private, and sends it to the endorsing peers as before. These simulate it but send the private data of the response to the committing peers of the authorized organizations through the gossip protocol. Next, they respond to the client with the response of the transaction that includes public data - if any -, the hash of the private key and the private data.

Ordering phase. It is in this phase, the client sends (broadcast) a well-formed transaction message, with the hash, to the Ordering Service, and this is processed normally, included in a block and sent to all the committing peers of the Channel (not only the from authorized organizations).

Validation phase. In this phase all Channel peers receive the block, but only the authorized nodes will work with the transactions with private data and will make the corresponding validations.

Ledger update phase. In this phase, authorized peers will store the private data in their SideDB, as long as the corresponding validations have passed successfully.

6.2 HYPERLEDGER FABRIC NETWORK DESIGN

The Fabric network infrastructure is constructed with elements provided by all the organizations such as the peers, and on top of this infrastructure, the Fabric network elements are built, like the channels and the chaincodes. The proposed Hyperledger Fabric architecture design is based on three client organizations (Figure 9):

- ValenciaPort Network
- THPA Network
- Shared common Network

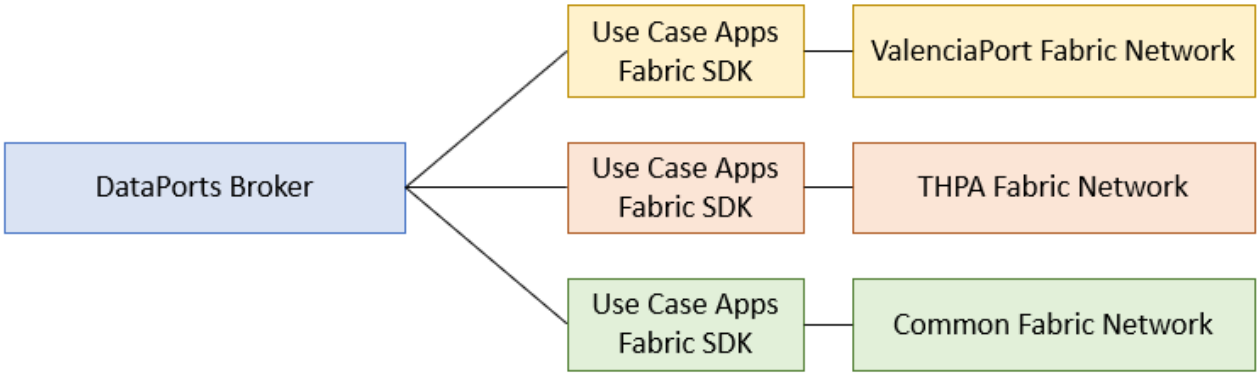


Figure 9 – High level diagram of the blockchain components and networks

This approach that divides the blockchain platform in three separate networks, isolating the development of the use cases individually, helping those in charge of developing and maintaining the testing environments. A third common network shared between all participants will provide a different communication channel for shared use cases, future implementations, and easier integration with the rest of the DataPorts platform.

The following legend applies to the diagrams in Figure 10, Figure 11, and Figure 12 that refer to Hyperledger Fabric technical design and network architecture. Each colour represents a different organization, and every box in the diagram is a Fabric component that will need to be deployed and configured individually. A triangle above a module signifies the Organisation that owns that module and needs to manage and configure it:

- Pi = Peer i
- Si = Chaincode installed in Peer i
- Li = Ledger of Channel i
- CAi = Certificate Authority of Organisation i
- NCi = Network Configuration
- CCNi = Channel configuration
- App = Individual application that communicates with the blockchain from/to the DataPorts platform.

