D4.2 Smart Contracts and DApps implementation tools I

Document Identification			
Status	Final	**Due Date**	31/07/2021
Version	1.0	**Submission Date**	30/07/2021

Related WP	WP4	**Document Reference**	D4.2
Related Deliverable(s)	D2.3, D3.3	**Dissemination Level (*)**	PU
Lead Participant	INNOV	**Lead Author**	Nikos Kapsoulis
Contributors	ICCS/NTUA, i2CAT, HOLO, FILL	**Reviewers**	Francesco Iadanza (ENG) Verena Stanzl (FILL)

Keywords:
Smart Contracts, DApps, Edge, Cloud, Distributed Ledger Technology, Blockchain Network, Service-Level Agreements, Orchestration

(*) Dissemination level: **PU**: Public, fully open

Document Information

List of Contributors	
Name	Partner
Nikos Kapsoulis	INNOV
Estela Carmona Cejudo	i2CAT
August Betzler	i2CAT
Verena Stanzl	FILL
Nour Fendri	HOLO

Document History			
Version	Date	Change editors	Changes
0.1	20/05/2021	Nikos Kapsoulis (INNOV)	Table of Contents formation
0.2	11/06/2021	Nikos Kapsoulis (INNOV)	Initial contributions in all sections
0.3	29/06/2021	Nikos Kapsoulis (INNOV)	Technical contributions and diagrams
0.4	30/06/2021	Estela Carmona Cejudo (i2CAT), August Betzler (i2CAT)	Contributions in data models
0.5	06/07/2021	Verena Stanzl (FILL)	Contributions in data models
0.6	06/07/2021	Nikos Kapsoulis (INNOV)	Ready for internal review
0.7	15/07/2021	Francesco Iadanza (ENG), Verena Stanzl (FILL)	Internal review
0.8	16/07/2021	Nikos Kapsoulis (INNOV)	Ready for quality review
0.9	27/07/2021	Carmen San Román (ATOS)	Quality assurance review
1.0	30/07/2021	Lara López (ATOS)	Final version to be submitted

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	Nikos Kapsoulis (INNOV)	16/07/2021
Quality manager	Carmen San Román (ATOS)	27/07/2021
Project Coordinator	Lara López (ATOS)	30/07/2021

Table of Contents

List of Tables

List of Figures

List of Acronyms

Abbreviation / acronym	Description
DApp	Decentralized Application
DLT	Distributed Ledger Technology
Dx.y	Deliverable number y belonging to WP x
OBU	On-Board Unit
RSU	Road Side Unit
SLA	Service-Level Agreement
Tx.y	Task number y belonging to WP x
UC	Use Case
VRU	Vulnerable road users
WP	Work Package

Executive Summary

This document accompanies the delivery of T4.2 "Smart Contracts and DApps on edge and cloud" results during the first phase. This is the first iteration of the deliverable and the prototype DLT where the trusted nodes and privacy rules that apply under its schema are introduced. The entire prototype is lead towards supporting the needs of the edge and cloud environments as demanded by the use cases.

Task 4.2 focuses on the deployment of the smart contracts and DApps in a trusted environment of reliable blockchain nodes. Every Pledger component that demands and realizes blockchain activities is supported by the task features and modules while maintained by the Pledger DLT. Furthermore, the task results are designed to handle the interoperability and scalability framework of the project between smart contracts and SLAs.

The delivered output on this iteration defines the basic foundation of the task. Every module that is already built or is scheduled to is supported and hosted under the installed prototype of the first iteration. Pledger DLT comprises an orchestrated permissioned blockchain that is dedicated to the project with trusted blockchain nodes where smart contracts and DApps reside. Pledger DLT incorporates private environments that are providing confidential blockchain activity among network users.

1 Introduction

1.1 Purpose of the document

The purpose of deliverable D4.2 is to provide the insights and capabilities of the underlying DLT deployment, the completed network integration along with the applicable functionalities supporting smart contracts and DApps for the current and future release.

This deliverable accompanies the prototype during the first iteration of WP4 and provides the necessary installation and usage guidelines of the demonstrator.

1.2 Relation to other project work

The prototype and the deliverable are guided by the work performed in WP2 about the architecture (D2.3) and WP3 about the SLA tools and components (D3.3).

The prototype constitutes the integrated foundation that will serve core demands in smart contracts and DApps in the next iteration. The prototype provides a permissioned blockchain network for Pledger with trusted nodes and privacy rules that apply under its schema. Also, the first version introduces orchestrated nodes while paving the way for smart contract governance.

1.3 Structure of the document

This document is structured in six major chapters:

Chapter 2 presents the functional overview of the prototype.

Chapter 3 elaborates on the technical description of the prototype, its specific components architecture and data models.

Chapter 4 presents the installation and usage guide of the prototype.

Chapter 5 presents the demonstration of the prototype with the description of scenarios.

Chapter 6 presents the conclusions and next steps for this initial iteration of the prototype.

2 Functional description

Pledger DLT is a blockchain network specifically deployed for the project and its main purpose is to host and support the other task 4.2 modules along the SLA subsystem triggers and use cases adaptive demands.

