


Tailored IoT & BigData Sandboxes and Testbeds for Smart,
Autonomous and Personalized Services in the European
Finance and Insurance Services Ecosystem



D4.9 – Permissioned Blockchain for Finance and Insurance - III

Revision Number	3.0
Task Reference	T4.3
Lead Beneficiary	UBI
Responsible	Konstantinos Perakis
Partners	ENG GFT HPE IBM INNOV SIA UBI UNIC
Deliverable Type	Report (R)
Dissemination Level	Public (PU)
Due Date	2021-12-31
Delivered Date	2021-12-22
Internal Reviewers	GFT, LXS
Quality Assurance	INNOV
Acceptance	WP Leader Accepted and Coordinator Accepted
EC Project Officer	Beatrice Plazzotta
Programme	HORIZON 2020 - ICT-11-2018
	This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no 856632

Contributing Partners

Partner Acronym	Role ¹	Author(s) ²
UBI	Lead Beneficiary	Konstantinos Perakis, Dimitris Miltiadou
INNOV	Contributor	John Soldatos, Nikos Kapsoulis, Antonis Litke
IBM	Contributor	Fabiana Fournier, Inna Skarbovsky
GFT	Internal Reviewer	Ernesto Troiano
LXS	Internal Reviewer	Pavlos Kranas
INNOV	Quality Assurance	Filia Filippou

Revision History

Version	Date	Partner(s)	Description
0.1	2021-11-05	UBI	ToC Version
0.2	2021-11-18	UBI	Initial contributions on Section 3 and 5
0.3	2021-11-22	UBI, INNOV, IBM	Contributions on Section 3
0.4	2021-11-29	UBI, INNOV, IBM	Updated contributions on Section 3
0.5	2021-12-03	UBI	Updated contributions on Sections 2, 3, 4, 5 and 6
0.6	2021-12-06	UBI	Finalisation of section 2, 3, 4, 5 and 6
0.7	2021-12-08	UBI	Consolidation of the first version
1.0	2021-12-09	UBI	First Version for Internal Review
1.1	2021-12-15	GFT	Internal Review
1.2	2021-12-15	LXS	Internal Review
2.0	2021-12-20	UBI	Version for Quality Assurance
3.0	2021-12-22	UBI	Version for Submission

¹ Lead Beneficiary, Contributor, Internal Reviewer, Quality Assurance

² Can be left void

Executive Summary

The document at hand, entitled “D4.9 - “Permissioned Blockchain for Finance and Insurance - III”, constitutes the final report of the efforts and the produced outcomes of Task T4.3 “Distributed Ledger Technologies for Decentralized Data Sharing” of WP4. Hence, its main objective is to deliver the updated and final documentation of the INFINITECH Blockchain applications and the INFINITECH Blockchain network that hosts these applications. To this end, the deliverable builds on top of the outcomes and knowledge extracted in the previous iterations, deliverables D4.7 and D4.8, in order to provide the final report of the work performed on M27.

Toward this direction, the scope of the current report can be described in the following axes:

- Present the results of the thorough analysis of the key characteristics and offerings of the blockchain technology, as well as the role of the blockchain technology within the INFINITECH RA. During this analysis the different characteristics offered by the various implementations and different approaches in the blockchain technology were analysed and presented. Through the comprehensive analysis that is performed the role of the blockchain technology within the INFINITECH RA is defined. Although the results remained unchanged from the previous iteration, they are included in the deliverable for coherency reasons.
- Provide the final documentation of the blockchain applications which are implemented within the context of the INFINITECH project. In detail, two INFINITECH Blockchain applications are presented, namely the Consent Management and the Know-Your-Customer (KYC) / Know-Your-Business (KYB) applications. For each application, the business need which is covered and the rationale behind their design and implementation is documented. Furthermore, their final design specifications that includes their internal architecture is presented along with their key offerings, the supported use cases and the corresponding sequence diagrams.
- Document the final technical specifications of the fully functional Consent Management and the Know-Your-Customer (KYC) / Know-Your-Business (KYB) applications. In particular, the implementation details are presented by documenting the implemented services and functions, as well as the interactions between the services along with their source code structure. In addition to this, the offerings and features of the applications are presented by providing a walkthrough from the user’s perspective.
- Document the final design and implementation details of the INFINITECH blockchain network. The core aspects of the INFINITECH blockchain network are presented, which includes the network topology, the nodes and their roles in the network, the deployed services and key blockchain components hosted on each node, as well as their interactions. Although the INFINITECH blockchain network remained unchanged from the previous iteration, it is included in the deliverable for coherency reasons
- Document the list of baseline technologies and tools which are leveraged for the implementation of the blockchain network and applications. This list is composed of a set of dominant open-source software, libraries and frameworks.

The current deliverable provides the required updates and optimisations from the previous iteration and constitutes the final report of the Task 4.3 and concludes the activities of the specific task.

Table of Contents

1	Introduction	8
1.1	Objective of the Deliverable.....	8
1.2	Insights from other Tasks and Deliverables	9
1.3	Structure.....	9
2	The Blockchain technology	11
2.1	An overview of the blockchain technology	11
2.2	The role of blockchain technology in INFINITECH RA	13
3	INFINITECH Blockchain Applications.....	16
3.1	Consent Management.....	17
3.1.1	Business need & Rationale.....	17
3.1.2	Description of the solution	19
3.1.3	Use cases.....	23
3.1.4	Sequence Diagrams.....	31
3.1.5	Implementation of the Consent Management.....	36
3.1.6	Consent Management Application overview	43
3.2	Know Your Customer / Know Your Business.....	57
3.2.1	Business Need & Rationale	57
3.2.2	Description of the solution	57
3.2.3	Use cases.....	61
3.2.4	Sequence Diagrams.....	63
3.2.5	Implementation of the Know Your Customer / Know Your Business.....	65
3.2.6	KYC/KYB Application overview.....	68
3.3	Tokenization	70
3.3.1	Business need & Rationale.....	70
3.3.2	Description of the solution	70
4	The INFINITECH Blockchain Network.....	72
5	Baseline Technologies and Tools	76
6	Conclusions	77
	Appendix A: Literature.....	79

List of Figures

Figure 1: High-level architecture of the Consent Management solution.....	21
Figure 2: Consent Management Data Schema diagram	21
Figure 3: Use Case CMS-1 sequence diagram (customer).....	31
Figure 4: Use Case CMS-1 sequence diagram (external financial institution)	31
Figure 5: Use Case CMS-2 sequence diagram	32

Figure 6: Use Case CMS-3 sequence diagram	32
Figure 7: Use Case CMS-4 sequence diagram	33
Figure 8: Use Case CMS-5 sequence diagram	33
Figure 9: Use Case CMS-6 sequence diagram (update).....	34
Figure 10: Use Case CMS-6 sequence diagram (withdrawal).....	34
Figure 11: Use Case CMS-7 sequence diagram	35
Figure 12: Use Case CMS-8 sequence diagram	35
Figure 13: Use Case CMS-9 sequence diagram	36
Figure 14: Consent Management chaincode structure	37
Figure 15: Consent Management System source code structure	40
Figure 16: Consent Management System – List of consents (customer view)	44
Figure 17: Consent Management System – List of consents (financial institution’s view)	45
Figure 18: Consent Management System – New consent request	47
Figure 19: Consent Management System – New request pre-approval or rejection	48
Figure 20: Consent Management System – Customer review and approval	49
Figure 21: Consent Management System – External Financial Institution approval	50
Figure 22: Consent Management System – Update consent	51
Figure 23: Consent Management System – Withdraw consent.....	53
Figure 24: Consent Management System – Expired consents	54
Figure 25: Consent Management System – Consent complete history	55
Figure 26: Consent Management System – Access Control and Search	56
Figure 27: High-level architecture of the KYC/KYB solution	59
Figure 28: KYC/KYB Data Schema diagram	60
Figure 29: Use Case KYC/KYB-1 sequence diagram	63
Figure 30: Use Case KYC/KYB-2 sequence diagram	64
Figure 31: Use Case. KYC/KYB-3 sequence diagram	65
Figure 32: KYC/KYB Chaincode structure.....	65
Figure 33: KYC/KYB Application code structure	67
Figure 34: Clients view	68
Figure 35: New Client View	69
Figure 36: Permissions View	69
Figure 37: The final INFINITECH Blockchain network	74

List of Tables

Table 1: Consent Management Data Schema details (1)	22
Table 2: Consent Management Data Schema details (2)	22
Table 3: Consent Management Data Schema details (3)	23
Table 4: Consent Management Data Schema details (4)	23
Table 5: Consent Management Use Case CMS-1	23
Table 6: Consent Management Use Case CMS-1 (2)	24
Table 7: Consent Management Use Case CMS-2	25
Table 8: Consent Management Use Case CMS-3	25
Table 9: Consent Management Use Case CMS-4	26
Table 10: Consent Management Use Case CMS-5	27
Table 11: Consent Management Use Case CMS-6	28
Table 12: Consent Management Use Case CMS-7	29
Table 13: Consent Management Use Case CMS-8	29
Table 14: Consent Management Use Case CMS-9	30
Table 15: KYC/KYB Data Schema details (1)	60
Table 16: KYC/KYB Data Schema details (2)	60

Table 17: KYC/KYB Use Case KYC/KYB-1	61
Table 18: KYC/KYB Use Case KYC/KYB-2	61
Table 19: KYC/KYB Use Case KYC/KYB-3	62
Table 20: INFINITECH Blockchain network details	75
Table 21: Baseline Technologies and Tools	76
Table 22: Conclusions (TASK Objectives with Deliverable achievements)	77
Table 23: Conclusions – (map TASK KPI with Deliverable achievements).....	78

Abbreviations/Acronyms

Abbreviation	Definition
AES	Advanced Encryption Standard
AI	Artificial Intelligence
AML	Anti-Money Laundering
API	Application Programming Interface
BFT	Byzantine Fault-Tolerant
CA	Certificate Authority
CFT	Crash Fault-Tolerant
CRUD	Create Read Update Delete
DLT	Distributed Ledger Technology
DoA	Description of Action
GDPR	General Data Protection Regulation
IoT	Internet of Things
KYB	Know-Your-Business
KYC	Know-Your-Customer
M	Month
MSP	Membership Services Provider
PSD2	Revised Payment Service Directive
RDBMS	Relational Database Management System
RA	Reference Architecture
UUID	Universally Unique Identifier

1 Introduction

The scope of deliverable D4.9 “Permissioned Blockchain for Finance and Insurance - III” is to document the efforts undertaken within the context of T4.3 “Distributed Ledger Technologies for Decentralized Data Sharing” of WP4. The deliverable D4.9 is prepared in accordance with the INFINITECH Description of Action and constitutes the third and final iteration of the work performed under Task 4.3. To this end, the document at hand has two-fold purpose: a) to provide the final and complete documentation of the specifications of the permissioned blockchain infrastructure that is exploited in the INFINITECH platform, b) to document final design specifications and the implementation details of the blockchain applications that were developed in the course of the project along with a detailed walkthrough of the user interface of the produced applications.

The blockchain technology has been embraced by the INFINITECH platform and holds a crucial part of the overall platform’s offerings. The advancements of the blockchain technology and its novel offerings were leveraged within INFINITECH in order to design and deliver blockchain empowered scenarios for the financial and insurance sectors. The scope of the usage of the blockchain technology within the INFINITECH platform has a dual purpose. At first, to enable the financial institutions to optimise and enhance their current core operational services and processes with novel solutions that can drive the innovation and the efforts for significant cost reductions and increased performance. Furthermore, the designed solutions provide the basis for the design and implementation of cutting-edge services, products and offerings to their clients.

The deliverable follows the structure of the previous iterations and is building directly on top of the previous results that were documented in deliverables D4.7 and D4.8, enhancing and optimising the blockchain infrastructure was designed and integrated in the INFINITECH Reference Architecture (INFINITECH RA), as well as delivering the designed and implemented blockchain-enabled solutions on top of this blockchain infrastructure. The deliverable presents the final fully-functional solutions that can be exploited by the INFINITECH project’s pilots, as well as the stakeholders of the financial and insurance sectors.

1.1 Objective of the Deliverable

The purpose of this deliverable is to report the final outcomes on the work performed within the context of Task 4.3 till M27 that the task concludes its activities. As the specific deliverable constitutes the third and final iteration, its main focus to provide the updated information that supplements the information documented on the previous iteration, highlighting the updates where needed, and to deliver the final fully functional blockchain applications that are deploy on top of the INFINITECH blockchain network.

Hence, the main objectives of the deliverable can be summarized as follows:

- To document the results of the thorough analysis of the key characteristics and offerings of the blockchain technology, as well as the role of the blockchain technology within the INFINITECH RA. While those results remained unchanged from the previous iteration, they are documented in this deliverable for coherency reasons.
- To deliver the final fully functional blockchain applications which leverage the innovative offerings of the blockchain technology. At first, the business need and the rationale behind each application is documented. The final design specifications are also reported highlighting the updates introduced. Then, the updated use cases and the corresponding sequence diagrams are presented. Following the use cases and sequence diagrams presentation, the implementation details of the final blockchain applications are documented. Finally, a walkthrough of the delivered applications is presented.

- To present the final design specifications of the INFINITECH blockchain network. The details of the designed and implemented blockchain network which is deployed and leveraged by the implemented blockchain applications are presented. The INFINITECH blockchain network remained unchanged from the previous iteration however it is included in the deliverable for coherency reasons also.
- To document the final list of technologies and tools which are leveraged for the deployment of the INFINITECH blockchain network and the blockchain applications.

The specific deliverable concludes the activities of Task 4.3 in accordance with the INFINITECH Description of Action. Hence, it constitutes the final documentation of the work performed and includes all the necessary updates and optimisations that were introduced from M21 till M27.

1.2 Insights from other Tasks and Deliverables

The deliverable D4.9 is released in the scope of WP4 “Interoperable Data Exchange and Semantic Interoperability” activities and documents the final outcomes of the work performed within the context of T4.3 “Distributed Ledger Technologies for Decentralized Data Sharing”. The task is tightly interconnected with the outcomes of WP2 “Vision and Specifications for Autonomous, Intelligent and Personalized Services” in which the overall requirements of the INFINITECH platform were defined. In detail, the outcomes of Task 2.1, as presented in deliverable D2.1 and D2.2 that reported the collected user stories of pilots of the project and the extracted user requirements, were provided as input in T4.3. Furthermore, the specification of the technologies that constitute the fundamental building blocks of the INFINITECH platform and the elicited technical requirements that are linked to these building blocks, as reported by the outcomes of T2.3 in deliverable D2.5 and D2.6, were also provided as input in T4.3.

Last but not least, the outcomes of T2.7, that formulated the INFINITECH Reference Architecture (INFINITECH RA) and that serve as the blueprint for the development, deployment and operation of Big Data, AI and IoT in the finance and insurance sectors, are directly related with the work performed in this task, as the reported outcomes of this report related to the blockchain network and the blockchain applications are integral parts of the INFINITECH platform. Finally, the work reported in this deliverable is tightly connected with the work performed in T4.4 and T4.5 of WP4 as the output of Task 4.3, and especially the designed blockchain network serves as the basis in the activities of both T4.4 and T4.5.

1.3 Structure

This document is structured as follows:

- Section **Error! Reference source not found.** introduces the document, describing the context of the outcomes of the work performed within the task and highlights its relation to the rest of tasks and deliverables of the project.
- Section 2 provides the results of the analysis of the blockchain technology, presents the key characteristics and main components of the technology, and defines the role of the blockchain technology in the INFINITECH RA.
- Section 3 presents the final and complete designed specifications of the blockchain applications, the use cases addressed by each application, and the corresponding sequence diagrams of each use case. Furthermore, it documents the technical details of the implemented blockchain applications and a walkthrough of the delivered blockchain application.

- Section 4 documents the updated details of the blockchain network which is leveraged by the blockchain applications, by presenting the updated network topology along with the updated list of services of each node and their interactions.
- Section 5 presents the list of baseline technologies and tools utilised in the implementation of the described network and applications.
- Section 6 concludes the document.

2 The Blockchain technology

Updates from D4.8:

The particular section remained unchanged from the previous version. It presents the key characteristics and offerings of the blockchain technology while also defining the role of the blockchain technology within the INFINITECH Reference Architecture.

2.1 An overview of the blockchain technology

Blockchain is a distributed digital ledger of cryptographically signed transactions that are grouped into blocks, which in turn are cryptographically linked to each other after validation and undergoing a consensus decision. The addition of new blocks increases the tamper resistance of the older ones and are replicated across the copies of the ledger within the network, resolving automatically any possible conflicts through a set of established rules [1]. In this sense, blockchain is a continuously growing, distributed, shared ledger of uniquely identified, linked transaction records organised in blocks that are sealed cryptographically with a digital fingerprint generated by a hashing function and are sequentially chained through a reference to their hash value [2]. Blockchain technology became extremely popular as the basis of cryptocurrencies as well as its implementations such as Bitcoin and Ethereum, which are digital currencies that were designed to work as medium of exchange incorporating secure and verifiable cryptographically signed transactions and controlled cryptocurrency unit generation.

In general, the blockchain technology is composed of multiple technologies related to cryptography, peer-to-peer networks, identity management, network security, transaction processing, (distributed) algorithms and more, that are all leveraged in order to formulate an immutable transaction ledger which is maintained by a distributed network of peer nodes formulating the blockchain network. The key characteristics of the blockchain technology can be grouped as follows [3]:

- *Decentralised:* One of the core characteristics of the blockchain is its decentralised and distributed nature across multiple number of nodes (peers) that provides extensibility, scalability, confidentiality, flexibility and resilience to attacks or misuse.
- *Immutable:* Any transaction record is immutable and reserved forever. Hence, full transactional history is maintained and all records are cryptographically secure. This fact safeguards that the underlying data cannot be tampered and are attestable.
- *Transparent:* The transaction data that formulate the blocks are transparent to each node and each node can introduce an update, based on a set of rules, increasing the transparency and trustworthiness of the technology.
- *Autonomy:* One of the core characteristics of the blockchain is the autonomy offered within the peer network that is regulated by the consensus protocols, where each node can safely and securely transfer and update data. Since the ledger (or actually a copy of the ledger) is shared among multiple nodes (peers), the transparency and trustworthiness are also increased.
- *Open Source:* The blockchain technology is an open source technology with multiple blockchain implementations and variations being available; sustained by various communities and ecosystems, that can be leveraged upon needs.

At a high level, the blockchain technology exploits well-established computer network mechanisms and cryptographic primitives such as cryptographic hash functions, digital signatures, asymmetric-key cryptography, certificate authority mixed with record keeping concepts (such as append only ledgers) [1]. Nevertheless, the blockchain technology has a set of key concepts that includes the distributed

ledger that is composed by blocks containing the transaction records, the smart contracts or chaincode and the consensus model.

The heart of the blockchain is the distributed ledger in which all transaction records that are published within the blockchain network are stored in the form of blocks. Being by nature decentralized, the technology takes advantage of both the distributed ownership and the distributed physical architecture of a distributed ledger. Each peer maintains its own copy of the ledger, ensuring that is synced and updated with the same data. As the blockchain network is designed and is operating in a peer-to-peer mode, the blockchain network has an increased resilience to the loss of any node. Every new transaction is checked and verified among all peers before it is accepted and inserted into the ledger and it is referenced to the previous block, enabling an integrity check of invalid transmitted transactions or nodes. Finally, with the utilisation of the cryptographic mechanisms the distributed ledger is tamper evident and tamper resistant.

The nodes that are participating in the blockchain network can be characterised as publishing and non-publishing nodes. The candidate transactions are submitted to the blockchain network via its user through the interacting applications and services. However, it is the role of the publishing node(s) only to publish a block in the blockchain network that contains these transactions via the gossip data dissemination protocol, once they have been validated and authenticated, that will be received by the rest of the (publishing and non-publishing) nodes which in turn will validate and authenticate the received block and accept it in order to be inserted finally in the ledger.

To facilitate all the operations performed in the ledger within the blockchain network, the smart contracts (or chaincode) are leveraged. Smart contracts are the trusted distributed applications that are deployed within the nodes of the blockchain network and encapsulate the business logic of the blockchain applications. Smart contracts include the agreements that the participants of the blockchain network have formulated with regards to the generation of new facts that are added to the ledger and that will update the current and historical state of the facts that are already stored in the ledger. In this sense, the smart contracts enable the creation of new transactions by the users of the blockchain network by invoking the smart contracts' functions. The smart contracts facilitate the controlled access to the ledger, offering a layer of abstraction on top of the aspired transactions, encapsulating and simplifying all the relevant information while also ensuring their compliance with the underlying legal agreements, as well as the automation of the several aspects of the transactions. The implementation and execution of smart contracts varies depending on the blockchain implementation with most popular cases being Ethereum's smart contracts and Hyperledger Fabric's chaincode.

One of the most critical concepts of blockchain technology is the consensus model that is utilised in order to validate a transaction and to keep the ledger transactions synchronized across the blockchain network. Hence, the consensus model undertakes the validation and approval of the candidate transactions and ensures that the copies of the ledger, that are kept within the nodes of the blockchain network, are updated with the same transactions and in the same order. As the blockchain network is composed of multiple nodes, it is very likely that many publishing nodes will compete at the same time to publish new nodes. Additionally, conflicts might be created by nodes publishing new blocks at approximately the same time. Hence, it is evident that a method is required to ensure that transactions will be written to the ledger at the same order as they generated, as well as that malformed or malicious transactions are rejected. For this reason, the blockchains depending on their implementation specifications exploit different consensus models that are available in computer science such as the CFT (crash fault-tolerant) or BFT (byzantine fault-tolerant) ordering, while at the same time large research effort is spent on this topic towards the definition of further alternative consensus models capable of better addressing this issue with less trade-offs.

The blockchain implementations can be characterised and grouped into two major high-level categories based on the permission model applied on the blockchain network, the *permissionless*

blockchain networks and the *permissioned blockchain networks*. Permissionless blockchains are based on open and public blockchain networks where anyone is capable of publishing new blocks or read the blocks of the blockchain anonymously and without granting any permission from any authority. Hence, the implementation of the permissionless blockchains dictates that they are open and available to anyone and anyone can issue new transactions in new blocks and read the transactions included in the existing blocks, thus write and read the ledger. To prevent the malicious usage of the blockchain, these implementations employ a consensus model that requires from the participants to expend or maintain resources through a “mined” native cryptocurrency or through transaction fees when it comes to publishing new blocks. The most common consensus models employed are the “proof of work” or “proof of stake” that are rewarding the participants of new blocks that conform the consensus protocol with a native cryptocurrency. The well-established examples of permissionless blockchains are Bitcoin and Ethereum.

