NGI ONTO CHAIN

Blockchain for the Next Generation Internet

D3.5 FRAMEWORK SPECIFICATION 2

02/11/2022

Grant Agreement No.: 957338
Call: H2020-ICT-2020-1

Topic: ICT-54-2020
Type of action: RIA

D3.5 FRAMEWORK SPECIFICATION 2

WORK PACKAGE	WP3
TASK	T3.3
DUE DATE	31/8/2022
SUBMISSION DATE	02/11/2022
DELIVERABLE LEAD	IEXEC
VERSION	1.0
AUTHORS	Souvik Sengupta (IEXEC) Anthony Simonet-Boulogne (IEXEC) Vlado Stankovski (UL) Alberto Ciaramella (IS) Marco Ciaramella (IS)
REVIEWERS	Petar Kochovski (UL) Thanasis Papaioannou (AUEB)
ABSTRACT	This deliverable provides updated specification of the ONTOCHAIN framework and architecture (compared to the initial architecture described in D3.3 and D3.4), of its components (including those developed by third parties during Open Call 1 and Open Call 2) and the specification of the ONTOCHAIN pilot deployment which will be used to evaluate the project results.
KEYWORDS	Decentralisation, blockchain, trustworthy content, data traceability, trustworthy knowledge exchange, privacy protection, web semantic, service interoperability

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Document Revision History

Version	Date	Description of change	List of contributor(s)
0.1	01/12/2021	Table of content & initial draft	Anthony Simonet-Boulogne
0.2	21/08/2022	v0.2 second version	Souvik Sengupta
0.3	23/08/2022	update OC1 short projects	Souvik Sengupta
0.4	25/08/2022	update OC1 long projects	Souvik Sengupta
0.5	26/08/2022	modify the introduction section	Anthony Simonet-Boulogne
0.6	31/08/2022	update the OC2 short projects	Souvik Sengupta
0.7	15/09/2022	update the OC2 long projects	Souvik Sengupta
0.8	10/10/2022	address the internal reviewers comments	Anthony Simonet-Boulogne
0.9	16/10/2022	update the section 4 for pilot network description	Souvik Sengupta
1.0	31/10/2022	final modification and address rest of the reviewers comments	Souvik Sengupta

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

DISCLAIMER

The information, documentation and figures available in this deliverable are written by the "Trusted, traceable and transparent ontological knowledge on blockchain ONTOCHAIN " project's consortium under EC grant agreement 957338, and do not necessarily reflect the views of the European Commission. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein. The information in this document is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability. Moreover, it is clearly stated that the ONTOCHAIN Consortium reserves the right to update, amend or modify any part, section or detail of the document at any point in time without prior information.

The ONTOCHAIN project is funded by the European Union's Horizon 2020 Research and Innovation programme under grant agreement no. 957338.

COPYRIGHT NOTICE

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

The ONTOCHAIN Consortium is the following:

Participant number	Participant organisation name	Short name	Country
1	EUROPEAN DYNAMICS LUXEMBOURG SA	ED	LU
2	UNIVERZA V LJUBLJANI	UL	SI
3	IEXEC BLOCKCHAIN TECH	IEXEC	FR
4	INTELLISEMANTIC SRL	IS	IT
5	ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS – RESEARCH CENTER	AUEB	EL
6	ELLINOGERMANIKO EMPORIKO & VIOMICHANIKO EPIMELITIRIO	GHCCI	EL
7	F6S NETWORK LIMITED	F6S	IE

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

EXECUTIVE SUMMARY

This document is deliverable "D3.5 Framework Specification 2" of the ONTOCHAIN project funded under the Horizon 2020 Research & Innovation program "ONTOCHAIN– Trusted, traceable and transparent ontological knowledge on blockchain".

The framework specification was produced by Task 3.3, after the execution of ONTOCHAIN Open Call #1 "Research" and Open Call #2 "Protocol Suite & Software Ecosystem Foundations"; it consolidates the results of the twenty projects funded and executed between March 2021 and September 2022 and integrates them within a single infrastructure design. Parts of this document will be provided to applicants as input to applicants of ONTOCHAIN Open Call #3 ("Applications & Experimentation").

The framework specification defines the software components that will compose the ONTOCHAIN ecosystem, the application programming interfaces (APIs) that will allow these individual components to communicate, and the user-facing interfaces. that the ecosystem will provide (graphical user interfaces and APIs).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

TABLE OF CONTENTS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

LIST OF FIGURES

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

LIST OF TABLES

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

ABBREVIATIONS

APIs	Application Programming Interfaces
DID	Decentralized IDentity
DLT	Distributed Ledger Technology
ERC	Ethereum Request for Comments
NGI	Next Generation Internet
OC	Open Call for participation
P2P	Peer-to-Peer
PKI	Public Key Infrastructure
SDK	Software Development Kit
EVM	Ethereum Virtual Machine

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

1 INTRODUCTION

This first version of the ONTOCHAIN framework specification is the combined result of the initial architecture designed before Open Call 1 (OC1), and of revisions made during the execution of the third party projects funded under OC1. This framework specification is thus the result of the collaboration between the consortium, the funded third parties and the Advisory Board. Then during the Open Call 2 (OC2), the OC2 participants developed protocol suite & software ecosystem foundations of the ONTOCHAIN project. The main objective for formulating this document is to provide common guidelines and recommendations to participants of OC3 regarding important design choices that go beyond their own project.

The rest of this deliverable is organized as follows:

- Chapter 2, provides the first version of the ONTOCHAIN architecture, as revised after the execution of OC1; the architecture defines the high-level services and components that will implement the ONTOCHAIN vision and describes the services that were implemented by third parties during OC1;
- Chapter 3, defines application programming interfaces for the ONTOCHAIN ecosystem, i.e. the APIs that will be presented to application developers and users of ONTOCHAIN services; these APIs also define primitives useful to developers of ONTOCHAIN (including participants funded through OC2), e.g. for integrating services and for implementing interoperability between services;
- Chapter 4 describes the planned and developed pilot deployment of the ONTOCHAIN ecosystem which has built on top of the iExec Ethereum sidechain and on top of geo-distributed hardware provided and hosted by the consortium members.
- Chapter 5 provides concluding remarks for this deliverable.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

2 ONTOCHAIN ARCHITECTURE

The ONTOCHAIN software ecosystem consists of a novel protocol suite grouped into highlevel application protocols, such as data provenance, reputation models, decentralised oracles, market mechanisms, ontology representation and management, privacy aware and secure data exchange, multi-source identity verification, value sharing and incentives and similar, and core protocols that include smart contracts, authorisation, certification, event gateways, identity management and identification, secure and privacy aware decentralised storage, data semantics and semantic linking.

2.1 ARCHITECTURE DESIGN

For enabling scalability, openness and high performance, the consortium employs a modular approach. The original layered approach introduced before OC1 has been revised to take into account the progress made during the execution of OC1 funded projects. The revised architecture is introduced earlier in D3.4 FRAMEWORK SPECIFICATION [1] and depicted in Figure 1; three modules are on the left side, and each builds on the functionalities offered by the lower layers:

- Applications;
- Ontologies;
- Distributed Ledgers.

On the right hand side are cross-layer modules, which mainly focus on interfaces and interoperability. The colour scheme shows at which stage of the project each module is developed:

- OC1 focused on building the foundations for the ONTOCHAIN ecosystem;
- OC2 integrated the results of OC1 into a platform comprising building blocks for developing ONTOCHAIN applications;
- OC3 will develop said applications.

At the top layer lie different next-generation application solutions, such as trustworthy web and social media, trustworthy crowd-sensing, trustworthy service orchestration, decentralised online social networks, which tackle today's Internet problems that can be built upon the use cases in Trustworthy Information Exchange and Trustworthy and Transactional Content Handling. Each of the use cases is built upon combined functionalities from the Application Protocols layer, such as Data Provenance, Reputation Models, Decentralised Oracles, etc. The modules at the Application Protocols

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

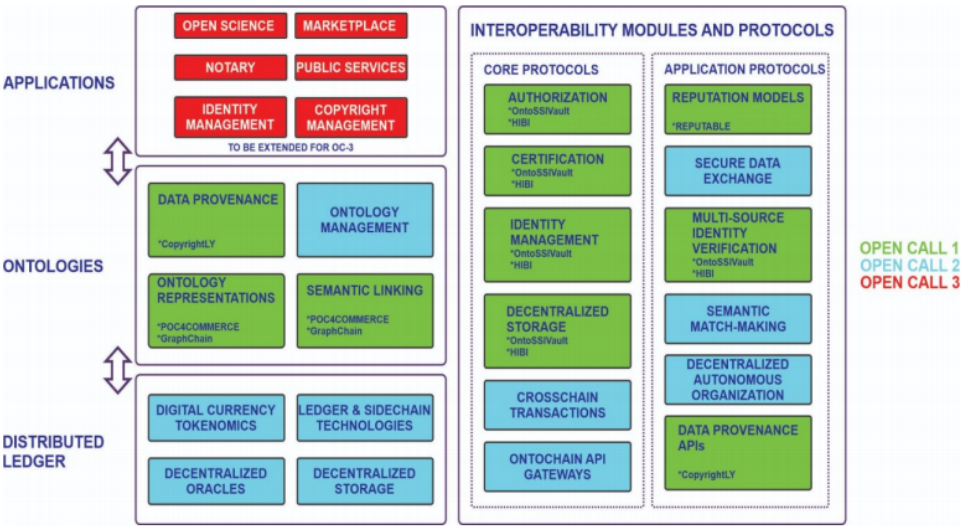

FIGURE 1: ONTOCHAIN ARCHITECTURE AND COMPONENTS DIAGRAM AS REVISED AFTER OPEN CALL 1.

layer themselves are built upon core Blockchain-based services at the Core Protocols layer, such as Smart Contracts, Identity Management, Secure and Privacy-Aware Decentralised Storage, Certification, Authorisation and Data Semantics. The Core Protocols modules employ low-level Distributed-ledger functionality, i.e. Digital Currency, Distributed Storage interfaces and cross-chain functionalities, which lie on combined proprietary, corporate and public resources.

The *Distributed ledger* module will provide a distributed and decentralized execution environment for the whole ecosystem; the *Ontologies* module will provide novel, trusted services for managing web ontologies in a fully trusted and secure fashion; the *Applications* module will be the focus of OC3, although OC1 already produced valuable pilot use-cases. The *Interoperability modules and protocols* has multiple functions; first, it will act as the backbone for interconnecting the other modules; second, it provided the ONTOCHAIN Gateway API, i.e. the main entry point for application developers; last, it will provide essential services and building blocks to all of the modules and to the applications, e.g. decentralised storage, identity management and data certification.

2.1.1 Distributed Ledgers

The ONTOCHAIN software ecosystem will rely on several interconnected Blockchains in order to ensure performance and scalability metrics that satisfy a business context.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

The first benefit of this multi-chain environment is to better scale applications by hosting the transactions and data from different business domains in different chains. An additional benefit is to isolate applications that may need it, in order to preserve the confidentiality of transactions when necessary (e.g. in the context of private industrial consortium). ONTOCHAIN will provide native interoperability solutions for exchanging information and assets between the different chain (see 2.1.4).

The main ONTOCHAIN Blockchain will be based on Ethereum[1], which is currently the industry standard for Blockchain-based decentralised applications[2]. Ethereum also features several initiatives to support cross chains operations (e.g. Polkadot[3], Polygon[4]) that ONTOCHAIN can monitor and eventually integrate for this purpose. The initial pilot deployment will rely on the iExec Bellecour network, which is an Ethereum sidechain based on the POA Network[5] stack (see Section 4).

In addition to Blockchains based on Ethereum and on the Ethereum Virtual Machine, ONTOCHAIN necessarily offer some features that are not compatible with the EVM; At this time, two services are in this situation, GraphChain which is entirely compatible with Ethereum smart contracts but relies on a modified client (see 2.2.1) and Gimly ID, which is built on top of the Oasis Protocol[6] (see 2.2.4).

- *Digital currency tokenomics* Beyond the current hype revolving around Bitcoin, Ethereum and over 5,000 altcoins, the potential for social change of what is now being called "Blockchain 2.0" is appearing more and more clearly. For example, cryptocurrencies are praised for allowing cheap and fast money transfer to the 1.7 billion people who are excluded from the banking system around the world, or as a stable alternative to devalued fiat currencies. One very interesting aspect for the Next Generation Internet is the possibility of programming complex self-executing transactions in Smart Contracts. Integrated with ONTOCHAIN's provenance and reputation mechanisms, a crypto token will guarantee a fair compensation to every contributor who participates in the ecosystem.
- *Sidechains & Layer-2 technologies* Blockchain offers a plethora of features, such as traceability, transparency, pseudo-anonymity, democracy, automation, decentralization and security. Despite these promising features the technical scalability of the network is still a key barrier which can put a strain on the adoption process, especially for real business environments. The complexity and the specialization of all the different real-world ONTOCHAIN applications will lead practitioners to use multiple ledger technologies for implementing different solutions. This will enable higher performance and scalability, while enabling different business logics, access meth-

[1] https://ethereum.org/fr/
[2] https://entethalliance.org/
[3] https://polkadot.network/
[4] https://polygon.technology/
[5] https://www.poa.network/
[6] https://oasisprotocol.org/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

ods and governance models that require specific chains. This component will build the main ONTOCHAIN and associated clients based on Ethereum, which is a stable and well tested system for data transactions and has a cost-effective network for the operation of its applications.

- *Decentralized oracles* By essence, smart contracts must run entirely isolated and cannot access data from the external world on their own. Oracles are software components which primary purpose is to collect external data and input it to smart contracts. Because oracles bring arbitrary data to the Blockchain they create a major vulnerability, and the trustworthiness of the imported data is extremely difficult for them to assess. This component will provide trustful and trustless oracle prototypes that are capable of interacting with the ONTOCHAIN infrastructure and providing necessary data for the operation of its applications.
- *Decentralized storage* Various decentralised repositories, such as Peer-to-Peer and Content Distribution Networks have existed for decades. Lately, the ENTICE[7] project developed a series of repositories for the storage of Virtual Machine and container images, including their fragments and it was optimised for delivery upon request. The interfaces used are S3 like[8]. In recent years, with the emergence of Blockchain, we have witnessed a new wave of participatory storage repositories that can help address the security and privacy needs, and may help store practically any kind of data, for example, services such as STORJ[9] and Filecoin [10]. This component will integrate new storage services that will help store private data in encrypted and decentralised ways, manage data replicas for reliability and Quality of Service, while balancing the trade-offs with the storage costs.

2.1.2 Ontologies

Specific domain ontologies have been developed or reused by ONTOCHAIN projects.