In Pledger functional architecture, the DLT is graphically placed in the Support layer as shown in Figure 1. The actual blockchain deployment relays on a decentralized architecture that spans across different networks and locations.

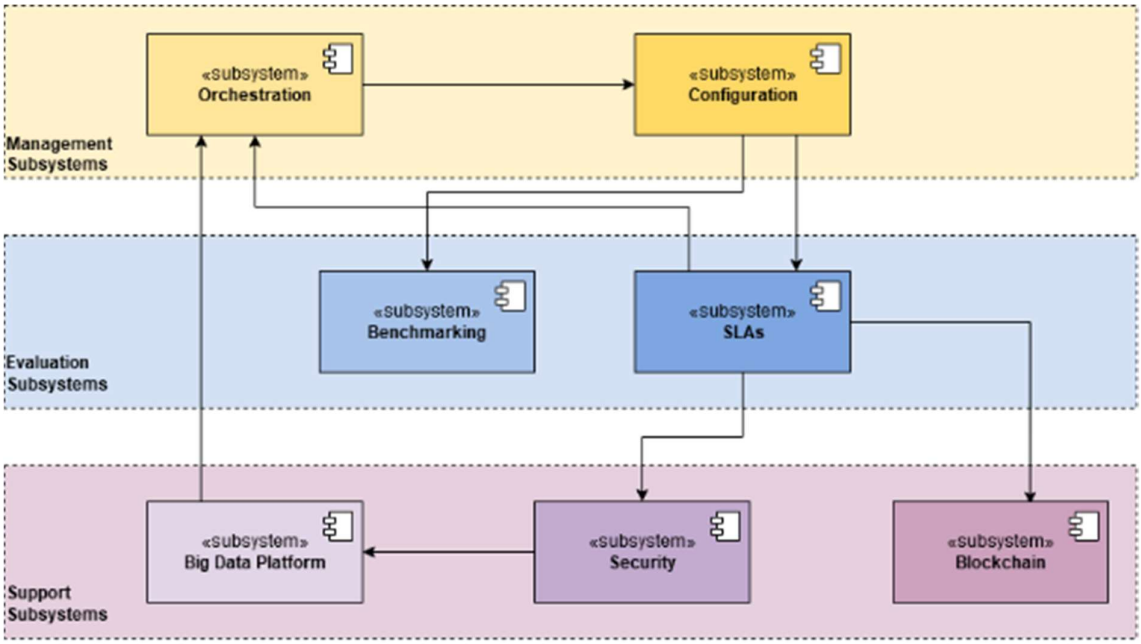

Figure 1: Pledger Core Subsystem [1]

As mentioned, Pledger provides its own blockchain, taking under consideration features such as openness, access, speed, security, consensus mechanisms, etc. As extracted from the architecture deliverable (D2.3), Figure 2 shows a functional insight of the Blockchain subsystem components and their relations.

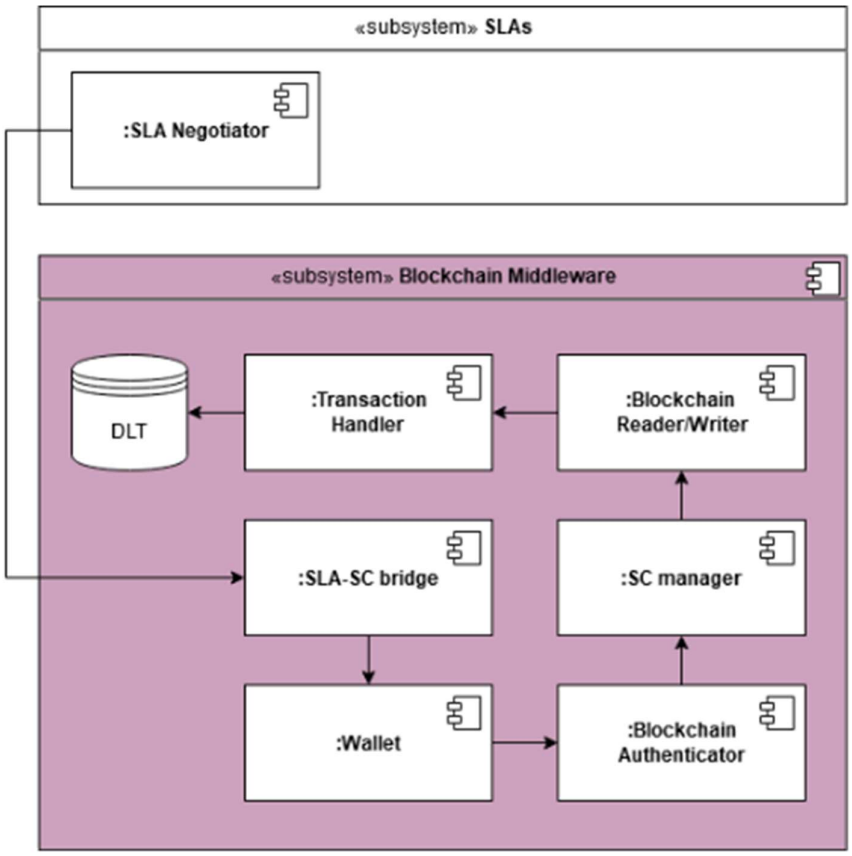

Figure 2: Blockchain subsystem component diagram [1]