On the other hand, the permissioned blockchain networks are regulated blockchain networks where only authorised users, by a specific authority as defined within the network specifics, are able to maintain the underlying blockchain, while read access and publishing of new transactions are also restricted and regulated. In this sense, the permissioned blockchain networks are formulated only by a set of known, identified and verified participants whose access rights and roles are regulated by an agreed governance model defined by the participants of the networks providing a certain degree of trust and security for all generated transaction records. As the identities of the participants of the network are known and trusted, the consensus models that are employed for publishing new blocks do not require the expense or maintenance of resources. In permissioned blockchain networks the consensus models exploited are usually faster and less computationally expensive, as the mining operations are eliminated and more traditional consensus protocols are adopted, such as the CFT or BFT protocols. Permissioned blockchain networks allow the tight control and protection of the underlying blockchain, and the level of trust between each participant of the network can be reflected on the consensus model that will be used or regulated by the access rights to the data that each participant can obtain. Furthermore, the authorisation of each participant can be revoked in the case of misuse or withdrawal of trust.

Blockchain technology has an enormous potential that has been noticed by several industries that are looking forward to exploit its various advantages and offerings in order to introduce new services, products and offerings to their clients or to rejuvenate their internal processes and legacy systems towards a better performance, reduction of financial costs and increase of trust between the partners involved in business transactions. The mostly adopted area is the cryptocurrencies area where Bitcoin and Ethereum are the most notable cases. Additionally, blockchain is adopted in financial services, insurance services, supply chain, energy trading, sales, digital music, anti-counterfeiting, domain name services and videogames, among others. For the financial and insurance services in particular, the blockchain technology has found many potential use cases for providing blockchain-enabled banking and insurance services that optimise various back-office processes, removing various intermediaries and disrupting various operational processes towards the financial cost reductions and the expansion of the portfolio of offered services to their customers.

2.2 The role of blockchain technology in INFINITECH RA

Within the INFINITECH Reference Architecture (INFINITECH RA), as presented in deliverables D2.13 and D2.14, the blockchain technology has a dual presence and can be exploited in different ways, depending on the scope of the use case that INFINITECH RA aims to address.

Hence, on the one hand, blockchain can be considered as an additional data source type at the infrastructure layer, from which data are accessed and collected with the use of the respective sophisticated data collection mechanisms. On the other hand, blockchain can be considered as a cross-cutting service positioned in the central layer of the INFINITECH RA, in which decentralised

applications tailored to the needs of the financial and insurance sectors can be developed and exploited. The blockchain-enabled decentralised applications can be utilised to apply use cases that can optimise, and even revolutionize, core operational processes of the financial or insurance institutions, decreasing their costs and increasing radically their performance and efficiency by reducing or eliminating the need for manual processing or manipulation, while at the same time augmenting their level of trust. Within the context of the WP4, the focus is on the development of such decentralised applications that will showcase the potentials of the blockchain technology.

As with many other industrial sectors, financial and insurance sectors are highly regulated with strict legislations and processes in which security and trust are fundamental aspects. While the blockchain technology is considered as the dominant candidate to disrupt all of the financial and insurance sectors' processes, as it is promising to mitigate the cost of trust and increment the security level in these processes and even business models [4], not all blockchain implementations are deemed as ideal candidates. The reason for this lies on the specific characteristics offered by the two major approaches followed in the blockchain technology, the permissionless blockchain and the permissioned blockchain.

While the permissionless blockchain, with public open networks to which anyone can participate and interact in an anonymous manner, has been adopted in the case of cryptocurrencies, when it comes to more enterprise-oriented use cases, such as the banking institutions or other financial institutions, different requirements arise and are related mainly to the privacy and confidentiality of the data, as well as the underlying business or financial transactions stored in the blockchain, the regulated and strictly controlled access to the blockchain network, the performance of the network with high throughput and low latency, and most importantly the hard requirement of the identifiable and pre-approved identity of the participants of the blockchain network. For all these reasons, among the two major approaches, only the permissioned blockchain technology is considered as the appropriate candidate solution and is exploited within the INFINITECH RA.

Towards this end, the consortium decided to exploit the Hyperledger Fabric open source enterprise-grade permissioned distributed ledger technology (DLT) platform [5] that is one of the most active projects of the Hyperledger project founded by the Linux Foundation. Hyperledger Fabric has been designed specifically for enterprise use and delivers a set of key differentiating capabilities over other popular distributed ledger or blockchain platforms. One of the main differentiations of Fabric is its highly modular and configurable architecture that promotes the innovation, versatility and optimization for a broad range of industry use cases including banking, finance and insurance [6]. Based on its modular and configurable architecture, Fabric guarantees a high level of confidentiality, resiliency, flexibility, and scalability.

Through its pluggable ordering service, consensus can be achieved with multiple implementations, such as the CFT or BFT and more, based on the requirements of a specific use case or deployment. Fabric offers a private and permissioned blockchain network where all participants can be enrolled based on their cryptographic entities, through a set of pluggable trusted membership service providers that are supported, and can be tailored to the needs of the deployment. Fabric provides its own ordering service implementation named Raft. Raft ordering service is a CFT that is based on an implementation of Raft protocol in the etcd distributed key-value store and it constitutes the first step toward Fabric's development of a BFT ordering service.

Although Hyperledger Fabric's Raft ordering service is CFT based on an implementation of Raft protocol in etcd, it constitutes the first step toward Fabric's development of a byzantine fault tolerant (BFT) ordering service.

In Fabric, smart contracts, referred as chaincode, are operating in an isolated container environment, such as Docker, and can be written in standard programming languages, such as Go and Node.js. Smart contracts offer the required interfaces that are exploited by applications outside of the blockchain network in order to interact with distributed ledger providing the required level of abstraction, as well

as increased level of privacy and confidentiality. To further promote privacy and confidentiality, Fabric enables the creation of channels in which the participants own a separate ledger of transactions from the rest of the blockchain network that is visible only to the participants of the channel. Finally, it provides the feature of private data, where collections of data can only be visible and accessible to a portion of the participants of a specific channel.

It is acknowledged that the combination of all these key differentiating capabilities, sets Fabric as one of the best performing platforms in transaction processing and transaction confirmation latency platforms. Its pluggable architecture enables its exploitation in a variety of different use cases of the financial and insurance sectors that are characterized as highly complex and restrictive sectors.

Towards this end, to address the needs of the financial and insurance sectors, the designed distributed applications will address core use cases of the financial and insurance sectors exploiting the benefits of the permissioned blockchain technology. In this sense, the designed distributed applications will effectively leverage the underlying permissioned blockchain infrastructure provided by Fabric that is designed in accordance to the needs and requirements of the stakeholders of the specific sectors. The design specifications of the designed distributed applications are presented in detail in Section 3 of the current deliverable, while the details of the designed underlying permissioned blockchain network that these distributed applications will be deployed, are documented in Section 4.

3 INFINITECH Blockchain Applications

Updates from D4.8:

The particular section documents the necessary updates related to the advancements on the implementation of the INFINITECH Blockchain applications. In detail, the following documentation is introduced per blockchain application:

- *Consent Management: The documentation of the business need and rationale behind the developed solution (section 3.1.1), the update of the Use Cases and Sequence Diagrams (section 3.1.3 and 3.1.4), the update of implementation details of the final version of the Consent Management (section 3.1.5) and the presentation of the Consent Management Application (section 3.1.6).*
- *Know Your Customer (KYC) / Know Your Business (KYB): The documentation of the business need and rationale behind the developed solution (section 3.2.1), the update of implementation details of the final version of the KYC/KYB (section 3.2.5) and the presentation of the KYC/KYB Application (section 3.2.6).*
- *Tokenization: The updated short documentation of the work performed for the specific use case, as well as the details of the future work (Section 3.3).*

The benefits of the blockchain technology are leveraged with the development of trusted distributed applications that are deployed within the nodes of the blockchain network. In the blockchain technology, the trusted distributed applications are developed as smart contracts or chaincode in the Fabric terminology. The role of a chaincode is to generate the new facts that will be added to the ledger, which maintains all the facts related to the current and historical state of a set of business objects, in order to introduce the appropriate changes in both the current and historical state. In this sense, the chaincode defines the transaction logic that is responsible for the evolution of the state of the business objects. Furthermore, the chaincode provides a set of interfaces that are utilised by the applications outside of the blockchain network in order to interact with the distributed ledger.

Hence, in the design of any blockchain-enabled solution that effectively addresses the needs of a specific business operation, process or service, the main components included are the external to the blockchain network application and the chaincode that is deployed on the blockchain network. In the design specifications of both components, a specific business logic is encapsulated, whose aspects are clearly defining the permitted and required operations which are executed for a use case of the business operation.

With regards to the design of the chaincode, it is usually written in the GO programming language and it explicitly defines the governance rules for any type of business object utilised in the use case. The structuring of the source code of the chaincode can vary as there is no strict norm of how the source code should be constructed, rather than best practises formulated by the blockchain development communities. However, the basis of the source code is the definition of the business objects on top of which this chaincode will perform all the operations and the functions that will undertake the execution of these operations. Thus, as for every specific business operation different business objects are defined, the whole chaincode is tailored to the needs of each business operation.

Nevertheless, within the context of the INFINITECH project and specifically Task 4.3, the developed chaincode will adhere to the following structuring of the source code that is based on the logical schema of the Data Processing Components as defined in the deliverable D2.5:

- *Blockchain Reader: The main purpose of this component is to enable the fetching of the requested data from the blockchain ledger. Depending on the underlying use case, the design specifications may vary. However, the context of the functions of the component will remain the same.*

- *Blockchain Writer*: The main purpose of this component is to facilitate the submission of new transactions to the blockchain ledger. As with the blockchain reader, the design specification for each use case may vary, nevertheless the context of the functions of the component will remain unaffected.
- *Smart Contract Executor*: The main purpose of this component is to encapsulate the business logic of the designed use case and execute the smart contracts on the blockchain ledger. Hence, the design specifications of each use case are tailored to the needs of the aspired operation and service and they will orchestrate the use case execution.
- *Blockchain Authenticator*: The main purpose of this component is to perform the authentication of the blockchain network user in order to grant access to a specific channel of the blockchain network. The implementation of the specific component is almost generic and will be utilised across all the implemented use cases.
- *Blockchain Encryptor*: The main purpose of this component is to perform the encryption of the data that are involved and produced within the smart contract execution utilising the AES 256 encryption. The implementation of the specific component is almost generic and will be utilised across all the implemented use cases.
- *Blockchain Decryptor*: The main purpose of this component is to perform the decryption of the data that were encrypted by the Blockchain Encryptor. Again, the implementation of the specific component is almost generic and will be utilised across all the implemented use cases.

In the following sections, the design specifications of the blockchain applications that will be developed within the context of Task 4.3, are presented in detail. In total, two different blockchain applications are presented. For these two applications, namely the Consent Management and the Know-Your-Customer (KYC) / Know-Your-Business (KYB), a thorough description of the addressed business operation is presented, highlighting the use of the blockchain technology on each application accompanied by the high-level architecture of the application. Furthermore, the details of each specific use case of the business operation, that is supported by the blockchain application, is presented. In addition to this, for each use case of the business operation, the corresponding sequence diagram that illustrates the interactions between the involved stakeholders and the components is documented. Finally, a walkthrough of the developed applications for the two solutions is presented. The section concludes with a short description of the application related to tokenization use case that is currently formulated within the context of Task 4.4. However, the thorough documentation of this specific use case will be documented within the context of the deliverable D4.12 in accordance with the INFINITECH Description of Action (DoA).

3.1 Consent Management

Updates from D4.8:

The updates on this final iteration involve the Business need & Rationale section (3.1.1) that documents the motivation behind the implementation of the developed application, as well as small enhancements in the Use Cases (3.1.3) and Sequence Diagrams (3.1.4) sections in order to better address the needs of the finance and insurance sectors. In addition to this, the Implementation section (3.1.5) has been updated to depict the latest advancements of the development phase and the introduction of the Consent Management Application Overview section (3.1.6) that provides a walkthrough of the implemented application.

3.1.1 Business need & Rationale

The banking sector has entered a new digital innovation era which is mainly driven by the newly introduced Revised Payment Service Directive (PSD2) [7] that reshapes the specific sector acting as a catalyst for the new wave of innovative financial services. This new directive was initially introduced in 2017 and became effective as of December 2020 as an amendment of the previous directive

Payment Service Directive (PSD) that was established in 2007 as a European legislation composed of a set of laws and regulations for electronic payment services in the European Union (EU) and the European Economic Area (EEA) [8]. PSD2 is fostering innovation with the requirement to open the access to the financial (or banking) data, currently maintained by the banks, to third party providers (TPPs) in conjunction with strong customer authentications and the requirement of an authorized and registered by the European Banking Authority (EBA) payment license. With the enforcement of the PSD2 directive, novel financial services can be introduced and offered to customers through the usage of third party APIs which can be leveraged by TPPs introducing what is referenced as the Open Banking initiative. The main rationale behind the introduction of PSD2 by the European Commission is the establishment of a single European payment market that will be driven by innovation, transparency and security through the close collaboration of banks and fintech innovative institutions [9]. In accordance to a recent report by KPMG [10], the Open Banking initiative enables banking customers to authorize third parties to obtain access to their financial information, such as bank account data, in order to either collect account information or to initiate payments.

Nevertheless, the Open Banking initiative introduces many challenges for the banking sector since it imposes the requirement to share personal and sensitive information of their customers to TPPs in order to be compliant with PSD2. However, another important directive was put into force by the European Commission that constitutes the main pillar law for the data protection and data privacy of the European Union, namely the General Data Protection Regulation (GDPR) [11]. GDPR became effective as of May 2018 and imposes a set of strict privacy requirements related to data collection, processing and retention of personal and sensitive information of individuals. In a nutshell, GDPR enables individuals to have complete control of their personal and sensitive information by providing them the right to provide or withdraw their consent to any third party that will legitimately access and process their personal and sensitive information. In addition to this, strict security and privacy preserving requirements are imposed to third parties that access and process their personal and sensitive information.

As it is obvious, the banking sector is obliged to adhere to both directives by finding the proper balance between the data sharing towards innovation and the respective data protection which is required at the same time. In detail, the banks in order to comply with the PSD2 directive they should enable the access to the personal and sensitive information of their customers to TPPs in a way which the security of this information is guaranteed in a GDPR compliant manner. Hence, the Open Banking initiative imposes several challenges to the banks as Open Banking will expand traditional banking data flows, placing the customer at its core and in control of their banking data, including their personal information, as reported by Deloitte [9]. However, as reported also by Ernest & Young, when properly implemented in harmony, PSD2 and GDPR enable banks to better protect and serve consumers, move beyond compliance and to seize new opportunities for growth [12]. Furthermore, through Open Banking and PSD2 the banks have acknowledged the unique opportunity to expand and strengthen their business models, offer new novel services to their customers and be able to extract valuable information from their customer's patterns of behaviour [13].

The pillar of Open Banking is the data sharing between involved institutions in order to optimise the effectiveness and efficiency of existing services, reduce costs and strengthen the relationships of businesses with their clients. However, when it includes sharing of personal and sensitive information of customers, data privacy requirements also arise, as explained above. The proper balance between PSD2 and GDPR is achieved by the consent of the customer of the bank to share their personal and sensitive information to TPPs. Consent constitutes one of the six legal grounds of lawful processing of personal data, in accordance to GDPR [11], and it is imperative that is given by a statement or by a clear affirmative action. To this end, the need arises for robust and efficient consent management by the banks.

Consent management is referred to process of handling the complete lifecycle of consents provided by an individual (such as the customer of the bank) for the legit processing and/or sharing their

personal data to another party (such as the TPPs). Since PSD2 is fostering digital innovation and integration, the appropriate process is the digital consent management as it constitutes the bridge between PSD2 and GDPR. This is acknowledged also by the Open Banking Working Group of Euro Banking Association (EBA) that reports that digital consent management lies at the heart of any possible solution for the challenges ahead [14] as it is enabling the banks and TPPs to leverage the offered by PSD2 services in a GDPR compliant manner.

The main aim of a consent management process is to facilitate tracking, monitoring and management of the personal and sensitive information collection and/or processing from the moment of opt-in to the data erase in GDPR compliant manner. The process facilitates the complete control over this information to the individuals by offering them the right to provide (or withdraw) their consent to third parties in order to collect and process their personal and sensitive information at any point and with immediate effect. An effective consent management system is enabling the execution of digital consent management by implementing all the required operations during the complete consent management lifecycle, such as the consent collection, storage, usage, update and opt-out operations. Its main goal is to act as the mediator between the individuals and the third parties regulating the formulation, updated and withdrawal of consents, as well as their storage in a secure and trusted data storage modality.

3.1.2 Description of the solution

Collaborative data sharing between customers, banks and other organizations in accordance with PSD2, enables application of advanced analytics over particular datasets and intelligent support tools for better understanding customers and their financial relationships, thus being critically important in today's financial markets. However, the development of intelligence support tools which will support new customer services that solve business problems, such as improved Know-Your-Customer (KYC) processes and consequently Anti Money Laundering (AML), credit scoring and fraud detection services that can be built upon them, is highly dependent on the customers permission to share data. As a result, the requirement for a trusted and secure sharing mechanism of customer consent arises. In this sense, banks also identify granular permission consent as a key enabler of trust which is vital to maximise data sharing and ensure customers are comfortable with sharing data.

In this blockchain application, we aim at exploiting the blockchain technology and specifically the permissioned blockchain in order to develop a decentralised and robust consent management mechanism, that will enable the sharing of the customers' consent to exchange and utilise their customer data across different banking institutions. Blockchain technology, and its latest advancements, appears as a compelling technology to overcome the underlying challenges of trust improvement due to its decentralised nature and immutability, as well as the impossibility of ledger falsification. The integrity of customer data processing consents and their immutable versioning control are protected by the blockchain infrastructure. The blockchain-enabled consent management mechanism will enable the financial institutions to effectively manage and share their customers' consents in a transparent and unambiguous manner, enabling them to inform the customers at any time about:

- a) any customer data they are managing upon their consent
- b) the status of their consent (active or revoked/withdrawn)
- c) the recipients (financial institutions or peers) of their customer data upon their consent
- d) the purpose (or even legal basis) and time period of their customer data sharing to the recipient (financial institutions or peers)

In the same transparent and unambiguous manner, the customers of the financial institution will:

- a) be constantly informed for all the requests for sharing their customer data
- b) be able to activate or revoke their consents

- c) be constantly aware of the active consents they have given to each specific recipient

The blockchain-enabled consent management mechanism guarantees the integrity of the consents through the usage of the blockchain technology, as well as cryptographic techniques and digital signatures incorporated in it. Besides the consent records and their integrity, the blockchain technology will be used to securely maintain the consent history providing the complete consents' versioning. Through the blockchain technology, immutable versioning control is provided that is capable of obtaining the latest version of the consent, as well as the previous versions of the consent and their valid periods, in an indisputable manner. Thus, the blockchain is capable of storing the consents and their complete update history in a secure and trusted manner. In this way, both the financial institutions, as well as their customers are protected as both the consents' integrity and validity periods are secured.

With regards to the consents and their validity period, two different approaches are considered. The first approach provides the ability of an "once off" consent, in which the validity period of the consent is a predefined period. Once this period expires, the consent is automatically revoked and new consent (or an updated consent) is required in order for the recipient to be able to access the customer's data. An example of the cases considered for this consent type is the customer's consent for sharing of KYC data between two financial institutions (banks) for the scenario of a loan origination / application or an account opening. The second approach provides the ability of a permanent (or "regular") consent that has an infinite validity period and it is only invalidated if the consent is withdrawn. An example of the cases considered for this consent type is the Peer-to-Peer (including Person to Person, Person to Organisation, Organisation to Organisation cases) customer data sharing consent in which a customer utilises an interface of a mobile application to select a set of its specific customer data (such as specific accounts, specific transactions or alerts) that they would like to share with an individual person or a group of persons. It should be noted that the consent management mechanism does not save or distribute any customer data. Customer data are exchanged using the respective secured APIs of the financial institution. Instead, the blockchain based consent management mechanism maintains the minimum information that is required in order to formulate a consent agreement between the customer and the respective financial institution.

Figure 1 depicts a high-level architecture of the proposed solution. As depicted, the core elements of the proposed solution are the *Consent Management System*, the *File Storage* and the *Blockchain infrastructure*, while the key stakeholders that are involved and are interacting with the proposed solution are the *internal financial institution* that collects and has access to its customer data, the *customer of internal financial institution* whose data are collected by the internal financial institution, and finally the *external financial institution* or peer that aspires to obtain the data of the customer of internal financial institution upon the formulation of a valid and legitimate consent that is formulated between the three parties.

In this architecture, the Consent Management System is the mediator between the involved parties. It receives and processes the requests for a consent formulation from the external financial institution or peer to the internal financial institution, and consequently the customer of the internal institution. By interacting with all three involved parties and upon acceptance of the details and terms of the consent by all of them, the final consent receipt is formulated and stored in digital form within the File Storage. At the last and most crucial step, the minimum information that is required from the formulated consent form is entered in the blockchain infrastructure by invoking the deployed chaincode that is responsible for the generation of a new transaction in the underlying ledger. Additionally, the Consent Management System, is consulting and retrieving the information stored in the blockchain infrastructure, by invoking the deployed chaincode in order to formulate an access control decision depending on the existence and validity of a consent when access to customer data via the internal financial institution's APIs is requested or to retrieve the complete history of consents for a customer upon request.

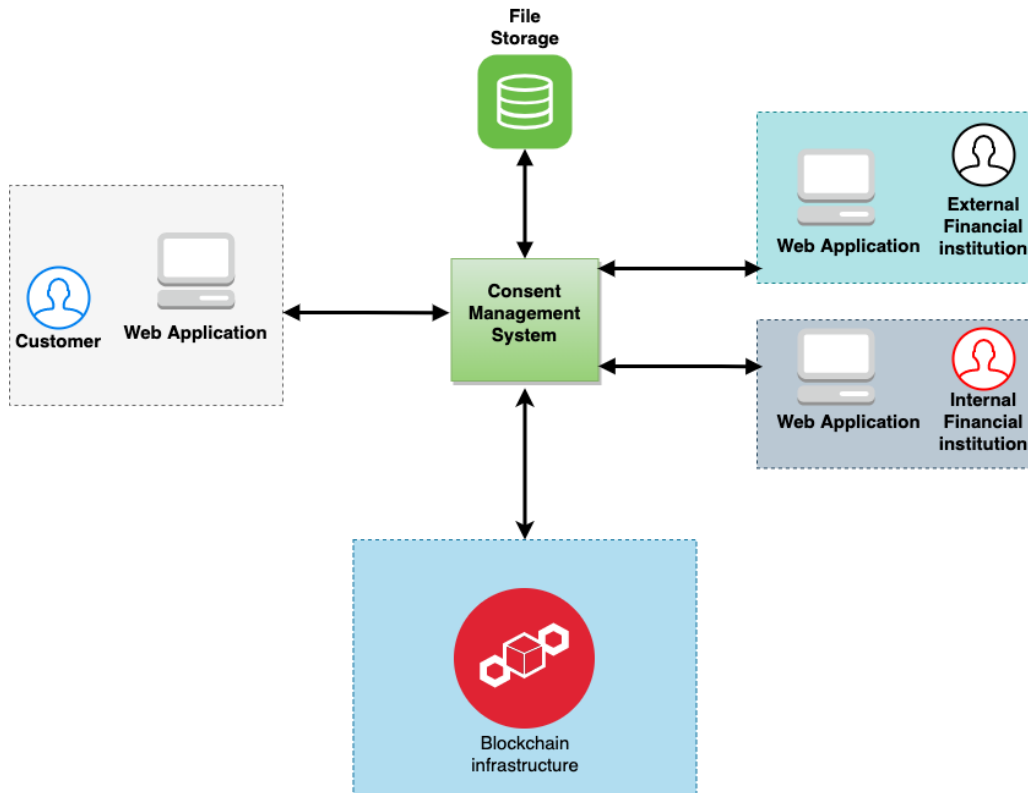


Figure 1: High-level architecture of the Consent Management solution

One of the core aspects of the design specifications is the definition of the business objects for which the current and historical state will be maintained and updated through the functions of the chaincode. To this end, for the Consent Management application, the initial data schema which defines the core business objects has been defined. The definition was based on the Consent Receipt specification that is proposed by the Kantara Initiative [15] which has been adapted in terms of terminology in order to be aligned with the EU GDPR legislation. Figure 2 depicts the initial data schema of the Consent Management application, while the details of all the entities are documented in **Error! Reference source not found.** to Table 4.