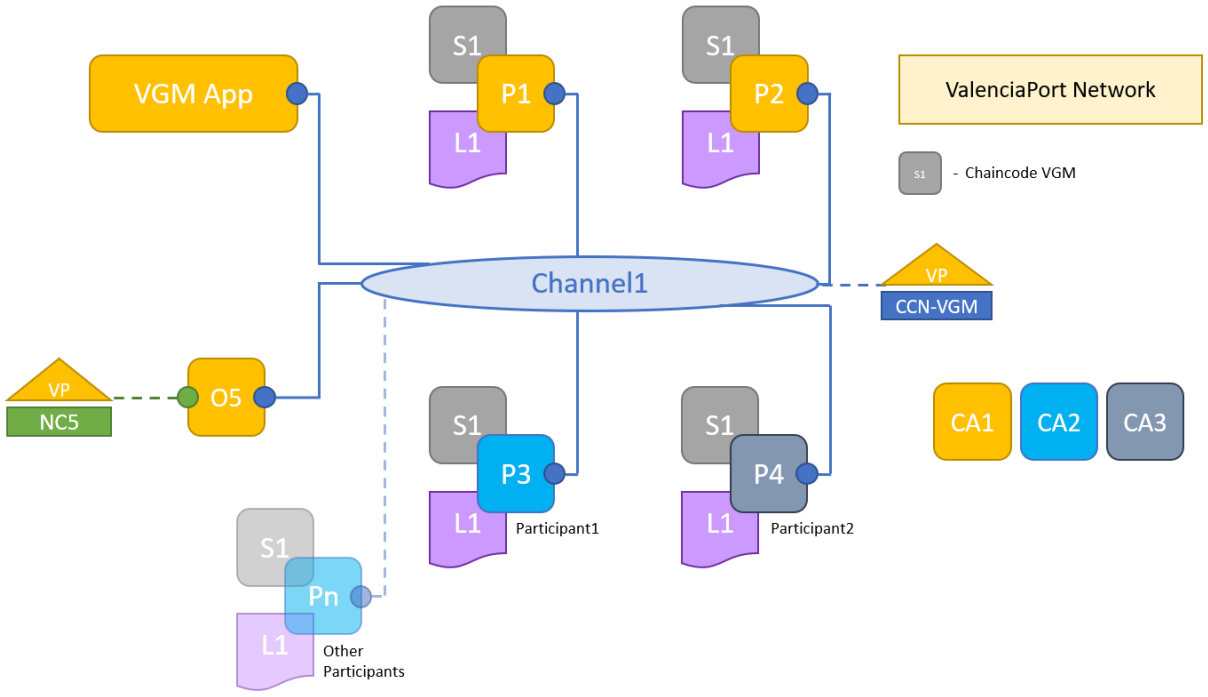


Figure 10 – ValenciaPort Network

In the ValenciaPort Network, the VGM use case will be deployed. The network consists of a single channel with two Endorsing Peers for the ValenciaPort ORG. Additional ORGs can be easily added if new participants external to Valencia Port Authority need to join the network. A single Ordering Service will take care of transactions in the channel. Every ORG has its own Certificate Authority in the network.

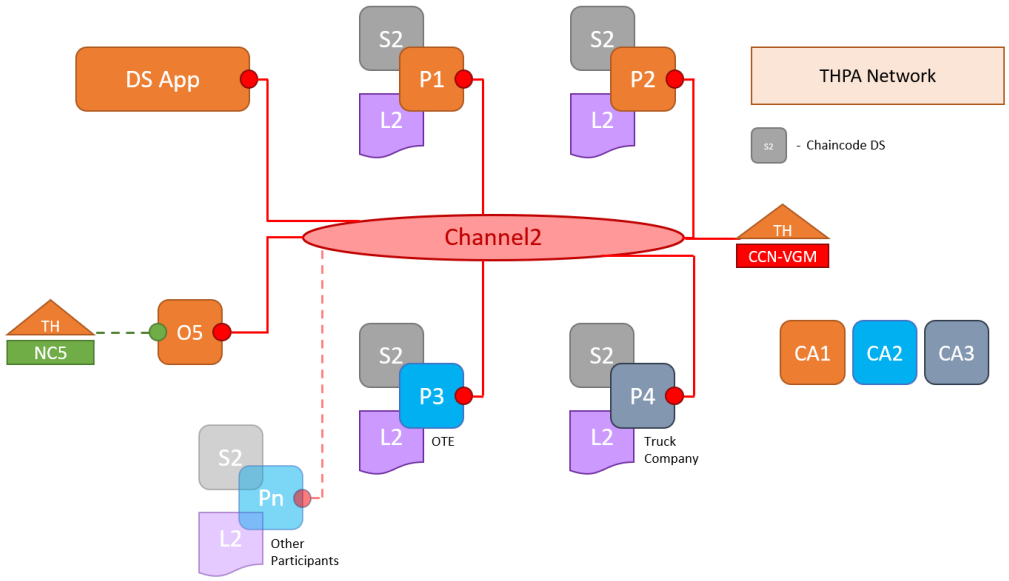


Figure 11 – THPA Network

Similarly, the THPA Network will incorporate the Data Strategy use case. A single channel administered by THPA contains the chaincode and registers all transactions in the network. Two Endorsing Peers for THPA ORG are the minimum requirement, with the possibility of seamless creation of new ORGs for new participants of the network. A single Ordering Service handles all transactions, with every ORG having its own Certificate Authority.

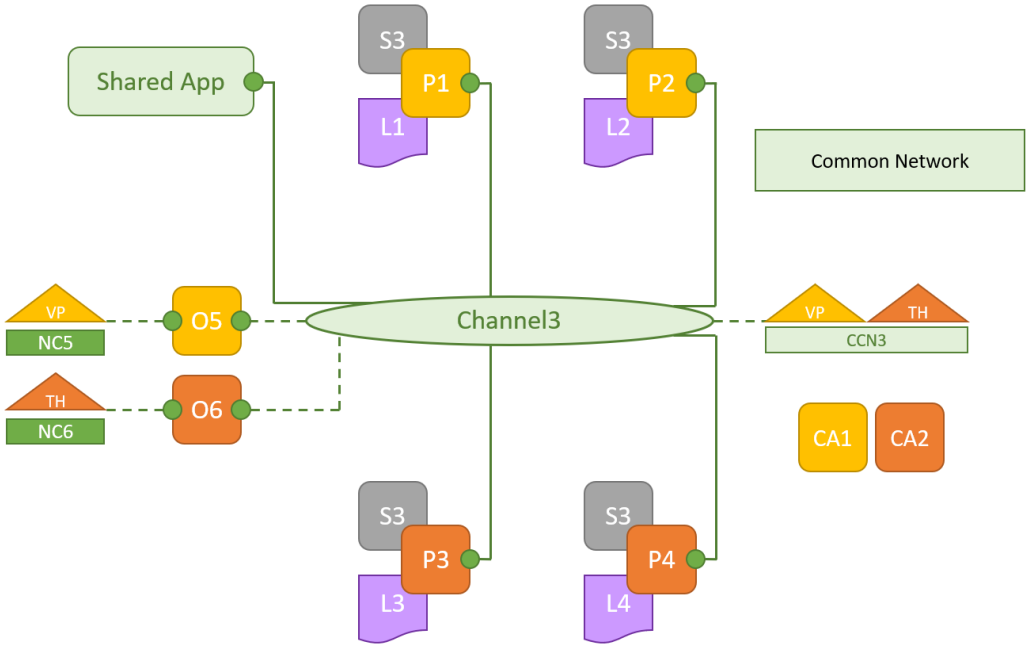


Figure 12 – Common Network

The Shared network will have both ValenciaPort and THPA as participants and will be used for the global use cases developed for the DataPorts platform. Each organization will have two Endorsing Peers each with its

own CA. In order to comply with best practices, in the common network, each organization will have its own Ordering Service.