The list of the Blockchain subsystem functional components implemented and hosted on the DLT and their roles is reported in the Table 1:

Table 1: Blockchain subsystem Components [1]

ID	Component	Functionality
FC.5.1	**Blockchain Authenticator**	This component grants access to specific data of the blockchain network, exploiting the corresponding credentials information of the components or users that are requesting the access, i.e., private keys and certificates.
FC.5.2	**Blockchain Reader/ Writer**	This component fetches the requested data from the blockchain and submits transactions on it. The module consists of various methods that describe the way and technique or query specification the information is read from the blockchain. Different data need different kinds of reading and fetching, hence the module is constituted from several sub-modules.
FC.5.3	**Wallet**	This component manages the credential information of a blockchain participant entity (either it is a component, administrator, end user or other). The module is the main method with which an entity is present inside the blockchain network through its transaction activities and smart contract or DApps deployments.

ID	Component	Functionality
FC.5.4	**SLASC**	The SLA-SC bridge links the SLA contractual terms with the blockchain system. The component describes the methods and functions that define the connection between SLAs and smart contracts in the project while it introduces the one-to-one representation of the SLA terms into smart contract specific contractual terms, being executable code that is triggered by external systems.
FC.5.5	**Smart Contract Manager**	This component is responsible for all the activities regarding smart contracts, from deployment to maintenance. The component is managing the various deployed smart contracts inside the blockchain network while its main goal is to holistically sustain the Pledger system of smart contracts inside the Pledger blockchain.
FC.5.6	**Transaction Handler**	This component administers and maintains the transaction flows of the blockchain network. The module subtly checks and confirms the transactions validations and endorsements on the background, while at the same time it offers an apparent role of a gatekeeper where errors or other malfunctions occur.
FC.5.7	**DLT**	Pledger blockchain network.

For this first release, the Blockchain constitutes an integrated solution for Pledger that serves the triggered smart contracts and deployed DApps. The prototype provides a permissioned blockchain for Pledger with secure and trusted nodes incorporating private environments for confidential transactions. In terms of functional components, the "Blockchain Reader/Writer" along with the "Transaction Handler" and the "Smart Contract Manager" are packaged and deployed on the DLT. For the next iteration, the "Blockchain Authenticator", "Wallet" and "SLASC" along with the enhanced "Smart Contract Manager" are planned to be delivered.

3 Technical description

3.1 Baseline technologies and dependencies

The baseline technologies that are used within the prototype are described in Table 2:

Table 2: Baseline technologies and dependencies

Name	Description	Version
Hyperledger Blockchain Framework & Middleware	Blockchain framework provided by Innov-Acts Ltd. that offers platform and tool integration with ease while leveraging from Hyperledger blockchain properties of private and confidential transactions, smart contracts and DApps.	latest
Docker [2]	Container engine for containerized apps running on the edge.	1.18+
Kubernetes [3]	Kubernetes is an open source orchestration software for deploying, scaling, and management of containerized applications. This tool is licensed under Apache License 2.0. Kubernetes is now managed by the Cloud Native Computing Foundation (CNCF).	Vanilla 1.19+
Golang [4]	Go is an open source programming language that makes it easy to build simple, reliable, and efficient software.	1.14+
Node.js [5]	Asynchronous event-driven JavaScript runtime that is designed to build scalable network applications.	12+

3.2 Components Architecture

The implemented components of the prototype, i.e. the "Blockchain Reader/Writer", the "Transaction Handler" and the "Smart Contract Manager" are packaged and installed on the DLT. In Figure 3, the architecture of the prototype is depicted.

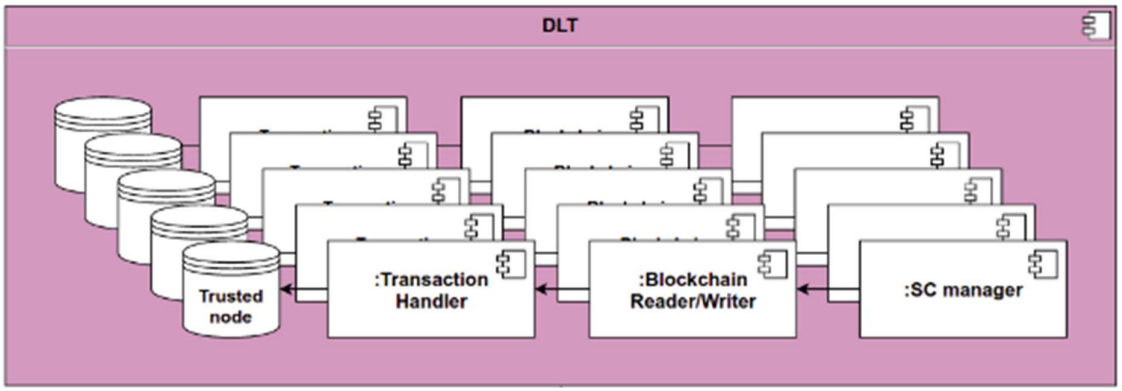

Figure 3: Pledger blockchain network high-level architecture

The DLT constitutes the Pledger blockchain network and hosts the environment and integration capabilities that are necessary for all the system operations. In this iteration of the prototype, the deployment procedure for the network and its hosted components is reformulated in order to be orchestrated in a single cluster. The dedicated mechanism for enabling smart contracts and DApps for Pledger focuses on the prototype interoperability with the other core components of the project and is mentioned in 3.2.2. Before analysing each of the aforementioned parts of the prototype, the main