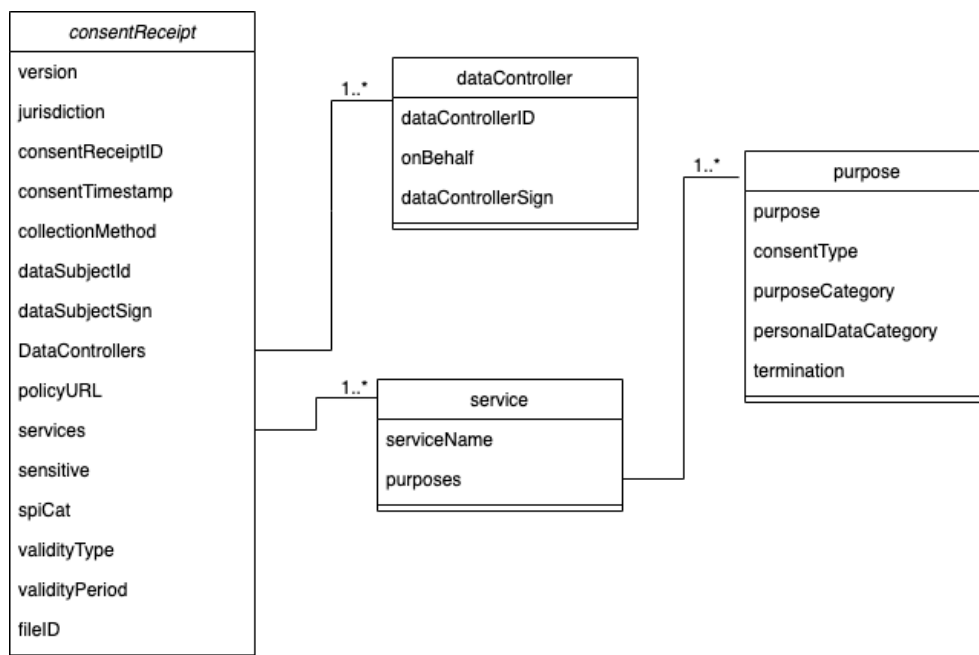


Figure 2: Consent Management Data Schema diagram

Table 1: Consent Management Data Schema details (1)

consentReceipt		
Name	Type	Short description
<i>version</i>	String	REQUIRED: The version of the consent specification to which a receipt conforms
<i>jurisdiction</i>	String	REQUIRED: The jurisdiction(s) applicable to this transaction i.e. EU and/or any EU-national
<i>consentReceiptID</i>	String	REQUIRED: The unique identifier of each Consent Receipt in UUID-4 format
<i>consentTimestamp</i>	String	REQUIRED: Date and time of the consent transaction in ISO 8601 format
<i>collectionMethod</i>	String	REQUIRED: A description of the method by which consent was obtained, i.e. the Consent Management System
<i>dataSubjectID</i>	String	REQUIRED: The unique identifier of the Data Subject in UUID-4 format
<i>dataSubjectSign</i>	String	REQUIRED: The Data Subject's digital signature in base64 format.
<i>dataControllers</i>	Array [dataController]	REQUIRED: An array of dataController items (see Table 2)
<i>policyURL</i>	String	REQUIRED: A link to the DataController's privacy statement/policy and applicable terms of use in effect when the consent was obtained, and the receipt was issued.
<i>services</i>	Array [service]	REQUIRED: An array of Service items (see Table 3)
<i>sensitive</i>	Boolean	REQUIRED: Indicates whether the consent interaction contains personal data that is designated sensitive or not sensitive
<i>spiCat</i>	Array [categories]	REQUIRED: A listing of categories where personal data collected is sensitive. The array must contain the categories if sensitive is TRUE
<i>validityType</i>	String	REQUIRED: The validity type of the consent, either ONCE_OFF or PERMANENT.
<i>validityPeriod</i>	String	REQUIRED: Date and time of the consent expiration in ISO 8601 format. Required when validityType is ONCE_OFF.
<i>fileID</i>	String	REQUIRED: A reference to the file stored in the File Storage
<i>status</i>	String	REQUIRED: Represents the current consentReceipt status. Possible values are ACTIVE, WITHDRAWN, EXPIRED

Table 2: Consent Management Data Schema details (2)

dataController		
Name	Type	Short description
<i>dataControllerID</i>	String	REQUIRED: The unique identifier of the data controller in UUID-4 format.
<i>onBehalf</i>	Boolean	OPTIONAL: True if a data processor is acting on behalf of a data controller.

<i>dataControllerSign</i>	String	REQUIRED: The Data Controller’s digital signature in base64 format.
---------------------------	--------	---

Table 3: Consent Management Data Schema details (3)

service		
Name	Type	Short description
<i>serviceName</i>	String	REQUIRED: The service or group of services being provided for which personal data is collected. The ID of the service for which consent for the collection, use, and disclosure of personal data is being provided.
<i>purposes</i>	Array [purpose]	REQUIRED: An array of Purpose items (see Table 4)

Table 4: Consent Management Data Schema details (4)

purpose		
Name	Type	Short description
<i>purpose</i>	String	OPTIONAL: A short, clear explanation of why the personal data is required
<i>consentType</i>	String	REQUIRED: The type of the consent used by the Data Controller as their authority to collect, use or disclose personal data. The accepted values are EXPLICIT or IMPLICIT.
<i>purposeCategory</i>	String	REQUIRED: The reason the Data Controller is collecting the personal data. The acceptable values are based on the (CISWG) Wiki page [16].
<i>personalDataCategory</i>	String	REQUIRED: A list of defined personal data categories. Personal data category should reflect the category that will be shared as understood by the Data Subject. The acceptable values are based on the (CISWG) Wiki page [16].
<i>termination</i>	String	REQUIRED: Conditions for the termination of consent. Link to policy defining how consent or purpose is terminated.

3.1.3 Use cases

The following sections provide the detailed documentation of all the use cases encapsulated in this blockchain application, describing in detail all information of each use case.

3.1.3.1 Use case CMS-1: Register Customer in the Consent Management System

In order for customers to be able to receive consent requests, it is mandatory that they are registered in the Consent Management System, creating their profile that will be used in order to receive consent requests. The process is completed by the administrator of the Consent Management System. The customer profile shall include the minimum customer discovery and communication details required in order to receive a consent request. The profile information and any consequent private information is not disclosed to any interested party and is not saved in the blockchain infrastructure.

Table 5: Consent Management Use Case CMS-1

Stakeholders involved:	Customer, Internal Financial Institution
-------------------------------	--

Pre-conditions:	A customer willing to provide his/her consent to share his/her customer data upon his/her approval.
Post-conditions:	A customer is registered in the Consent Management System, his /her profile is created and is able to receive consent requests from the Internal Financial Institution.
Data Attributes	Required data based on the Consent Management Data Schema.
Normal Flow	<ol style="list-style-type: none"> 1. The administrator registers the customer to the Consent Management System by filling in the required information explained in the Data Attributes above 2. The registration of the customer automatically enables the customer to receive new consent requests
Pass Metrics	<ol style="list-style-type: none"> 1. Customer profile is available in the Consent Management System
Fail Metrics	<ol style="list-style-type: none"> 1. No customer profile is available in the Consent Management System

In the same manner, the external financial institution or the peer is registered by the administrator of the Consent Management System in the Consent Management System in order for the external financial institution to be able to initiate consent requests to a customer of the Internal Financial Institution.

Table 6: Consent Management Use Case CMS-1 (2)

Stakeholders involved:	External Financial Institution, Internal Financial Institution
Pre-conditions:	An External Financial Institution interested in obtaining the consent of customer in order to access his/her customer data upon his/her approval
Post-conditions:	An External Financial Institution is registered in the Consent Management System, his /her profile is created and is able to initiate consent requests to Internal Financial Institution.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	<ol style="list-style-type: none"> 1. The External Financial Institution is registered to the Consent Management System by the administrator of the Consent Management System by filling in the required information explained in the Data Attributes above
Pass Metrics	<ol style="list-style-type: none"> 1. External Financial Institution profile is available in the Consent Management System
Fail Metrics	<ol style="list-style-type: none"> 1. No External Financial Institution profile is available in the Consent Management System

3.1.3.2 Use Case CMS-2: Customer receives a request to provide new consent for sharing his/her customer data

The stakeholder wishing to gain access to the customer data issues a new consent request to the Consent Management System specifying the details of the requested customer data (i.e. specific data, period of usage). The customer receives a notification with all the required information for the consent request in a proper way that it allows him/her to review the request.

Table 7: Consent Management Use Case CMS-2

Stakeholders involved:	Customer, Internal Financial Institution, Banking institutions, Peers, Financial organisations
Pre-conditions:	<ol style="list-style-type: none"> 1. A customer profile is available in the Consent Management System capable of receiving new consent requests 2. A new request for customer data is issued by a banking institution or financial organisation to the Internal Financial Institution
Post-conditions:	The customer is notified for the new consent request in order to review and decide about giving his/her consent or denying the access.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	<ol style="list-style-type: none"> 1. External Financial Institution issues a new consent request in the Consent Management System in order to obtain the consent of a customer of the Internal Financial Institution including all the required information described in the data attributes above. 2. Internal Financial Institution pre-approves the new request for the consent of the customer 3. Customer receives the new request in a proper format through the web-based interface provided by the Consent Management System so that he/she can review and decide whether to provide his/her consent or deny the access to his/her customer data.
Pass Metrics	<ol style="list-style-type: none"> 1. The customer is informed of the new consent request and is able to formulate a decision.
Fail Metrics	<ol style="list-style-type: none"> 1. The customer is not informed for the new consent request

3.1.3.3 Use Case CMS-3: Definition of the consent

The customer reviews and possibly alters the details and conditions of the consent request before formulating his/her decision to give his/her consent or deny the access. In the case of approval, the final consent is defined by the customer and it is submitted to the Consent Management System. In the case of the denial, the request is blocked and the interested party is informed and the processing is finished.

Table 8: Consent Management Use Case CMS-3

Stakeholders involved:	Customer, Internal Financial Institution
Pre-conditions:	A new consent request from an interested party is provided to the customer.
Post-conditions:	The customer formulates the consent form that is ready to be signed by both parties.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	<ol style="list-style-type: none"> 1. The customer reviews, edits and finalises the consent form details (based on the data attributes above). He/she is able to check and alter the proposed attributes and details of the request in a user-friendly way through the web-based interface provided by the Consent Management System. 2. In case of approval: The customer provides his/her consent form to the Consent Management System in order to be signed. 3. In case of denial: The customer denies the request; the interested stakeholder is informed and the process is finished.
Pass Metrics	<ol style="list-style-type: none"> 1. In case of approval: The consent form is ready to be signed by both parties. 2. In case of denial: The interested stakeholder is informed and the process is finished.
Fail Metrics	<ol style="list-style-type: none"> 1. In case of approval: No consent form is available 2. In case of denial: The process is still open

3.1.3.4 Use Case CMS-4: Signing of the consent by the interested parties

Once the consent form is ready, the Consent Management System provides it to both parties (the customer and the interested stakeholder) in order to be digitally signed. The Consent Management System collects the digitally signed consent form from both parties.

Table 9: Consent Management Use Case CMS-4

Stakeholders involved:	Customer, Internal Financial Institution, Banking institutions, Peers, Financial organisations
Pre-conditions:	A valid consent form from a customer is available.
Post-conditions:	The signed consent form from both parties.
Data Attributes	Required data based on the Consent Management Data Schema

Normal Flow	<ol style="list-style-type: none"> 1. The consent form is sent to the customer by the Consent Management System through the web-based interface provided by the Consent Management System in order to be digitally signed. 2. The customer provides the digitally signed consent form to the Consent Management System. 3. The consent form is sent to the interested party by the Consent Management System in order to be digitally signed. 4. The interested party provides the digitally signed consent form to the Consent Management System.
Pass Metrics	<ol style="list-style-type: none"> 1. The Consent Management System obtains the digitally signed consent form from both parties.
Fail Metrics	<ol style="list-style-type: none"> 1. The Consent Management System cannot obtain the digitally signed consent form from both parties.

3.1.3.5 Use Case CMS-5: Consent form is entered into blockchain

Once the digitally signed consent form is available, the Consent Management System interacts with the blockchain infrastructure in order to create a new transaction that will be introduced in the blockchain.

In the case of the “once off” consent, where a specific validity period is defined, the Consent Management System is internally handling the validation of consent time period by creating and monitoring the specific timer in order to perform the validation of consent time period. Use Case 1.9 describes the handling performed when the consent time period expires.

Table 10: Consent Management Use Case CMS-5

Stakeholders involved:	N/A
Pre-conditions:	The digitally signed consent form from both parties is available.
Post-conditions:	A new transaction containing all the information of the consent form is available in the blockchain infrastructure.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	<ol style="list-style-type: none"> 1. The Consent Management System retrieves the digitally signed consent form from both parties. 2. The Consent Management System interacts with the blockchain infrastructure and enters the new consent form in the ledger in the form of a new transaction. 3. In the case of the “once off” consent the proper timer is started in the Consent Management System.

Pass Metrics	1. The new transaction containing all the information of the consent form is available in the blockchain infrastructure.
Fail Metrics	1. No new transaction is available in the blockchain infrastructure.

3.1.3.6 Use Case CMS-6: Consent update or withdrawal

The consent form can be updated or withdrawn at any time by any of the involved parties. In the case of the update, the previously described steps Use Case 1.3 to Use Case 1.5 are re-executed and the status of the consent remains “active”. On the other hand, the withdrawal of the consent is internally translated to “invalidation” of the smart contract and the status of the consent is set to “withdrawn”. In the case of the “once off” consent, the associated timer is restarted when the consent is updated. On the other hand, when the consent is withdrawn the associated timer is stopped.

In all these processes, the consent history is maintained in the blockchain infrastructure with the context of the smart contract. Through the transactions all versions of the consents and their complete update history is maintained.

Table 11: Consent Management Use Case CMS-6

Stakeholders involved:	Customer, Internal Financial Institution, Banking institutions, Peers, Financial organisations
Pre-conditions:	A transaction with status active containing all the information of the consent form is available in the blockchain infrastructure.
Post-conditions:	A new transaction containing the latest information of the consent form is available in the blockchain infrastructure.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	<ol style="list-style-type: none"> 1. The customer or the External Financial Institution initiates the consent update process by defining the consent form (in the case of withdrawal the corresponding status is defined). 2. The Consent Management System retrieves the new consent form and sends it to both the customer and the interested stakeholders. 3. The consent form is digitally signed by both parties and provided to the Consent Management System. 4. The Consent Management interacts with the blockchain infrastructure and introduces the updates with a new transaction. 5. In the case of the “once off” consent, the timer is either restarted (update) or stopped (withdrawal) in the Consent Management System.
Pass Metrics	1. A new transaction containing the latest information of the consent form is available in the blockchain infrastructure.
Fail Metrics	1. There is no new transaction with updated consent information.

3.1.3.7 Use Case CMS-7: Access Control based on the consent forms

During a data access request to the underlying data management system, the data management system is consulting the Consent Management System in order to validate the consent status between the requesting party and the customer whose data are requested. By utilising the transactions formulated in the previous steps, the Consent Management System is able to formulate the access control decision.

Table 12: Consent Management Use Case CMS-7

Stakeholders involved:	Internal Financial Institution, Banking institutions, Peers, Financial organisations
Pre-conditions:	A transaction containing all the information of the consent form is available in the blockchain infrastructure.
Post-conditions:	The access control is formulated based on the consent status contained in the latest transaction.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	<ol style="list-style-type: none"> 1. Upon a data access request, the data management system initiates a request to the Consent Management System to check the consent status between the customer and the requesting party. 2. The Consent Management System retrieves the relevant transaction from the blockchain infrastructure in order to validate the status of the consent for the requesting party. 3. The Consent Management System formulates the approval or denial decision and informs the data management system.
Pass Metrics	1. The data access requests are correctly validated.
Fail Metrics	1. The data access requests are incorrectly validated.

3.1.3.8 Use Case CMS-8: Retrieve complete history of consents

The stakeholders are able to be constantly informed of all the consents given to each specific recipient, as well as of the complete history of the consents. The transaction performed contains all the different versions of the consents of the customer or the external financial institution besides the latest one. In this sense, the stakeholders are able to retrieve at any time his/her consent history per specific stakeholder, for all stakeholders or all the consents given by the customers for a stakeholder.

Table 13: Consent Management Use Case CMS-8

Stakeholders involved:	Customer, Internal Financial Institution
Pre-conditions:	At least a transaction containing all the information of the consent form is available in the blockchain infrastructure.
Post-conditions:	The customer is able to retrieve the complete history of consents by the Consent Management System.

Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	<ol style="list-style-type: none"> 1. The stakeholder initiates a request to the Consent Management System to retrieve the complete list consents (active, withdrawn, expired) per specific stakeholder along with their history or for a specific stakeholder. 2. The Consent Management System interacts with the blockchain infrastructure and retrieves the relevant transactions in order to compile a list of consents that customer has granted. 3. The stakeholder is able to check the requested list of consents in a user-friendly way through the web-based interface provided by the Consent Management System.
Pass Metrics	<ol style="list-style-type: none"> 1. The stakeholder retrieves the request list of consent he/she has granted.
Fail Metrics	<ol style="list-style-type: none"> 1. The stakeholder is able to retrieve the request list of consent he/she has granted or has been granted.

3.1.3.9 Use Case CMS-9: Expiration of the validity period

In the case of the “once off” consent, the validity period is set to a predefined time period. The moment that a transaction is created in the blockchain infrastructure for this specific type of consent, the Consent Management System creates and monitors the specific timer in order to perform the validation of consent time period. Once this timer is expired, the Consent Management System creates a new transaction for the specific consent with the status set to “expired”.

Table 14: Consent Management Use Case CMS-9

Stakeholders involved:	N/A
Pre-conditions:	A transaction with status active containing all the information of the consent form is available in the blockchain infrastructure whose timer has expired.
Post-conditions:	A new transaction containing the latest information of the consent form is available in the blockchain infrastructure with status set to “expired”.
Data Attributes	Required data based on the Consent Management Data Schema
Normal Flow	<ol style="list-style-type: none"> 1. Upon the expiration of the validity timer, the Consent Management System retrieves the latest transaction for the specific consent by interacting with the blockchain infrastructure. 2. The Consent Management introduces the updates with a new transaction where the status is set to “expired”.
Pass Metrics	<ol style="list-style-type: none"> 1. A new transaction containing the latest information of the consent form is available in the blockchain infrastructure.
Fail Metrics	<ol style="list-style-type: none"> 1. There is no new transaction with updated consent information.

3.1.4 Sequence Diagrams

In the previous section, all the relevant use cases of the designed blockchain application were documented. The following sections present the sequence diagrams for each use case, depicting the interactions between the stakeholders and the components of the designed solution, as well as the interactions between the various components of the designed solution.

3.1.4.1 Register Customer in the Consent Management System

In Use Case CMS-1, the customer of the internal financial institution is registered in the Consent Management System by the administrator in order to create the associated profile that will receive consent requests by interacting with the web interface of the Consent Management System and providing the profile details. In the same manner, the representative of the external financial institution or the peer is registered by the administrator in the Consent Management System in order to create the associated profile that will initiate consent requests to the internal financial institution in order to get access to the internal financial institution’s customer’s data.

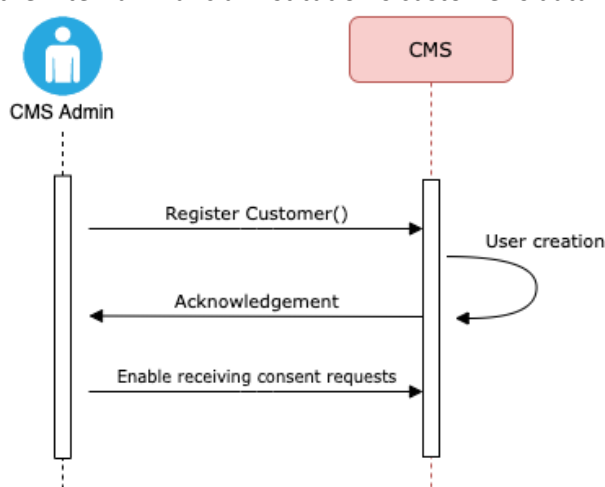


Figure 3: Use Case CMS-1 sequence diagram (customer)

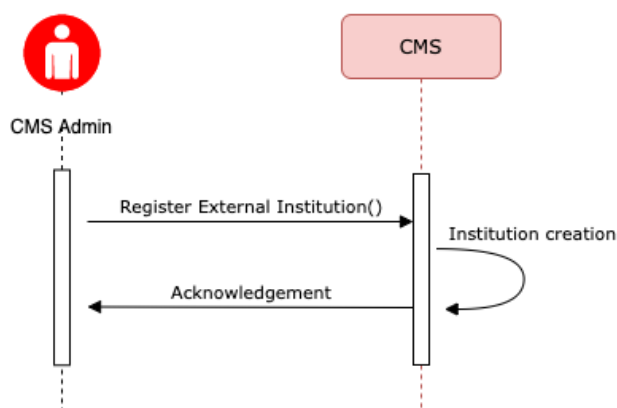


Figure 4: Use Case CMS-1 sequence diagram (external financial institution)

3.1.4.2 Customer receives a request to provide new consent for sharing his/her customer data

In Use Case CMS-2, the external financial institution or peer initiates a request to receive the customer’s consent in order to access his/her data. Upon the approval of the internal financial institution’s administrator, the Consent Management System interacts with blockchain components

in order to check the existence of a consent in the ledger by querying and reading the query results upon decryption. In the case of absence of a valid and active consent, the Consent Management System is informed to transmit the new request to the customer. In case of existence of a valid and active consent, the Consent Management System is informed and approves the request to access the requested data.