Client organizations are supposed to already have “Users”, “DataSources” and “Applications”. In Fabric all users are identified by X.509 Certificates Issued by a “CA” and each organization must provide “peers” and an administrative user/role, “Org Admin”, in order to manage the CA and the peers. Applications should be updated to use the DataPorts Blockchain through the “DataPorts Connector”, which will access the Organization’s peers and the “Data Sources”. Each peer provided by the Organization must have its own state database.

6.2.1 Infrastructure

The needed infrastructure will depend on the docker containers19 and resources created from the previous section diagrams.

In general, the system needs some shared resources, two containers per peer, and one for the Ordering Service or the CA.

6.2.1.1 Shared Resources

Within Docker, Hyperledger Fabric needs a network definition to keep all the pods isolated from other containers running on the machine.

Also, three shared volumes must be defined:

- CryptoVolume: this volume will contain all the certificates and private keys to be the system functional. In a full production environment, each Organization should have its own volume containing its own keys, but in this stage, it is no necessary. This volume needs less than 50MB
- ScriptVolume: this volume will contain some utility scripts which help the operation of the blockchain network. This volume needs less than 10MB.
- ChainCodeVolume: This volume will be used to share the chain codes with the CLI, so it will be easier to manage the installation. Recommended 500MB for storing several versions.

As Hyperledger uses docker, it provides some images and binaries who needs hard disk space. Needed at least 2 GB of free space, recommend 4GB.

Lastly on the shared components is the CLI Container, this container will act as the interface of the network. Needs 50 milicores (max 100), 100 MB RAM and 1GB of free space.

Resource	CPU	RAM	Storage
CryptoVolume	0	0	50 MB
ScriptVolume	0	0	10 MB
ChainCodeVolume	0	0	500 MB
Images	0	0	4 GB
CLI container	100 mc	100 MB	1 GB
Total	**100 mc**	**100 MB**	**5.5 GB**

Table 3 – Hardware requirements for the shared resources

[19] https://www.docker.com/resources/what-container

6.2.1.2 Valencia Port Case

The use case of Valencia port uses three CAs, One Ordering Service and four Peers. Each peer needs two containers, the peer itself and the database.

Resource	Units	CPU	RAM	Storage
CA	3	100 mc	50 MB	200 MB
Ordering Service	1	200 mc	100 MB	100 MB
Peer- peer	4	500 mc	200 MB	100 MB
Peer-CouchDB	4	200 mc	200 MB	500 MB
	Total	**3300 mc**	**1850 MB**	**3.1 GB**

Table 4 – Hardware requirements for the ValenciaPort use case

Considering a computer for this Use case, the recommended configuration is:

- 4 Core CPUS or more.
- 4 GB of Ram or more (2 GB plus OS requirements)
- 560 GB of HD or more (8.6 GB plus OS, plus extra data)

6.2.1.3 ThPA Use Case

The use case of ThPA uses three CAs, one Ordering Service and four Peers. Each peer needs two containers, the peer itself and the database.

Resource	Units	CPU	RAM	Storage
CA	3	100 mc	50 MB	200 MB
Ordering Service	1	200 mc	100 MB	100 MB
Peer-Peer	4	500 mc	200 MB	100 MB
Peer-CouchDB	4	200 mc	200 MB	1 GB
	Total	**3300 mc**	**1850 MB**	**5.1 GB**

Table 5 – Hardware requirements for the ThPA use case

Considering a computer for this Use case, the recommended configuration is:

- 4 Core CPUS or more.
- 4 GB of Ram or more (2 GB plus OS requirements)
- 60 GB of HD or more (10.6 GB plus OS, plus extra data)

6.2.1.4 Common Network

The use case for the shared network uses two CAs, two Ordering Services and four Peers. Each peer needs two containers, the peer itself and the database.

Resource	Units	CPU	RAM	HD
CA	2	100 mc	50 MB	200 MB
Ordering Service	2	200 mc	100 MB	100 MB
Peer-Peer	4	500 mc	200 MB	100 MB
Peer-CouchDB	4	200 mc	200 MB	250 MB

	Total	**3400 mc**	**1900 MB**	**2 GB**

Table 6 – Hardware requirements for the Shared Network

Considering a computer for this Use case, the recommended configuration is:

- 4 Core CPUS or more.
- 4 GB of Ram or more (2 GB plus OS requirements)
- 40 GB of HD or more (7.5 GB plus OS, plus extra data)

6.2.2 Network governance

To add a new organization in any network, the organization must create the basic resources, such as the peers and the CA and some starting users, and then the Network Administrator will configure the existing channels for accepting the user on the consensus definition. Last the Organization Administrator will join the peers to the channels.

When an Organization wants to add a new peer it needs to create the certificates of the peer, the containers required of the peers (peer and state database) and publish the peer in a fixed DNS inside the TCP/IP network of the fabric network. In a similar way this was done when the organization was included on the network. After the node is started, the administrator must send a network update in order to include the node on the consensus mechanism.

After the node is deployed in the network, is time for it to join it the existing channels. It is a process as easy as issuing a join command to the peer with the channel file generated by the Orderer Peers when the channel is created.

New channels can be created by creating a channel configuration file, “configtx.yaml”, defining the organizations who can join the channel, endorsement policies, and consensus configuration.

Then the Network administration must create the channel transaction and sign it. Once signed, the Network Administrator must send it to an Orderer Peer to create it. This operation will create the block file needed to join peers. After the channel is created the peers must join the channel.

Chaincodes must be installed in Endorsing Peers that are participants of the intended channel. Once every Peer has a copy of the chaincode, it can be instantiated in a channel. This operation will initialize the chaincode in the channel for every participant to make calls to it and propose transactions. The Channel Administrator will need to undertake this operation.