security and privacy pillars as initially inherited by Hyperledger [6] are described in Table 3 for completeness.

Table 3: Main security and privacy pillars of Hyperledger

Property	Description
Transaction Immutability	Transaction immutability is the functional quality of the blockchain to remain unaltered and unchanged. The blockchain data is kept in sequential blocks forming a chain and each block uses a cryptographic principle or a hash value to keep the data consistent.
Nodes state agreement	In fault-tolerant distributed systems the participants need to agree on the same state. The nodes state agreement is the quality method that applied on all the nodes that participate in the network in order to decide finally.
Permissioned environment	A permissioned blockchain provides a supplementary access control layer that enhances the level of security and improves the role management over public blockchain systems.

3.2.1 DLT orchestrated

The Pledger DLT is an orchestrated blockchain framework. In order to adapt to the project's orchestration and container life-cycle, the prototype of the first iteration is adopting an orchestration nature for the network and each of the hosted components. The deployment of trusted nodes and DApps inside the DLT are respectively modified and re-organized in order to follow the orchestration character of the prototype. Table 4 describes the main phases of the orchestration procedure that Pledger DLT maintains.

Table 4: DLT orchestration main phases

Phase	Description
Migration	The necessary artifacts of the DLT are modified in order to adapt to the YAMLs specifications [7].
Integration	The blockchain framework is pre-configured in order to be deployable on a single cluster.
Deployment	Every DLT component is installed and deployed on Kubernetes.

3.2.2 Smart Contract Triggering

In order to intercommunicate securely with the Pledger DLT, a mechanism that can reach the inside environment of the blockchain is required, namely Smart Contract Triggering. The current version consists of the Smart Contract Deployment System (SCoDeSy) and the SLASC Bridge which are described in sections 3.2.2.1 and 3.2.2.2 respectively.

3.2.2.1 Smart Contract Deployment System

The Smart Contract Deployment System (SCoDeSy) constitutes the main mechanism that deploys the smart contracts on the orchestrated DLT. SCoDeSy is an automated procedure that is delivering smart contracts through a pre-defined life-cycle. The component is created specifically for the project and Table 5 describes the stages "Package", "Installation", "Approval" and "Submission" that are required to be completed sequentially during the procedure.

Table 5: SCoDeSy stages

Stage	Description
1. Package	The smart contract is packaged and delivered to the DLT nodes.
2. Installation	The packaged smart contract is installed on the respective nodes of the DLT.
3. Approval	Every participant of the DLT must approve the smart contract in order to be validated and ready for submission.
4. Submission	The smart contract is submitted on the blockchain and the included business intelligence is enabled on-chain.

The current status of SCoDeSy constitutes mainly from stages 1. Package and 2. Installation, while both next stages are scheduled for the next iteration of the prototype. In principle, the development of each stage completes on two major steps. The first one relates to the successful creation of the work as described in Table 5, and the second step consists of the stage orchestration, as described in DLT orchestration (Table 4).

3.2.2.2 SLASC Bridge

The SLASC Bridge links the SLA contractual terms with the blockchain system. This component describes the methods and functions that define the connection between SLAs and smart contracts while introduces the one-to-one representation of the SLA terms into smart contract specific contractual terms, namely packaged software executable at specific point in time.

The SLASC Bridge processes events received from the SLA app and executes the corresponding transactions on the blockchain while the possible integration via streaming events is studied and planned for the next iteration [8].

The SLASC Bridge intercommunicates with particular DApps that include the business intelligence of the interaction. Each event that triggers the specific DApp on-chain executes the respective smart contracts workflow followed by the management of the corresponding information on the ledger and returning the required response when necessary. In Figure 4, the high-level architecture of the SLASC component is depicted.