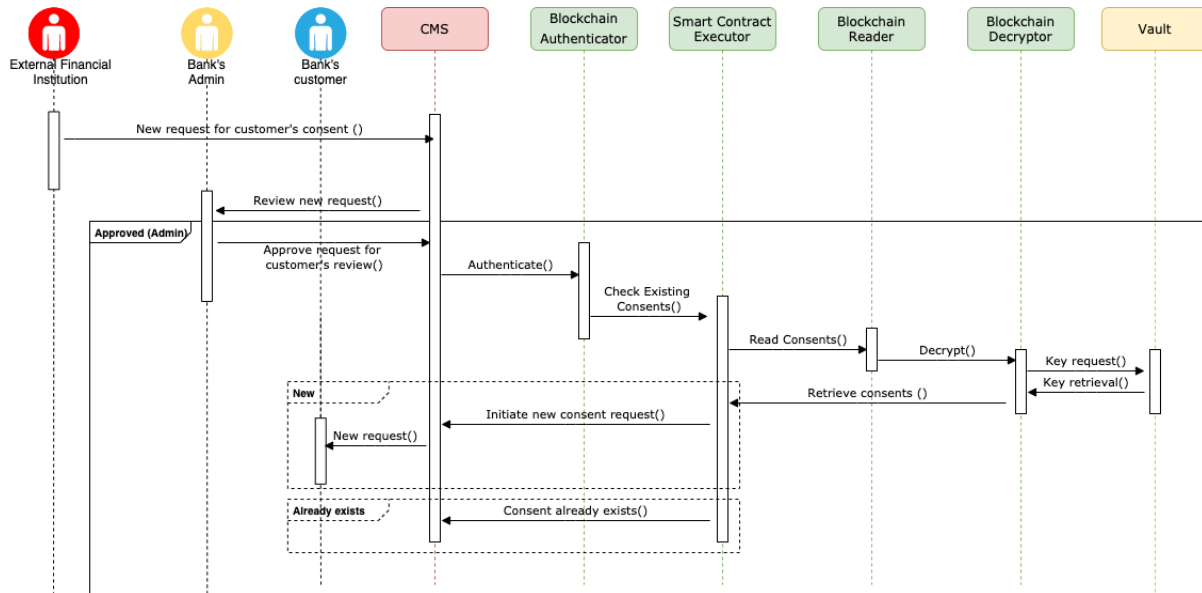


Figure 5: Use Case CMS-2 sequence diagram

3.1.4.3 Definition of the consent

In Use Case CMS-3, the customer receives and reviews the request and in the case of approval, defines the details and conditions of the consent in the Consent Management System and the candidate consent form is created. In the case of denial, the Consent Management System informs the external financial institution accordingly.

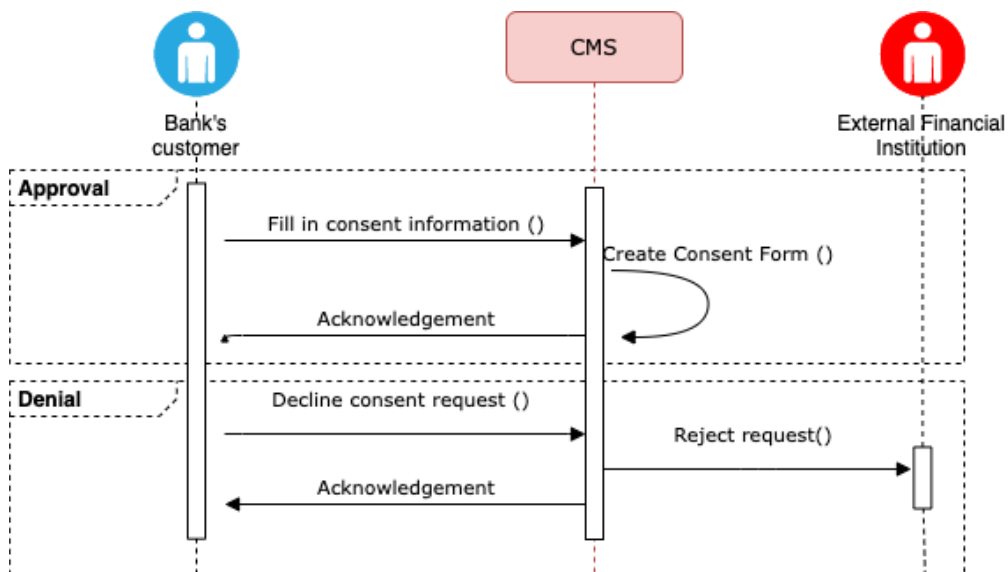


Figure 6: Use Case CMS-3 sequence diagram

3.1.4.4 Signing of the consent by the interested parties

In Use Case CMS-4, the Consent Management System provides the candidate consent form to both the customer and external financial institution and collects the digitally signed consent from both parties.

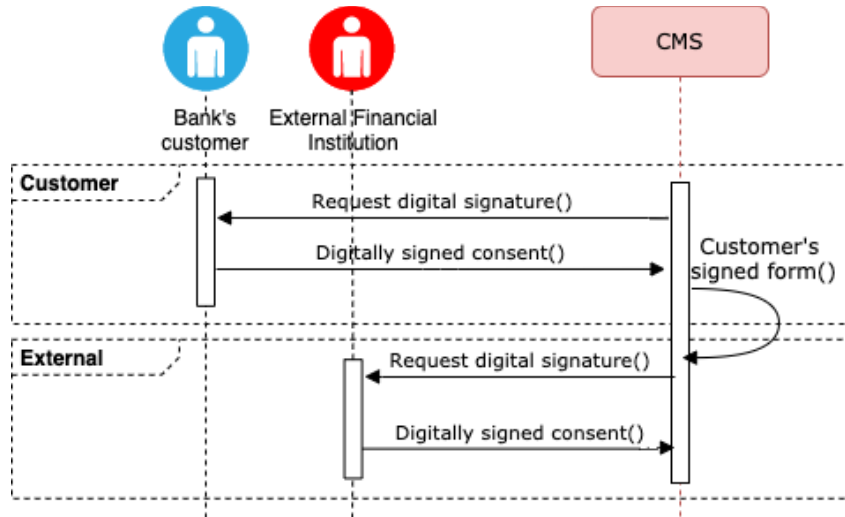


Figure 7: Use Case CMS-4 sequence diagram

3.1.4.5 Consent form is entered into blockchain

In Use Case CMS-5, the Consent Management System interacts with the blockchain components in order to write the new consent by executing the respective chaincode, encrypting the transaction and writing the new transaction in the ledger. In the case of “once off” consent, the Consent Management System initiates the internal timer. Finally, the Consent Management System stores the formulated consent form in the File Storage.

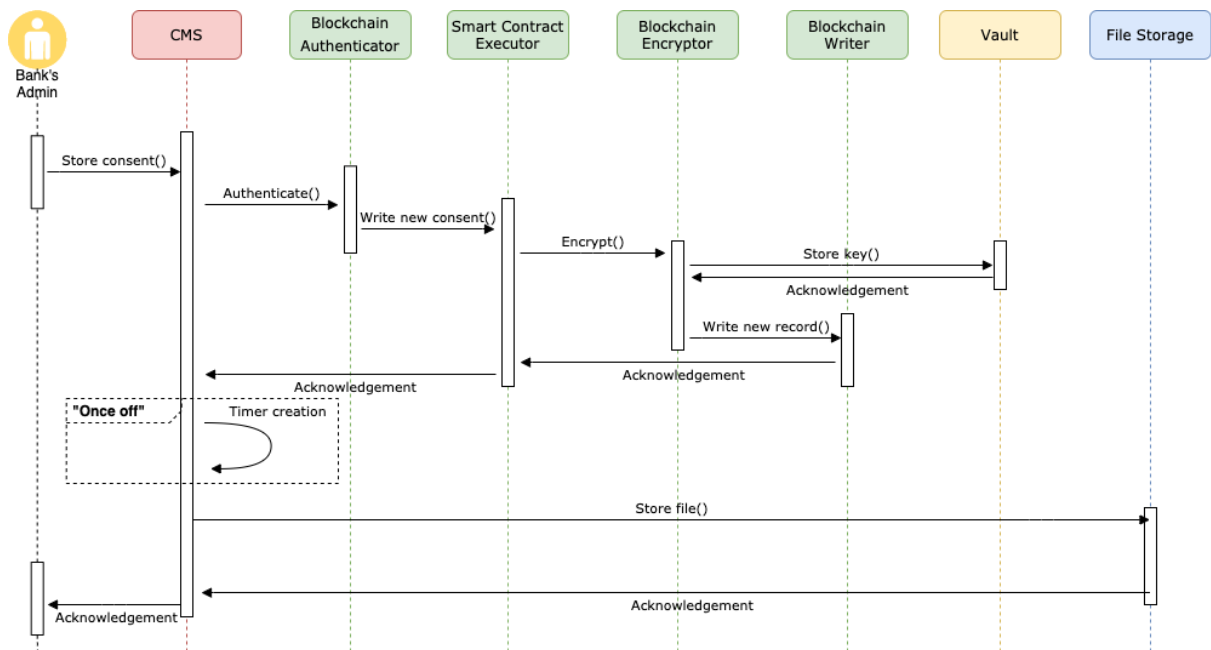


Figure 8: Use Case CMS-5 sequence diagram

3.1.4.6 Consent update or withdrawal

In Use Case CMS-6, the update or withdrawal at any time by any of the stakeholders of the consent is handled. In the case of an update, the Consent Management System orchestrates the execution of the sequence diagrams described for Use Case CMS-3 till Use Case CMS-5 in order to generate a new transaction based on the updated consent form that is digitally signed by both parties. In the case of withdrawal, the Consent Management System interacts with the blockchain components in order to retrieve the existing consent from the ledger and update the consent status through a new transaction in the ledger that depicts the new status. The following sequence diagram displays this use case which remains the same when the request is initiated by an external financial institution.

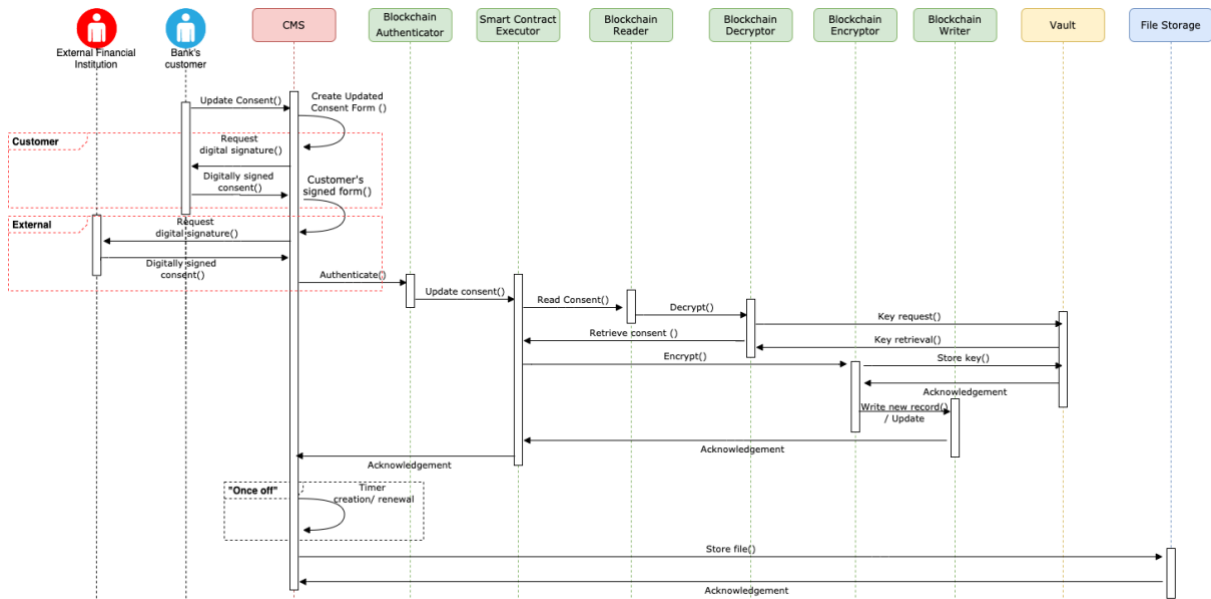


Figure 9: Use Case CMS-6 sequence diagram (update)

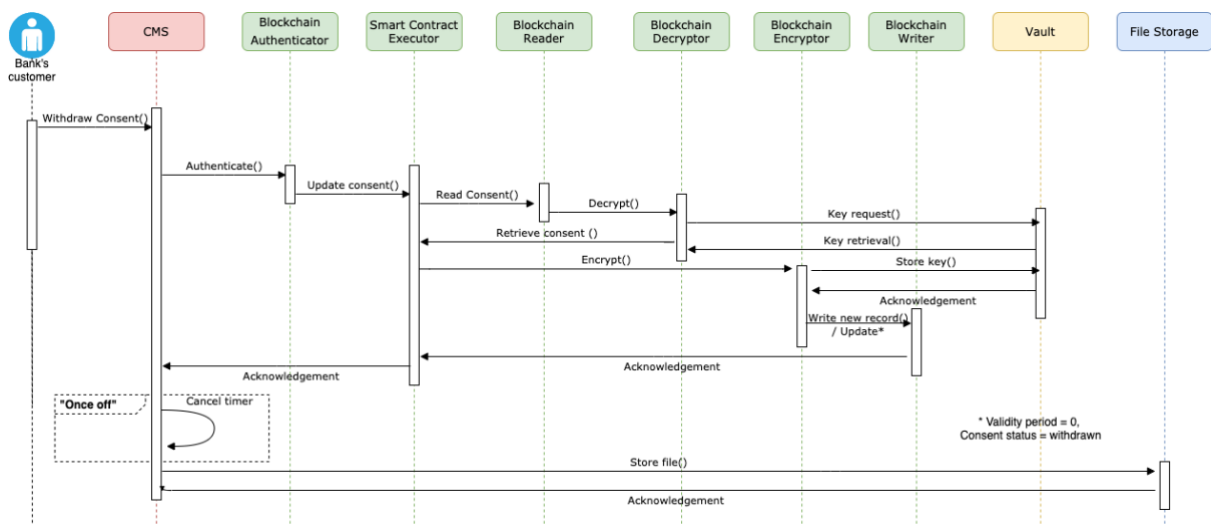


Figure 10: Use Case CMS-6 sequence diagram (withdrawal)

3.1.4.7 Access Control based on the consent forms

In Use Case CMS-7, the Consent Management System receives a new data access request and formulates an access control decision based on the existence of a valid and active consent. To achieve

this, it interacts with the blockchain components in order to query the ledger for the existence of a consent between the involved parties for the specific data.

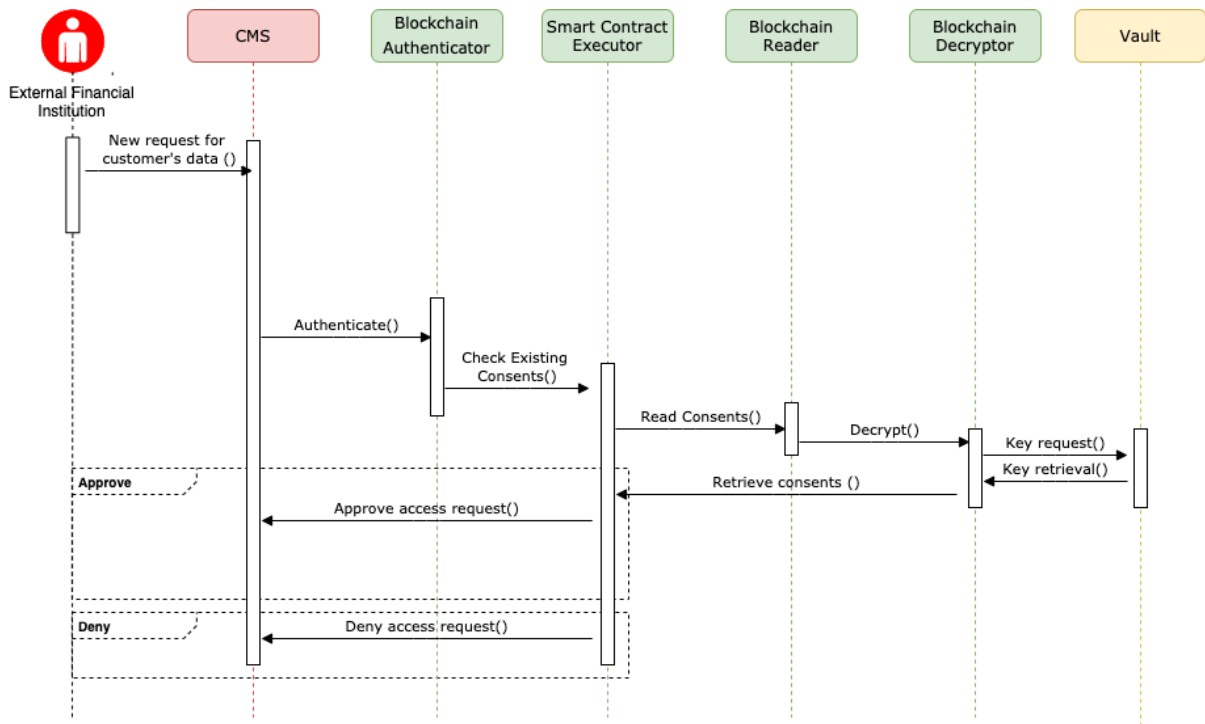


Figure 11: Use Case CMS-7 sequence diagram

3.1.4.8 Retrieve complete history of consents

In Use Case CMS-8, the Consent Management System receives a new request to retrieve all the consents that a stakeholder has given to each specific recipient of his/her data or obtained along with their completed history. To achieve this, the Consent Management System interacts with the blockchain components in order to query the ledger and retrieve the required information. The following sequence diagram displays this use case which remains the same when the request is initiated by an external financial institution.

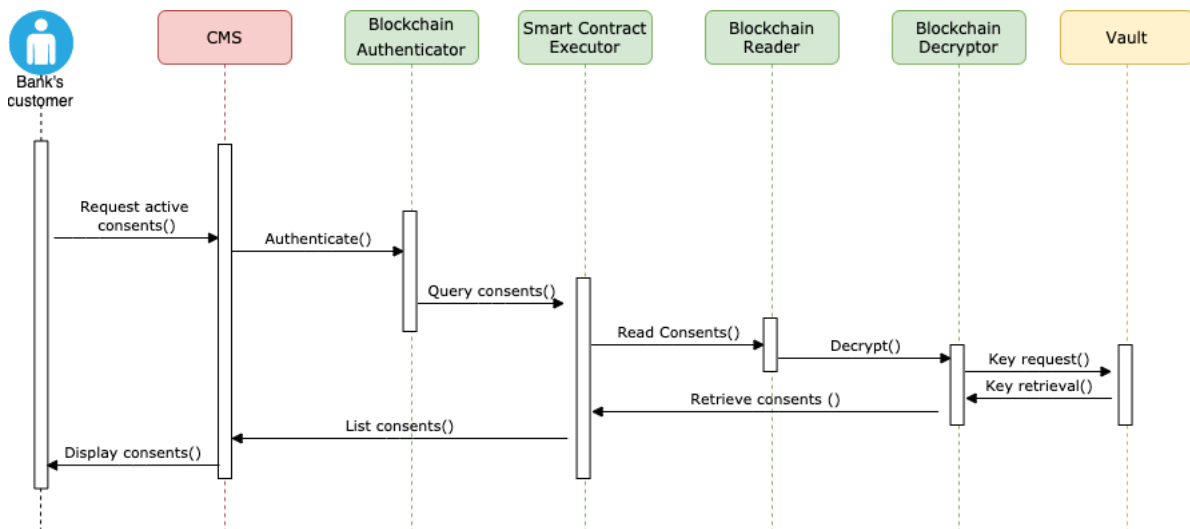


Figure 12: Use Case CMS-8 sequence diagram

3.1.4.9 Expiration of the validity period

In Use Case CMS-9, before the validity period of a “once off” consent expires, the Consent Management System informs the customer in order to initiate an update of the consent as described in Use Case CMS-6. In the case where the timer expires, the Consent Management System interacts with the blockchain components to retrieve the consent and update the consent status via a new transaction.

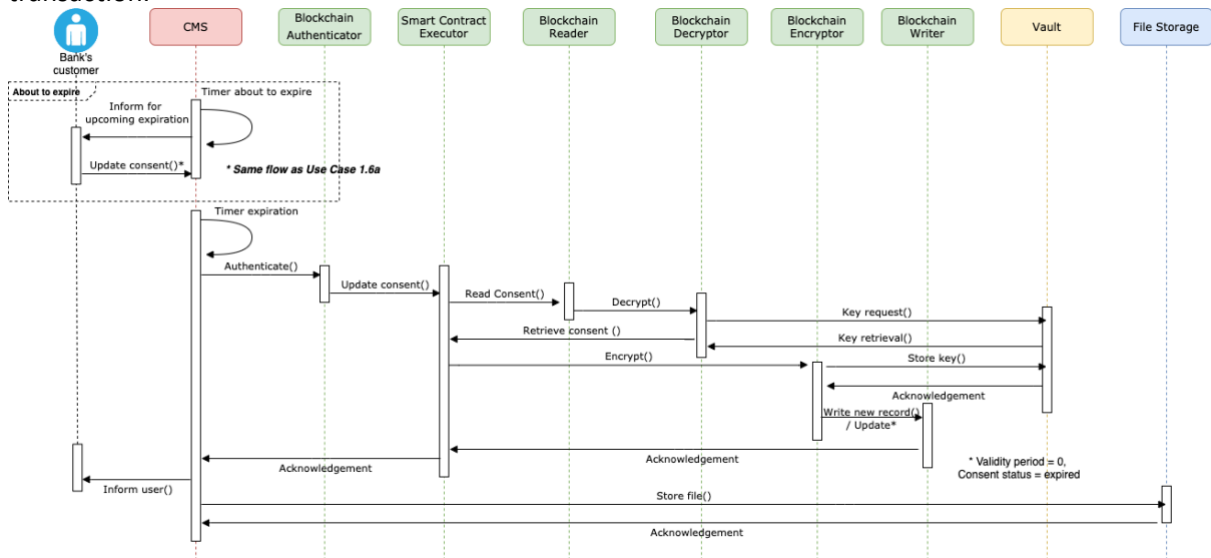


Figure 13: Use Case CMS-9 sequence diagram

3.1.5 Implementation of the Consent Management

In this section, the implementation details of the final version of the Consent Management are presented providing the necessary updates from the initial documentation that was provided in deliverable D4.8. The designed and implemented version constitutes the fully-functional version which implements the complete set of the use cases described in Section 3.1.2. The initial prototype version that was presented in deliverable D4.8 has been enhanced in order to: a) support the additional use cases as per the design specifications documented in the previous section that were not included in the initial prototype and b) the enhancements introduced on the existing use cases in order to better address the requirements of the already implemented use cases. The scope of the current section is to present in a clear and coherent way the final implementation details of the fully-functional version by performing a walkthrough of the implemented functions and the structure of the source code.

Based on the design specifications of the Sections 3.1.2 and 3.1.3, as well as the data schema defined in Section 3.1.1, the chaincode is structured at a component level as depicted in Figure 14.