Of these, the project PROF4COMMERCE was specifically focused to develop ontologies, in this case for supporting e-commerce applications. More specifically, this project developed 3 domain ontologies, as extensions of well know ontologies, and more specifically:

- OC-Found, building on the ontology OASIS, for supporting the semantic description of the stakeholders of the ONTOCHAIN ecosystem, including supply chain of resources, digital identities of agents, quality valuation processes;

[7] http://www.entice-project.eu/
[8] https://aws.amazon.com/fr/s3/
[9] https://www.storj.io/
[10] https://filecoin.io/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

- OC-Commerce, building on the ontology GoodRelations, for supporting the semantic description of offering, auctions and commercial activities;
- OC-Ethereum, building on the ontology BLONDiE, for supporting the semantic description of Ethereum Blockchain, smart contracts and tokens.

The project CopyrigthLY reused the Copyright Ontology, which is a background of the group deliverying this project. This ontology, available on https://rhizomik.net/ontologies/copyrightonto. Models the fundamental building blocks of the copyright domain, including rights, creations, and actions, are also available there.

The project KnowledgeX developed an ontology for matching data science problems with skills of data scientist.

The project GraphChain developed We have developed the Graph metadata ontology, a lightweight ontology that can be used to represent and interchange provenance information and other metadata generated inside the Graphchain ecosystem. The ontology is mainly intended for describing data in the default graph.

2.1.3 Applications

Applications will be the focus of ONTOCHAIN Open Call 3, which is already kick-off to open on the month of October, 2022. At this time ONTOCHAIN features a few complete applications, (e.g., KnowledgeX, BOWLER, CARECHAIN, PXC, etc.) have been developed.

2.1.4 Interoperability modules and protocols

Core protocols

- *Authorizations* Blockchain has stimulated the idea of self-sovereign digital identity, and few commercial services have already emerged[11]. Various Role-Based Access Control (RBAC) systems have also existed in the course of the past decades. With ONTOCHAIN one could easily see systems where a patient is self-identified on the Blockchain, while a medical doctor gains access to the medical records based on their role (e.g. surgeon, general practitioner).
- *Certification* Certification refers to the confirmation of certain characteristics of an object, person, or organization. For example, a government may decide to offer certificates to cloud providers that have verified GDPR-compliant handling of pri-

[11] https://www.ibm.com/blogs/Blockchain/category/trusted-identity/self-sovereign-identity/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

vate citizens' data. In such cases, certificates can be issued on-chain, and can be used as conditions for performing specific transactions, for example, using AI methods to analyse private data. The specific conditions can be implemented within a Smart Contract to govern the GDPR-handling of private citizens' data only on certified cloud providers.

- *Identity management* Self-sovereign Digital Identity is a specific area of research that is overreaching to be addressed solely by the present project. Nevertheless, ONTOCHAIN technologies and solutions can be used to address parts of the digital identity puzzle. There are two conflicting requirements that drive this development. First is the ability to identify oneself in specific interactions, such as withdrawing money in a bank, and another is to still preserve one's privacy, for example of the health data or the web browsing or buyer's habits. This is a feasible endeavour. However, it is necessary to invest more in technologies like ONTOCHAIN to make it happen.

- *Decentralized storage* Various decentralised repositories, such as Peer-to-Peer and Content Distribution Networks have existed for decades. In recent years, with the emergence of Blockchain, we have witnessed a new wave of participatory storage repositories that can help address the security and privacy needs, and may help store practically any kind of data in virtualized block devices. In the near future, one could imagine new storage services that can help store private data in encrypted and decentralised ways, that can help manage data replicas for reliability and Quality of Service, while balancing the trade-offs with the storage costs.

- *Crosschain transactions* A Crosschain transaction module facilitates the interoperability between two relatively independent Blockchains. In other words, this module will enable the functionalities for the Blockchains to speak to one another Blockchain. Crosschain implementation is mainly represented by asset swap and asset transfer, which is both an essential part of the Blockchain world. With the help of this module, the limitations of a single chain can be avoided.

- *ONTOCHAIN API Gateway* This module will support connections between the ONTOCHAIN Blockchain and the outside world, including other Blockchains. Part of its duty will be to help programmers in the upper layers make trade-offs about how much data is stored on-chain, by supporting pointers to offchain decentralized storage, such as IPFS. The module will provide several low-level Application Programming Interfaces (APIs) in the form of Smart Contracts, as well as several higher-level wrappers for at least three programming languages that are commonly used by developers e.g. JavaScript, Java and Python. The interfaces will be generic and extensible in order to allow connections with different ledger technologies in the future, while only external Ethereum-based chains will be supported during the course of the project because of its important community of adopters and developers and because its ecosystem already contains most of the software components that sub-

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

projects will require (e.g. off-chain Computing, Decentralized Oracles). However, sub- grantees will be discouraged from producing designs relying on concepts and optimisations that are specific to any particular Blockchain.

Application protocols

- *Reputation models* This module will provide the functionality of building different decentralized reputation models over the Blockchain infrastructure. The basic building blocks of a reputation system are an approach for casting assessments/votes for a particular subject (person/data/fact), an approach for recording the history of votes per subject and an approach for summarizing votes into a single reputation metric per subject. One important problem with reputation systems is weak identities, referred to as "cheap pseudonyms", through which multiple attacks can be employed, such as ballot stuffing, bad naming, negative discrimination, sybil attacks, reputation whitewashing, reputation milking and more. Existing solutions include reputation cold start (which introduces a new set of problems) and making pseudonym change more costly. Emerging decentralized reputation mechanisms built upon the Blockchain will enable stronger user identities without sacrificing anonymity. Different reputation models can be defined to assess different aspects, such as data source trustworthiness, data credibility, service trustworthiness, etc. This module is built upon Identity Verification mechanisms.

- *Secure Data Exchange* Secure data exchange comprises the functionality of exchanging data among distributed parties, while verifying the ownership of the data and access rights, authenticity of transacted parties, the integrity of the data exchanged and the confidentiality of the data through Blockchain underlying mechanisms. Most often, off-chain data will be exchanged in data transactions, while on-chain data will store public cryptographic keys and access control lists based on which elevated data access to different portions of data is authorized for specific transacted parties.

- *Multi-source Identity Verification* This module seeks to register and verify individual digital identities of physical objects (in addition, abstract objects, physical and abstract persons) via trusted data from multiple sources. Attributes represent a fundamental part of any ontological concept. In particular, various AI methods can be introduced to operate on sensing data (IoT based, sensors, cameras and similar) so that an assertion can be made whether an individual belongs to a specific ontological concept (e.g. car, chair, container image, the person Elisabeth, and similar).

- *Semantic Match-Making* This module will find correlations between any goods or services that can be described with an ontological representation. Semantically related entities of ontologies can be used for different tasks such as ontology merging, query answering, data translation, etc. Ontology matching is a requirement for finding compatible offer and demand (buy and sell orders) in semantic-based marketplaces. To guarantee the fairness of the transactions in the marketplace, the match-

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

ing process should be fair to every party, e.g. preventing exclusion, censorship, price manipulation and fraud.

- *Decentralized Autonomous Organization* Based on the encoding rules, the decentralized autonomous organization (DAO) module can run a Blockchain protocol entirely and autonomously with the help of smart contracts. Thus, this module circumvents the need for human intervention or centralized coordination and helps to build a trustless system.
- *Data Provenance APIs* This module will provide application programming interfaces for querying and presenting provenance information from ONTOCHAIN about on-Chain and off-Chain data (pointers to data stored outside of ONTOCHAIN). Provenance information will include the complete trail of transactions that resulted in a record, including links to the programs that were run (e.g. address of smart contracts, signature of AI models when available), to the input data that was processed and to the contributors who ran the programs or provided original information.

2.2 COMPONENTS DESCRIPTION

The system architecture described in the previous section will be realised by integrating software components developed by successful applicants to the ONTOCHAIN open calls for contribution. The following seven projects have already been selected and developed by third parties under ONTOCHAIN OC1 and will be integrated into the final ONTOCHAIN ecosystem; this section briefly introduces each of these components and describes the interfaces they will provide for integrating with the other ONTOCHAIN components.

In particular, these interfaces should be read as specifications for Topic A1 "Service Integration (Gateway APIs) for ONTOCHAIN applications" of ONTOCHAIN OC3.

2.2.1 GraphChain: a framework for on-chain data management for ONTOCHAIN

Description GraphChain is a framework for on-chain data management for ONTOCHAIN which implements decentralised On-chain graph management technologies, including the ability to perform usual graph operations. GraphChain proposes a radically different approach: instead of encapsulating the semantic data into Blockchain blocks, they propose to design and implement the Blockchain mechanisms on top of semantic data. The GraphChain solution provides different functionalities such as:

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

- Hashing of subgraphs for the on-chain graph structures;
- Procedural smart contracts with access to the on-chain semantic data;
- Identification, authorisation and data provenance for the on-chain data;
- Sharding mechanisms and strategies.

The whole idea of GraphChain is adding a new level of trust without sacrificing availability, query ability and performance of graph databases so the solution can be integrated in any software ecosystem that uses traditional LPG databases.

Application Programming Interfaces In the GraphChain ecosystem, the core element is an Ontonode, which is a single OntoSidechain node. The general idea of Ontonode operation is similar to every Blockchain network. The Blockchain nodes process transactions and achieve consensus over data that represent them. Ontonode is based on the combination of Ethereum and RDF-star triplestores. The modification strategy for the Ethereum client node has been thoroughly presented in [2].

Module *Ontopod* is one of the most important sub-elements of Ontonode. It is an RDF-star compliant graph database that stores all named graphs protected and distributed by Blockchain network. The services offer by the Ontopod are exposed by the REST APIs. Below the detailed descriptions of each REST API calls has been presented:

POST `/ontonode/deploy?contract=type` Deploy the contracts which are essential for proper work of Ontonode. Takes the type of contract as an argument. If the operation is successful then it returns the value "OK" and HTTP 200. Otherwise it will returns "error" and HTTP code 400.

GET `/ontonode/info?size=graph_count` Get metadata about the graphs which are uploaded to Graphchain. This call takes the number of graph as the argument and returns the JSON with graph metadata.

GET `/nodes/get?size=count` Get the information about Ontonodes. Takes the number of desired Ontonodes as the argument and returns HTTP code 200 in case of success. Otherwise, it returns HTTP 400 for erroneous call or HTTP 404 if the information of the Ontonode is not found.

POST `/nodes/adding?nodeId=id` Add a new Ontonode. The function takes the node ID as the argument and returns HTTP code 200 if successful. Otherwise, it will return HTTP code 400 for erroneous call or HTTP 404 if the information of the Ontonode is not found.

POST `/nodes/update?nodeId=id` Modify the Ontonode. Similarly to the previous function, it takes a node ID as the argument and returns HTTP code 200 for success.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Otherwise, it returns HTTP 400 for erroneous call or HTTP 404 if the information of the Ontonode is not found.

Module *Ontoshell* is another important sub-element of Ontonode. It is a set of end-points and interfaces. It is a crucial component because all Blockchain interactions, which are not internal, work on this layer. The most standard way of interaction with Ontonode is through REST API (Graph update, retrieval, storing and deleting).

PUT `/rdf-graph-store?graph=graph_uri&user=user_addr`
POST `/rdf-graph-store?graph=graph_uri&user=user_addr` Both calls upload RDF graph from the request body to GraphChain. These calls take the URI of a graph and the Ethereum address of a user as arguments. Successful execution of these calls return "OK" and HTTP code 200. Otherwise, it returns HTTP code 400 for erroneous call or HTTP 415 for wrong serialization.

GET `/rdf-graph-store?graph=graph_uri` Return the latest version of a RDF graph. The function takes the URI of the target graph as the argument. For unsuccessful search operation, the function returns HTTP error code 404.

DELETE `/rdf-graph-store?graph=graph_uri` The delete operation does not remove anything, instead it adds an empty graph to the Triplestore and points to it as the latest version. The argument is the URI of the graph to remove and returns HTTP code 200 for successful call. Otherwise, it returns HTTP code 400 for erroneous call or HTTP 404 if the information of the graph is not found.

HEAD `/rdf-graph-store?graph=graph_uri` Same as GET but returns an empty response body. Similar to ASK SPARQL query. Takes the URI of the retrieved graph as the argument and returns HTTP code 200 for successful call. Otherwise, returns HTTP 400 for erroneous call or HTTP 404 if the information of the graph is not found.

2.2.2 CopyrightLY: Decentralised Copyright Management for Social Media

Description CopyrightLY is an application that helps managing ownership and rights in the ONTOCHAIN ecosystem. From claiming authorship of content or data, to linking these claims to evidence off-chain or associating reuse terms that once agreed set the reuse conditions among the involved parties.

CopyrightLY proposes a system capable of rooting on-chain copyright transactions, especially NFTs, on copyright claims that can by tied to evidence and validated on court. These set of evidence, together with the opportunity to make complaints and use incentives to curate them, makes it possible to build a scalable and community driven

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

content ownership layer.

The central part of the CopyrightLY's architecture is on-chain and based on a set of smart contracts, which take care of registering Manifestations (i.e. authorship claims), Complaints (to denounce authorship claims potentially fraudulent) and Reuses (used to attach reuse terms to a manifestation, including the initial offer, the negotiation steps and the final reuse agreement, if reached).

Application Programming Interfaces To facilitate interoperability and dealing with a complex domain as copyright management, CopyrightLY is based on the use of semantic technologies and the formal conceptualisations provided by the Copyright Ontology.

CopyrightLY's functionality is made available through different mechanisms. If the aim is to read the state regarding authorship claims (manifestations), evidence accumulated or existing complaints, it is available mainly through a GraphQL API provided using The Graph:

- Manifestation (authorship claim) details;
- Manifestations (authorship claims) list;
- Upload Evidence details;
- Upload Evidence list.

On the other hand, it is also possible to access and modify CopyrightLY's state as stored in the Blockchain, interacting directly with the corresponding smart contracts. Notably, the services will be exposed by the REST API calls. Detailed of those calls are presented below:

POST `/api.studio.thegraph.com/query/1303/copyrightly/0.0.9` Is sending a GraphQL query to the endpoint. Taking GraphQL query as the argument and returning the result in JSON.

More details can be found in [3].

2.2.3 HIBI: Human Identity Blockchain Initiative

Description HIBI encompasses scalable Blockchain, decentralised legal reputation and identity systems and interoperable semantic web technologies. HIBI is provided to developers as a modular SDK for adding specific features to an application. All of the features are based on the eIDAS standard for qualified electronic signatures and require the NFC scans of a legal EU identification document. With HIBI, developers

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

can:

- Support eID-based authentication in their applications;
- Let users sign transactions and documents using their physical eID card;
- Link the legal identity of a user to a Blockchain-based public key;
- Perform key backup and recovery using a legal ID document.