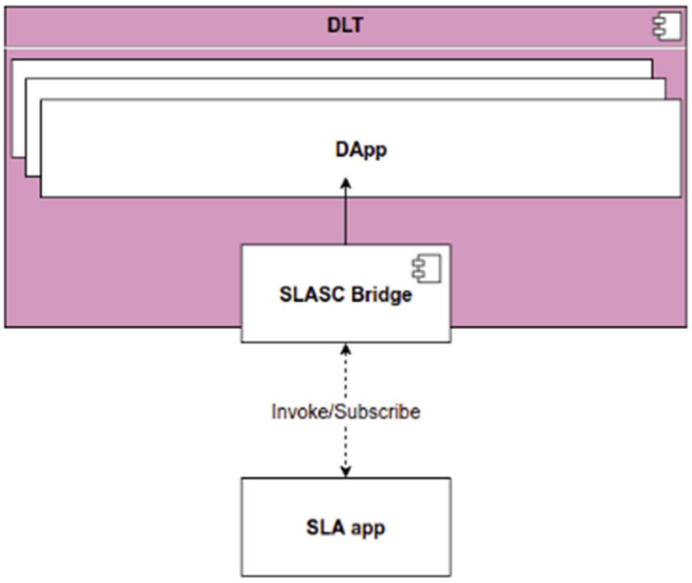

Figure 4: SLASC Bridge high-level architecture

3.3 Interfaces provided

This first version of the DLT endpoint capabilities is currently studying the necessary interoperability mechanisms that will ease the integration with the emitted events by the related subsystem functional components. An initial approach on the main endpoints structure to connect with the DLT will be provided mainly through the SCoDeSy and SLASC components (3.2). The defined endpoints will be set chiefly to leverage the asynchronous communication functionalities with the other Pledger components.

3.4 Data models

In this section the goal is to describe the respective data models and interactions in order to pave the way for the invocation of the next iteration smart contracts and UC DApps.

3.4.1 SLAs

The SLA data model and their functional states are described in Section 3.4 of the deliverable D3.3 "QoS and SLA assessment and negotiation tools I" [8].

3.4.2 Secure handshake

Head-Mounted Devices (HMDs) are limited in their processing and computation abilities and thus struggle to provide a smooth experience for the user when the application size (polygon amount) gets too high. To mitigate this, it has to either reduce the quality or reduce the frame rate and both situations cause the user to have a bad experience. The Pledger AR Workspace solves this by running the intensive calculations on a Server (Laptop, etc…) with a powerful GPU and displays the results on the Client (HMDs). For this peer-to-peer connection to be established between the Server and Client, an initial exchange of sensitive data needs to take place. To secure this handshake (signaling), the exchange will be secured through the DLT by sending the sensitive data using the Whisper protocol that will be enabled on the DLT nodes.

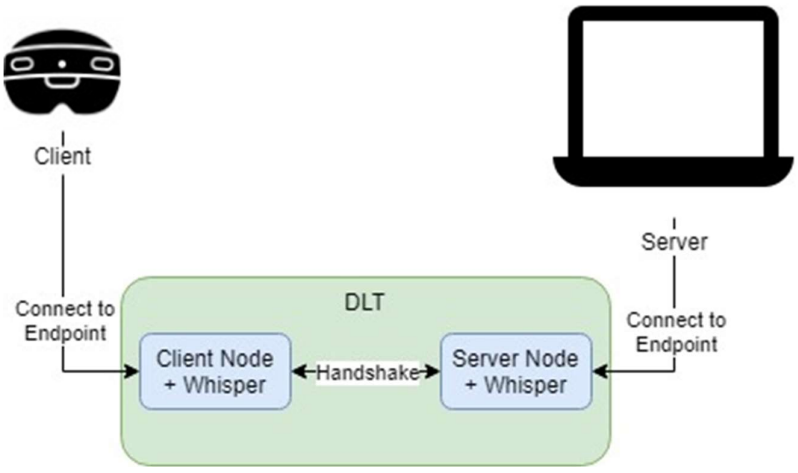

Figure 5: Whisper handshake in UC1

The handshake process has been successfully installed in a local blockchain with the Whisper protocol (Shh) activated. The next step would be to integrate this with the Pledger DLT nodes where an access point (Endpoint) allows to send messages securely between the Server (Laptop) and the Client (Hololens2). To further secure the process, a reverse proxy is planned to be set up on the corresponding endpoints in order to upgrade the protocol from HTTP to HTTPS.

Table 6: UC1 Handshake entries

Key	Value (type)	Description
Version (Server)	String	Signaling Version
SDP Offer (Server)	String	Negotiate the session's parameters (sent to client)
ICE Offers (Server)	String	Standard method of NAT traversal (sent to client)
SDP Offer (Client)	String	Negotiate the session's parameters (sent to server)
ICE Offers (Client)	String	Standard method of NAT traversal (sent to server)

3.4.3 Risk detection, notification and event log

The most relevant events in UC2 are those related to VRUs approaching a "danger zone" near a tram stop, and the arrival of trams to the same stop. Every time a VRU enters the transmission range of the RSUs, it starts sharing its position with the vehicular software stack to which applications can subscribe. Whenever a tram or bicycles are approaching the stop at the same time, the risk detection app will be triggered, raising an alert message that will be broadcast to all VRUs within the danger zone. The algorithm responsible for this detection of risks allows to define two areas of detection for the bicycles: the distant area (tens of meters before the tram stop) and the close area right next to the stop. By distinguishing these areas where bicycles can be located in, two levels of risk detection are defined: the early detected risk and detection and the imminent risk detection, respectively.