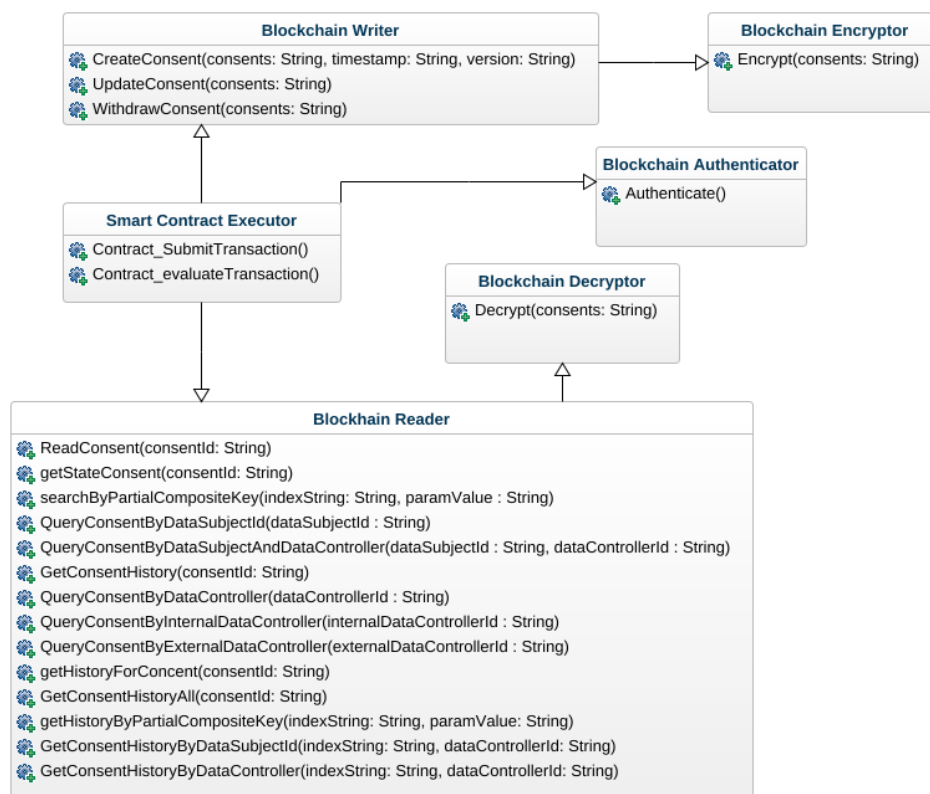


Figure 14: Consent Management chaincode structure

The *Blockchain Writer* component undertakes the responsibility to submit new transactions to the blockchain ledger. In the Consent Management case, this refers to transactions for a new consent, an updated consent or a withdrawn consent, and it is implemented with the following functions:

- `CreateConsent(consents string)`: The specific function is performing the necessary actions in order to create a new consent receipt into the ledger by updating the world state with a new record and appending new blocks at the end of the ledger. The function receives the data containing the consent information, constructs them into the appropriate format and interacts with the `Encrypt()` function to encrypt them. Once the data are encrypted they are inserted into the ledger.
- `UpdateConsent(consents string)`: The specific function undertakes the responsibility of updating an existing consent receipt into the ledger by updating the world state with a new record and appending new blocks at the end of the ledger. The function receives the updated consent information, constructs it into the appropriate format and encrypts the information via the `Encrypt()` function. Finally, the encrypted information is appended into the ledger.
- `WithdrawConsent(consents string, timestamp string, version string)`: In the same manner as with the `UpdateConsent` function, the specific function withdraws an existing consent by updating the world state with a new record and appending new blocks at the end of the ledger. The new information is also encrypted via the `Encrypt()` function before it is appended into the ledger.

The *Blockchain Reader* component provides the necessary functions to read the ledger state, fetch a particular consent by id, query the ledger state given different parameters to fetch a set of consents matching those parameters, and get the consents history (history of all updates for particular consent data entity or data entities). The implemented functions are as follows:

- `ReadConsent (consentId string)`: This function is responsible for reading the consent record with the given `consentId` from the ledger, interacting with the `Decrypt()` function to decrypt the consent information, converting it to the appropriate format and returning it to the invoker.
- `getStateConsent (consentId string)`: This is a helper function used in `ReadConsent()`, `searchByPartialCompositeKey()`, and the query functions to read a given consent record (identified by 'consentId' value) from the ledger.
- `searchByPartialCompositeKey (indexString string, paramValue string)`: This is a helper function intended to search the ledger by a partial composite key of consent entities. When storing the consent record in the `CreateConsent()` function, two composite keys are created and stored in the ledger along with the consent data entity itself: a composite key comprised of combining the 'dataControllerId' with 'consentId', and a composite key comprised of the combination of 'dataSubjectId' and 'consentId'. This enables fast searches of all consent records serviced by particular `dataControllerId`, or belonging to particular `dataSubjectId`. The function receives the particular index it needs to search the ledger by (either the `dataSubjectId` index or `dataControllerId` index), the partial value of the composite key to use in the search (the particular `dataSubjectId` or `dataControllerId`), and it returns a list of decrypted consent records matching this search.
- `QueryConsentByDataController (dataControllerId string)`: This function uses the `searchByPartialCompositeKey()` function to search for all consents belonging to the specified `dataControllerId`. It returns an array of matching decrypted consent entities.
- `QueryConsentByDataSubjectId (dataSubjectId string)`: This function uses the `searchByPartialCompositeKey()` function to search for all consents belonging to the specified `dataSubjectId`. It returns an array of matching decrypted consent entities.
- `QueryConsentByInternalDataController (internalDataControllerId string)`: This function uses the `searchByPartialCompositeKey()` function to search for all consents belonging to the specified `internalDataControllerId`. It returns an array of matching decrypted consent entities.
- `QueryConsentByExternalDataController (externalDataControllerId string)`: This function uses the `searchByPartialCompositeKey()` function to search for all consents belonging to the specified `externalDataControllerId`. It returns an array of matching decrypted consent entities.
- `QueryConsentByDataSubjectAndDataController (dataSubjectId string, dataControllerId string)`: This function uses the `searchByPartialCompositeKey()` function to search for all consents belonging to the specified `dataSubjectId` and serviced by the specified `dataControllerId`. It returns an array of matching decrypted consent entities.
- `GetConsentHistory (consentId string)`: This function utilizes the Fabric's built-in functionality to fetch the transaction history for the particular consent entity given by `consentId` value. It returns an array of decrypted consent records, each representing an updated state of the particular consent entity, along with the `txId` (transactionId) which updated it and the timestamp from when it was updated.
- `getHistoryForConcent (consentId string)`: This function is a helper function which retrieves the complete history of a consent with the `consentId` value. It returns an array of matching decrypted consent updates.
- `GetConsentHistoryAll (consentId string)`: This function utilizes also the Fabric's built-in functionality to fetch the transaction history for the all consents available on the system. It returns an array of decrypted consent records, each representing an updated state of each particular consent entity, along with the `txId` (transactionId) which updated it and the timestamp from when it was updated.

- `getHistoryByPartialCompositeKey(indexString string, paramValue string)`: This function is a helper function which returns an array of history records for a particular search string.
- `GetConsentHistoryByDataSubjectId(indexString string, dataSubjectId string)`: This function is a helper function which returns an array of history records for a particular `dataSubjectId`.
- `GetConsentHistoryByDataController(indexString string, dataControllerId string)`: This function is a helper function which returns an array of history records for a particular `dataControllerId`.

The *Smart Contract Executor* component provides the necessary functions for the execution of the smart contract on the blockchain ledger. In the Consent Management case, this refers to all the operations related to the read and write operations which are performed by the respective functions of the Blockchain Writer and Blockchain Reader components. The implemented functions are as follows:

- `Contract_SubmitTransaction()`: The specific function is performing the necessary actions for the execution of the smart contract by invoking the respective functions of the Blockchain Writer (*CreateConsent()*, *UpdateConsent()*, *WithdrawConsent()*) along with their required parameters based on the received request.
- `Contract_evaluateTransaction()`: The specific function is performing the necessary actions for the execution of the smart contract by invoking the respective functions of the Blockchain Reader (*ReadConsent*, *GetConsentHistory*, *QueryConsentByDataSubjectId*, *QueryConsentByDataSubjectAndDataController*,) along with their required parameters based on the received request.

The *Blockchain Authenticator* component is the component responsible for the authentication operations which are performed in order to ensure the controlled access of the respective users to the blockchain channel utilised by the Consent Management case. For the specific component, the following function is implemented:

- `Authenticate()`: The specific function performs the authentication of the user requesting access to the underlying blockchain channel. In the case where the appropriate keys and certificates are provided by the user, access to the specific channel is granted.

The *Blockchain Encryptor* component is the component that undertakes the responsibility for the encryption operations which are performed on the consent data prior to being inserted as new transactions to the blockchain ledger. For the specific component, the following function is implemented:

- `Encrypt (consents string)`: The specific function is performing the necessary actions in order to encrypt the consent data utilising AES 256 encryption. The function receives the data containing the consent information, encrypts them with a randomly generated encryption key which is later stored in a Vault.

The *Blockchain Decryptor* component is the component responsible for the decryption of the consent data which were encrypted by the *Blockchain Encryptor*. Hence, the specific component decrypts the encrypted data when they are read from the blockchain ledger. For the specific component, the following function is implemented:

- `Decrypt (consents string)`: The specific function is performing the necessary actions in order to decrypt the consent data when they are fetched from the ledger. The function receives the encrypted data containing the consent information and decrypts them with the appropriate encryption key as retrieved from the Vault.

As explained also in the architecture of the solution, the Consent Management System is the mediator between the involved parties, namely the external financial institution or peer, the internal financial institution and the customer of the internal institution. It interacts with the underlying blockchain infrastructure where the chaincode resides and provides all the required functionalities to the involved stakeholder in order to formulate the consent. To this end, the Consent Management System is composed of a set of services, as depicted in Figure 15, which are interacting via well-defined APIs in order to provide the required functionalities.



Figure 15: Consent Management System source code structure

The `ConsentManagementService` is a core service of the Consent Management System and is responsible for the main operations performed during the consent formulation. In detail, the service undertakes the creation, update or editing, as well as the deletion of a consent request, enabling the formulation of the consent through the various steps described in the use cases reported in the Section 3.1.2. The specific service is responsible for the formulation of the consent prior to being inserted into the blockchain ledger. Finally, it interacts with the `BlockchainService` in order to perform all the required operations on the underlying blockchain infrastructure. The specific service exposes the `ConsentManagementRestController` through which the various interactions with the rest of the services are realised. The implemented functions are as follows:

- `createConsent(consentEntity: Object)`: The specific function creates the new consent request based on the input received and stores it in the Consent Management System's database.
- `deleteConsent(consentId: Long)`: The specific function deletes an existing consent request from the Consent Management System's database.
- `fetchAllConsentsHistoryFromBlockChainHashMap()`: The specific function invokes `fetchAllConsentsHistoryFromBlockChain` function of the Blockchain service in order to retrieve the history of all consents written in the blockchain ledger. The result is returned in a

HashMap. The key of the HashMap is the consentReceiptId of every consent. The result is cached, in order to avoid querying blockchain ledger consecutively.

- `fetchConsentsFromDbByUser(accessToken: String)`: The specific function, given the user's access token, returns the consent requests that are stored in Consent Management System's storage. The role and the id of the logged-in user are extracted from the given access token. In case of a customer, the function fetches all the received consent requests that a specific customer of the internal financial institution has received. In case of a bank's superuser or branch, the function fetches all consent requests in which the logged-in entity participates as the internal/external financial institution.
- `fixExpiration()`: The specific function is executed daily at 10:00 am. It serves two main goals. Firstly, it locates all consents that are stored in the blockchain ledger and are about to expire (in less than 7 days), sending a respective notification to the involved entities. Secondly, for all the consents that have expired, it updates the blockchain ledger and then notifies the involved entities for the status change.
- `withDrawConsent(consentReceiptId: String)`: The specific function invokes the *withDrawConsent* function of the Blockchain Service in order to withdraw the consent identified by the provided consentReceiptId.
- `updateConsent(consentReceiptId: String, consentDto: Object)`: The specific function invokes the *updateConsent* function of the Blockchain Service in order to update a consent in the blockchain ledger, which is identified by consentReceiptId.
- `fetchConsentById(consentId: Long)`: The specific function fetches an existing consent request from the Consent Management System's storage based on the provided ConsentId.
- `editConsent(consentEntity: Object)`: The specific function modifies an existing consent request in the Consent Management System's storage based on the consent request data provided.
- `changeStatusToPreApproved(consentId: Long)`: The specific function updates the status of an existing consent request to PreApprove state. This operation is performed when the administrator of the internal financial institution pre-approves the newly received request from an external financial institution and as a consequence the request reaches the customer for review and approval.
- `changeStatusToReviewed(consentId: Long)`: The specific function updates the status of an existing consent request to Reviewed state. This operation is performed when the customer reviews the received consent requests, modifies it or approves it.
- `changeStatusToRejected(consentId: Long)`: The specific function updates the status of an existing consent request to Rejected state. This operation is performed when the customer reviews the received consent requests and rejects it.
- `changeStatusToApproved(consentId: Long)`: The specific function updates the status of an existing consent request to Approved state. This operation is performed when the external financial institution receives the reviewed and approved by the customer consent requests and approves it also.
- `fetchConsentsForBank(accessToken: String)`: The specific function firstly extracts from the specified access token, the user's id and role. Secondly, it retrieves from the Consent Management System's storage, all the consent requests in which the specified financial institution is involved. Thirdly, it invokes *fetchFromBlockchainByUserIDAndRole* function of the Blockchain Service in order to fetch all consents of the banking entity that are stored in the blockchain ledger. The results of the second and the third step are concatenated, cached and returned to the user.
- `fetchConsentsForCustomer(accessToken: String)`: The specific function firstly extracts from the specified access token, the customer's id. Secondly, it retrieves from the Consent Management System's storage, all the consent requests in which the given customer

is involved. Thirdly, it invokes *fetchFromBlockChainByUserIDAndRole* function of the Blockchain Service in order to fetch all consents of customer that are stored in the blockchain ledger. The results of the second and the third step are concatenated, cached and returned to the user.

- *fetchFromBlockchainByConsentId*(consentId: String): The specific function invokes the *fetchFromBlockchainByConsentId* function of the BlockchainService in order to retrieve a formulated consent from the blockchain ledger.

The BlockchainService is another core service of the Consent Management System that is responsible for the insertion, modification and retrieval of the consents to or from the blockchain ledger. While the actual consent is formulated by the ConsentManagementService starting from the initial consent request, once the consent has been formulated the BlockchainService is invoked to insert the formulated consent in the blockchain ledger. On the other hand, when the consents are displayed to the involved stakeholders, the BlockchainService is interacting with underlying blockchain infrastructure in order to retrieve the required information. Finally, any updates performed on the consent information are inserted into the blockchain ledger. The implemented functions are as follows:

- *finalizeConsent*(BlockchainConsentDTO: Object): The specific function receives the consent information as formulated by the ConsentManagementService and interacts with the blockchain infrastructure in order to insert it into the blockchain ledger.
- *fetchAllConsentsHistoryFromBlockChain*(): The specific function fetches the history of all consents, stored in the Blockchain ledger for the entity that logged in Consent Management System.
- *fetchFromBlockchainByConsentId*(consentId: String): The specific function retrieves a specific consent from the blockchain ledger based on the provided consentId.
- *updateConsent*(consentReceiptId: String, consentDto: Object): The specific function updates a consent in the blockchain ledger, which is identified by consentReceiptId. The update is based on the consentDto object. In addition, the cache entries of the three entities that are involved in the specific consent are evicted.
- *withdrawConsent*(consentReceiptId: String): The specific function withdraws a consent, which is identified by consentReceiptId, from the blockchain ledger. In addition, the cache entries of the three entities that are involved in the specific consent are evicted.
- *fetchFromBlockChainByUserIdAndRole*(userId: String, role:String): The specific function, given a userId and role, fetches from the blockchain ledger all consents that concern the specific entity. The result is cached, in order to rapidly fetch the necessary information from the cache, in case of potential refresh or changes that do not involve the ledger (i.e. creation of a new consent request).

The UserManagementService is responsible for the user management operations with regards to the users of internal financial institution, the customers of the internal financial institution and the users of the external financial institution. In this sense, it provides all the functionalities to create a financial institution, a customer or an external financial institution, as well as update their profile information at any time. The UserManagementService is leveraging the Keycloak technology that effectively covers all the user management and authorisation operations. The UserManagementService interacts with Keycloak (see section 5) via the following functions:

- *loginKeycloak*(loginDTO: Object): The specific function is responsible for the login in the Consent Management System. The user needs to specify his email and password, in the context of the LoginDTO object. The function uses the aforementioned credentials to obtain a JSON Web Token (JWT) authorization token from Keycloak.

- `getAuthKeycloakUser(accessToken: String)`: The specific function, given the JWT authorization token, which is returned by Keycloak when the user logs in, returns a User object that is utilized from the various functionalities of the Consent Management System.

3.1.6 Consent Management Application overview

As described in sections 3.1.3 and 3.1.4, the Consent Management System plays the role of the mediator between the involved parties (internal financial institutions, customers of the internal financial institutions and the external financial institutions) providing a mechanism for the complete lifecycle management of digital consents. It leverages the benefits of the blockchain technology offering at the same time a level of abstraction above the purely backend operations executed on the underlying distributed ledger. To this end, the Consent Management System via its novel and easy-to-use user interface enables the creation, update and withdrawal of consents, the retrieval of the complete history of the established consents and the access control assessment based on the formulated consents.

Upon the successful login, the users are presented with the list of consents where for each consent the relevant information is displayed. In particular, for each consent the following information is displayed: a) the unique id of the consent, b) the id of the external financial institutions which issued the request for the customer's consent, c) the id of the customer, d) the timestamp of the last performed action, e) the consent's status (Received, Reviewed, Active, Withdrawn, Rejected, Expired) and f) a set of actions such the view of the details of the consent and the complete history of the consent. Depending on the role of the user (customer or internal/external financial institution) a set of dynamic filters are also available in order to enable the effective filtering of the list of consents. In addition to this, all users are informed via real-time notifications constantly on any updates performed, such as the reception of a new consent request, the acceptance, update or withdrawal of consents by any involved party (Figure 16, Figure 17).

Consents

Consent Requests Active Consents Action Needed Consents

Consent Id	External User ID	Customer SSN	Customer Full Name	Date Updated	Status	Action
14a45549be	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/22	Rejected	
130bbf349b	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/22	Reviewed	
0a3375b7d4	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/03	Withdrawn	
0b5849be08	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/25	Withdrawn	
392ca2dab1	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/30	Withdrawn	
3af8451576	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/22	Expired	
44bfec0db2	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/01	Withdrawn	
4e02e38ea8	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/25	Expired	
54fc2f5163	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/30	Withdrawn	
5ac6ba4036	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/30	Withdrawn	

Items per page: 10 1 - 10 of 24 << < > >>

Figure 16: Consent Management System – List of consents (customer view)

D4.9 – Permissioned Blockchain for Finance and Insurance - III

The screenshot displays the 'Consents' management interface. At the top, there is a search bar for 'Consent Id' and a '+ Consent' button. Below the search bar are tabs for 'Consent Requests', 'Active Consents', 'My Consents', and 'Requested'. The main area contains a table with the following columns: Consent Id, External User ID, Customer SSN, Customer Full Name, Date Updated, Status, and Action. The table lists 10 consent records with various statuses such as 'Received', 'Rejected', 'Reviewed', 'Withdrawn', and 'Expired'. At the bottom right, there is a pagination control showing 'Items per page: 10' and '1 - 10 of 33'.

Consent Id	External User ID	Customer SSN	Customer Full Name	Date Updated	Status	Action
a6b6ed9de5	2021d870-659b-498e-963b-fc5e3c0ec045	123789	cust omer02	2021/11/17	Received	👁
70e83c9ab3	2021d870-659b-498e-963b-fc5e3c0ec045	123789	cust omer02	2021/11/17	Received	👁
1cb25ac0fe	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/22	Rejected	👁
14a45549be	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/22	Rejected	👁
130bbf349b	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/22	Reviewed	👁
904e159999	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/29	Rejected	👁
5a65c3e20b	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/12/02	Received	👁
0a3375b7d4	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/03	Withdrawn	👁 🔄
0b5849be08	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/25	Withdrawn	👁 🔄
0b94c187d1	2021d870-659b-498e-963b-fc5e3c0ec045	123789	cust omer02	2021/12/03	Expired	👁 🔄

Figure 17: Consent Management System – List of consents (financial institution's view)

The external financial institution can initiate a consent request by pressing the “Consent” red button on the right upper corner. At this point, a dynamic form is displayed to the user where all the details of the consent request should be filled-in before this request is issued to the internal financial institution for pre-approval (Figure 18).

Upon the creation of a new consent request, the internal financial institution is informed. The user of the internal financial institution is then able to review the new request and pre-approve it in order to reach the customer for the actual review. In addition to this, the user of the internal financial institution might reject the new request before it reaches the customer (Figure 19).

Once the new consent request has been pre-approved by the internal financial institution, the customer is informed of the new consent request via a new notification. The customer can select the new consent request (listed with the status Reviewed) and can view the details of the new consent request. The customer is able to change the terms and conditions of the requested consent, review it and finally digitally sign it in order to be sent to the external financial institution for digital signing also (Figure 20).

At the final step of the consent formulation, the external financial institution is notified for the response of the customer. Then, the external financial institution can review and finally accept or reject the final terms and conditions as set by the customer. If the terms and conditions are accepted, the consent is created and stored into the underlying blockchain infrastructure (Figure 21).

At any point, the users can initiate an update of an existing consent by selecting the desired consent, modifying the terms and conditions and sending the update request to other involved party (Figure 22). The updated consent has to be reviewed, accepted and signed by both parties in order to be valid and to be written in the blockchain. This will result into a new version of the consent also.

The screenshot displays the 'Create New Consent Request' form within the Infinitech Consent Management System. The form is overlaid on a sidebar containing a list of consent requests with their IDs. The form fields are as follows:

- SSN ***: A text input field with a red underline.
- Policy Url ***: A text input field.
- Validity Type**: Radio buttons for **Permanent** (selected) and **Once Off**.
- Sensitive**: Radio buttons for **True** (selected) and **False**.
- Sensitive Personal Data Categories: ***: A dropdown menu.
- Services**: A section containing:
 - Service**: A text input field for **Service Name ***.
 - Purposes for service**: A section containing a text input field for **Purpose Name ***.

At the bottom right of the form, there are three buttons: **Close**, **Save as Draft**, and **Submit**. The background sidebar shows a list of consent requests with IDs such as a6b6ed9d, 70e83c9a, 1cb25ac0, 14a45549, 130bbf34, 904e1599, 0a3375b7, 0b5849be, 0b94c187, and 1e208106. A '+ Consent' button is visible in the top right of the sidebar. The top navigation bar shows the Infinitech logo and the user 'nbg_master'.