6.2.3 Description of the components

APIs

The exposed APIs serve as an interface for external systems and users to interact with the solution. The implementation of the services proposed is Spring Boot. These services will be responsible for:

The REST APIs exposed are yet to be defined and will be based on the chosen uses cases and will be detailed in the upgrade document.

Blockchain Connector

The Java SDK for Hyperledger Fabric helps facilitate Java applications to manage the lifecycle of Hyperledger channels and user chaincode. The SDK also provides a means to execute user chaincode, query blocks and transactions on the channel, and monitor events on the channel.

The SDK acts on behave of a particular User which is defined by the embedding application through the implementation of the SDK's User interface.

The SDK also provides a client for Hyperledger's certificate authority. The SDK is however not dependent on this particular implementation of a certificate authority. Other Certificate authorities may be used by implementing the SDK's Enrolment interface.

Certification authority (CA)

As Hyperledger Fabric a private and authorized network, only identified and identifiable participants can make transactions and view them. To manage identities Fabric includes an MSP component (Membership Service Provider), an abstract component of the system that provides credentials to clients and their peers.

Hyperledger Fabric CA is the default Certification Authority (CA), issuing certificates based on PKI (Public Key Infrastructure) to organizations and their users. The CA issues a root certificate (rootCert) for each organization and an enrolment certificate (ECert) for each authorized user.

PKIs and MSPs work together: a PKI provides a list of identities and an MSP indicates which identities belong to the members of a particular organization that participates in the network.

Organizations can choose and implement an External Certification Authority. Hyperledger Fabric is compatible with many MSPs and a Hyperledger Fabric network can be controlled by multiple MSPs.

By default, the Fabric CA server and client store private keys in a PEM-encoded file, but they can also be configured to store private keys in an HSM (Hardware Security Module) via PKCS11 APIs. This behaviour is configured in the BCCSP (Blockchain Crypto Service Provider) section of the server's or client's configuration file.

In addition, The Hyperledger Fabric CA client or SDK may connect to a server in a cluster of Hyperledger Fabric CA servers and supports LDAP for user authentication. The client routes to an HA Proxy endpoint which load balances traffic to one of the fabric-ca-server cluster members.

All Hyperledger Fabric CA servers in a cluster share the same database for keeping track of identities and certificates. If LDAP is configured, the identity information is kept in LDAP rather than the database.

Finally, it will be seen later how certificates issued by CAs are at the heart of the transaction generation and validation process. Specifically, X.509 certificates are used in client application transaction proposals and smart contract transaction responses to digitally sign transactions. Subsequently the network nodes who host copies of the ledger verify that transaction signatures are valid before accepting transactions onto the ledger.

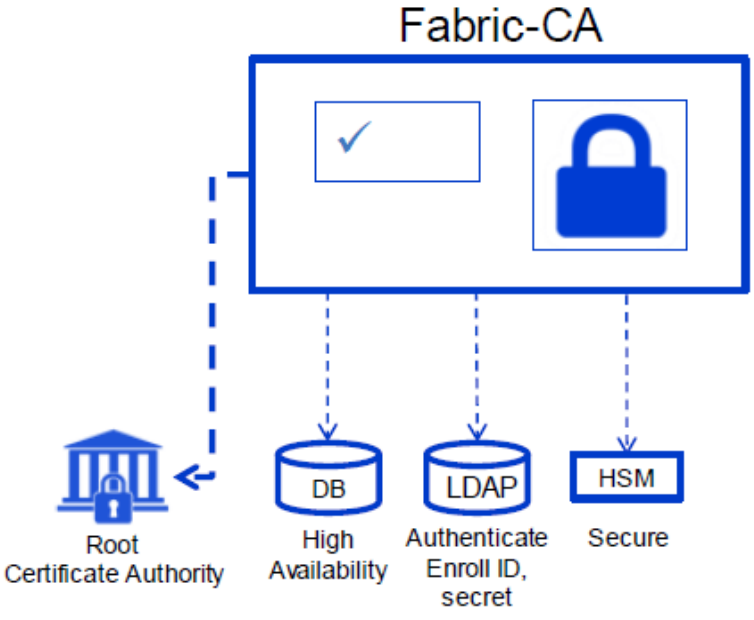


Figure 13 – Hyperledger Fabric Certificate Authority Architecture

Membership Service Provider (MSP)

The Membership Service Provider is a component that aims to provide an abstraction of a membership operation architecture.

More specifically, MSP is responsible for abstracting all the cryptographic mechanisms and protocols that manage the issuance and validation of certificates and the authentication of users. An MSP can define its own notion of identity and the rules by which those identities are governed (identity validation) and authenticated (generation of signatures and verification).

Hyperledger can be governed by one or more MSP. This provides the modularity of membership operations and interoperability through different architectures and membership standards.

The configuration of an MSP is communicated to all the channels in which the members of the corresponding organization participate through a channel. In addition to the MSP channel, peers, computers, and clients also maintain a local MSP to authenticate messages from members who do not have a channel and to define permissions on a particular component.

Each local MSP includes a keystore with a private key for signing transactions and a signcert with a public x.509 certificate. May also include TLS credentials and can be backed by a Hardware Security Module (HSM). In addition, an MSP can allow the identification of a list of identities that have been revoked

To configure an instance of the MSP, its configuration must be specified locally on each pair and payer and channels, to enable validation of client identity, par, order, and the respective signature (authentication) of all channel members.

At this time, the MSP will identify which root CA and which intermediate CAs are reliable to define the members of a trusted domain, such as an organization, and may do so by listing the identities of their members or identifying which CAs are authorized to issue. identities valid for its members or, more commonly, by a combination of both.