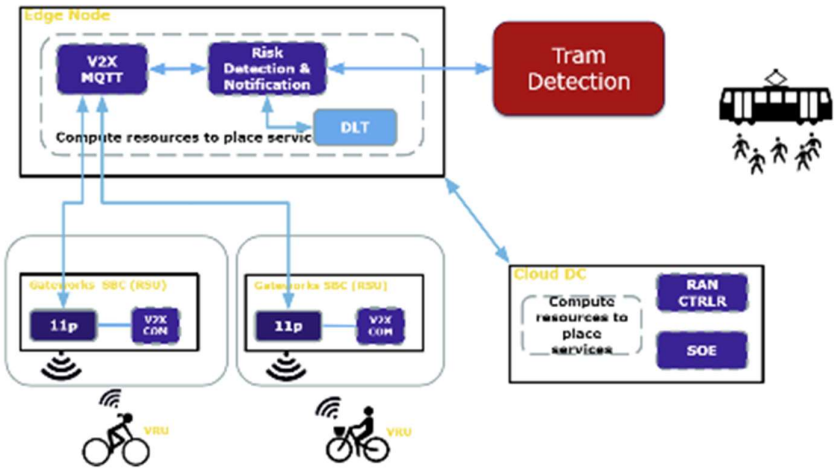

Figure 6: Block diagram of UC2

All these events (bicycle enters or leaves an area, tram arrives at or leaves the station, early or imminent risk detection) generate certain data that must be securely saved in an immutable way. Through the DLT the corresponding management of this kind of data can occur with focus on privacy. In fact, this information can include the actual event, but also metadata, such as the location of bicycles or the number of bicycles in the vicinity when an event happens. Since some of this data is sensitive, and in order to guarantee data privacy, relevant data is written on Pledger's DLT chain. Figure 6 represents the block diagram of UC2 and the interconnection needed between the DLT and the application-specific modules.

There are two distinct categories of data to be dumped on the DLT:

- Data about bicycle position. This information can be associated with one of the events mentioned previously and it can be dumped on the chain approximately every three minutes, and have a size of around 1MB.
- Alerts generated by the risk detection and notification system: these alerts are generated every time a risky situation is detected, and they are stored with additional metadata. These messages are dumped on the chain as required, and they have an approximate size of 100KB.

Table 7 summarises the essential set of data entries that should be written on the DLT chain:

Table 7: UC2 on-chain essential set of data entries

Log entry	Field 1 (type)	Field 2 (type)	Field 3 (type)
Tram enters station	Timestamp (string)	Count number of OBUs in the vicinity (integer)	Count number of pedestrians in the vicinity (integer)
Tram leaves station	Timestamp (string)	Count number of OBUs in the vicinity (integer)	Count number of pedestrians in the vicinity (integer)
VRU enters tram stop danger zone	Timestamp (string)	OBU ID (MAC address)	Type of VRU (string)
VRU leave tram stop danger zone	Timestamp (string)	OBU ID (int)	Type of VRU (string)
Risk detected	Timestamp (string)	OBU ID (int)	Type of VRU (string)
Collision imminent	Timestamp (string)	OBU ID (int)	Type of VRU (string)
Tram comes to a halt	Timestamp (string)	Count number of OBUs in the vicinity (integer)	Count number of pedestrians in the vicinity (integer)
External warning sign (VRU approaching tram stop danger zone)	Timestamp (string)	OBU ID (int)	Type of VRU (string)
External warning sign (VRU leaving tram stop danger zone)	Timestamp (string)	OBU ID (int)	Type of VRU (string)

3.4.4 Logging information about produced parts

In UC3, information to determine the process stability is gathered from the machine and analysed by analytical services, including average air consumption, average electrical consumption and parts produced. This information gathered about parts produced by the machine including the start and end time as well as information about sub-processes and quality is classified as sensitive. Especially the number of parts produced as well as the information about the quality and resulting number of rejects allows a conclusion to be drawn about the ongoing process as well as how the business is doing. As this kind of data should be confidential among the machine builder and the client, the corresponding secure environment where the exchange occurs privately is provided by the Pledger DLT. This kind of information will be stored on-chain in order to enable access only to authorized parties, e.g. FILL as machine builder and the according customer owning the process.

The analytical service of type Basic Analytics (Parts Produced) gathers relevant data from the message broker, analyses it and pushes the result back to the RabbitMQ, from where it should be stored on the chain. This process is illustrated in Figure 7.

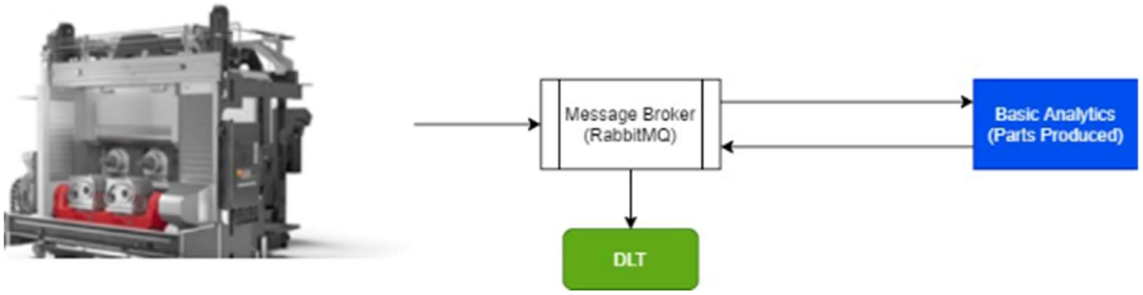

Figure 7: Parts produced data in UC3

The analytical services' result is one document object containing all relevant information about the produced part. This document constitutes one log entry on the blockchain with all information to uniquely identify a part. In Table 8, a summary of this information is given. The amount of subprocesses depends on the type of machine, for Pledger the machine has four subprocesses, where examples of two are given. These values are arrays to account for repeated subprocesses.