Figure 18: Consent Management System – New consent request

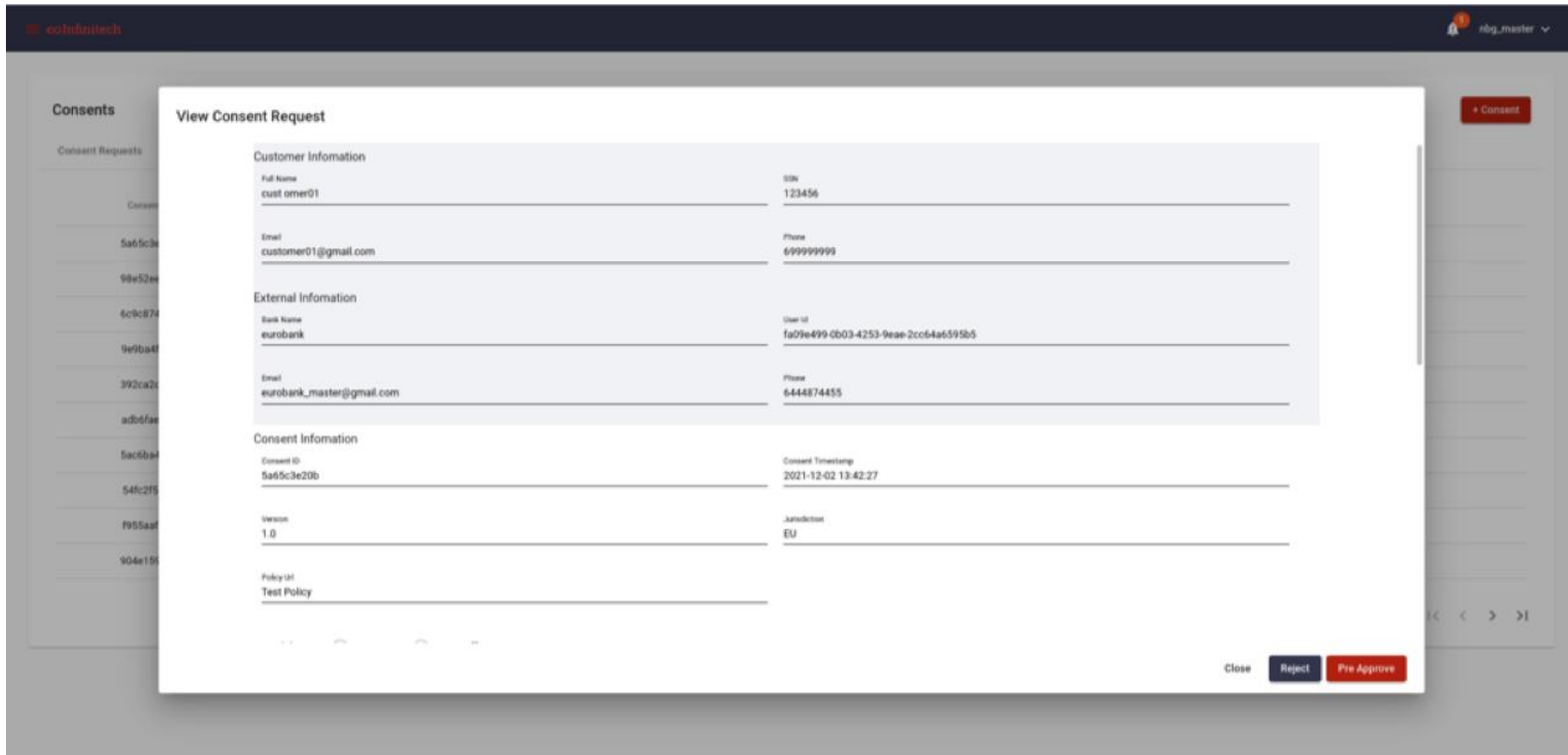


Figure 19: Consent Management System – New request pre-approval or rejection

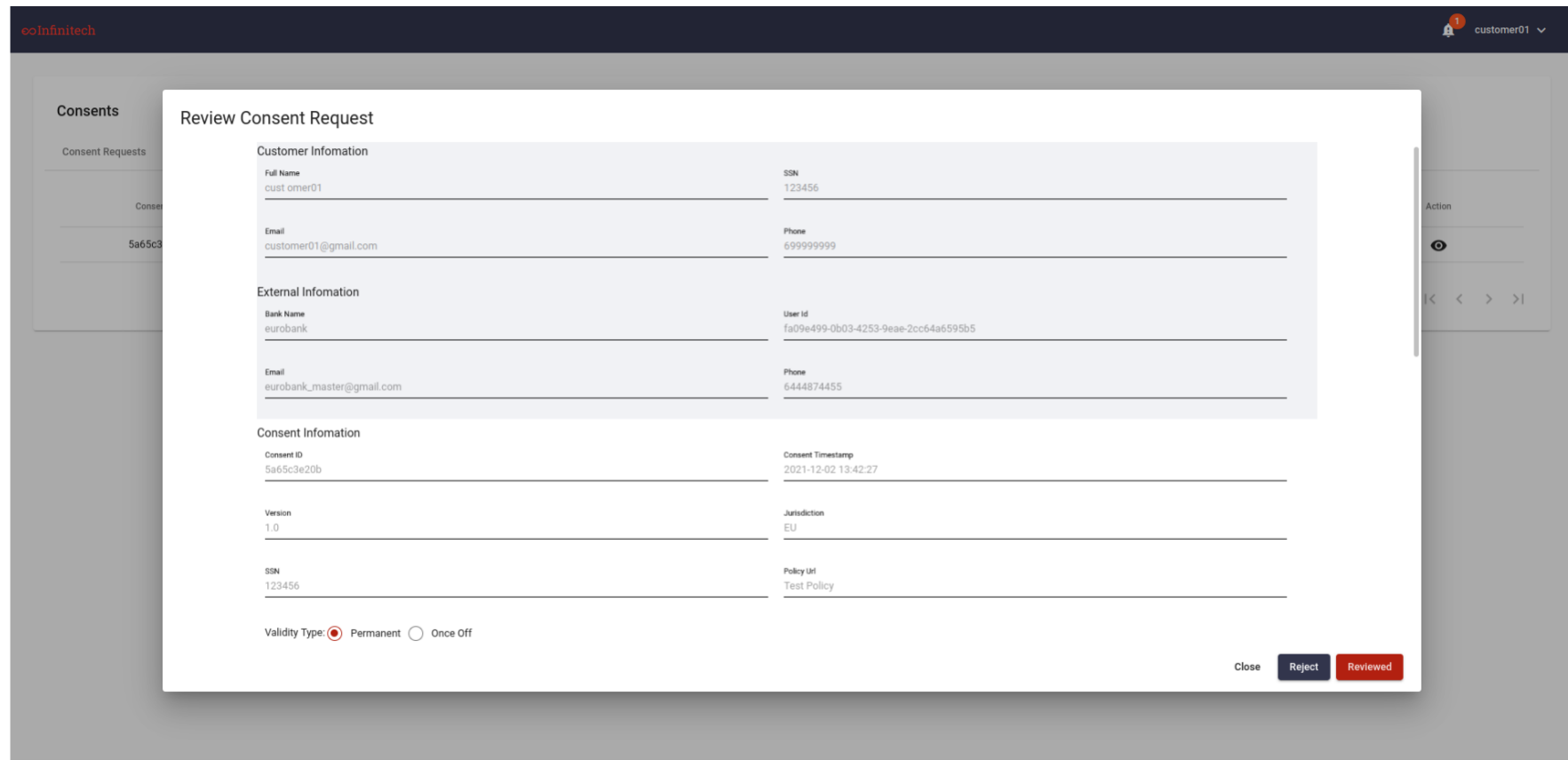


Figure 20: Consent Management System – Customer review and approval

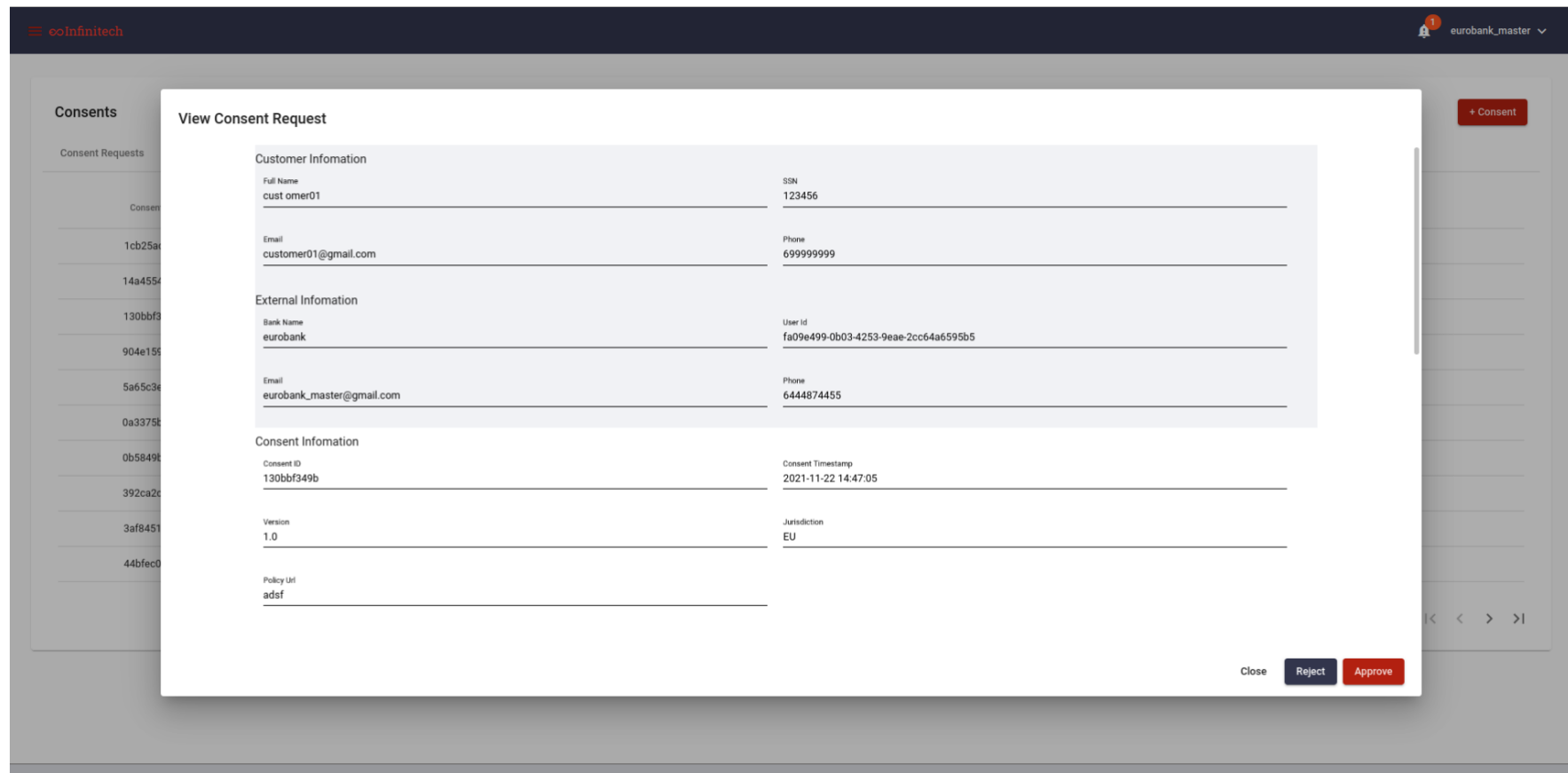


Figure 21: Consent Management System – External Financial Institution approval

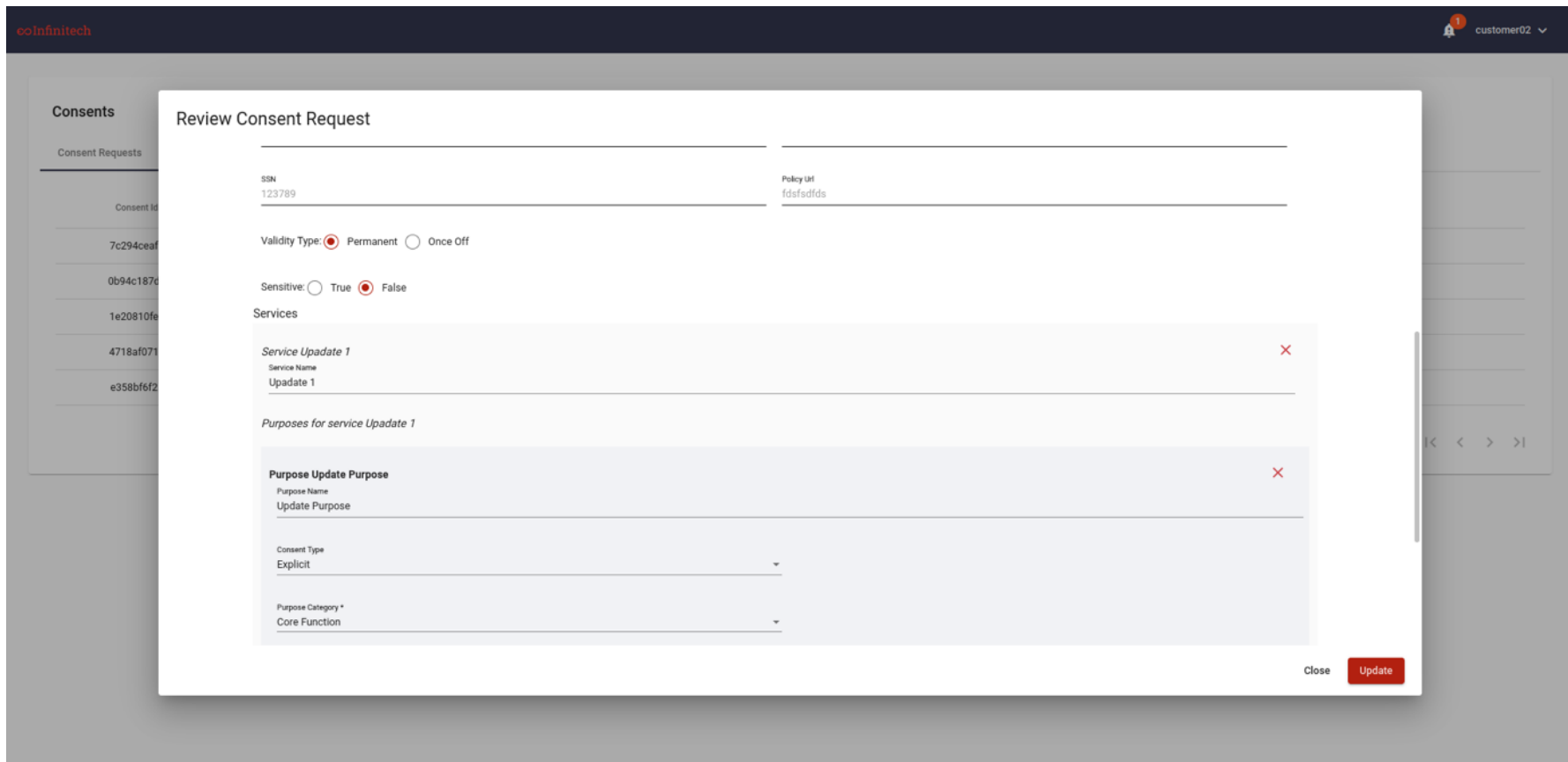


Figure 22: Consent Management System – Update consent

Besides the update, the users are able to withdraw an existing consent by selecting the Withdraw option. At this point, the Consent Management System informs the user that this action is permanent and ask for a confirmation. Upon the confirmation of the user, the status of the consent is set to “Withdrawn” and a new version of the consent is written in the blockchain (Figure 23).

For the “once off” consents, in which the validity period of the consent is a predefined period, the user informed via a notification that the validity period is about to expire providing him/her the option to initiate an update before the actual expiration. If the validity period expires, the consent status is changed to Expired and a new version of the consent is written in the blockchain (Figure 24).

At any point, the users can access the complete history of each consent by selecting the “Show history” button of the entry in the list of consents. The option will display the different versions of the consent in a time-line format where the user can have an overview of the details of each version of the consent (Figure 25).

Finally, the internal financial institution can leverage the complete lifecycle management of the consent in order to formulate an access control decision. In particular, the user of the internal financial institution can consult the Consent Management System in order to validate the consent status between the requesting party and the customer whose data are requested by utilising the enhanced search capabilities of the Consent Management System that performs queries on the blockchain records with different parameters such as the customer id and the external financial institution id or their combination (Figure 26).

Video Link of Solution in INFINITECH Marketplace:

<https://marketplace.infinitech-h2020.eu/webinar/blockchain-enabled-consent-management>

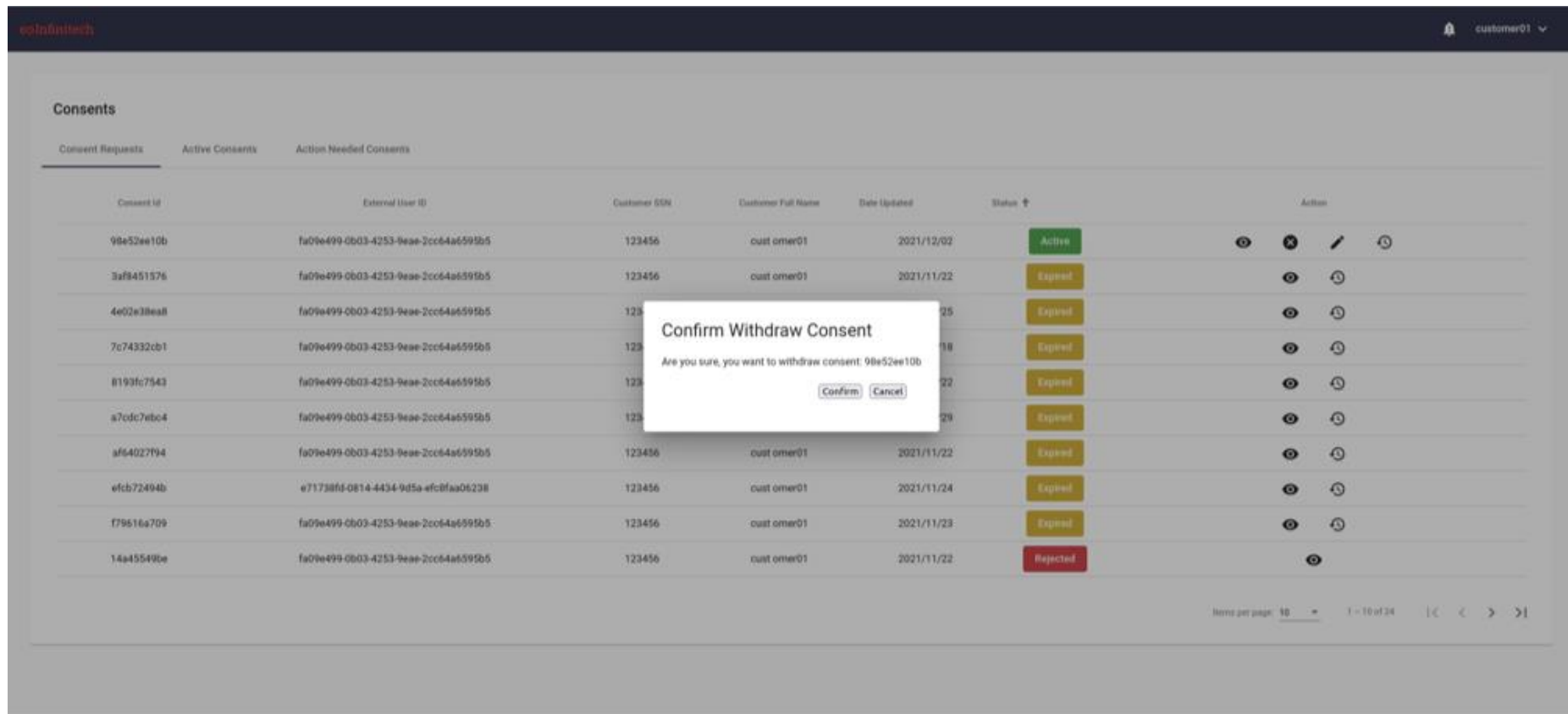


Figure 23: Consent Management System – Withdraw consent

D4.9 – Permissioned Blockchain for Finance and Insurance - III

The screenshot shows the 'Consents' section of the eoinfinittech system. It features three tabs: 'Consent Requests', 'Active Consents', and 'Action Needed Consents'. The 'Consent Requests' tab is selected. Below the tabs is a table with the following columns: Consent Id, External User ID, Customer SSN, Customer Full Name, Date Updated, Status, and Action. The table contains 10 rows of data. Two rows have a status of 'Expired' (yellow background), and the other eight rows have a status of 'Withdrawn' (grey background). The 'Expired' rows are for Consent Ids 3af8451576 and 4e02e38ea8. The 'Action' column contains icons for viewing details and refreshing the data.