However, an MSP is not only able to list who is a network participant or member of a channel. He can also identify specific roles within the organization that represents the MSP and establish the basis for defining access privileges in the context of a network and channel.

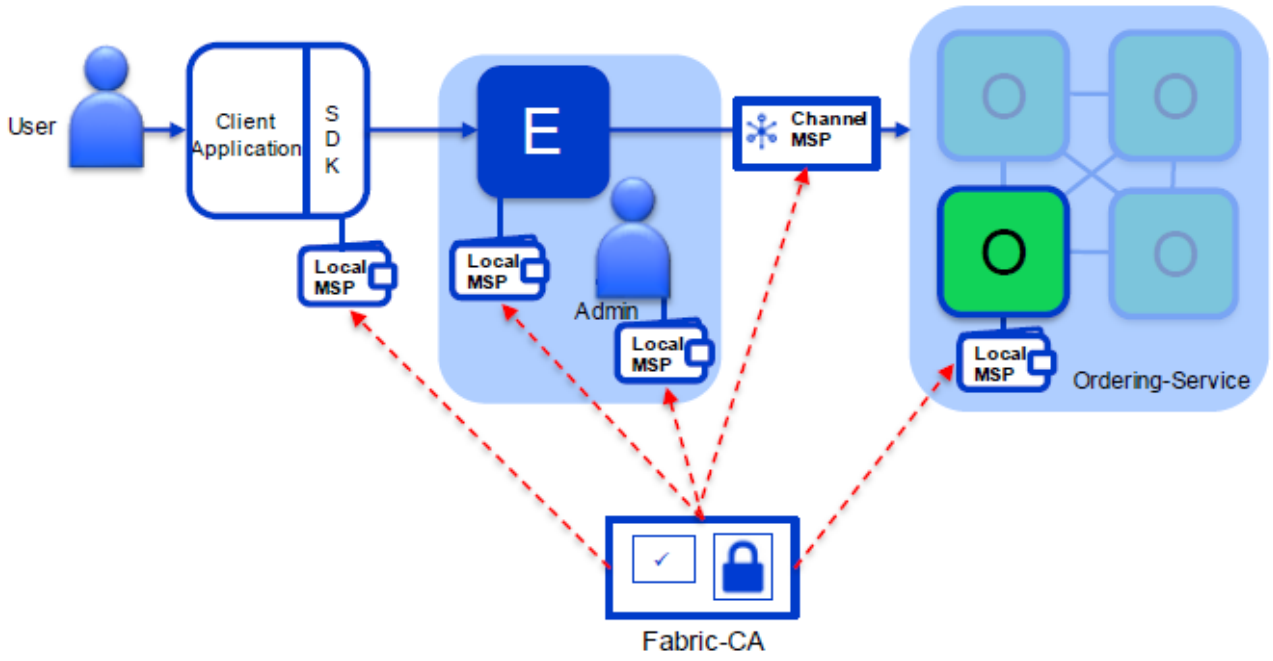


Figure 14 – Relationship between CAs and MSPs

Channels

The channels are an overlay of the chain of blocks that isolate the data and provide confidentiality; only members who are duly authenticated to access that channel can interact with it. Each channel has its own ledger that contains only the transactions that are made in that channel.

When a new channel is created, in its start block or genesis block, the specific rules that will govern this channel are established: participants, smart contracts (chaincode), consensus, certification authorities, endorsement nodes, etc.

Each transaction on the network is executed on a channel, where each party must be authenticated and authorized to transact on that channel. Each peer that joins a channel, has its own identity given by a Membership Services Provider (MSP), which authenticates each peer to its channel peers and services.

Orderer

The Orderer (also known as an "Ordering Service") is the node that approves the inclusion of transaction blocks into the ledger and communicates with committing and endorsing peer nodes which along with other nodes forms an ordering service.

It also provides advantages to the network at the level of performance and scalability, eliminating bottlenecks that can occur when execution and ordering are performed by the same nodes.

- **Orderer nodes and channel configuration**

In addition to their ordering role, Orderers also maintain the list of organizations that are allowed to create channels. This list of organizations is known as the "consortium", and the list itself is kept in the configuration of the "orderer system channel" (also known as the "ordering system channel"). By default, this list, and the channel it lives on, can only be edited by the Orderer admin and it is possible for an ordering service to hold several of these lists.
Orderers also enforce basic access control for channels, restricting who can read and write data to them, and who can configure them.

Configuration transactions are processed by the Orderer, as it needs to know the current set of policies to execute its basic form of access control. In this case, the Orderer processes the configuration update to make sure that the requestor has the proper administrative rights. If so, the Orderer validates the update request against the existing configuration, generates a new configuration transaction, and packages it into a block that is relayed to all peers on the channel. The peers then process the configuration transactions in order to verify that the modifications approved by the Orderer do indeed satisfy the policies defined in the channel.

Ledger

In Hyperledger Fabric, a ledger consists of two distinct, though related, parts – a world state and a blockchain. Each of these represents a set of facts about a set of business objects.

The world state is a database that contains the current value of the set of key-value pairs that have been added, modified or eliminated by the set of transactions that have been endorsed and committed in the chain of blocks. The world state makes it easier for a program to obtain the current value of those states, instead of having to calculate them through the entire transaction log. The world state can change frequently, since states can be created, updated, and eliminated.

The world state can be configured to support a variety of DBMSs (Data Base Management System). Fabric for now allows the use of LevelDB (default) and CouchDB.

As blockchain, is responsible of transaction log that records all the all sequential transactions that have resulted in the current the world state. Transactions are collected inside blocks that are appended to the blockchain, enabling you to understand the history of changes that have resulted in the current world state. The blockchain data structure is very different to the world state because once written, it cannot be modified; it is immutable but allows to see all the changes that resulted in the current value of the world state.