HIBI provides the user with the power and sovereignty of their keys, identifiers, and verifiable credentials. Therefore, it contradicts existing identity management systems that are usually based on centralised data silos managed by identity providers.

Application Programming Interfaces The SDK allows the client application to communicate with the HIBI services over remote API calls. For instance, the APIs let a user request their secrets to be backed up, or the DID on which some credential should get issued. The two SDK modules are *EVERKEY* and *EVERID*; the mothodes they provide to client applications for consuming HIBI services are listed hereafter.

Module *EVERKEY* Decentralized, Non-custodial Backup and Recovery Mechanism that is using eIDAS-compliant eIDs that can provide a pseudonym.

`store(key, userIdentifier, keyIdentifier)` Get a secret backed up using the HIBI SDK. There is no need to expose further APIs, as the rest of the system is provided by the eID infrastructure where the HIBI SDK itself receives data autonomously and the decentralized storage solution to post secrets.

`recover(userIdentifier, keyIdentifier)` Get a secret that came from the decentralized storage grid and was reassembled within the HIBI SDK.

`retrieveIdentifier()` Perform the eID Authentication. The user needs to scan their ID with an NFC-enabled device. After entering the correct PIN, the eID is unlocked and can be used for EVERKEY and EVERID.

Module *EVERID* With this REST API, it is possible to call the EverID service. The service allows for deriving a Verifiable Credential and get it issued, as well as get the Qualified Electronic Signature (QES) from the issuer, and verify the credential.

POST `/everkey.id/v1/getVC` Return a Verifiable Credential that includes the provided identity attributes. The argument is the name of the user and returning verifiable Credential in JSON-LD form.

POST `/everkey.id/v1/verifyVC` Verify the Verifiable Credential. Takes the verifiable credential as argument and returns a boolean.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

GET `/everkey.id/v1/cert` Obtain the X.509 Issuer certificate to verify the VC of the user.

GET `/everkey.id/v1/getTcTokenURL` Obtain the endpoint to perform the eID authentication.

More details can be found in [4].

2.2.4 Gimly ID: an SSI application suite

Description Gimly ID is a fully self-sovereign identity solution that brings trust and usability to users without compromising security and privacy of the ecosystem and its members. Gimly ID centers around the mobile application, which offers a passwordless single-sign on experience and selective disclosure of data by leveraging decentralised identifiers (DIDs) and Verifiable Credentials (VCs) and a sovereign data vault. The Gimly ID mobile app can be used as a standalone solution, aiming for full interoperability with other SSI conformant solutions.

Application Programming Interfaces Gimly ID also includes a range of web portals and developer tools. These allow for advanced data management, simplified deployment of the SSO and SSI functionalities in existing systems, the creation and management of DIDs and VCs, and the governance and analytics of deployed ecosystems.

The complete software suite consists of the following React native mobile application and of several web applications:

1. Gimly ID mobile App: authentication and SSI data vault. Used for password-less login, management of credentials and sovereign data, managing data access permissions.
2. SSO login button: sed to enable SSO with the Gimly ID app into web applications and web portals.
3. Sovereign MyData web portal: advanced sovereign data management.
4. Developers web portal: allows developers, organisations, companies to implement the password-less SSO flow with the Gimly ID app.
5. Organisations identity admin portal: managing DIDs and VCs, including the creation, issuing and revocation of credentials.
6. Ecosystem governance portal: allows for defining policies and rules for identities, verifications.

More precisely, three new module has been introduced in addition with the previous

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

modules. In this section we are going to describe the API calls for those three new modules.

Module *DID Auth SIOP* SIOP v2 is an extension of OpenID Connect to allow End-users to act as OpenID Providers (OPs) themselves. Using Self-Issued OPs, End-users can authenticate themselves and present claims directly to the Relying Parties (RPs), typically a webapp, without relying on a third-party Identity Provider. This makes the solution fully self sovereign, as it does not rely on any third parties and strictly happens peer 2 peer, but still uses the OpenID Connect protocol. Below, all the methods related to this module has been thoroughly discussed.

`createURI` Create a signed URL encoded URI with a signed SIOP authentication request. Taking `SIOP.AuthenticationRequestOpts` as argument and returning promise `<SIOP.AuthenticationRequestURI>`.

`verifyJWT` Verify a SIOP Authentication Request JWT. Throws an error if the verification fails. Returns the verified JWT and metadata if the verification succeeds.

`createJWTFromRequestJWT` Creates an Authentication Response object from the OP side, using the Authentication Request of the RP and its verification as input together with settings from the OP. The Authentication Response contains the ID token as well as optional Verifiable Presentations conforming to the Submission Requirements sent by the RP.

`verifyJWT` Verifies the OPs Authentication Response JWT on the RP side as received from the OP/client. Throws an error if the token is invalid, otherwise returns the Verified JWT.

`resolver.resolve` Resolves the DID to a DID document using the DID method provided in didUrl and using DIFs did-resolver and Sphereons Universal registrar and resolver client.

`getResolver` The DidResolution file exposes 2 functions that help with the resolution as well.

`resolveDidDocument` The DidResolution file exposes 2 functions that help with the resolution as well.

`createDidJWT` Creates a signed JWT given a DID which becomes the issuer, a signer function, and a payload over which the signature is created.

`verifyDidJWT` Verifies the given JWT. If the JWT is valid, the promise returns an object including the JWT, the payload of the JWT, and the DID Document of the issuer of the JWT, using the resolver mentioned earlier. The checks performed include general JWT decoding, DID resolution and Proof purposes. Proof purposes allows restriction of verification methods to the ones specifically listed, otherwise the 'authentication' verification method of the resolved DID document will be used.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Module *Gimly ID Mobile SDK* For the most up-to-date information, it is recommended to read the documentation: https://github.com/Gimly-Blockchain/gimly-id-app-sdk.

`createEncryptedWallet` This API creates a wallet. Taking pin as the argument and returning a string value.

`createDid` It creates a DID and taking identityData and Authority as arguments and returning IdentityData.

`getUserDid` This call returning the value of IdentityData and resolves a DID.

`updateDID` It updates the DID document by taking arguments as IdentityData, permission info, parental info of the corresponding DID and Authority related info.

`createCredential` Create a credential.

`verifyCredential` Verify the credential.

`createPresentation` Create a presentation.

`signPresentation` Sign a presentation.

`verifyPresentation` Verify a presentation.

`verifyUnsignedPresentation` Verify an unsigned presentation.

Module *Universal DID resolver/registrar client* This is another module related with the SDK. For more details it is recommended to read the online documentation: https://github.com/Sphereon-Opensource/did-uni-client.

`Registrar.create` Create a DID and obtain its JSON representation as a string.

`Registrar.update` Update a DID and obtain its JSON representation as a string.

`Registrar.deactivate` Deactivate a DID and obtain its JSON representation as a string.

`Resolver.resolve` Resolve a DID document.

Notably, for this project no services have been exposed through REST API calls as it is aimed at Peer-to-Peer communication. More details are provided in [5].

2.2.5 Reputable: a Provenance-aware Decentralised Reputation System for Blockchain-based Ecosystems

Description Reputable delivers a cross-platform privacy-aware reputation system which leverages Blockchain technology to achieve decentralised, verifiable calculation

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

of reputation scores. Further it enables interaction with end users and systems through a secure, reputation analytics dashboard to facilitate user verification as seamless integration with other systems and services.

Within Reputable, the reputation data consists of two different types. Firstly, it is the individual user feedback i.e. the feedback provided by the users when contacted to share their experiences with a service/seller/marketplace. Secondly, it is the aggregate reputation score which is calculated using the individual user feedback. As these two types of data are linked with each other, the linkage is preserved and utilised it to achieve verifiable reputation scores.

Application Programming Interfaces Reputable provides three different interfaces:

- Query reputation score for a seller
- Query historical reputation score for a seller
- Verify reputation calculation for a user

As the reputation modelling system is expected to interact with external services and users in an off-chain arrangement, Reputable envisions leveraging decentralised oracles to integrate reputation data with the on-chain provenance mechanism. Notably, the developer did not build any APIs for the SDK module. They have exposed their services through REST APIs. An example of these services has presented below. For more details, it is recommended to follow [6].

Module *Aggregator* This module is responsible to calculate aggregate reputation score using the individual user feedback. The aggregator is a crucial component of the REPUTABLE system.

GET `/localhost:3000/reputation` Aggregate the reputation score for a seller. Takes the identifier of the seller as the argument and returns the seller's reputation score.

POST `/localhost:3000/reputation_score` Post a seller's aggregate score using seller id and user scores. Taking sellerID and individual reputation scores, as arguments.

GET `/localhost:3000/verify_reputation` Verify the reputation of a service/seller. Returns the receipt highlighting on-chain record/hash of reputation feedback.

GET `/localhost:3000/individual_score` Get the individual score of a buyer/consumer.

GET `/localhost:3000/individual_scores` Get the individual scores relating to a buyer's aggregated score and returning the individual scores of buyers who rated the specified seller.

GET `/localhost:3000/token_used` Query the aggregator to find out if a token has previously been used. Takes the token as an argument and returns a Boolean value.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

2.2.6 KnowledgeX: Trusted data-driven knowledge extraction

Description KnowledgeX is a trustworthy marketplace for data science. This means data owners can outsource data science tasks to independent contractors without risking data misuse. Independent data scientists can bid on proposed tasks without getting prior access to confidential data.

Currently data markets are hampered by confidentiality requirements due to competitive (e.g., cost data) or regulatory (e.g., personal data) considerations. Data scientists have to be employed in-house or are contractually restricted by non-disclosure stipulations, which tend to be ambiguous and costly to enforce.

KnowledgeX aims to solve this problem via a process where data privacy, and contract fulfillment are technologically guaranteed, so the need for non-disclosure agreements does not arise. Therefore, KnowledgeX leverages the following technologies:

Emerging technologies:

- Blockchain and Smart Contracts
- Decentralised Cloud Computing
- Trusted Execution Environments

Established technologies:

- Web Applications
- Public Key Infrastructure
- Microservice Backend

Application Programming Interfaces According to the ONTOCHAIN project proposal, different higher-level applications and interfaces will be built through several open calls. These applications and interfaces can leverage KnowledgeX' REST API for identity management, microservices and interface to on-chain interactions. The corresponding details of the REST APIs for each individual services are presented in [7].

> **Module *1 (USER DETAILS SERVICE)*** Rest API of the user details service is used by frontend user interface to store personal information of the user. The API allows to modify information such as address, billing data or user description. This service stores users IDs shared between the services and allow to detect user Type to render the corresponding views.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Module *2 (EXECUTION SERVICE)* Execution service rest API is used by frontend user interface to trigger and manage execution/exploration jobs, datasets, programs as well as blockchian orders or execution results. The Blockchain Middleware use this API to persist the execution data gathered from iExec.

Module *3 (GIG SERVICE)* Gig Service API is used by User service to persist new users and by frontend application to manage gigs and offers.

Module *4 (BLOCKCHAIN MIDDLEWARE)* For this service, no external integration is allowed. Some examples of API calls for exposing this services are as follows:

GET `/<base_uri>/blockchain-middleware/iexec/deal/id/results` Taking id of execution job as argument and returning execution results.

POST `/<base_uri>/blockchain-middleware/iexec/executionJob` Taking the whole JSON containing the task execution details and stores execution job details in iExec and starts execution.

POST `/<base_uri>/blockchain-middleware/contracts` This REST call stores contract in Blockchain.

2.2.7 POC4COMMERCE: Making ONTOCHAIN practical for eCommerce

Description POC4COMMERCE innovates the ontological representation of Blockchain-oriented digital commerce by integrating and extending the most representative ontologies for modelling, participants, in particular commercial actors, offers, products, and tokens emitted on the Ethereum Blockchain as digital representation of exchanged assets: providing a semantic descriptions of smart contracts and related transactions, in particular of smart contracts related with tokens trading and associated with commercial means.

POC4COMMERCE reuses and extends the most suitable and relevant ontologies in the domain, namely, OASIS for the representation of commercial participants and smart contracts, GoodRelations for representing commercial offers, and BLONDiE for describing Ethereum essential elements: all these ontologies are conjoined and extended to also cover the gap missing from the literature on the representation of digital tokens, smart contracts, digital identities, valuation mechanisms, and auctions.

Application Programming Interfaces POC4COMMERCE delivers a set of three modular ontologies describing each semantic compartment of eCommerce, from participants, assets, and offerings, to supply chains, smart contracts, and digital tokens. The

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

ontological stack is released together with the SPARQL queries implementing the defined competency questions that enable users and developers to meaningfully probe the knowledge bases they contribute to construct. Also, it delivers an API library consisting of two main modules: *OC-Generator*, in short OCGEN, providing basic APIs for generating agent ontological rep- resentations in the mentalistic notion of agent behaviors fashion, adopted by the foundational ontology OC-Found. The second one is *OC-Commerce Search Engine*, in short OCCSE, providing APIs that realize the core of a semantic search engine and reasoning system for querying the ONTOCHAIN knowledge bases.

The modules OCGEN and OCCSE have been implemented in Python, version 3.7. Specifically, OCGEN adopts the RDFLib[12] version 5 API library[13] to generate RDF fragments, whereas OCCSE adopts the API library OWLReady 2[14] version 0.34 to realize the query engine. Whereas RDFLib provides low-level APIs for generating RDF triples in an extremely efficient way, OWLReady 2 provides an higher level interface for integrating OWL reasoners and performing SPARQL queries compliant with the latter version 1.1.

Module *OCGEN* The OCGEN main object can be instantiated first by creating three RDFLib ontology objects, one for the ontology hosting the agent behaviors, one for the ontology hosting the agent templates, one for the ontology hosting data, then by calling the constructor of the class OCGEN. The module will store RDF graphs in the correct ontology depending on the type of operation required. It is also possible to create or load one single ontology for storing both behaviors, templates, and data. The OCGEN module provides several methods to deal with semantic web agents in the OASIS ontology fashion:

`createAgentTemplate()` is a void method creating an agent template given agentTemplateName as input parameter, namely a string representing the agent template name.

`createAgentBehaviorTemplate()` is a void method creating a behavior template to be associated with an agent template.

`connectAgentTemplateToBehavior()` is a void method associating a behavior template to an agent template.

`createAgent()` is a void method creating a real agent where MyAgent is a string representing the agent name.

`createAgentBehavior()` is a void method creating a concrete behavior to be associated with a concrete agent.

[12] https://pypi.org/pro ject/rdflib/5.0.0/
[13] RDFLib version 6 has currently a bug preventing the OCGEN prototype from loading remote ontologies.
[14] https://pypi.org/pro ject/Owlready2/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

`connectAgentToBehavior()` is a void method associating a concrete behavior to a concrete agent.