Table 8: UC3 on-chain data entries of one part

Key	Value (type)	Description
Start	DateTime	Start of part (Loading)
Stop	DateTime	Stop of part (Unloading
CycleTime	Double	Cycletime
TargetCycleTime	Double	Targetcycletime
Quality	Integer	Info whether quality of part is OK (1) or not OK (0)
ElectricityConsumption	Array of Doubles	Electricity consumed by the machine
Subprocess1Starts	Array of DateTimes	Starting dates for Subprocess 1
Subprocess1Stops	Array of DateTimes	Stopping dates for Subprocess1
Subprocess1Times	Array of Doubles	Durations of Subprocess1
Subprocess2Starts	Array of DateTimes	Starting dates for Subprocess2
Subprocess2Stops	Array of DateTimes	Stopping dates for Subprocess2
Subprocess2Times	Array of Doubles	Durations of Subprocess2
SpindleNumber	Integer	Number of the spindle which processed the part
NOK reason 1	Boolean	Reason 1 for part being not OK
NOK reason 2	Boolean	Reason 2 for part being not OK
Code	String	Unique identifier code of the part
Name	String	Type of the part (e.g. cylinder head)

4 Installation and usage guides

4.1 Requirements

In order to create, orchestrate and deploy the prototype, the following technologies are required:

- Docker 1.18+
- Bash 4+
- Kubernetes 1.19+

Additionally, tools like **jq** [9] are needed along with sophisticated configuration of the necessary manifests that are completing the migration to Kubernetes.

4.2 Installation

The prototype is deployed using the Pledger CI/CD platform by accessing the respective images from the dedicated private Docker registry.

The current version has been deployed on a single cluster and is planned to be hosted on multiple ones by the end of the project.

4.3 Usage

For the current iteration the network and the respective components are installed and available on the ENG Kubernetes cluster. In the next iteration, the usage of the defined DLT will be clarified through the deployment of the UC DApps.

4.4 Licenses

Apache License 2.0 [10]

4.5 Source code repository

The source code repository of the DLT is planned to be accessible online within the public git repository of Pledger.

5 Demonstration

5.1 Scenario description

Scenario 1:

Objective: Create and manage a permissioned blockchain dedicated for Pledger.

Pre-configuration:

1. Create initial network topology.
2. Initiate Validation Service and Membership Service Providers.
3. Migrate to Kubernetes.

Script:

1. Start the blockchain network.
2. Start blockchain visualisation.

Scenario 2:

Objective: Configure use case specific private environments.

Pre-configuration:

1. Create private submissions tool.
2. Migrate tool installation to Kubernetes.

Script:

1. Initiate private networks for the corresponding intercommunications.
2. Monitor networks activity through blockchain visualisation.

5.2 Validation and Verification

The first iteration of the Blockchain subsystem implements the following main functional components of the subsystem. They constitute the base blockchain environment of the Support Subsystem that create the dedicated Pledger DLT network capable of hosting the corresponding smart contracts and DApps.

- FC.5.2 Blockchain Reader/Writer
- FC.5.5 Smart Contract Manager (incubation)
- FC.5.6 Transaction Handler
- FC.5.7 DLT

The respective System Use Cases (SUC) that are defining the interrelation and interactions of the subsystem's internal components and functions with external types of actors are listed below for completeness (D2.3 Pledger Overall Architecture **Error! Reference source not found.**).

- SUC.27 Create SLA template
- SUC.28 Update/delete SLA template
- SUC.29 Monitor compliance with SLAs
- SUC.30 Get SLA violation notification
- SUC.31 Accept SLA
- SUC.32 Pay for services
- SUC.33 Request refund

In the next iteration, the parametric smart contracts and the hosted UC DApps will be served by the deployed blockchain ecosystem.

5.3 Demo

In this section, a demonstration showing the implemented functionalities on top of the orchestrated Pledger DLT is presented. Validation and Verification of the prototype solution is achieved by confirming the integration with as well as the usability by the core Pledger components while focusing on the performed output through the serving of the UC DApps.

The following figures depict the prototype functionalities and capabilities of the current iteration of the task as mentioned in the scenarios' descriptions (section 5.1).

Scenario 1:

Objective: Create and manage a permissioned blockchain dedicated for Pledger.