Consent Id	External User ID	Customer SSN	Customer Full Name	Date Updated	Status	Action
14a45549be	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/22	Rejected	
130bbf349b	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/22	Reviewed	
0a3375b7d4	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/03	Withdrawn	
0b5849be08	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/25	Withdrawn	
392ca2dab1	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/30	Withdrawn	
3af8451576	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/22	Expired	
44bfec0db2	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/01	Withdrawn	
4e02e38ea8	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/25	Expired	
54fc2f5163	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/30	Withdrawn	
5ac6ba4036	fa09e499-0b03-4253-9eae-2cc64a6595b5	123456	cust omer01	2021/11/30	Withdrawn	

Items per page: 10 1 - 10 of 24

Figure 24: Consent Management System – Expired consents

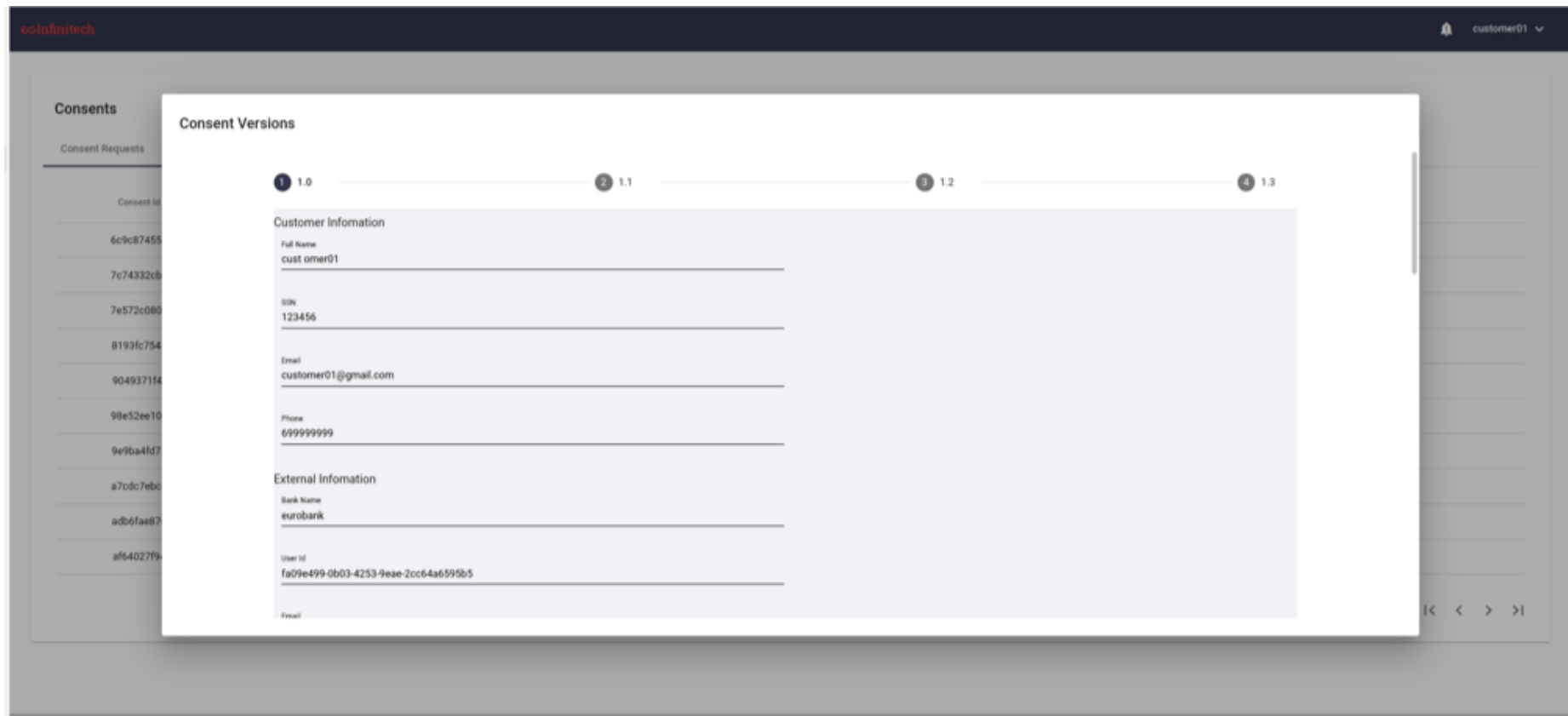


Figure 25: Consent Management System – Consent complete history

D4.9 – Permissioned Blockchain for Finance and Insurance - III

The screenshot displays the 'Consents' management interface. At the top, there is a search bar labeled 'search by Consent Id' with a dropdown menu showing options: 'Consent Id', 'Customer SSN', 'External Id', and 'External Id and Customer SSN'. A '+ Consent' button is located in the top right corner. Below the search bar is a table with the following columns: Consent Id, Customer SSN, Customer Full Name, Date Updated, Status, and Action. The table contains 10 rows of data. The status column uses color-coded buttons: blue for 'Received', red for 'Rejected', green for 'Reviewed', and teal for 'Pre-Approved'. 'Withdrawn' items are shown in grey buttons. The Action column contains icons for viewing details and refreshing the record.

Consent Id	Customer SSN	Customer Full Name	Date Updated	Status	Action
a6b6ed9de5	2021d870-659b-498e-963b-fc5e3c0ec045	cust omer02	2021/11/17	Received	👁️
70e83c9ab3	2021d870-659b-498e-963b-fc5e3c0ec045	cust omer02	2021/11/17	Received	👁️
1cb25ac0fe	fa09e499-0b03-4253-9eae-2cc64a6595b5	cust omer01	2021/11/22	Rejected	👁️
14a45549be	fa09e499-0b03-4253-9eae-2cc64a6595b5	cust omer01	2021/11/22	Rejected	👁️
130bbf349b	fa09e499-0b03-4253-9eae-2cc64a6595b5	cust omer01	2021/11/22	Reviewed	👁️
904e159999	fa09e499-0b03-4253-9eae-2cc64a6595b5	cust omer01	2021/11/29	Rejected	👁️
5a65c3e20b	fa09e499-0b03-4253-9eae-2cc64a6595b5	cust omer01	2021/12/08	Reviewed	👁️
7c294ceaf0	2021d870-659b-498e-963b-fc5e3c0ec045	cust omer02	2021/12/08	Pre-Approved	👁️
0a3375b7d4	fa09e499-0b03-4253-9eae-2cc64a6595b5	cust omer01	2021/11/03	Withdrawn	👁️ 🔄
0b5849be08	fa09e499-0b03-4253-9eae-2cc64a6595b5	cust omer01	2021/11/25	Withdrawn	👁️ 🔄

At the bottom right of the table, there is a pagination control showing 'Items per page: 10' and '1 - 10 of 34' with navigation arrows.

Figure 26: Consent Management System – Access Control and Search

3.2 Know Your Customer / Know Your Business

Updates from D4.8:

The updates on this final iteration involve the Business need & Rationale section (3.2.1) that documents the motivation behind the implementation of the developed application. In addition to this, the Implementation section (3.2.5) has been updated to depict the latest advancements of the development phase and the KYC/KYB Application Overview section (3.2.6) is introduced which provides a walkthrough of the implemented application.

3.2.1 Business Need & Rationale

In current financial relations, Know Your Customer (KYC) and Know Your Business (KYB) regulations expect that client participants, either individuals or entire enterprises, undertake verification of their identity. In particular, a KYC or KYB mechanism ensures that the identification and validation of a customer unfolds against international policies and laws set by governments, central banks and important financial institutions. Each financial organization is capable of estimating the risks involved when aiming sustainability on a new business partnership. In Finance field of Anti-Money Laundering (AML) processes, every financial organization is obligated to install KYC and KYB procedures during the time that they onboard a new customer. As both the customer profile data and the relevant regulations and rules are subject to changes over time, the respective maintenance and amendments of the stored information become more complex. Moreover, the currently adopted centralized systems are threatened by next generation attacks of data protection and cyber-security that establish cheaper to launch and focused cyber-assaults that are led by more advanced adversaries each year [17].

Blockchain technology and particularly permissioned blockchains are able to offer protection and guarantees to the relevant KYC and KYB procedures through their decentralized nature [18] [19]. Decentralization achieves a specific and structural system architecture where the available information is replicated across all blockchain network nodes. In particular, information integrity remains solid and cannot be modified by taking control of one or more participant nodes of the network. Thus, single points of failure do not appear in such structures. The distributed ledger software technology is able to separate all the important information inside the corresponding permissioned network that operates with well-defined and organized access control. Specifically, every party is accepted into the network by an invitation from the inside dedicated agents which enables the party to employ the relevant blockchain activities.

Client data is stored safely on the permissioned ledger where transparency is established to legal network parties. Both customers and enterprises execute the dedicated kinds of CRUD functionalities (Create, Read, Update, Delete) on the specified data under pre-configured and regulatory standards. For instance, the various capabilities of permissioned ledgers empower dissimilar application rules that isolated selected parties in an inside network of higher data privacy which is part of the initial permissioned blockchain network. Immutability of the data and optimized privacy control are qualities that govern inside the described technological framework while they provide legal client data security and governance along with legitimate data administration by financial entities [20] [21].

3.2.2 Description of the solution

The Know Your Customer (KYC) / Know Your Business (KYB) policies in state-of-the-art financial relations expect that customer parties, either individuals or entire corporations, endeavour verification of their identity. Thus, each financial organization is able to estimate the risks involved with sustaining a new business-customer partnership. In this context, together with the wider Finance

field of Anti-Money Laundering (AML) procedures, every financial institution establishes KYC and KYB operations at the time they register a new customer [18].

The KYC/KYB blockchain application resolves the implementation of such industry mechanisms by utilizing blockchain technology as a basic background infrastructure. Security, immutability and controlled transparency are employed inside large enterprise blockchain networks, while they offer efficiency of tasks and effectiveness of transactions within the corporation and its members. Ultimately, blockchain technology simplifies the emerging use cases and directly addresses any possible issues that are created.

Particularly, a KYC/KYB mechanism ensures that the identification and verification of a customer occurs against national and international regulations and laws set by governments, commissions, central banks and financial associations. As both the customer profile information and the relevant laws and rules are subject to changes over time, their update and maintenance become complicated. Moreover, their centralized systems are exposed to data protection and cyber-security risks, which become less expensive to launch while they are led by more sophisticated adversaries, year after year [17].

Blockchain technology and particularly permissioned blockchain networks are capable of providing security to the KYC and KYB processes through decentralization. The concept of decentralization mainly exploits the idea that the information is replicated across all network nodes, and thus sabotaging one or more nodes cannot harm the information integrity and a single point of failure is avoided. In particular, the permissioned blockchain technology promises to keep that sensitive information inside a private network where only privileged parties can access it with an insider invitation. Thus, the customer information is kept safe on a private ledger that offers transparency to a privileged group of legal network participants. Both the customer and the organization are able to perform create, read, write, delete (CRUD) operations on the data under pre-defined access control policies. The various features of permissioned blockchains enable different policy applications that are able to, for instance, separate legal parties into a higher privacy network running inside the initial private one. Improved privacy control and data immutability rule inside the aforementioned technological scenario while they ensure legitimate customer data protection and management together with proper administration of this data by financial enterprises [20].

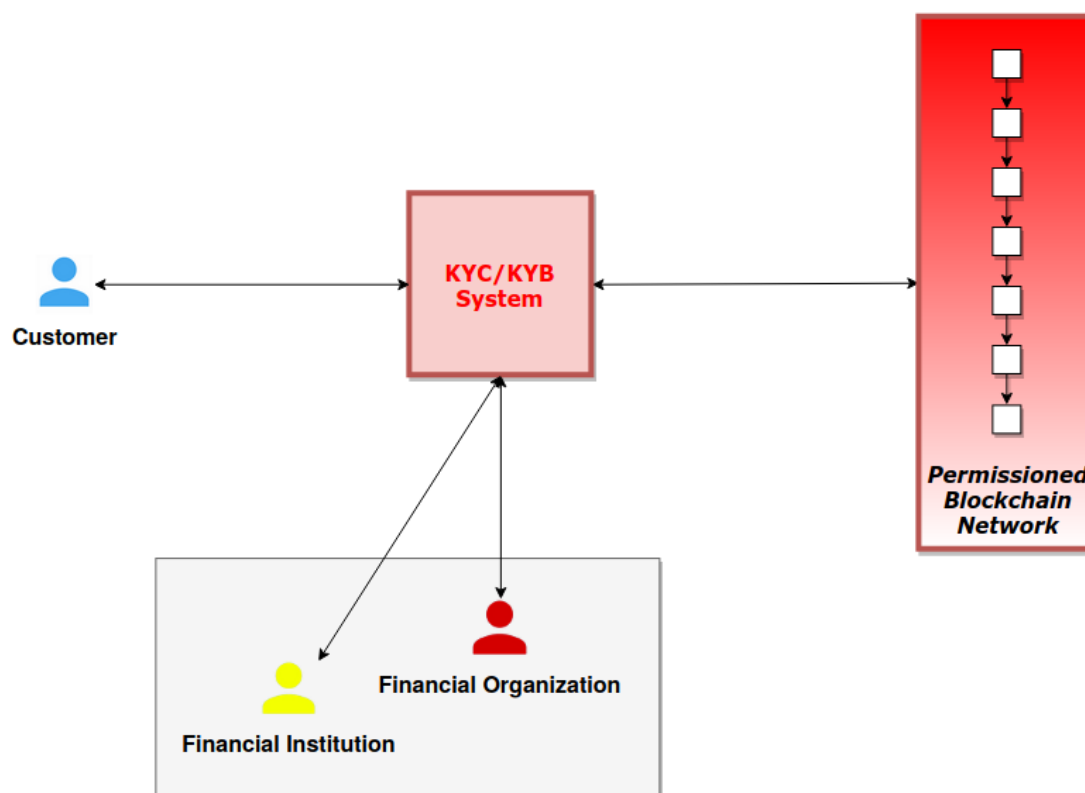


Figure 27: High-level architecture of the KYC/KYB solution

Figure 27 depicts a high-level architecture of the proposed solution. In general, the customer participant (light blue actor), being either an individual or an entire organization, needs to acquire financial services that are offered by the financial institution (yellow actor). For the completion of this transaction, the financial institution requests that the customer identity information is documented in KYC/KYB data after it is legally verified. In that case, the customer participant uploads their KYC/KYB documentation to the KYC/KYB System that interacts with the permissioned blockchain network and stores this information on-chain. When another financial organization (red actor) requires to start business relations with the same customer (light blue actor), the latter's KYC/KYB information is easily and time-efficiently obtained through the access rights provided by the financial institution (yellow actor) which is already having business partnerships with this customer. It is important to clarify that the customer using this kind of technology has declared their consent of sharing their information among privileged participants of the network, while their information privacy is guaranteed. In this system, data security and efficient data management and maintenance rule, for the united underlying blockchain infrastructure with common data structures offers stability, security, sustainability and time-efficiency.

The design specifications of any blockchain application is tightly related to the definition of the business objects for which the relevant functions of the chaincode will be implemented towards the maintenance and update of their current and historical state. In this sense, to facilitate the implementation of the KYC/KYB application, the data schema has been defined. The data schema is depicted in Figure 28, while the details of all the entities are documented in Table 15 and Table 16.

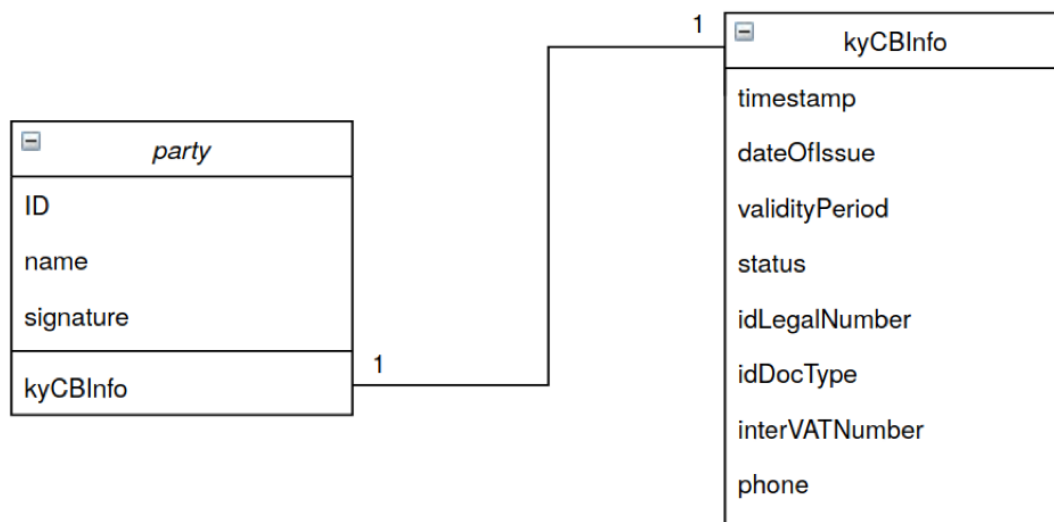


Figure 28: KYC/KYB Data Schema diagram

Table 15: KYC/KYB Data Schema details (1)

party		
Name	Type	Short description
<i>ID</i>	String	REQUIRED: The identification number used to uniquely identify network participants
<i>name</i>	String	REQUIRED: The name of the participant
<i>signature</i>	String	REQUIRED: The digital signature of the participant
<i>kyCInfo</i>	String	REQUIRED: The KYC/KYB pointer to the participant’s KYC/KYB record

Table 16: KYC/KYB Data Schema details (2)

kyCInfo		
Name	Type	Short description
kyCInfo	Integer	REQUIRED: The KYC/KYB documentation pointer number
<i>timestamp</i>	String	REQUIRED: The date the KYC/KYB documentation is modified
<i>dateOfIssue</i>	String	REQUIRED: The date the KYC/KYB documentation is first accepted
<i>validityPeriod</i>	String	REQUIRED: The period until which the KYC/KYB information is valid
<i>status</i>	Boolean	REQUIRED: The validity of the KYC/KYB documents (either valid or invalid)
<i>idLegalNumber</i>	String	REQUIRED: The alphanumeric number of the legal identity of the participant
<i>idDocType</i>	String	REQUIRED: The document type with which the party is legally approved
<i>interVATNumber</i>	String	REQUIRED: The international VAT identification number of the party in alphanumeric number form
<i>phone</i>	Integer	REQUIRED: The communication phone to contact a participant representative

3.2.3 Use cases

The following sections provide the detailed documentation of all the use cases encapsulated in this blockchain application describing in detail all information of each use case.

3.2.3.1 Use Case KYC/KYB-1: Assertion of customer identity documents

In this use case, the customer asserts their identity documents to the financial organization through the KYC/KYB process. In particular, a customer who represents either an individual or an entire corporation, acknowledges the KYC or KYB information in order to initiate business relations with the financial institutions and organizations of the network. In this context, each enterprise participant of the permissioned blockchain network ensures the legitimacy of their customer and, thus, new business relationships are established.

Table 17: KYC/KYB Use Case KYC/KYB-1

Stakeholders involved:	Customer
Pre-conditions:	1. A private blockchain network is set up.
Post-conditions:	1. The Customer's KYC/KYB information is stored on-chain.
Data Attributes	1. The Customer's KYC/KYB documents
Normal Flow	1. The Customer uploads their KYC/KYB documentation on the blockchain ledger. 2. The documentation information is successfully stored on-chain.
Pass Metrics	1. The documentation information is successfully stored on-chain.
Fail Metrics	2. There is no Customer to upload their KYC/KYB documents.

3.2.3.2 Use Case KYC/KYB-2: Read access to customer identification information

In this use case, as a legal network participant, a financial organization has *read access* to any customer's KYC/KYB documents information. In particular, inside the permissioned blockchain network, each party is able to read the corresponding ledgers. In this context, each financial organization has *read access* to the data that is stored on-chain. By a simple *read access* request, the financial organization can fetch the information of a customer they are interested in. Additionally, upon using the system, each customer approves that their data may be accessed by the financial organization they will initiate business relations with.

Table 18: KYC/KYB Use Case KYC/KYB-2

Stakeholders involved:	Financial organization
Pre-conditions:	1. The financial organization is a legal participant of the network.

Post-conditions:	1. The financial organization has <i>read access</i> control over the requested KYC/KYB information.
Data Attributes	1. The financial organization's network attributes 2. The requested KYC/KYB documents
Normal Flow	1. The financial organization requests for <i>read access</i> of a specific customer's KYC/KYB documentation. 2. The requested documentation information access is successfully granted.
Pass Metrics	1. The requested documentation information is successfully granted.
Fail Metrics	1. The financial organization is not eligible to access the requested documentation information.

3.2.3.3 Use Case KYC/KYB-3: Sharing of customer KYC/KYB documents

In this use case, the financial organization A shares the KYC/KYB document information of customer B with the financial organization C through the secure and private blockchain network. In particular, each of the financial organizations participating in the blockchain network is eligible for accessing the data stored on-chain. However, depending on the different access control rights granted by the time of joining the network, there exists the case where different organizations are qualified to access different data. In this context, together with the initial customer consent, a financial organization may grant access to another organization or institution for a specific customer KYC/KYB documentation.

Table 19: KYC/KYB Use Case KYC/KYB-3

Stakeholders involved:	Financial organization
Pre-conditions:	1. The financial organizations A and C are legal participants of the network. 2. The requested customer B has submitted their KYC/KYB document information.
Post-conditions:	1. The financial organization C has <i>read access</i> control over the requested KYC/KYB information.
Data Attributes	1. The financial organizations' A and C network attributes 2. The requested KYC/KYB documents
Normal Flow	1. The financial organization A requests for sharing of a specific customer's KYC/KYB documentation with a financial organization C. 3. The requested documentation information access is successfully granted.
Pass Metrics	1. The requested documentation information is successfully granted.
Fail Metrics	1. The financial organization A is not eligible to share the requested documentation information.

3.2.4 Sequence Diagrams

In the previous section, all the relevant use cases of the designed blockchain application were documented. The following sections present the sequence diagrams for each use case, depicting the interactions between the stakeholders and the components of the designed solution, as well as the interactions between the various components of the designed solution.

3.2.4.1 Assertion of customer identity documents

In Use Case KYC/KYB-1, a customer provides their documentation information in order to be eligible for business partnerships with the participating financial organizations. In this use case, a customer actor, which is either an individual or an entire enterprise, acknowledges the requested KYC or KYB documents in the KYC/KYB System by uploading them. This action is translated into an internal request that propagates the information inside the blockchain network, though the different blockchain components. Finally, the data is safely stored on-chain, i.e. on the private ledger, and further actions and use cases can emerge afterwards (see Use Case KYC/KYB-2 and Use Case KYC/KYB-3).

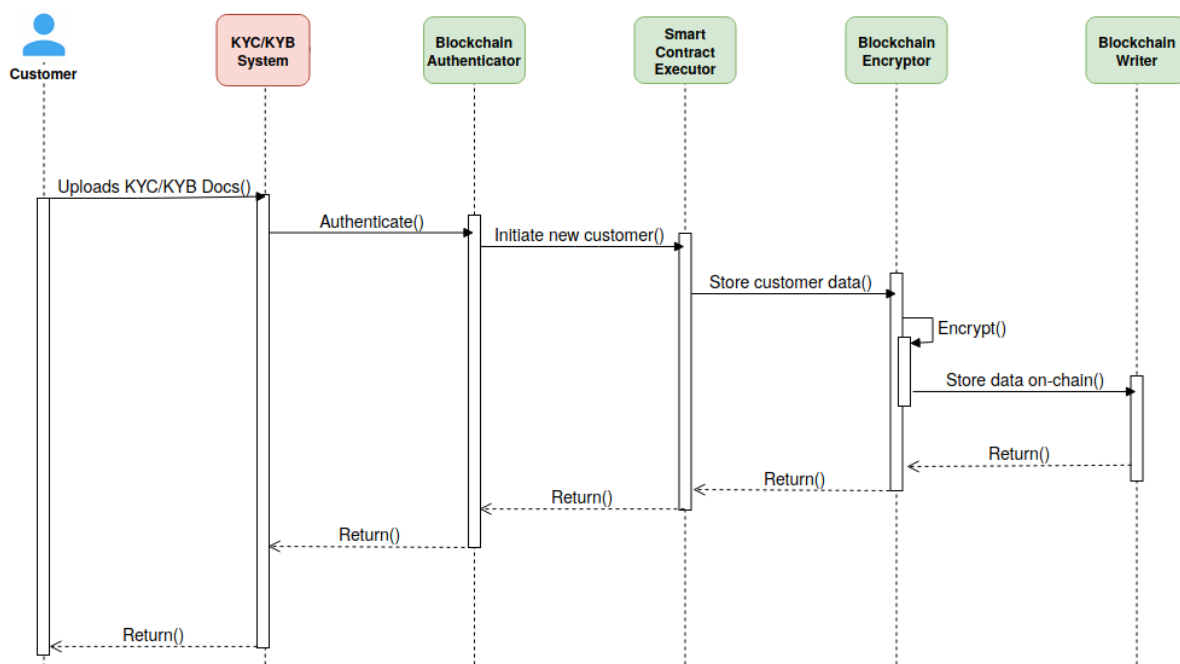


Figure 29: Use Case KYC/KYB-1 sequence diagram

3.2.4.2 Read access to customer identification information

In Use Case KYC/KYB-2, the blockchain network parties are constituted of financial organizations and are able to inspect the KYC/KYB documentation information of customers. Through the private and secure network, the different organizations obtain access to a customer's KYC/KYB submitted data in order to estimate whether to establish business relationships with them or not. The customer data submission is already accompanied by the customer's consent of using the KYC/KYB information by the legal parties of the network, i.e. the financial organizations. As in Figure 30, through the sequence of the blockchain components the *read access* is propagated to the financial organization that requested it.