`createAgentAction()` is a void method creating an agent action that is associated with an agent behavior.

Module *OCCSE* Before instantiating the OCCSE, a repository manager and a reasoner interface are required. To create a repository manager it is sufficient to create an object of type RepositoryManager, passing a list of IRIs representing the repositories that will be loaded into the OCCS triple store.

`RepositoryManager()` returns the object RepositoryManager, where [repository1, repository2, ...] is a list of IRIs representing the repository to load.

`ReasonerInterface()` returns the ReasonerInterface, where reasonerName is either pellet or hermit values.

`OCCSE()` instantiates the OCCSE module, where repos- itoryManager and reasonerInterface are the repository manager and the reasoner interface, respectively, as created before.

The details of other methods are describe in [8]. Notably, for this project no REST services are included in current implementation.

2.2.8 ONTOSPACE

Description Ontospace's value proposition contains the innovative use of RDF graphs with Blockchain to provide trusted environment for improved performance of structural data storage with the use of the graphs. It also reiterates the innovative concept of Ontospace ecosystem, that shall be thought of as a multichain network of networks based on Blockchain Layer-2 principles. The construction of Ontospace has direct benefits for ONTOCHAIN large scale pilot, because it provides a working ecosystem in which various networks can be launched and coexist under the fundamental principles of the construction of Layer2 networks. The project comprises of four different program modules. Brief description of those modules are given below:

Module *Blockchain* developed on the Ethereum network. Notably, the project deployed the hyperledger Besu Ethereum client which allows the direct access to graph database.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Module *Ontoshell* is a collection of bunch of REST APIs. Importantly, these REST API services are the main gateway for interacting with Graphchain. This module is also responsible for synchronization between deployed Ontonodes.

Module *Ontopod* is the main storage component of the Graphchain. This module has been developed based on the Blazegraph triplestore database.

Module *VCFIH* module is responsible for implementing Vicious Circle Free Interwoven Hashing algorithm.

Application Programming Interfaces No SDK has been implemented for this project, therefore no APIs have been defined for accessing the SDK. However, for accessing the services, a few REST APIs are developed. Brief description of them are given in the follows:

POST `/rdf-graph-store?graph=graph_uri&user=user_addr` takes arguments as URI of graph to be stored on Graphchain and Ethereum address of user for uploading a graph. For the successful operation, this API helps to upload the RDF graph from the request body to Graphchain.

PUT `/rdf-graph-store?graph=graph_uri&user=user_addr` takes arguments as URI of graph to be stored on Graphchain and Ethereum address of user for uploading a graph. For the successful operation, this API helps to modify and upload the RDF graph from the request body to Graphchain.

GET `/rdf-graph-store?graph=graph_uri` takes URI of graph as argument which would be retrived from Graphchain and returns the latest version of an RDF graph.

DELETE `/rdf-graph-store?graph=graph_uri` takes URI of graph as argument which would be removed from Graphchain.

HEAD `/rdf-graph-store?graph=graph_uri` takes URI of graph to be retrieved from Graphchain. Similar as GET operation but without content in response body. Same to ASK SPARQL query.

POST `/deploy?contract=type` takes the contract info which to be deployed to the Blockchain, as argument. Helps to deploy contracts needed for proper work of Ontonode.

GET `/info?size=graph_count` takes number of graph records to be returned as argument and returns the JSON with graph metadata which has to be uploaded to Graphchain.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

GET `/nodes/get?size=count` takes number of information (ontonodes) as argument for displaying information about ontonodes.

POST `/nodes/adding?nodeId=id` takes nodeid as argument and it helps to add the new ontonode.

POST `/nodes/update?nodeId=id` takes nodeid for modifying the ontonode.

2.2.9 ADOS

Description ADOS (AirTrace Decentralized Oracle System) originally stems from AirTrace platform, developed by Cubic Fort Consultores during 2021 and now available as a closed beta for our early clients. As was briefly introduced in the Executive Summary, and also detailed in later sections, AirTrace is a platform that allows, both visually as well as programmatically to quickly and easily deploy IoT networks supporting Blockchain. Users (IoT system integrators and IoT SaaS platforms) simply choose the IoT devices to deploy among the growing database of IoT System Integrators, and after a few configuration steps, the resulting interfaces (RestFUL API, MQTT, etc.) are generated, so that integrators can easily support them in their IoT projects. Since data injected to the Blockchain comes from IoT devices measuring physical magnitudes (temperature, air quality, etc.) data can potentially be subject to different attacks that may corrupt their integrity. The real utility of ADOS in the platform AirTrace (as well as other potential platforms that in the future might make use of ADOS for other non-necessarily blockchain-related anomaly detection schemes) is that it helps potential audits later to verify the reliability of data when already stored in the Blockchain. This increases transparency of data and improves the reliability of Blockchain and IoT systems by improving the core element: reliability of data and its credibility. ADOS helps to maximize data reliability by proving that anomaly detection algorithms can be used in distributed oracle systems in order to have an extra layer that can convey important information to enhance data credibility before injecting into the Blockchain. In fact, this approach based on Graph Neural Networks, for that it is possible to embed information not only about given sensors but also from external oracles (out of the IoT network itself) that can play an important role when spotting anomalies thanks to inherent correlations. Among other advantages, this system can be pretty fast in comparison with other approaches like the ones shown before where complex schemes (reputation, voting, etc.) are associated to longer computing times and higher delays.

Application Program Interfaces

Module *ADOS* is wrapped by AirTrace main Restful API infrastructure. It contains three main HTTP methods that are used to send IoT readings to ADOS. All readings will be associated to a specific link that will accumulate all readings before sending

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

them to the blockchain along with the computed DQF (Data Quality Factor).

POST `/api.airtrace.io/v1/ados/<link>` takes specific link where IoT data is submitted. Specifically, this API reads body from the POST request and its submission is cumulated during the commitment interval, where Id_transaction corresponds to the current interval slot.

GET `/api.airtrace.io/v1/ados/<link>/<id_transaction>` takes specific link from where to retrieve data and associated transaction id for the current reading. In returns the it gives the DQF associated to each sensor reading cumulated in the current window.

GET `/api.airtrace.io/v1/ados/<link>` takes specific link where IoT data is retrieved. Finally it returns the URI, which gives all available id_transaction completed or ongoing so far associated to link, which can be reviewed later.

2.2.10 CARECHAIN

Description CARECHAIN, offer a platform where companies can provides services of insurance contracting, advice and risk management. CARECHAIN addresses the following challenges: 1) these insurance contracting companies, to enhance services, reduce managing costs and provide secure online mechanisms, and 2) competitor companies seeking comparative advantages to enter this market, within the scope of ONTOCHAIN ecosystem. The idea behind CARECHAIN project is design and implement an environment for the execution of smart contracts for parametric microinsurance based on the distributed ledger, to guarantee users the application of coverage when meeting contract conditions, in search for new market niches and allowing the revitalization of the economy caused by COVID19 pandemic. Microinsurances and similar financial operations are one of the most security and trust demanding sections for online operations, creating a platform for microinsurance contracting and managing automated service, with security, traceability and trust using DLT technologies is expected to have a great impact.

Application Program Interfaces No SDK has been implemented, therefore no APIs have been developed related with the SDK. However, a vast number of end-points have been developed to interact with backend blockchain services, smart contracts, oracles, access management modules, sensors.

Module *APIs for Blockchain services* A set of endpoints has been developed to interact both in the backend and directly with the deployed ERC735 and ERC930 Smart Contracts. In the following image you can see all the endpoints designated for this function. The method root/get returns the address of the owner of the Smart

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Contract. Through the `onlyRoot` modifier defined in the smart contract, access to add or remove new administrators is restricted.

GET `/api/v1/blockchain/admin/get` This API call return if an account is admin.

POST `/api/v1/blockchain/root/transfer` This API transfers the ownership of the contract.

POST `/api/v1/blockchain/claim/changeStatus`

GET `/api/v1/blockchain/claim/get` This API call returns the data of a claim, either generic or specific.

POST `/api/v1/blockchain/claim/grantAllowanceSigner` This API call gives permissions to a user to sign a contract.

GET `/api/v1/blockchain/claim/isAllowanceSinger` This API call returns if an account is allowanceSigner.

POST `/api/v1/blockchain/claim/revokeAllowanceSigner` Removes a user's signing permissions.

POST `/api/v1/blockchain/claim/sign` This API call helps a user signs a claim to which he has permissions.

POST `/api/v1/blockchain/representative/addGenericClaim` This API call add generic claim (representative).

POST `/api/v1/blockchain/representative/addSpecificClaim` This API call add specific claim.

POST `/api/v1/blockchain/representative/grantAdmin` Gives administrator permissions to an user (wallet address).

POST `/api/v1/blockchain/representative/revokeAdmin` Remove administrator permissions to an user (wallet address).

GET `/api/v1/blockchain/root/get` This API call return the address of the root.

POST `/api/v1/blockchain/supplier/addClaim` This API call add specific claim (supplier).

POST `/api/v1/blockchain/supplier/modifyClaimCondition` This API call modify an existing claim.

POST `/api/v1/blockchain/supplier/modifyClaim` This API call changes a condition of an existing claim.

Module *APIs for Oracles* For registering a new oracles, the project coordinators have used a new type of variables for defining of an oracle, which is active in iExec sidechain. Also, when someone wants to store a new oracle, it will be first registered in the database.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Module *APIs for Roles* The method `newRepresentative` modifies a user role to representative. A user will have representative permissions and will be able to create microinsurance templates.

POST `/api/v1/roles/newRepresentative` Turns a user into a representative.

POST `/api/v1/roles/newSupplier` This API call turns a user into a Supplier.

POST `/api/v1/roles/revokeRepresentative` This API call revokes a user's representative role.

POST `/api/v1/roles/revokeSupplier` This API call revokes a user's Supplier role.

Module *APIs for Sensors* For sensors, the project coordinators have used Web of Thing Description directory JSON for the definition of the sensor. When someone wants to store a new oracle, it will be registered in the database.

GET `/api/v1/microservices/get` This API call gets all the microservices.

POST `/api/v1/sensors/addData` This API call registers a new sensor data.

POST `/api/v1/sensors/deploy` Deploys a sensor.

GET `/api/v1/sensors/get_all` This API call returns all sensors data.

GET `/api/v1/sensors/get` Get a sensor data.

POST `/api/v1/sensors/register` This API call register a new sensor.

Module *APIs for Auth* Login method will give the user access to the platform by logging in.

GET `/auth/context` This API call authenticate context.

POST `/auth/custom_logout` This API call performs custom Logout (close any session knowing the ID).

POST `/auth/email/verify` This API call sends request to verify an account.

POST `/auth/login` This API call performs login operation with username and password.

POST `/auth/logout` This API call performs logout operation.

POST `/auth/signup` This API call creates an account using email and password.

POST `/auth/tfa` Two factor authentication login. Input the one-use code.

For more details of the API implementation please follow the link and go to the Swagger API documentation tool.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

2.2.11 BOWLER

Description Smart contract development is software-intensive, requiring specific and scarce talent which is costly and prone to human risks. BOWLER offers a model-driven Web-IDE to allow (a) faster time to market (KPI: 1-statement smart-contract deployment in <1hr), (b) easier evolution through reusable models, (c) fostering standardization and thus interoperation amongst the entire ONTOCHAIN ecosystem. Indeed, the BOWLER will provide end-to-end support through its web-enabled IDE to reuse pre-existing model specifications (blueprint), model them and generate deployable Solidty code. In this way, the BOWLER can be used by any member of the BOWLER ecosystem to quickly deliver trustworthy smart contract solutions. Once more blueprints are added to the Bowler, higher levels of standardization and interoperation between smart-contract solutions of ONTOCHAIN partners can be reached. Notably, the project is composed of five different program module. A brief description of those modules have been given below.

Module *AKB Portal* pre-existing component characterizing the AstraKode Blockchain platform, allows users to register, authenticate and authorize themselves to the platform, and access low-code environments for developing blockchain solutions. It will therefore allow access to the Bowler Smart Contract Visual IDE. It also integrates a payment gateway that will allow the solution to be purchased and services delivered in SaaS mode.

Module *Bowler Smart Contract Visual IDE* is the visual development environment that will allow users to model the smart contracts and use the blueprint templates. This is a sophisticated black box module that allows to configure itself according to a specific definition of the components characterising the metamodel and the view files of a specific DSL (in the case of Bowler related to the creation of Smart Contracts). The content will be enriched by integrations with the semantic search module.

Module *AKB Service Layer* is the module provided by the AKB platform to manage projects, users, subscriptions, and all services required to operate the platform. This layer acts as the entry point and links with the big part of the other modules of the service layer.

Module *AKB Generator Engine* can receive the composite model instance reflecting a specific use case and, by means of two other fundamental elements, the processors and the generation cartridges, is able to trigger a sophisticated templating

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

engine capable of producing the desired source code.
The processors, defined for every single component, determine the quantity and type of output files resulting from code generation. The MDA-cartridges instead, contain exactly functions and macros, that contain all the elements and mechanisms required by a particular technology (Solidity, Javascript, Typescript, ...), to be used to dynamically compose the contents of all output files.
One feature we believe is essential to the generation process is its complete platform independence. The Platform Specific Model will be built through all the elements described above, enriched using place-holding techniques, of platform specific details.

Module *Semantic Engine* offers the ontological extensions planned for the scope of BOWLER, stemming from the predefined baselines.

Application Program Interfaces This project does not have an SDK. The project services will be provided to the final users by means of visual development environments. Also, no Services have been implemented within the scope of this project. So, no APIs have been provided. However, specific APIs have been provided to manage models and generate smart contracts. The overview of these APIs have been presented bellow:

POST `http://localhost:8080/AstraKode-BE_WS/rs/public/generate` this api call post an XML instance string (the smart contract model) to the handler and returns the solidity generated file.

POST `http://localhost:8080/AstraKode-BE_WS/rs/public/upload` this call post an XML file to the handler, which allows to handler to add additional ticket types.

GET `http://localhost:8080/AstraKode-BE_WS/rs/public/get_rdf` this method returns full RDF for the XML files known to the prototype/system.

GET `http://localhost:8080/AstraKode-BE_WS/rs/public/sqarql` this call helps handler to run the SparQL engine, using a query provided. The input parameter query accepts a basic text / string value, with a SparQL query.