Smart Contracts (Chaincode)

Whereas a ledger holds facts about the current and historical state of a set of business objects, a smart contract defines the executable logic that generates new facts that are added to the ledger.

In Hyperledger Fabric, smart contracts are called chaincode, they can be written in standard programming languages, such as Golang or Node.js and eventually in other programming languages such as Java. Smart contracts handle the logic of business agreed by the network participants; is a software that runs in a Docker container protected and isolated from the approval process of the nodes, enforces the rules to read or modify the key-value pairs or other world state information. Chaincode initializes and manages the general ledger status through the transactions sent by the applications.

The state created by a chaincode has an exclusive scope for that chaincode and cannot be accessed directly by another chaincode. However, within the same network, giving the corresponding permission, a chaincode can invoke another chaincode to access its state.

Most importantly however, the execution of a smart contract is much more efficient than a manual human business process.

Hyperledger Fabric users often use the terms smart contract and string code interchangeably. In general, an intelligent contract defines the transaction logic that controls the life cycle of a business object contained in the global state. Then it is packaged in a chain code that is then implemented in a blockchain network. In this way, smart contracts could be considered as governing transactions, while Chaincode manages how smart contracts are packaged for implementation.

Endorsement

Associated with every chaincode is an endorsement policy that applies to all of the smart contracts defined within it. An endorsement policy describes the conditions by which a transaction can be endorsed. It indicates which organizations in a blockchain network must sign a transaction generated by a given smart contract because a transaction can only be considered valid if it has been endorsed according to its policy.

However, other policies can be defined to identify who can query or update the ledger or add or remove participants from the network. Normally, policies should be agreed in advance by the consortium of organizations in a blockchain network, but this can be change. Indeed, policies themselves can define the rules by which they can be changed. Even, it is also possible to define custom endorsement policy rules over and above those provided by Fabric.

Each chaincode is deployed with an Endorsement Policy. While the **ESCC** (Endorsement System Chaincode) is responsible for signing the response of the proposal response on the endorsing peer, the **VSCC** (Validation System Chaincode) validates the endorsements. These chaincodes are to be deployed in the three networks without any modification.

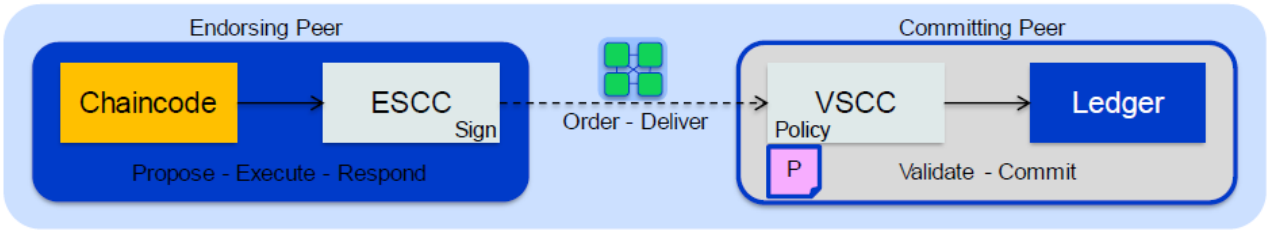


Figure 15 – Difference between Endorsing and Committing Peers in Hyperledger Fabric

7 CONCLUSIONS AND FUTURE WORK

The conducted analysis shows that Hyperledger Fabric is the most suitable technology for the use cases shown. Fabric is able to leverage all the potential benefits of blockchain technology and apply them in an enterprise environment where business rules can be automated, and governance of the network is shared among different partners.

The network design with three separate networks provides independence to the pilots with isolated environments to test and deploy applications, whereas the shared network can be used for global use cases and scenarios.

In the foreseeable future, the following topics must be discussed in order to continue the design and development of the blockchain components:

- Determination of the virtualization containers and hardware distribution for the blockchain components to be deployed.
- Provisioning of the cloud-hosted virtual machines.
- Further analysis of the interconnection with the rest of components of the DataPorts platform (e.g. the broker).
- Determination of the number and content of the chaincodes to be deployed in the individual networks.
- Further explaining the asset chaincode lifecycle.
- Detailing the role of the shared network and the operations to be performed in it.
- Align with the IDS Reference Architecture Model when possible.
- Determine the datasets – and metadata derived from them – to be included in the platform.

8 REFERENCES AND ACRONYMS

8.1 REFERENCES

[1] V. Buterin, The Ethereum Whitepaper, 2014.

[2] N. Szabo, The Idea of Smart Contracts, 1997.

[3] A. H. a. S. M. Patrick McCorry, “[BITCOIN '18] Smart Contracts for Bribing Miners”.

[4] J. Chase, Quorum Whitepaper, 2016.

[5] V. Authors, Hyperledger Fabric Official Documentation, 2015.

[6] R. G. Brown, The Corda Whitepaper, 2018.

[7] G. Bernstein, Multiversion Concurrency Control-Theory and Algorithms, 1983.

[8] S. Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, 2009.