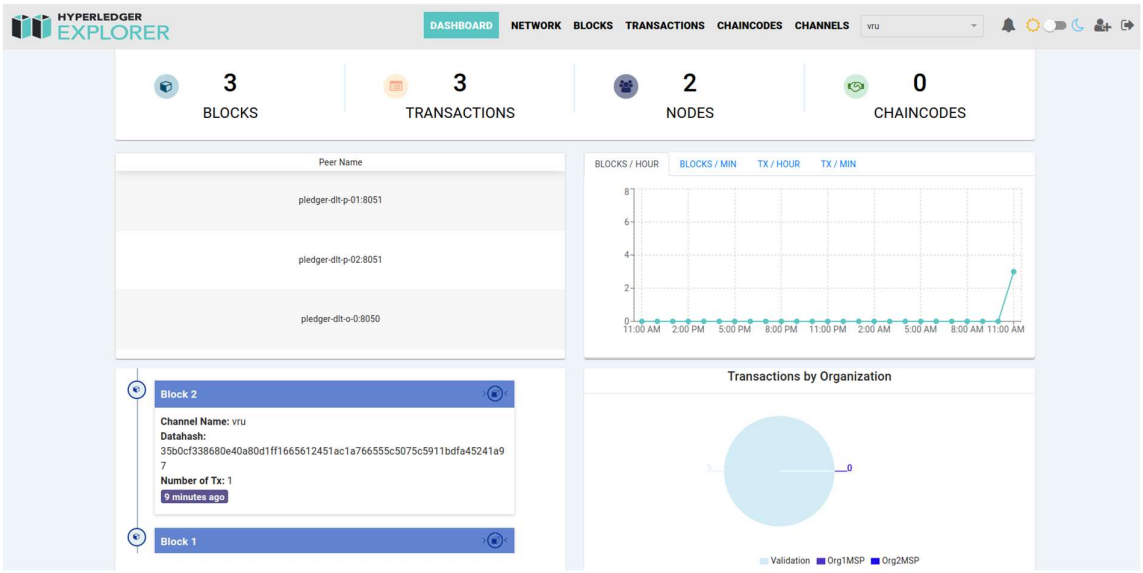

Figure 8: Orchestrated DLT initial deployment on Kubernetes: Pledger DLT

Scenario 2:

Objective: Configure use case specific private environments.

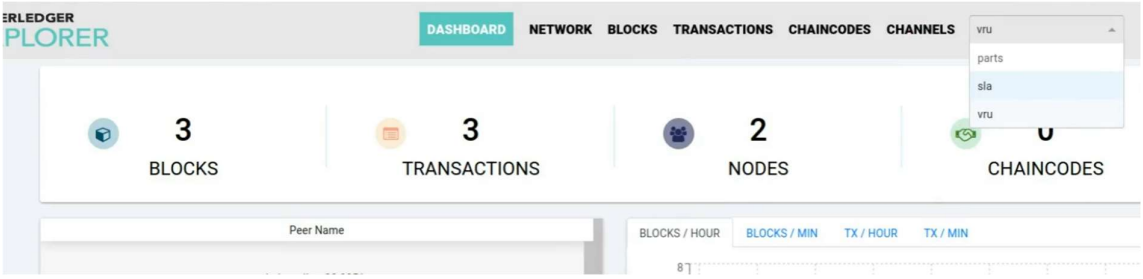

Figure 9: Pledger DLT private environments

6 Conclusions and next steps

This document accompanies the delivery of the first iteration of T4.2 "Smart Contracts and DApps on edge and cloud" and the work done for the period M6-M18. The output prototype comprises a secure and permissioned blockchain network, namely the Pledger DLT, with its underlying functionalities and components. The trusted nodes of the network supporting all the appropriate capabilities for transactions, smart contracts and DApps on this and the next iteration are offered together with the enabled privacy rules. This work includes an orchestrated blockchain platform specific to Pledger that serves the middleware architecture methods and the use cases. The Pledger DLT enhances the SLA on-chain applicability and constitutes an integrated environment that completes the SLA events workflow while the SLA integration is due to be complete by M24. The goal of the entire prototype is to support the requests and demands of the Pledger core components and subsystems that are involved in blockchain activities by M33.

7 References

[1] PLEDGER D2.3 - Pledger Overall Architecture v1.0. Voutyras Orfefs. 2020

[2] DOCKER, Docker home. https://www.docker.com, retrieved 14/07/2021.

[3] KUBERNETES, Kubernetes home. https://kubernetes.io/, retrieved 14/07/2021.

[4] GOLANG, Golang home. https://golang.org/, retrieved 14/07/2021.

[5] NODE.JS, Node.js home. https://nodejs.org, retrieved 14/07/2021.

[6] HYPERLEDGER, Hyperledger home. https://www.hyperledger.org/, retrieved 14/07/2021.

[7] YAML, YAML™ Specification Index. https://yaml.org/spec/, retrieved 14/07/2021.

[8] PLEDGER D3.3 QoS and SLA assessment and negotiation tools I. Antonio Castillo. 2021

[9] JQ home, https://stedolan.github.io/jq/, retrieved 14/07/2021.

[10] APACHE LICENSE, https://www.apache.org/licenses/LICENSE-2.0, retrieved 14/07/2021.