2.2.12 MFSSIA

Description In recent years, it has been seen the emergence of identity authentication systems to enable access to public- and private systems that render life easier. For example, in Estonia, the ID card has been essential for establishing the well known e-governance infrastructure. However, over more than the last two years, it has also seen the considerable erosion of citizen's liberties and freedoms with QR-code pass-

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

ports that are a planned foundation for attaching a Chinese style social-credit score system. Since such a development was considered undesirable, at least until 2019, it is important to enable multi-factor self-sovereign identity authentication (MFSSIA) that re-empower individuals for accessing in a trusted way information systems and to also engage in the future in the so-called machine-to-everything (M2X) economy. For this project and as per the implementation guide, the MFSSIA project focuses on enabling trusted cross-blockchain connection establishment. Given the pre-existing scholarly peer-reviewed research results about MFSSIA, the goal of this project is to map these results into the available ONTOCHAIN technology stack. The decentralized knowledge graph (DKG) solution has been used for expressing challenges and also to capture contexts for which responses to the challenges must be detected. The iExec Cloud is on the one hand the deployment infrastructure and on the other hand also delivers oracles to MFSSIA lifecycle for confirming the validity of responses to the configured challenge sets.

Application Program Interfaces

Module *MFSSIA Module APIs* is defined as a REST API that includes functions for retrieving the data from DKG (in the current context business consensus, security license and gateway). The modules API encapsulates the complexity of communication with DKG nodes and provides a simple way for oracles to request and process the DKG data.

`get_business_consensus(string businessConsensusInfo)` this function retrieves the data about business contract from the DKG node and returns its hash to the iExec oracle.

`get_security_license(string securityLicenseInfo)` first this function checks if requested data is valid and then queries the DKG instance that contains information about requested security licenses. If a respective security license is valid and the requested data is present in the data contained in the DKG instance then the function returns this data to the oracle.

`get_gateway(string gatewayInfo)` this function first validates the requested data and if the request is valid then the function queries the DKG instance for the existence and availability of these requested gateways. If such gateways exist and are available then the function returns this info to the oracle.

Module *REST APIs FOR SERVICES* Which is a communication adapter between MFSSIA oracles and DKG nodes. This module is supposed to simplify the business logic ofiExec oracles. All endpoints are available under <base_url>/v1/mfssia. As MFSSIA is in the development phase, we do not yet have a production environment in place, therefore, the base url of the API is defined as a parameter <base-url>. The project coordinator did not use any specific headers, and they expect the DKG node to return json content.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

GET `/business-contract/<contract-id>` takes id info of a contract to retrieve and it return the hash of the business contract.

GET `/security-license/<license-nr>` takes number of security license as argument and returns information about the security license.

GET `/gateway/<gateway-type>` takes the information about the type of the gateway and it retrieves the gateway details from DKG.

The API for the iExec oracles has 3 functions. All endpoints are defined as GET HTTP calls with one (1) integer input parameter that defines a unique identifier of the business contract. Checking the business contract returns the hash of the DKG data while calling the security license and the gateway endpoints will return info about the security license and the gateway.

2.2.13 NFTSwap

Description The project expect a massive market development based on the new approach. Potentially the system and its upcoming copy cats will gather the majority or at least a vast portion of the entire NFT-market due to its obvious benefits in the prediction and valuation of formerly unvalued items/assets. The project prediction-system will pave the path for the availability of new assets onchain. With its open-source approach the blockchain-agnostic solution will be used on additional chains opening new markets and new fields of interest. The solution architecture of NFTSwap project can be compared to the smart contract architecture of UniSwap. There is a central factory contract (Registry) that deploys individual NFT markets. Instead of deploying ERC20 contracts, the factory contract is an ERC1155 contract that registers all individual tokens used by the deployed markets. The architecture of NFTSwap is mainly comprises of Central TokenRegistry, a few NFT-Markets, Router and the User Interface.

Module *Token Registry (ERC1155)* helps to tracks markets and bull and bear tokens, deploys contracts for new markets.

Module *NFT-Market* is mainly responsible to trad of Bull and Bear tokens within the scope of this project.

Module *Router* makes it easier to interact with PiSwap protocol.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Module *User Interface* is the tool, which helps to the users to interact with the NFT-Markets.

Application Program Interfaces No SDK implementation has been done for this project. Therefore, no service APIs are offered by this project. Also, the smart contract implementation technically does not offer a specific REST API for the implementation. The smart contract API can be generated from the source code allowing to interface with it. Mainly two types of publicly callable functions of the *PiSwapRegistry* and *PiSwapMarket* contracts, have been developed. Below, a thorough description of them has been presented.

Module *PiSwapRegistry* this kind of functions mainly helps to register the smart contracts in the existing Blockchain network. A detailed functionalities of this kind of functions have been presented below.

`function WETH() external view returns (address);` Returns the address of the WETH contract for the chain the protocol is deployed on.

`function createMarket(address tokenAddress, uint256 tokenId) external returns (address market);` Creates a new market for a specific NFT identified by the NFT contract address and token id. The function returns the address of the newly created market.

`function mint(address to, uint256 amount, TokenType tokenType) external;` Mints new bull, bear or liquidity tokens to a specific address. Only callable by markets.

`function burn(address from, uint256 amount, TokenType tokenType) external;` Burns bull, bear or liquidity tokens from a specific address. Only callable by markets.

`function deposit(uint256 amount) external;` Deposits WETH into the registry contract and returns an equal amount of WETH1155 to the sender.

`function withdraw(uint256 amount, address to) external;` Burns WETH1155 from the sender and returns an equal amount of WETH to the specified address.

`function beneficiary() external view returns (address);` Returns the address of the beneficiary for the protocol fee.

`function fee() external view returns (uint256);` Returns the protocol fee in myriad taken when Bull and Bear tokens are minted or burned.

`function oracleLength() external view returns (uint256);` Returns the amount of blocks taken into consideration for the NFT value oracle.

`function getMarketForNFT(address tokenAddress, uint256 tokenId)`
`external view returns (address market);` Returns the market address for a specific NFT identified by the NFT contract address and token id. Reverts if market does not exist.

`function marketExists(address tokenAddress, uint256 tokenId) external view returns (bool);` Returns true if a market exists for a specific NFT identified by the NFT contract address and token id.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Module *PiSwapMarket* this type of functions enables the interactions with the marketplace.

`function registry() external returns (address);` Returns the contract address for the registry contract.

`function underlyingNFT() external view returns (address tokenAddress, uint256 tokenId, NFTType nftType);` Returns contract address, token id and NFT type (ERC721 | ERC1155) for the underlying NFT.

`function mint(Arguments.Mint calldata args) external returns (uint256 amountIn, uint256 amountOut);` Mints new Bull and Bear tokens according to the token formula as described in the whitepaper. Returns amount of ETH given in and amount of Bull and Bear tokens minted.

`function burn(Arguments.Burn calldata args) external returns (uint256 amountIn, uint256 amountOut);` Burns Bull and Bear tokens according to the token formula as described in the whitepaper. Returns amount of Bull and Bear given in to burn and amount of ETH returned.

`function addLiquidity(Arguments.AddLiquidity calldata args) external returns ( uint256 liquidityMinted, uint256 amountBull, uint256 amountBear );` Adds liquidity to the liquidity pool. Returns the amount of liquidity tokens minted to the liquidity provider, and amount of bull and bear tokens provided to the liquidity pool. The amount of ETH provided to the liquidity pool is passed as an argument and does not change, therefore it's not returned. The initial liquidity provider sets the initial NFT value using the amount of Bull and Bear tokens provided. All subsequent liquidity providers provide bull and bear tokens according to the current ratio in the liquidity pool.

`function removeLiquidity(Arguments.RemoveLiquidity calldata args) external returns ( uint256 amountEth, uint256 amountBull, uint256 amountBear );` Burns liquidity tokens from a liquidity provider and returns the proportionate amount of ETH, Bull and Bear tokens to the provider.

`function swap(Arguments.Swap calldata args) external returns (uint256 amountIn, uint256 amountOut);` Swaps one token into another. Liquidity tokens cannot be swapped with the liquidity pool. Bull and Bear tokens can be swapped with ETH and with each other. Returns amount in for the token in and amount out for the token out. Token in and token out are defined in the args struct.

`function sellNFT(Arguments.NFTSwap calldata args) external returns (bool);` Sells an NFT for ETH with the contract for the current value set by the market. Returns true on success.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

`function buyNFT(Arguments.NFTSwap calldata args) external returns (bool);` Buys an NFT held by the contract for ETH for the current value set by the market. Returns true on success.

`function mintOutGivenIn(uint256 amountIn) external view returns (uint256 amountOut);` Returns the amount of Bull and Bear tokens minted given amount of ETH.

`function mintInGivenOut(uint256 amountOut) external view returns (uint256 amountIn);` Returns the amount of ETH in given the amount of Bull and Bear tokens to mint.

`function burnOutGivenIn(uint256 amountIn) external view returns (uint256 amountOut);` Returns the amount of ETH returned given an amount of Bull and Bear tokens to burn.

`function burnInGivenOut(uint256 amountOut) external view returns (uint256 amountIn);` Returns the amount of Bull and Bear tokens to burn given the desired amount of ETH.

`function swapOutGivenIn( uint256 amountIn, TokenType tokenIn, TokenType tokenOut )`
`external view returns (uint256 amountOut);` Returns the amount of tokenOut for a given amount of tokenIn when swapping tokens.

`function swapInGivenOut( uint256 amountOut, TokenType tokenIn, TokenType tokenOut )`
`external view returns (uint256 amountIn);` Returns the amount of tokenIn required to receive a given amount of tokenOut when swapping tokens.

`function lockedEth() external view returns (uint256);` Returns the amount of locked ETH available that can be used to swap NFTs.

`function nftValueAccumulated() external view returns (uint256);` Returns the current value of the NFT used by the contract when buying or selling NFTs. This value is accumulated over a period of time to make it more resilient to manipulation.

`function swapEnabled() external view returns (bool);` Returns true if the amount of locked ETH exceeds available to swap NFTs exceeds the current NFT value for a single NFT.

`function nftValueAvg(uint256 amount) external view returns (uint256);` Returns the average NFT value over the last amount of blocks. This function is meant to be used by other smart contracts integrating with the PiSwap protocol.

`function nftValue() external view returns (uint256);` Returns the most recent NFT price. This value is less resilient to manipulation.

`function getOracleEntry(uint256 index)external returns (uint256 price, uint256 timestamp);` Get an entry from the oracle array. Returns the price at the returned timestamp.

`function oracleLength() external view returns (uint256);` Returns the total amount of NFT value entries in the oracle.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

2.2.14 NFTWATCH

Description Through advanced ML and semantic based technology, NFTWATCH helps to analyse and classify not only the structured data provided by the artist, but also unstructured data from different sources, including the piece of art itself (ML based image recognition and classification). We will use both the data collected and the data generated to propose multi-facet search and visual discovery of the NFT world. NFTWATCH also enables the facilities to the NFT owners for checking their authenticity, and match offchain and onchain. The data will be available not only through a GUI, but also by a REST API. By creating a complete ontology around NFT data fed by multiple sources associated with online visual data explorer, NFTWATCH is helping to fight the IP fraud and structuring what may become the biggest art market in a very near future.There are four logical components in the NFTWATCH solution corresponding to the four main functions of the applications:

- Collect NFTs, their metadata and associated media.
- Enrich collected data and align them to the target model/ontology.
- Store collected and enriched data.
- Search among all data.

The four main components are: 1. Collect component; 2. Enrich component; 3. Storage component; and 4. Front components.

Module *Collect component* is responsible to localize NFTs on marketplaces and to harvest all information's about it: the resource itself (images, videos, sounds, text), id, the metadata's attach to the resource, smart contract and onchain metadata.Spcial note; in accordance with the updated scope of the project, NFTWatch will only manage text and Image resources.

Module *Enrich Component* aims to extract all relevant information's from unstructured data using dedicated and appropriated algorithms. As we plan to consider different natures of asset (image, video, sound, and text) and because we design an open and flexible architecture, per nature, we propose to provide an interface to different processors. For this project, we will use general components and only on some text fields (NLP) and the image resource (Classification, features extraction). For both cases, machine learning approaches will be used.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Module *Storage component* responsible to store the outputs of enrichment stage, in an appropriate format. Mainly two kind of storage can be used within the scope of this project.

- Document oriented storage (Search engine) which have a good performance in indexing and querying.
- Graph oriented storage which is the most versatile kind of storage to authorize advanced queries without knowing the queries at the beginning.

In the future, decentralized storage of some of contents (post enrichment only) will be considered (not for this project). The model used into Neo4J is the common model. To expose the data into the targeted ontology, the project intended to use neosemantics , a Neo4J extension.

Module *Front Components* This components can be comprises of:

- A REST API behind for legacy apps
- A simple graphical interface to illustrate the features of the platform based on the REST API.

Application Program Interfaces No SDK has been deployed for this project; however to opt the services from the NFTWatch there are several APIs have been defined, which are described below:

Module *NFTWatch REST API* enables to query NFTWatch database in order to retrieved :

- NFT metadata including description, metadata and NFT confidence score
- NFT collection metadata including confidence score
- a list of NFT corresponding to classification criteria
- a list of NFT realized by an artist

GET `/asset/<contract_address>/<token_id>?output_format=` takes address of a contract, identifier of the NFT related to the contract and information about the output_format as arguments for returning all metadata associated to a NFT.

GET `/assets?filters=&cursor =` it returns a paged list of NFTs with their metadata.

GET `/assets?rdfgraph=&cursor =` returns a paged list of NFTs with their metadata.

GET `/collection//<contract_address>` returns all the metadata associated to a collection, especially: i) A confidence score; ii) The number of tokens in the contract

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

GET `/collection//<marketplace>/<collection name>` returns same as the above therefore it gives all the metadata associated to a collection, especially: i) A confidence score; ii) The number of tokens in the contract

GET `/collections?cursor=` returns a paged list of contract address. Next: A cursor to be supplied as a query param to retrieve the next page; Previous: A cursor to be supplied as a query param to retrieve the previous page.