8.2 ACRONYMS

Acronyms List	
DLT	Distributed ledger Technologies
P2P	Peer to peer
EVM	Ethereum Virtual Machine
CA	Certificate Authority
DB	DataBase
ETA	Estimated Time of Arrival
MSP	Membership Service Provider
WSDB	World State Database
SDK	Software Development Kit
PKCS	Public-key Cryptography Standard
HSM	Hardware Security Module
MVCC	Multi-Version ChainCode
ESCC	Endorsement System ChainCode
VSCC	Validation System ChainCode

Table 7 – Acronyms

Annex A – Use Cases Data Sources

- **VGM Use Case (VPF)**

- **User:**
  - id: *string*
  - email: *string*
  - password: *string*
  - salt: *string*
  - passwordResetToken?: *string*
  - passwordResetExpires?: *date*
  - disabled: *boolean*
  - approved: *boolean*
  - fullName: *string*
  - acronym?: *string*
  - picture?: *string*
  - companyVAT: *string*
  - companyName: *string*
  - privacyTermsAgreed: *boolean*
  - legalTermsAgreed: *boolean*
  - roles: []
    - role: enumeration (sysAdmin, shipper, vgmAgent, scaleOperator, haulier, shippingAgent, guest)
  - emailSubscriptions: []
    - subscription: string
  - tokens: []
    - accessToken: string
    - kind: string
    - login: string
    - date?: date
- **Company:**
  - id: string
  - VAT: string
  - businessName?: string
  - name: string
  - companyType: enumeration (haulier, shipper, scaleOperator, vgmAgent, shippingAgent)
  - address?: string
  - city?: string
  - postalCode?: string
  - country?: string
  - phone?: string
  - users: []

        - id: string
    - businessRelations: []
        - companyId: string
        - VAT: string
        - businessName?: string
        - name: string
        - companyType: enumeration ()
        - address?: string
        - city?: string
        - postalCode?: string
        - country?: string
        - phone?: string
- **AccountingEntry**: []
    - id: string
    - companyId: string
    - type: enumeration (credit, debit)
    - quantity: number
    - date: date
    - operationCode: string
    - vgmId: string
- **Billing: []**
    - id: string
    - invoiceId: string
    - date: date
    - amount: number
    - remarks: string
    - status: enumeration (generated, validated, paid)
    - hash: string
    - link: string
- **Scale**
    - id: string
    - companyId: string
    - name: string
    - type: enumeration (public, private)
    - active: boolean
    - price: number
    - openingTime?: string
    - gaugeEntity?: string
    - gaugeDate?: date
    - location?: string
    - address?: string

- city?: string
- postalCode?: string
- country?: string
- phone?: string
- gpsLocation?: string

- **Vehicle**
  - id: string
  - companyId: string
  - type: enumeration (truck, semitrailer)
  - plate: string
  - grossWeight: number (kilograms)
  - tankVolume?: number (litres)
  - active: Boolean
- **VGM**
  - id: string
  - status: enumeration (requested, weighted, errors, finalised)
  - company:
    - id: string
    - VAT: string
    - name: string
    - userId: string
    - request:
    - requestDate: date
    - requestNumber: string
    - requestLocator: *string*
  - weightMethod: enumeration (calculated, weighted)
  - bookingNumber: string
  - containerNumber: string
  - containerType?: string
  - gateInOrder?:
    - gateInReference?: string
    - gateInLocator?: string
  - weight?:
    - weighDate: date
    - scaleId: string
    - price: number
  - weightOperator?:
    - VAT: string
    - userId: string
  - vgmResponsible?: string
  - shipper?:

        - name?: string
        - VAT?: string
        - address?: string
        - city?: string
        - postalCode?: string
    - haulier?:
        - name?: string
        - VAT?: string
        - address?: string
        - city?: string
        - postalCode?: *string*
    - vgmAgent?:
        - name?: string
        - VAT?: string
        - address?: string
        - city?: string
        - postalCode?: *string*
    - shippingAgent?:
        - name?: string
        - VAT?: string
        - address?: string
        - city?: string
        - postalCode?: *string*
    - shippingLineSCAC?: string
    - vgm?: number
    - scaleWeight?: number
    - scaleMethod?: enumeration (composition, container)
    - scaleValid?: boolean
    - scaleCertificationId?: string
    - transport?:
        - truckPlate?: string
        - truckId?: string
        - semitrailerId?: string
        - semitrailerPlate?: string
        - tankLevel?: enumeration (low, medium, full)
        - fuelWeight?: number
        - transportReference?: *string*

- **Consignment Note Use Case (VPF)**

- **User:** (User personal data should not be stored on-chain)
  - id: string
  - email: string
  - password: string
  - salt: string
  - passwordResetToken?: string
  - passwordResetExpires?: date
  - disabled: boolean
  - approved: boolean
  - fullName: string
  - acronym?: string
  - picture?: string
  - companyVAT: string
  - companyName: string
  - privacyTermsAgreed: boolean
  - legalTermsAgreed: boolean
  - roles: []
    - role: enumeration (sysAdmin, shipper, vgmAgent, scaleOperator, haulier, shippingAgent, guest)
  - emailSubscriptions: []
    - subscription: string
  - tokens: []
    - accessToken: string
    - kind: string
    - login: string
    - date?: date
- **Company:**
  - id: string
  - VAT: string
  - businessName?: string
  - name: string
  - companyType: enumeration (haulier, shipper, scaleOperator, vgmAgent, shippingAgent)
  - address?: string
  - city?: string
  - postalCode?: string
  - country?: string
  - phone?: string
  - users: []
    - id: string

- businessRelations: []
  - companyId: string
  - VAT: string
  - businessName?: string
  - name: string
  - companyType: enumeration ()
  - address?: string
  - city?: string
  - postalCode?: string
  - country?: string
  - phone?: string

- **ConsignmentNone:** PDF/A format
  - Consignment note number
    1. Sender
    2. Consignee
    3. Place of delivery of the goods
    4. Place and date of taking over the goods
    5. Documents attached
    6. Marks and nos
    7. Number of packages
    8. Method of packing
    9. Nature of goods
    10. Statistical number
    11. Gross weight in kg
    12. Volume in m3
    13. Sender's instructions
    14. Instructions as a payment for carriage
    15. Cash on delivery
    16. Carrier (name, address, country)
        - Transport reference
        - Vehicle plate
        - Semi-trailer plate
        - Distance
    17. Successive carriers (name, address, country)
    18. Carrier's reservations and observations
    19. Special agreements
    20. To be paid by
    21. Established in