2.2.15 DESMO-LD

Description DESMO-LD project aims to provide a fully integrated distributed solution for consuming IoT external data, enriched with Web of Things semantics and data model, inside the ONTOCHAIN. This addresses the call's objectives of designing new trustful decentralized Oracles to poll semantic data from off-chain data sources. Besides, DESMO-LD introduces novel strategies to solve the above mentioned problems thanks to the heavy deployment of standard ontology and semantic oriented consensus algorithms for data quality and trustiness.The system architecture of the DESMO-LD is divided between an on-chain part, consisting of a set of smart contracts, and an off-chain part with the Oracle DApp and the Web of Things Thing Description Directory (TDD). In DESMO-LD different types of clients that may be interested in using the system: a classic smart contract, a complete full-stack DApp, or even a pure web3 frontend application. In particular, the environments span from on-chain deployments (i.e., smart contracts) and off-chain services and applications. Most of the core services will be developed during this project time frame but some of those may be controlled by different parties (i.e., VAIMEE s.r.l. may deploy and control some off-chain services, but it encourages third parties to deploy and control their own). The components are following:

- DESMO-LD iExecDOracle
- DESMO-LD Hub
- Oracle DApp
- Thing Description Directory

Application Program Interfaces Notably, no SDK has been implemented for the DESMO-LD project. Whereas, for ensuring to offer the services DESMO-LD has only one off chain component which has open REST API endpoints. The name of the component is **Thing Description Directory**. In below we are going to presents the APIs and their functionalities:

GET `https://tdd.desmo.vaimee.it/things` this API call retrieve all thing description.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

PUT `https://tdd.desmo.vaimee.it/things/id` this API call takes the ID of the created thing description and returns the created thing description.

POST `https://tdd.desmo.vaimee.it/things` this call takes the ID which to be created and it returns the created thing's description.

GET `https://tdd.desmo.vaimee.it/things/id` takes the ID of the created Thing description and in return it gives the thing description.

PUT `https://tdd.desmo.vaimee.it/things/id` by taking the ID of the created Thing description it helps to update the thing description.

PATCH `https://tdd.desmo.vaimee.it/things/id` this API call partially update the thing description by taking the ID of the created thing.

DELETE `https://tdd.desmo.vaimee.it/things/id` taking created Thing ID as an argument, it deletes a thing description.

GET `https://tdd.desmo.vaimee.it/search/jsonpath?query=query` this call takes the JSON path query string as an arguments and returns the path result for successfully performing the syntactic search.

2.2.16 PRINGO

Description Leveraging the blockchain capabilities to design new market mechanisms, this sub-project provides a platform aimed at escaping from huge difference in efficiency between free markets in the private vs common sectors in certain digital industry verticals, by realigning incentives in common goods economies so that private agents obtain benefits as significant as in private goods markets if their actions provide real value to the commons.

To limit the scope of the sub-project, the sub-project owners' propose an initial focus towards developing a robust link between common goods and the videogames industry. As last years boom in Non-Fungible Tokens (NFT) proved, users love owning digital assets (a song, a sword in a video game), and trading them in decentralized open markets. The rules that govern these markets are public (based on open-source code deployed to a blockchain), allowing entire decentralized economies to be built around them. Then provides a platform, which is ease to use for common goods curators and video game developers, whereby gamers can obtain direct profit from improvements of the real curated goods. By doing so, it aims at significantly improving the funding of certain common goods as well as to improve the retention, loyalty and commitment of their contributors.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

In PRINGO, to support some particular use cases (i.e., CGCs) through a control panel dashboard, it integrated with an API implementation which integrates both governance and Freeverse evolving NFTs technology. In addition, another role-based view of the dashboard supporting game users use cases also integrated with both Freeverse API and governance layer (in this occasion to check update status of the NFTs. Finally the SDK layer to support video-game companies integration; either using a reference implementation client with Freeverse to ease the companies engagement with the project or modular specifics clients that could fit into the game development workflow itself like C# for Unity or C++ for Unreal Engine as example. The PRINGO platform consists of a set of smart contracts deployed on a low-environmental impact blockchain, alongside nodes for the corresponding Living-NFT Layer 2, together with a set of tools to facilitate interaction/integration to different actors. Mainly these module are: PRINGO-backend, PRINGO-SDK, PRINGO-frontend.

Application Program Interfaces For PRINGO, there are two kind of APIs, can be found. One for accessing the SDK and the other one kind of APIs is responsible for providing various services. In below, those have been thoroughly described.

Module *REST APIs for SDK* provides a simple interface to manage the different project workflows; it is intended to be consumed both by the marketplace operators, common good curators using REST APIs described in the next section. In addition this interface mainly allows video game developers to interact with Freeverse NFT API. Below, the related functions of this interface have been presented thoroughly.

`getNonceCount` this function call returns the user nonce inside a particular Freeverse universe.

`getAssetNonceCount` this function call returns the asset nonce by taking the Freeverse universe identifier as argument.

`getAssets` this function fetch the assets information by taking index information, number of assets, and asset owner's delegate key as arguments.

`getAssetsByAttribute` this function filters assets by attribute.

`getAssetById` this function call takes asset ID as argument and returns an asset corresponding to the identifier.

`createAsset` taking asset information including metadata as arguments and it helps to create a new asset.

`updateAsset` taking asset ID (which has to be updated) as an argument and updates it according its identifier.

`createBuyNow` this function call helps to post a buy now order in freeverse second layer.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

`createBuyNowPayment` takes the asset identifier as arguments to allow buy an asset, which is previously put for sale.

`setDropPriority` this function helps the specific flow for the Freeverse marketplace. Allows ordering NFTs in drops view.

`getAuthToken` this function returns an authentication token from Freeverse API.

`updateAssetBulkByAssetId` takes asset IDs, properties, metadata and nonce as arguments for updating a bulk set of assets.

Module *REST APIs for services* to be consumed by the marketplace and the common good curator control panel.

GET `https://<base_uri>/api/v1/asset` it returns NFT assets from Freeverse.

POST `https://<base_uri>/api/v1/asset` this call helps to create a new NFT asset.

POST `https://<base_uri>/api/v1/asset/bulk/update` it helps to bulk update asset operation. This endpoint is useful to bulk update desired attributes on a range of assets at the same time

GET `https://<base_uri>/api/v1/asset/asset_id` this API call returns an NFT Asset from Freeverse filtering by asset id.

PUT `https://<base_uri>/api/v1/asset/asset_id` helps to update an NFT asset.

POST `https://<base_uri>/api/v1/asset/asset_id/buyNow` this API call helps to buy an asset that is on sales state.

POST `https://<base_uri>/api/v1/asset/asset_id/drop` this API call creates a specific flow for the Freeverse marketplace. In that platform there a Drops tab so Freeverse use the priority field to order the assets in that tab.

GET `https://<base_uri>/api/v1/asset/asset_id/plantLocation` it helps to get plant location real data from Plants for the Planet data endpoint.

GET `https://<base_uri>/api/v1/asset/asset_id/project` this API call corresponds with the getting real data from Plants for the Planet data endpoint.

POST `https://<base_uri>/api/v1/asset/asset_id/putForSale` for successful operation of this API, it helps to put an asset for sale for a certain amount of time, and the first buyer to pay acquires the asset.

POST `https://<base_uri>/api/v1/proposal` this API call helps to post the new proposal. Mainly this API is considered as the end point for governance proposal creation.

PUT `https://<base_uri>/api/v1/proposal` this API call corresponds with the governance proposal sponsoring, voring and processing endpoint (depending on the action specified in the body request).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

GET `https://<base_uri>/api/v1/asset/asset_id/project` for successful operation of this API call, it returns project real data from plants for the planet data endpoint.

GET `https://<base_uri>/api/v1/member/member_id` this API call helps to fetch the governance member information and status.

2.2.17 PXC

Description PolyCrypt (PXC) contributes to the overall objective of the ONTOCHAIN project to create a software ecosystem for the next generation Internet/social networks and for vital sectors of the European economy. It provides a secure, scalable and open cross-chain layer which is ready to connect ONTOCHAIN with any existing and future blockchain systems. PolyCrypt comprises of four different architectural components: 1. Blockchain infrastructure; 2. Server components; 3. Frontend; and 4. Software development kit.

Application Program Interfaces In essence, SDK provides a state channel network client, to interact with a state channels network. Once the client has been set up, it can be used to propose channels to other network peers, accept channel proposals, send updates on those channels and eventually settle them. The SDK offers a holistic function set which is growing continuously. Notably, the package client contains the Perun State Channel network protocol implementation. It provides a Client that exposes the central API to communicate with a channel network. The Client provides Channel controllers to interact with individual channels. In the below, those function calls have been given:

`NewChainNotReachableError` constructs a ChainNotReachableError and wraps it with the actual error message.

`NewTxTimedoutError` by taking the transacrtion ID and type as the arguments, this function call constructs a TxTimedoutError and wraps it with the actual error message.

`AdjudicatorEventHandler` represents an interface for handling adjudicator events.

`BaseChannelProposal` contains all data necessary to propose a new channel to a given set of peers. It is also sent over the wire. BaseChannelProposal implements the channel proposal messages from the Multi-Party Channel Proposal Protocol (MPCPP).

`BaseChannelProposalAcc` contains all data for a response to a channel proposal message. The ProposalID must correspond to the channel proposal request one wishes to respond to. Participant should be a participant address just for this channel instantiation. The type implements the channel proposal response messages from the Multi-Party Channel Proposal Protocol (MPCPP).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

`ChainNotReachableError` indicates problems in connecting to the blockchain network when trying to do on-chain transactions or reading from the blockchain.

`Channel` is the channel controller, progressing the channel state machine and executing the channel update and dispute protocols. Currently, only the two-party protocol is fully implemented.

`ChannelMsg` this function call returns all messages that can be routed to a particular channel controller.

Besides these function calls many others APIs have been developed for different function calls which are responsible for accessing different module of these projects. More details about SDK APIs for PXC can be found at this link: https://pkg.go.dev/perun.network/go-perun@v0.9.0/client#pkg-overview.

2.2.18 GEONTOLOGY

Description GEONTOLOGY aims to develop a network protocol that promotes regulation of data exchange, trust, consent management, reputation and security as contribution for the emerging Data Economy Ecosystem built on Blockchain technology. In details, GEONTOLOGY proposes an innovative protocol called Proof of Offset (POO) to enable a higher control and limit data access by geo-location, accountability, data exposition minimization, data semantic annotation that guarantee cross-domain data re-use and higher awareness about data protection. POO algorithm will be provided a deterministic mechanism to validate the geo-location from nodes requesting data in order to validate its legal jurisdiction. In addition, POO algorithm will be specialised to detect the use of relay nodes such as proxies and other entities that could manipulate the network location (IP address).

GEONTOLOGY aims to create significant evidence of the benefits of the proposed protocol. as part of an emerging data economy for the safety data reuse, accountability and trust via new models that guarantees the satisfaction of rules, contracts and agreements in terms of quantity (accountability), geo-location (legal constraints beyond European Union and Swiss borders), and data usage (data minimization and rights to revoke permissions) that enable and promote new collaboration models driven by users acceptance, compliance with new regulation frameworks, community-driven reputation schemes and integral data management. Finally, GEONTOLOGY has defined a methodology that guarantees the development of prototypes and demonstrators that could be valorised by industry. The results will be validated over a large-scale geo-distributed and interconnected testbed offered by ONTOCHAIN, PlanetLab/Fed4FIRE and/or public cloud, in order to demonstrate how it works over Internet and also over Internet Agnostic Networks.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Geontology brings several services and nodes that interact between them coordinated by a core component called Geo-Localization Service in charge of distributing the tasks and analysing the result provided by the nodes.

- Client: node of the network whose geolocation is to be determined by the system. Usually, it will be a client trying to access a web service, but it will be also developed the approach of a LwM2M IoT device trying to connect to a server to a bootstrap server.
- Service: represents the web service where the client wants to access, it can be a web service or an IoT server.
- Geo-Localization Service: that makes available the necessary geolocation data of clients and the entire network infrastructure to obtain said information.
- Nodes: connection point of the network that is part of the infrastructure provided by the Geo-location Service and that will try to establish connections with the client that is given by the service.

Application Program Interfaces Notably no SDK has been developed for the users' interaction; however services are established over HTTP. This service, for this explanations sake, hosted in the https://app.swaggerhub.com/apis/CTORRALBA/GEONTOLOGY_ORCHESTRATOR/1.0.0#/.

POST `/geontology/api/monitor/registration` once this request is received, the orchestrator executes the 2 phases of POO algorithm to verify the requester geolocation. Once verified, it will either allow or deny the registration depending on the final conclusion on its geolocation. If allowed, this user will be considered a monitor and will take part in future verification for new coming users.

POST `/geontology/api/monitor/validateRegistration` once the monitor has finished the training process, it will send its computed coefficients to the orchestrator so that they can be stored.

DELETE `/geontology/api/monitor/disconnect` a request from a monitor to disconnect itself from the Geontology service as a monitor.

POST `/geontology/api/monitor/geolocateData` as phase 1 of POO algorithm focuses on geolocating an objective, a verification for this objective is created and a selected set of subscribed monitors will have to ping the objective and post the results in the service for this verification. These results are the Proof of Offset.

POST `/geontology/api/monitor/verify` the monitor chosen as a verificator for the specified verification posts its results in regards to POO's phase 2 to the orchestrator.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

POST `/geontology/api/objective/registration` a client who wishes to have their geolocation verified must register in the GEONTOLOGY service as an objective in the first place.

GET `/geontology/api/objective/initPhaseOne` once an objective is registered in the service, it is ready to have its geolocation verified staring with phase 1 of the algorithm, which is focused on estimating a geolocation for the objective by applying a multilateration algorithm to the latencies measured by the monitors.

GET `/geontology/api/objective/initPhaseTwo` once phase 1 has ended, a closest monitor to the geoposition estimated in phase 1 is choosen as the verificator for this verification and is given a set of IPs called bots from the estimated country to ping. This same set of IPs is sent to the objective to ping them as well.

2.2.19 DKG

Description The DKG Knowledge Tool Stack developed within the Ontochain project is bringing novelty in easy access to the market of existing knowledge and data resources, by enabling

- Provisioning unique cryptographic tokens for each knowledge asset to facilitate knowledge exchange
- Enabling a knowledge wallet system, designed to easily interface with existing data stores and enable secure and trusted data sharing
- Knowledge marketplace tooling, enabling anyone to build domain specific, yet interoperable knowledge marketplaces
- Knowledge marketplace tooling, enabling anyone to build domain specific, yet interoperable knowledge marketplaces

The tools are intended to interoperate and several of the illustrated functionalities can be implemented by utilizing multiple of the tools in concert. However, for the purpose of clarity we introduce each of the tools independently. The DKG knowledge tool stack is developed based on the OriginTrail Decentralized Knowledge Graph and it is composed of three main components: 1. Knowledge Marketplace; 2. Knowledge Tenders; and 3. Knowledge Wallet.

Application Program Interfaces

Module *APIs for SDK:* Knowledge tokens implement a blockchain-based SDK API communication. Third-party components such as DEX, general smart contracts, Fair swap protocol implementation, and the DKG are not presented in the section.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Knowledge tokens presents blockchain-based module that implements decentralized exchange.

`mintTokens` this function mints the tokens with provided amount, type, and symbol. For both fungible and non-fungible tokens, the sender becomes the owner of the tokens.

`transfer` this function is a proxy function for which helps to transfer tokens.

`approve` this proxy function stands for approval.

`totalSupply` this function call returns total supply of a kToken.

`balanceOf` this proxy function stands for both ERC20 and ERC721 tokens.

Module *REST APIs For Services:* Knowledge wallets, marketplaces, tenders implement REST API communication. The following is a list of the components with REST API definitions.

Module *Knowledge marketplace (kMarket):* Knowledge marketplace presents hosted interface, implementing knowledge services that interact with underlying kMarket smart contracts and DKG. The idea is that this interface can be used by anyone as it utilizes decentralized networks as it's backend.

POST `http://localhost/advertise` taking unique identifier of an asset, resolves the UAL on the network and optionally stores assets metadata locally.

POST `http://localhost/search` retrieves all UALs associated with the keyword from the network and optionally load the assets from the local cache. Optionally will be used for more generic queries.

POST `http://localhost/withdraw` taking UAL as argument, this function call withdraw tokens acquired through sale (by the seller).

POST `http://localhost/complain` by taking purchase ID and misbehavior information, this function call initiates misbehaviour procedure.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Module *Knowledge tenders (kTender):* Knowledge tenders interfaces presents hosted interface that implement tender services, that interact with underlying kTender smart contracts and DKG. The idea is that this interface can be used by anyone as it utilizes decentralized networks as it's backend.

POST `http://localhost/advertise` taking unique ID for an asset and price for data submission related information; this function call resolves the UAL on the network and store assets metadata locally.

POST `http://localhost/search` taking keyword associated of an asset and price for data submission related information; this function retrieves all UALs associated with the keyword from the network and local cache that matches the price.

POST `http://localhost/submit` this function call retrieves UAL associated with the keyword from the network and stores it locally.

POST `http://localhost/withdraw` taking UAL as an argument this function call helps to withdraw tokens acquired through sale (by the buyer).

POST `http://localhost/complain` this function call initiates misbehaviour report procedure.

Module *Knowledge wallets (kWallet):* Knowledge wallet presents hosted interface that performs exchange data for tokens.

POST `http://localhost/create` by taking JSON-LD data as argument; this function call creates new asset graph on the DKG.

POST `http://localhost/update` this function call helps to update an asset graph state on the DKG.

POST `http://localhost/initiatePurchase` taking asset's identifier and purchase price information; this function call initiates purchase according to the fair swap.

GET `http://localhost/purchase` this call returns status of a purchase.

POST `http://localhost/tokenize` this function call is working like as an interface for tokenization (calls mintTokens of the kTokens contract and associates with UAL).

2.2.20 PS-SDA

Description The PS-SDA project focuses on enhancing the auditability and provenance of any personal data exchange transaction to strengthen trust and transparency in the digital economy. It helps organisations to leverage personal data while being transparent and legitimate in their data usage. It also empowers individuals to control

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

how their data is used and exchanged. The DEXA functions are available as microservices that can be plugged into existing systems, such as in iGrant.io SSI data exchange workflows. The core components are exposed as RESTFul APIs. The components can exist independently in any service provider agreement handling system. Espcially the Data Exchange Agreements (DEXA)

Application Program Interfaces For the PS-SDA project, no SDKs are required for the DDA implementation. Existing DA SDKs will be reused and be revised to support new flows. On the other hand, PS-SDA provides different service APIs which are classified as:

Module *Individuals*

GET `/individuals/data-agreements` this function call helps to view signed DAs.

GET `/individuals/data-using-service` it helps to view the shared data using services for a given organisation.

GET `/individuals/data-disclosure-agreements` it helps to view the DDA(s) in an organisation.

GET `/individuals/data-agreements/data_agreement_id/provenance_trail` this function calls helps to fetch provenance trail for a DA.

Module *Organisations (DS and DUS)*

POST `/organisation/data-disclosure-agreement` this function call helps to create DDA template.

GET `/organisation/data-disclosure-agreement` it list all published DDAs.

PUT `/organisation/data-disclosure-agreement/data_disclosure_agreement_id` Update signed DDA by ID.

DELETE `/organisation/data-disclosure-agreement/data_disclosure_agreement_id` this call helps to delete signed DDA by ID.

GET `/organisation/data-agreements` View signed DAs (Org. copy).

GET `/organisation/data-agreements/data_agreement_id/provenance_trail` it fetch provenance trail for a DA.

POST `/organisation/data-disclosure-agreements/data_disclosure_agreement_id/organisati` it offer a DDA to an organisation.

POST `/organisation/data-disclosure-agreements/data_disclosure_agreement_instance_id/accept` it helps to accept a DDA sent by an organisation.

POST `/organisation/data-disclosure-agreements/data_disclosure_agreement_instance_id/reject` this call helps to reject a DDA sent by an organisation.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

POST `/organisation/data-disclosure-agreements/data_disclosure_agreement_instance_id/terminate` it terminate a DDA sent by an organisation.

POST `/organisation/data-agreements/data_agreement_id/auditor/auditor_id/request-verify` it request verification of a DA instance by a third party auditor.

POST `/organisation/audit-requests` this call query audit requests sent.

Module *Auditors*

POST `/organisation/audit-requests` this function call query audit requests received.

POST `/auditor/audit-requests/audit_request_id/verify` this call Verify the digital signatures in DA.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

3 ONTOCHAIN GATEWAY API

This section describes the ONTOCHAIN Gateway API and its functionalities. The Gateway API will be the single-entry point for developers and users of ONTOCHAIN services. It is composed of three modules:

1. **Service Discovery** lets programs search search deployed ONTOCHAIN services with structured queries and access them directly;
2. **Accounts Management** provides functions related to user accounts and access rights:
3. **Data Storage** lets users and program upload data that can then be used by ONTOCHAIN services, and download certain data that has been produced by ONTOCHAIN services.

Each of these modules are described in the rest of this section, along with the details of the API functions they provide. These APIs where were defined by the ONTOCHAIN consortium and were part of the Open Call 3 material; they will be implemented by the BABELFISH project which was selected in OC3.

3.1 SERVICE DISCOVERY

In the third year of the project, ONTOCHAIN will integrate the several of the projects that were funded through OC1 and OC2 (see Section 2.2), which represents between 25 and 30 services to deploy on the pilot network by the end of 2023. The exploitation of ONTOCHAIN's results will bring even more services in the future, driving the need for machine-to-machine discovery mechanisms.

In this section we introduce the ONTOCHAIN Services catalog, which will service as a single entry point to accessing ONTOCHAIN services. The catalog will provide search mechanisms and return pointers to the service implementations directly to the clients, in the form of API endpoints when available. The search features will support advanced ontology-based queries so that consumer applications can search for services based on functionalities, implement their own selection algorithm and start consuming the services without requiring specific human interactions.

> **Module *Services catalog*** This module will provide a programmable way of discovering the services provided through the ONTOCHAIN ecosystem. Access to the functions will be authenticated and regular users will only have access to read operations. Write operations will be accessible only to administrators, unless noted in

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

the function's description.

GET `<base_uri>/list/count/page?filter=JSON` Obtain a list of all the available services as a JSON string. Arguments `count` (max. 100) and `page` are used to control the number of results. An optional argument can be provided to filter the types of desired services; the argument is a url-encoded JSON document that can contain every field of the resource description schema. Returns a JSON document with the desired result count (or less) and HTTP code 200 in case of success, HTTP 400 if the provided filter cannot be decoded, and code 401 if the user is not authenticated.

GET `<base_uri>/service/search?query=QUERY` Obtain a list of services which name or description match the provided search terms. The `QUERY` argument can contain one or several url-encoded terms. Returns a JSON document with matching project descriptions and HTTP code 200 in case of success, HTTP 400 if `QUERY` is absent, and code 401 if the user is not authenticated.

GET `<base_uri>/service/SERVICE_ID` Obtain the detailed description of a service as a JSON string. The argument is the service identifier that can be obtained with the `/list` function. Returns a JSON document and HTTP code 200 in case of success, HTTP 404 if the requested service cannot be found, and code 401 if the user is not authenticated.

POST `<base_uri>/service/` Add a new service to the catalog. The request body must contain a JSON document containing the complete service description. Returns HTTP code 200 in case of success, HTTP 400 in case the description has an incorrect format, and HTTP 401 if the user is not authenticated or does not have the required privileges.

PUT `<base_uri>/service/SERVICE_ID` Update the details of a service already stored in the catalog. The function takes the identifier of the service that must be updated as an argument and the request body must contain a JSON document containing the new service description. Returns HTTP code 200 in case of success, 400 in case the description has an incorrect format, 404 if the requested service does not exist, and code 401 if the user is not authenticated or does not have the required privileges.

DELETE `<base_uri>/service/SERVICE_ID` Removes the service pointed by argument `SERVICE_ID` from the catalog. The service is not actually removed, it will only be marked as deleted and it will no longer be part of the results of the `/list` and `/search` functions.

3.2 ORGANIZATION AND USER ACCOUNTS

Users in the ONTOCHAIN ecosystem can be either consumers or providers of services, or both. Users can be attached to an organization or be registered independently. The goal if this membership model is both to drive engagement, by providing ONTOCHAIN

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

with specific features (e.g. a user portal) and to manage accounting, i.e. payment for services and crypto-wallets.

Module *Accounts management* This module provides interfaces for creating users and organizations, and basic interactions with wallets. All API calls must be authenticated and write operations are accessible only to administrators, unless specifically noted.

POST `<base_uri>/organization/` Create a new organization with no users. The request body must contain the details of the organization as a JSON document (e.g. its name, short description, contact address). The call returns the identifier of the new organization and HTTP code 200 in case or success. The call can also return HTTP code 400 if the organization description is not a valid JSON document or if required fields are missing, and code 401 if the user is not authenticated.

PUT `<base_uri>/organization/ORGANIZATION_ID` Updates the information related to an existing organization. The identifier of the organization must be provided in the query string, and the request body must contain the new details of the organization as a JSON document. The call returns HTTP code 200 in case or success, code 400 if the organization description is not a valid JSON document or if required fields are missing, 404 if the organization does not exist, and code 401 if the user is not authenticated.

GET `<base_uri>/organization/ORGANIZATION_ID` Obtain the details related to an existing organization. The call takes the identifier of the organization as an argument and returns a JSON document containing the organization's information. The call returns HTTP error code 200 in case of success, 404 if the organization does not exist, and 401 if the user is not authenticated.

GET `<base_uri>/organization/ORGANIZATION_ID/list` Obtain the list of users that are members of an organization. The identifier of the organization is taken as the sole argument to the call. The call returns JSON array of objects describing the users. In case of success, the call return HTTP code 200; the call can also return code 404 if the organization cannot be found, and 401 if the user is not authenticated.

DELETE `<base_uri>/organization/ORGANIZATION_ID` Request the deletion of an organization. The organization will be market as deleted but no data will actually be removed, in order to preserve the history of the platform. If the organization contains users, the users will not be removed but will appear as independent. The only argument to the call is the identifier of the organization. In case of success, the call returns HTTP code 200. The call can also return code 404 if the organization does not exist and code 401 if the user is not authenticated or does not have the required privileges.

POST `<base_uri>/user/` Create a new user. The request body must contain the details of the user as a JSON document (e.g. its name, organization, email address,

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

phone number). The call returns the identifier of the new user and HTTP code 200 in case or success. The call can also return HTTP code 400 if the user description is not a valid JSON document or if required fields are missing.

PUT `<base_uri>/user/USER_ID` Updates the information related to an existing user. The identifier of the user must be provided in the query string, and the request body must contain the new details of the user as a JSON document. The call returns HTTP code 200 in case or success, code 400 if the user description is not a valid JSON document or if required fields are missing, 404 if the user does not exist, and code 401 if the requesting user is not authenticated.

GET `<base_uri>/user/USER_ID` Obtain the details related to an existing user. The call takes the identifier of the user as an argument and returns a JSON document containing the user's information. The call returns HTTP error code 200 in case of success, 404 if the user does not exist, and 401 if the requesting user is not authenticated.

DELETE `<base_uri>/user/USER_ID` Request the deletion of a user. The user will be market as deleted and all of associated information (except its identifier, in order to preserve the history of the platform) will be removed. The only argument to the call is the identifier of the user. In case of success, the call returns HTTP code 200. The call can also return code 404 if the user does not exist and code 401 if the requesting user is not authenticated or does not have the required privileges.

GET `<base_uri>/user/USERNAME/wallet` Obtain the details (chain, address, balance) of the user's wallets in the ONTOCHAIN ecosystem. The call takes the identifier of the user as an argument and returns the wallets information as a JSON document along with HTTP code 200 in case of success. The call can also return code 404 if the user does not exist or has been deleted, and code 401 if the requesting user is not authenticated or does not have the required privileges.

3.3 STORAGE SERVICE RESULTS/OUTPUT

Data storage is a special kind of service in the way they provide a communication medium between other services. In order to facilitate these interactions, the Data storage module provides high-level interfaces for storing and retrieving files and arbitrary blobs of data to and from providers, including external storage providers such as Amazon S3[15], Dropbox[16] and IPFS[17]. The underlying storage providers will be selected in the last year of the project, based on the requirements of the applications selected in OC3.

[15] https://aws.amazon.com/s3/
[16] https://www.dropbox.com/
[17] https://ipfs.io/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Module *Data storage* The module provides users and applications with high-level interfaces that are common to all of the supported backend storage services, both members of the ONTOCHAIN ecosystem, and external services. Storage providers can be searched in the Service catalog like any other service (see Section 3.1).

POST `<base_uri>/collection/` Create a new collection of objects, similar to a directory in a file system. The call takes a description of the collection (including the storage provider) in the JSON format as an argument. The call only creates a record for the collection, it does not store any data object. In case of success, the call returns the identifier of the new collection and HTTP code 200. In case of failure, the call returns code 400 (if the JSON document cannot be interpreted) and 401 if the user is not authenticated or does not have the required privileges.

PUT `<base_uri>/collection/COLLECTION_ID` Updates the information related to an existing collection of objects. The identifier of the collection must be provided in the query string, and the request body must contain the new details of the collection as a JSON document. The call returns HTTP code 200 in case or success, code 400 if the collection description is not a valid JSON document or if required fields are missing, 404 if the collection of object does not exist, and code 401 for unauthorized access.

GET `<base_uri>/collection/COLLECTION_ID` Obtain the details related to an existing collection of objects. The call takes the identifier of the collection of object as an argument and returns a JSON document containing the information of corresponding collection of objects. The call returns HTTP error code 200 in case of success, 404 if the collection of object does not exist, and 401 for the unauthenticated access request.

GET `<base_uri>/collection/list` Obtain the list of collection of objects. The call returns JSON array of objects describing the collection of objects. In case of success, the call return HTTP code 200; the call can also return code 404 if the collection of objects cannot be found, and 401 for unauthenticated access request.