22. Signature and stamp of the sender
23. Signature and stamp of the carrier
24. Goods received
    - Signature and stamp of the consignee

- **controlDocument:** PDF/A format
  - Sender / contractual shipper
  - Sender's remarks
  - Carrier
  - Carrier's remarks
  - Place of taking over the goods
  - Place of delivery of the goods
  - Nature of goods
  - Weight
  - Number of packages
  - Delivery date
  - Truck plate
  - Semi-trailer plate
  - Other vehicles plate (trucks and semi-trailers)
  - Consignee
  - Signature of sender
  - Signature of carrier
- **Documents:** (documents data should not be stored on-chain)
  - documentToken: string
  - documentType: enumeration (consignmentNote, controlDocument, proofOfDelivery)
  - content: binary
- **TransportDocument:**
  - id: string
  - createdAt: date
  - updatedAt: date
  - createdByToken?: string
  - transportMovementToken: string
  - senderToken?: string
  - carrierToken?: string
  - consigneeToken?: string
  - signedBySender?: boolean
  - signedByCarrier?: boolean
  - signedByConsignee?: boolean
  - deliveryDate?: date
  - consignmentNoteHash: string
  - controlDocumentHash: string

- tracingAudit: []
  - userToken: string
  - registeredAt: date
  - action: string
  - consignmentDocumentHash: string
  - controlDocumentHash: string

- **Data driven application for strategic and real time decisions (ThPA)**
  - From Terminal Operating System

FIELD NAME	TYPE
ARRIVAL_DATETIME	datetime
DEPART_DATETIME	datetime
CTYPE_CODE	string
CSIZE_CODE	string
LOADED	int (boolean)
ACTIVE	int (boolean)
DANGEROUS_CARGO	int (boolean)
WASTES_CARGO	int (boolean)
IMDG_CLASS_ID	string
CATEGORY_ID	integer
SHIPCAT_DESCRIPTION	string
DIRECTION	string
VO_CODE_IMP	string
VESSEL_NAME_IMP	string
VO_CODE_EXP	string
VESSEL_NAME_EXP	string
ORIGIN_PORT_ID	string
ORIGIN_COUNTRY	string
DESTIN_PORT_ID	string
DESTIN_COUNTRY	string

PERMIT_NO	integer
CONTAINER_ID	string
PACKAGE_ID	integer
UNITS	integer
DECLARED_WEIGHT	integer
CONTAINER_WEIGHT	integer
THEORETICAL_WEIGHT_SUM	integer
CRANE_WEIGHT	integer
VGM_WEIGHT	integer
CARGO_DESCRIPTION	string
TT_ID_IN	string
TT_ID_OUT	string
CS_CLASS_ID	integer
CS_ID	integer
COMPANY_ID	string
COM_COMPANY_ID	string
COM2_COMPANY_ID	string
CONTAINER_OWNER	string

- From Terminal Appointment System

FIELD NAME	TYPE
ID	integer
timeWindowID	integer
bookDate	integer (to be parsed as date/time)
userID	integer
companyID	integer
bookingNumber	integer
containersNumberExport	integer
containersNumberImport	integer
exportContainer1	string
exportContainer2	string
importContainer1	string
importContainer2	string
status	integer
changedBy	integer
changedDateTime	integer (to be parsed as date/time)
enterDateTime	integer (to be parsed as date/time)
origTimeWindowFrom	integer
timeWindowFrom	integer
origTimeWindowTo	integer
timeWindowTo	integer
timeWindowDate	integer (to be parsed as date)
isTransit	int (boolean)

- From Vessel Calls

FIELD NAME	TYPE
ship_descr	string
imo_code	integer
date_katapl	datetime
time_prosdesi	datetime
time_apodesi	datetime
start_work	datetime
end_work	datetime
departure_date	datetime
work_latin_descr	string
empor_latin_descr	string
cf_empor_action	integer
cf_empty	integer
cf_emforta	integer
cf_timh	number
cf_tonnoi	number
ship_type	integer
code	integer
descr_latin	string
class	integer
class_descr_latin	string

- From Gate Control

FIELD NAME	TYPE
vRegNo	string (license plate)
vMaker	string
vModel	string
vColour	string
vCompany	string
inACU	int (gate sensor ID)
inTime	bigint (UNIX timestamp)
outACU	int (gate sensor ID)
outTime	bigint (UNIX timestamp)
RFID	string (RFID tag)
vehicleType	int

- **Improve mobility of passengers, visitors, and professionals of the port (ThPA)**

Field Name	Data Type	Field Description
distinct_users	bigint	Number of distinct users
voice_in	bigint	Number of incoming voice calls
voice_out	bigint	Number of outgoing voice calls
sms_in	bigint	Number of incoming sms
sms_out	bigint	Number of outgoing sms
bytes_up	bigint	Total bytes uploaded
bytes_down	bigint	Total bytes downloaded
cell_municipality	string	Municipality which group of cells, that users for this time period used, are located
cell_lat	string	Latitude which group of cells, that users for this time period used, are located
cell_lon	string	Longitude which group of cells, that users for this time period used, are located
prev_cell_municipality	string	Municipality which group of cells, that users used for the previous time period, are located
prev_cell_lat	string	Latitude which group of cells, that users used for the previous time period, are located
prev_cell_lon	string	Longitude which group of cells, that users used for the previous time period, are located
time_period	int	Time period in Hours (e.g. 0, 1, 2…23)
event_date	string	Date of events
week_day	string	Day of the week
is_weekend	string	Flag if this date is weekend day
is_bank_holiday	string	Flag if this date is a bank holiday