DELETE `<base_uri>/collection/COLLECTION_ID` Request the deletion of a collection of object. The collection of object will be market as deleted but no data will actually be removed, in order to preserve the history of the platform. If the collection of object contains users, the users' information will not be removed but will appear as independent. The only argument to the call is the identifier of the collection. In case of success, the call returns HTTP code 200. The call can also return code 404 if the collection ID does not exist and code 401 if the requester (user) is not authenticated or does not have the required privileges.

POST `<base_uri>/object/` Create a new object. The call takes a description of the collection (including the storage provider) in the JSON format as an argument. The call only creates a record for the object. In case of success, the call returns the identifier of the new object and HTTP code 200. In case of failure, the call returns code

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

400 (if the JSON document cannot be interpreted) and 401 if the user is not authenticated or does not have the required privileges.

PUT `<base_uri>/object/OBJECT_ID` Updates the information related to an existing objects. The identifier of the object must be provided in the query string, and the request body must contain the new details of the object as a JSON document. The call returns HTTP code 200 in case or success, code 400 if the object description is not a valid JSON document or if required fields are missing, 404 if the object does not exist, and code 401 for unauthorized access.

GET `<base_uri>/object/OBJECT_ID` Obtain the details related to an existing objects. The call takes the identifier of the object as an argument and returns a JSON document containing the information of corresponding object. The call returns HTTP error code 200 in case of success, 404 if the object does not exist, and 401 for the unauthenticated access request.

DELETE `<base_uri>/object/OBJECT_ID` Request the deletion of a object. The object will be market as deleted but no data will actually be removed, in order to preserve the history of the platform. If the object contains users, the users' information will not be removed but will appear as independent. The only argument to the call is the identifier of the object. In case of success, the call returns HTTP code 200. The call can also return code 404 if the object ID does not exist and code 401 if the requester (user) is not authenticated or does not have the required privileges.

GET `<base_uri>/object/OBJECT_ID/read` Obtain the details related to an existing objects. The call takes the identifier of the object as an argument and returns information of corresponding object. The call returns HTTP error code 200 in case of success, 404 if the object does not exist, and 401 for the unauthenticated access request.

PUT `<base_uri>/object/OBJECT_ID/write` Updates the information related to an existing objects. The identifier of the object must be provided in the query string, and the request body must contain the new details of the object as a document. The call returns HTTP code 200 in case or success, code 400 if the object description is not a valid or if required fields are missing, 404 if the collection of object does not exist, and code 401 for unauthorized access.

POST `base_uri>/object/OBJECT_ID/USER_ID` Checks the access control for a particular user to access a particular object. If the user has the access grant for the accessing the object it returns HTTP code 200. Returns code 400 if the object description is not a valid or if required fields are missing, 404 if the object does not exist, and code 401 for unauthorized access

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

4 THE ONTOCHAIN PILOT NETWORK

Today, blockchain-based systems face many challenges related to scalability, privacy, security, and various other aspects. To solve these issues, many researchers have proposed different solutions. Among all these existing solutions, the sidechain concept [9] solves many existing blockchain issues and opens many new doors for using the blockchain in different application domains. Therefore, considering that fact, using the sidechain concept is beneficial for our project. Thus, we have adopted this concept for developing the pilot use-cases. Mainly, sidechains are secondary blockchains; they are connected with the other blockchain using the two-way peg methodology. Notably, the two-peg mechanism mainly enables the functionalities for bidirectional transfer of assets between the mainchain and sidechain at a fixed or pre-deterministic exchange rate. Remaining of this section, focusing on the two-peg mechanism, we are going to explain the overall deployment strategy for the pilot use-cases thoroughly.

The choice of relying on a sidechain over layer-2 solutions such as rollups is due to the lack of maturity of layer-2 chains at this time. However, the decision of relying on the EVM for the main ONTOCHAIN chain will ensure that upcoming updates to the Ethereum protocol and implementation (Ethereum 2.0) which are due to be deployed in 2022 and 2023 will benefit past, current and future ONTOCHAIN services and applications. Indeed, since almost all ONTOCHAIN services sit at the application layer, they will remain compatible with Ethereum updates (to the exception of GraphChain which relies on a modified Hyperledger Besu client which may require a small update in order to support Ethereum 2.0).

4.1 FOUNDATION OF PILOT NETWORK

The ONTOCHAIN network is based on the Ethereum Virtual Machines (EVMs) but not on the Ethereum network for various reasons. To begin, the consortium decided to build its blockchain-based pilot to test and validate various applications within the ONTOCHAIN framework to fulfil the ONTOCHAIN's vision and pre-setted objectives. The MoU has been signed between different consortium partners (i.e., iExec & IntelliSemantic, iExec & Athens University of Economics and Business, iExec & University of Ljubljana) for this purpose, and the SLAs have been defined for developing the ONTOCHAIN pilots. However, it is critical to determine the pilot's effective and efficient design before developing pilots. Notably, pilot designs' effectiveness and efficiency depend on the consensus protocol chosen, the number of blockchain nodes deployed, and the adoption of prominent governance policies. Furthermore, for the ONTOCHAIN framework, energy consumption is one of the critical points which has been considered before developing the pilots. Therefore, in the ONTOCHAIN ecosystem, we have developed our blockchain network based on the iExec provided Bellecour sidechain, considering the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Proof-of-Authority (PoA); more precisely, authorityRound (AuRa) consensus algorithm. In past studies, it has been shown that the PoA-based blockchain network consumes less energy for making the transactions compared to Proof-of-Work (PoW) or Proof-of-Stakes (PoS) or any other consensus-based blockchain network [10]. In the following section, we are going to present a more deep insight view of the pilot deployment.

4.2 STATUS OF THE ONTOCHAIN NETWORK

During this stage of the ONTOCHAIN, the consortium has decided to decentralize the iExec sidechain for developing the pilot. Therefore, three new validator nodes have been deployed at Athens University of Economics and Business, IntelliSemantic, and University of Ljubljana. Three new full nodes have also been deployed along with these validator nodes. Besides that, a bridge node will be deployed University of Ljubljana and soon will be added to iExec sidechain. Thus all the nodes are geographically scattered over the different region in Europe. The details specification of those nodes have been presented in tables 1 & 2, below. As the pilot has been developed by decentralizing the iExec Bellecour sidechain, therefore the new validator and full nodes are not responsible for performing the transactions, therefore they have not directly perform any transaction related tasks. Currently, all the transactions in iExec Bellecour is performing by some other dedicated validator and full nodes. In order to achieve the fully decentralized blockchain network, in future our plan is to extend the functionalities for each validator and full nodes and allow them to perform any transaction related tasks.

4.3 DEVELOPER'S LIABILITIES

Blockchain technology has advanced significantly in recent years. Blockchain has the potential to become a key technology of digital transformation as B2B, and B2C businesses increasingly shift to a digital market [11]. In the anonymous world of cross-border digital connectivity, it fosters trust and security for consumers, customers, trade and business partners. However, many risks [12] are associated with any blockchain-based ecosystem, which must be addressed before utilizing the blockchain technology in various verticle business domains (e.g., health, education, finance and Governmental). For that purpose, many European countries have already taken many resolutions and passed various laws both on the National and European levels [13]. The main objectives of adopting those laws are to provide proper guidance for attracting private-sector investors, ensure consumer protection and citizens rights, and provide safeguards against anticompetitive practices [13]. However, to make a sustainable, robust and more secure Blockchain ecosystem, it is necessary to define the strong policies for the Blockchain and Smart Contract developer [14]. Meanwhile, those policies could implicitly help define the roles and responsibilities of different participants of the

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Validator Node Info	University of Ljubljana	IntelliSemantic	Athens University of Economics and Business
Number of deployed node	1	1	1
Hosted over Physi-cal/Virtual platform	Cloud: The Academic and Research Network of Slovenia (ARNES)	Cloud: Linode provider	On-Premise
Month of deploy-ment	April 2022	May 2022	April 2022
Up-Time for Validator	6 months (100%)	5 monthsc(100%)	8 months (100%)
Hardware description	8GB RAM/100GB Storage/4 CPU cores	8 GB RAM/160 GB Storage/4 CPU cores	16GB RAM/500GB Storage (SSD)/ 4 CPU cores
Software dependen-cies	Docker and Docker-Compose, Veracrypt, Keys generation Dapp, PoA Smart Contracts	Docker and Docker-Compose, Veracrypt, Keys generation Dapp, PoA Smart Contracts	Docker and Docker-Compose, Veracrypt, Keys generation Dapp, PoA Smart Contracts
Number of Blocks Validated (mined)	362879	280809	310869
Number of Transac-tion executed	0	0	0

TABLE 1: VALIDATOR NODES RELATED INFORMATION.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Full Node Info	University of Ljubljana	IntelliSemantic	Athens University of Economics and Business
Number of deployed node	1	1	1
Hosted over Physi-cal/Virtual platform	Cloud: The Academic and Research Network of Slovenia (ARNES)	Cloud: Linode provider	On-Premise
Month of deploy-ment	February 2022	February 2022	April 2022
Up-Time for Full node	10 months (100%)	7.5 months (100%)	9 months (100%)
Hardware description	16GB RAM/ 80GB Storage/ 4 CPU cores	8 GB RAM/160 GB Storage/4 CPU cores	16GB RAM/1TB Storage (SSD)/ 4 CPU cores
Software dependen-cies	Docker and Docker-Compose, PoA, NetStat	Docker and Docker-Compose, PoA, NetStat	Docker and Docker-Compose, PoA, NetStat

TABLE 2: FULL NODES RELATED INFORMATION.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

Blockchain ecosystem [13]. In this section, we mainly focus on presenting the roles and responsibilities of Blockchain and Smart Contract developers.

4.3.1 Role and Responsibilities of Blockchain Developer:

The Blockchain developer has to perform several activities to make the ecosystem more sustainable and robust. They have to provide updates and fix the vulnerabilities or bugs upon their detection. They are in charge of optimizing code and developing new functionalities, which can be helpful for the solution provider.

4.3.2 Role and Responsibilities of Infrastructure Provider:

They are mainly responsible for delivering the infrastructure on which the Blockchain network will run. They are also in charge of maintaining, operating and providing security solutions to maintain the network infrastructure's resiliency. They also provide services to solution providers to deploy their applications over the network infrastructure.

4.3.3 Role and Responsibilities of Solution Provider:

The solution provider is often considered the designer of the business solution for the Blockchain ecosystem. They are mainly in charge of writing the business logic code, which the Blockchain developer uses to develop the Blockchain ecosystem. Also, they are the leading strategy provider for defining the token storing methodology and wallet management. In addition, they provide various transaction-related services to the users and end users. The solution provider should be able to demonstrate measures and controls in place to protect the information about the ownership of the assets, critical related information and the real identity of participants.

4.3.4 Smart Contract Developer:

The main challenge for Smart Contract developers is related to validating and auditing the Smart Contract code before its deployment to the Blockchain ecosystem. Therefore it is their responsibility to ensure the involvement of the employees from the business side for testing the Smart Contracts in order to check the fulfilment of their business purposes. Also, they need to properly define the controlling mechanism for automatic transaction and app execution. Also, it is the liability of the Smart Contract developer

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

to build a proper resolution process in case of incidents for the execution of the compromised or malfunctioned Smart Contracts.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

5 CONCLUSION

This document consolidates the outcomes of the twenty funded projects, which have been developed between March 2021 to September 2022. This document will be provided as the guideline material for developing the projects of Open Call #3 participants.

This deliverable thoroughly described the revised layered approach of the ONTOCHAIN framework and different software solutions, which have been developed during the Open Call #1 and Open Call #2. Mainly the structure of three previously envisioned modules (Applications, Ontologies, and Distributed Ledgers) and their functionalities have been modified and tuned to build within the ONTOCHAIN framework. More precisely, the development of the seven funded Open Call #1 and Open Call #2 projects added new functionalities of the ONTOCHAIN framework and helped for developing different components within the ONTOCHAIN framework. Moreover, documenting all the related information of the different components and their interfaces in this deliverable will help the Open Call #3 participants further extend those functionalities and add applications to the ONTOCHAIN framework.

The functions described in this deliverable can be the basis of Call 3 developed applications, as e-commerce, copyright management and data marketplaces.

Finally, we have documented the specifications and requirements for an ONTOCHAIN network supported by some ONTOCHAIN participants for deploying the pilot use-cases/demonstrators, which will be used to have a better insight of the results of this project.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338

REFERENCES

[1] Anthony Simonet-Boulogne et al. *D3.4 FRAMEWORK SPECIFICATION*. 2021. URL: https://ontochain.ngi.eu/sites/default/files/deliverables/D3.4-Framework-specification.pdf.

[2] Dominik Kuziski et al. "D4 Prototype Demonstration- Full Design Specification - GraphChain". In: *Deliverable* (2021).

[3] Roberto García et al. "D4 Prototype Demonstration- Full Design Specification - CopyrightLY". In: *Deliverable* (2021).

[4] Rebecca Johnson and Martin Martin Schaffner. "D4 Prototype Demonstration- Full Design Specification HIBI". In: *Deliverable* (2021).

[5] Caspar Roelofs et al. "D4 Prototype Demonstration- Full Design Specification - OntoSSIVault". In: *Deliverable* (2021).

[6] Junaid Arshad et al. "D4 Prototype Demonstration- Full Design Specification - Reputable". In: *Deliverable* (2021).

[7] Marcel Muller et al. "D4 Prototype Demonstration- Full Design Specification - KnowledgeX". In: *Deliverable* (2021).

[8] Giampaolo Bella et al. "D4 Prototype Demonstration- Full Design Specification POC4COMMERCE". In: *Deliverable* (2021).

[9] Adam Back et al. "Enabling blockchain innovations with pegged sidechains". In: *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains* 72 (2014).

[10] Abigael Okikijesu Bada et al. "Towards a green blockchain: A review of consensus mechanisms and their energy consumption". In: *2021 17th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE. 2021, pp. 503–511.

[11] Financial Conduct Authority. "Discussion Paper on distributed ledger technology". In: *DP17/3 (April 2017)< https://www. fca. org. uk/publication/discussion/dp17-03. pdf* (2017).

[12] Harish Natarajan, Solvej Krause, and Helen Gradstein. "Distributed ledger technology and blockchain". In: (2017).

[13] "Distributed Ledger Technologies & Blockchain - CSSF". In: (2022). URL: https://www.cssf.lu/wp-content/uploads/DLT_WP.pdf.

[14] Jenny Alexandra Triana Casallas, Juan Manuel Cueva-Lovelle, and José Ignacio Rodríguez Molano. "Smart contracts with blockchain in the public sector". In: (2020).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 957338