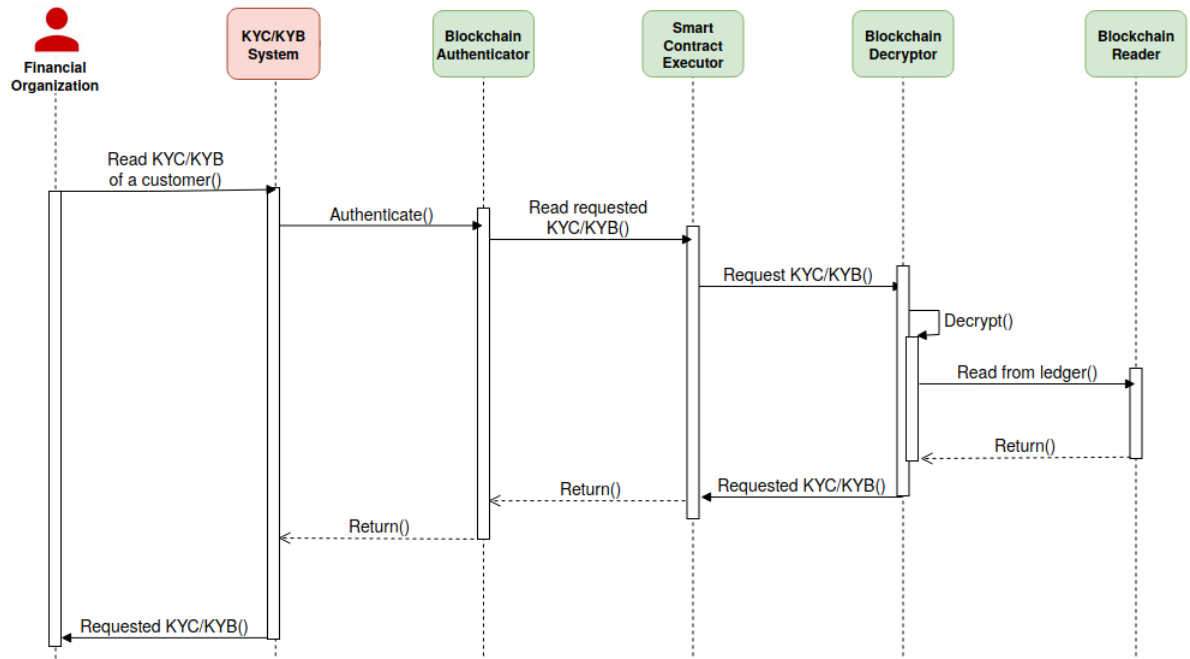


Figure 30: Use Case KYC/KYB-2 sequence diagram

3.2.4.3 Sharing of customer KYC/KYB documents

In Use Case KYC/KYB-3, a sharing of customer (B) information among participant organizations (A and C) takes place. Organization C obtains customer’s B information through the cooperation of organization A. Since the customer’s data already exists inside the secure and private blockchain network, organization C can request it indirectly. Organization A responds to the request by granting *read access* of customer B information to organization C. In such an efficient system, the customer and the financial organizations benefit from this kind of sharing, since the customer avoids re-entering their KYC/KYB data to a different system of a different organization, while the financial organizations speedily obtain customer information and make decisions upon the establishment of new business relations.

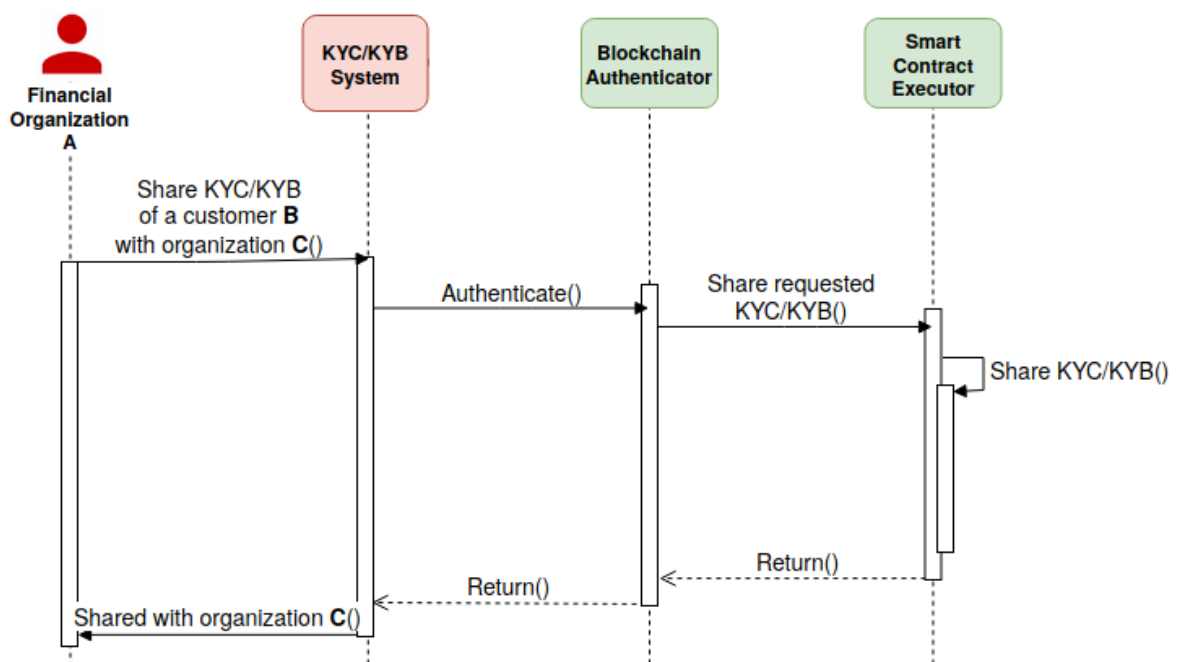


Figure 31: Use Case. KYC/KYB-3 sequence diagram

3.2.5 Implementation of the Know Your Customer / Know Your Business

This section explains the details of the implementation of the Know Your Customer / Know Your Business Use Case Demonstrator. The designed and developed solution has several important parts that are underlined below in order to present in a more clear and coherent way the built framework and the technological manifestation of the initial conceptualized architecture.

Based on the design specifications of the Sections 3.2.2 and 3.2.3, as well as on the data schema defined in Section 3.2.1, the chaincode is structured at a component level as depicted in Figure 32.

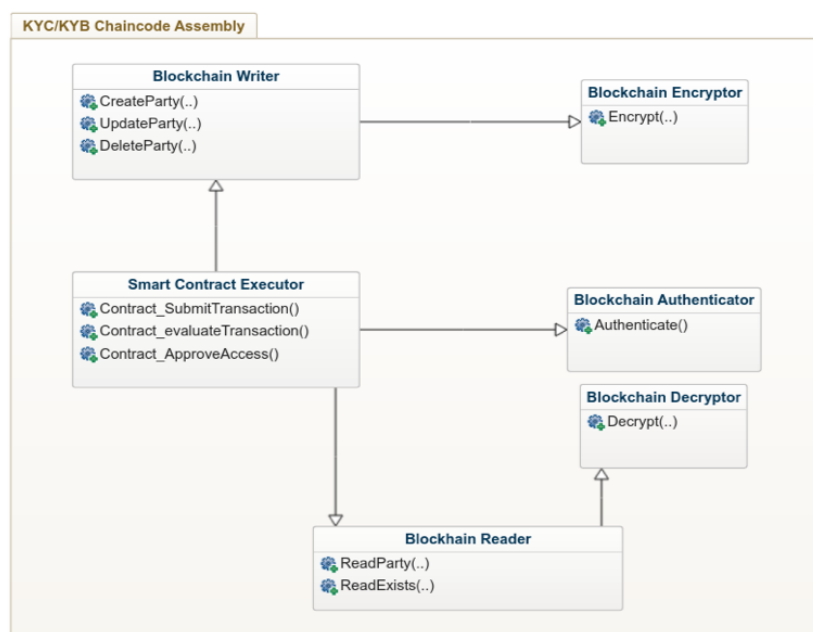


Figure 32: KYC/KYB Chaincode structure.

The *Blockchain Writer* component undertakes the responsibility to submit new transactions to the blockchain ledger. In the Know Your Customer / Know Your Business Use Case, this component refers to the transactions submitted for registering a new client on-chain, updating an existing one and withdrawing the KYC/KYB information of an existent participant. It is implemented with the following functions:

- `CreateParty(id string, name string, signature string, validityPeriod string, idLegalNumber string, idDocType string, interVATNumber string, phone int)`: The `CreateParty` function is creating a new client on the ledger by updating the world state with the received data and by appending new blocks at the end of the ledger. The "CreateParty" function receives the data containing the client's KYC or KYB information in each function parameter field and converts them into the appropriate format. Afterwards, the party data is organized in a single structure and it is finally submitted to the world state.
- `UpdateParty(id string, name string, signature string, validityPeriod string, idLegalNumber string, idDocType string, interVATNumber string, phone int)`: The `UpdateParty` function is changing the information of an existing client on the ledger by updating the world state with the newly received details of the participant. The `UpdateParty` function receives the data containing these details in each function parameter

field and as in `CreateParty` converts them into the appropriate format in order to submit them to the world state.

- `DeleteParty(id string)`: The `DeleteParty` function withdraws the information of an existing client on the ledger by updating the world state. The `DeleteParty` function receives the ID of the party that should be deleted and afterwards submits the new withdrawal transaction on the ledger.

The *Blockchain Reader* component is used in order to read the ledger state and fetch a particular party's KYC or KYB submitted information:

- `ReadParty(id string)`: This function is fetching the appropriate party KYC or KYB data from the ledger by querying the ID of the client. After a successful execution of this function, the demanded information is returned in JSON format.
- `PartyExists(id string)`: This function is requesting if a specific party exists on the ledger along with their KYC/KYB information. The party that exists is validated by its own ID, whereas if there is no relevant submission on the blockchain ledger the function claims on the negative.

The *Smart Contract Executor* component's main purpose is to consolidate the business intelligence of the defined use case and execute the chaincodes on the blockchain ledger.

- `Contract_SubmitTransaction()`: The function initiates the on-chain recording of new KYC/KYB information. This function sends the transaction to all dedicated peers of the respective channel in the blockchain network. In case that all of the aforementioned peers endorse the transaction, the endorsed proposal is sent to the ordering service in order to be committed by each of the peers to the ledger of the channel.
- `Contract_evaluateTransaction()`: The function is responsible for initiating the retrieval of KYC/KYB documents of a specific party. This function communicates with a single peer and the requested results will be returned.
- `Contract_ApproveAccess()`: The function is responsible for initiating the access approval of the KYC/KYB documents from financial organization A to B, i.e. the latter being the one that requested the data. This function alters the data view of organization B through the permission of organization A.

The *Blockchain Authenticator* component is responsible for performing the authentication of the blockchain network user in order to grant access to a specific channel of the blockchain network.

- `Authenticate()`: The function is responsible for the authentication of the organization user in order to access a specific channel of the blockchain network. This function receives the appropriate keys and certificates and allows connection of the organization user to the specified smart contract of the respective channel.

The *Blockchain Encryptor* component performs the encryption of the related data that are involved and produced within the smart contract execution.

- `Encrypt (party string)`: The function performs the aforementioned data encryption adopting AES-256 encryption.

The *Blockchain Decryptor* component performs the decryption of the data encrypted by the Blockchain Encryptor component.

- `Decrypt (party string)`: The function performs the reverse procedure of the *Encrypt* function of the Blockchain Encryptor component by decrypting the respective data.

With regards to the KYC/KYB web application, the following analysis presents the essential services of user interface (UI) framework and their functions that are hitherto developed for the user interaction with the on-chain part of the solution.

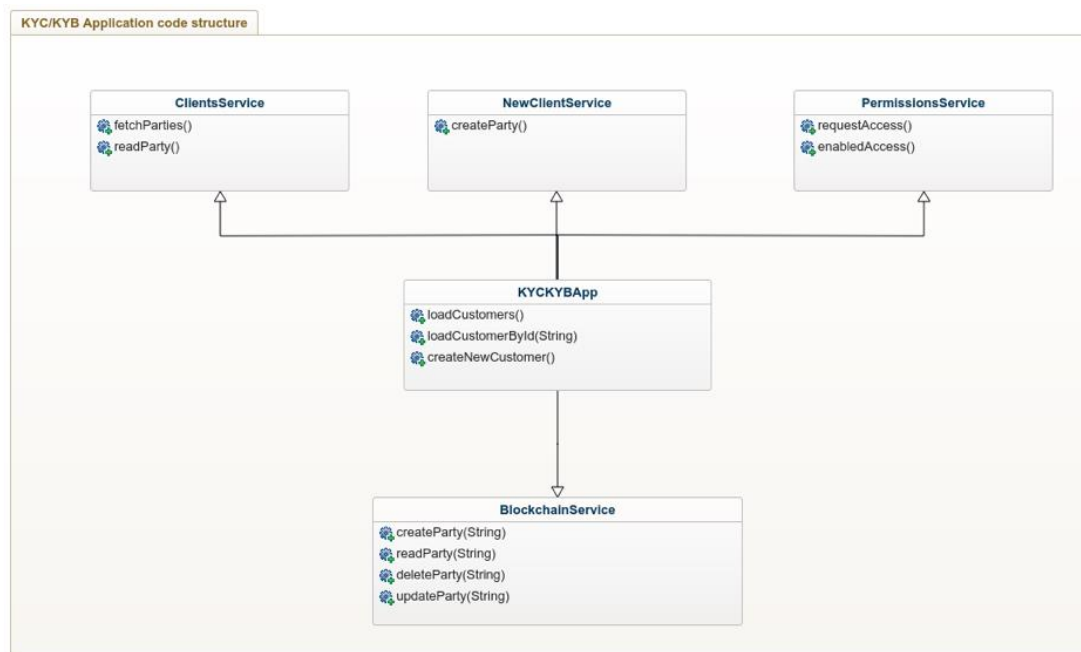


Figure 33: KYC/KYB Application code structure

The *Clients Service* is responsible for retrieving the requested KYC or KYB information from the blockchain ledger and for delivering it on the web interface of the participant channel organization. The implemented functions are as follows:

- `fetchParties()` : The specific function retrieves the KYC or KYB information of all the clients, as seen by the participant channel organization user.
- `readParty()` : The specific function retrieves the KYC or KYB information of a specific client, as seen by the participant channel organization user.

The *New Client Service* is responsible for delivering the KYC or KYB data of a newly inserted client on the specified blockchain endpoints in order to be submitted on the ledger. The implemented function is the following:

- `createParty()` : The specific function creates a new client record on the blockchain ledger by delivering the provided client KYC/KYB information.

The *Permissions Service* is responsible for switching the access control state of a financial organization. The implemented functions are as follows:

- `requestAccess()` : A financial organization requests access control for KYC/KYB information of another organization.
- `enabledAccess()` : The specific function returns in case access control is enabled for a channel organization.

The *KYC/KYB App* constitutes the main user web interface endpoint through which the end-user can initiate all the permitted actions and navigate into the application. The *KYC/KYB App* integrates with the Clients Service through the direct communication of the related messages and requests calling the dedicated Clients Service functions. In similar manner, the functions of New Client Service, Permissions Service and Blockchain Service are integrated. Also, the *KYC/KYB App* provides a governed angle of the application development with a holistic approach on the organization and coordination of the

corresponding activities and function calls. The KYC/KYB App high-level functions are implemented as follows:

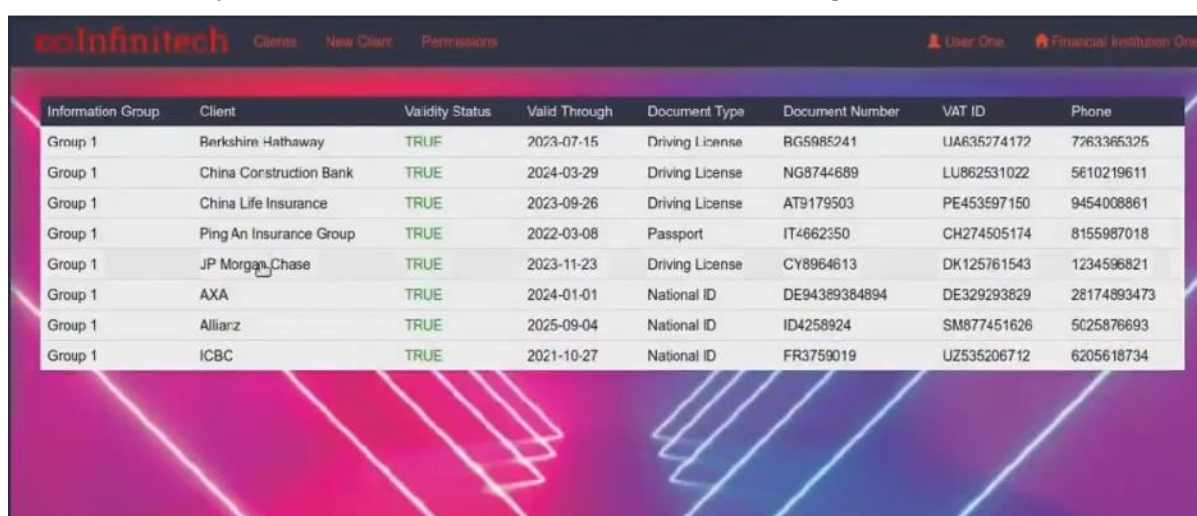
- `loadCustomers()` : This function is the starting point from where the end-user retrieves all the clients' KYC or KYB information. By triggering this function, the full stack integrated mechanism retrieves the requested data back to the user.
- `loadCustomerById()` : This function is exploited by the end-user in order to retrieve the KYC or KYB information of a specific client.
- `createNewCustomer()` : The specific function inserts the data of a new client in the system in order to be handled by the New Client Service.

The *Blockchain Service* is responsible for submitting new use case related data on the blockchain ledger once it is triggered by the web user interface. The functionalities immediately enable the corresponding chaincode functions. The *Blockchain Service* is materializing the straightforward communication of the respective application functions and procedure calls with the blockchain network and the dedicated chaincode structure (Figure 32). On every call that needs to reach the blockchain network, the *Blockchain Service* ensures that the appropriate confirmations enable the correct and legitimate triggering of the chaincode assembly functions. The *Blockchain Service* operating functions are implemented as follows:

- `createParty(string)` : The specific function prompts the respective chaincode function in order to submit the provided client KYC/KYB information on the blockchain ledger.
- `readParty(string)` : The specific function prompts the respective chaincode function in order to retrieve the required party KYC/KYB information from the blockchain ledger.
- `deleteParty(string)` : The specific function prompts the respective chaincode function in order to remove the requested client from the blockchain ledger.
- `updateParty(string)` : The specific function prompts the respective chaincode function in order to update any of the requested client information on the blockchain ledger.

3.2.6 KYC/KYB Application overview

With regards to the web application that is offered to the end-user, the following illustrations depict the relevant views information accompanied by their descriptions. In general, the last part of the implementation showcases in a practical way the important parts of the user interface framework that is hitherto developed for the user interaction with the blockchain ledger.



Information Group	Client	Validity Status	Valid Through	Document Type	Document Number	VAT ID	Phone
Group 1	Berkshire Hathaway	TRUE	2023-07-15	Driving License	BG5985241	UA635274172	7263365325
Group 1	China Construction Bank	TRUE	2024-03-29	Driving License	NG8744689	LU862531022	5610219611
Group 1	China Life Insurance	TRUE	2023-09-26	Driving License	AT9179503	PE453597150	9454008861
Group 1	Ping An Insurance Group	TRUE	2022-03-08	Passport	IT4662350	CH274505174	8155987018
Group 1	J.P. Morgan Chase	TRUE	2023-11-23	Driving License	CY8964613	DK125761543	1234596821
Group 1	AXA	TRUE	2024-01-01	National ID	DE94389384894	DE329293829	28174893473
Group 1	Allianz	TRUE	2025-09-04	National ID	ID4258924	SM877451626	5025876693
Group 1	ICBC	TRUE	2021-10-27	National ID	FR3759019	UZ535206712	6205618734

Figure 34: Clients view

In Figure 34, the specific Clients view displays the KYC/KYB information of the on-chain submitted participants as seen from the account of User One of Financial Institution One. In every Clients view the ledger information that the specific financial institution user is allowed to read is depicted as defined from the underlying blockchain mechanisms.

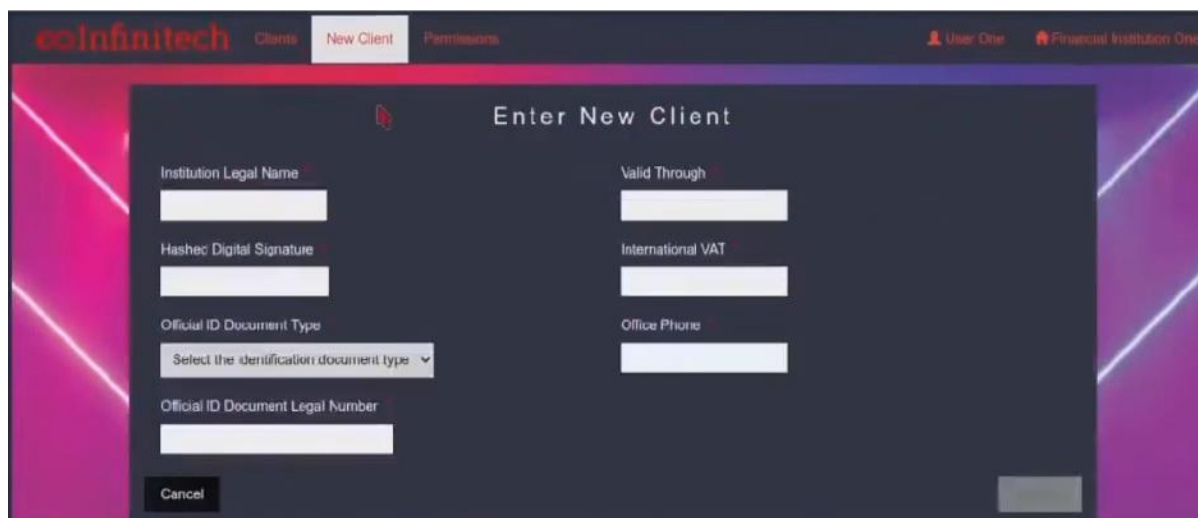


Figure 35: New Client View

In Figure 35, the New Client view depicts the user interface (UI) that a financial institution user exploits in order to register the KYC or KYB information of a new client. Upon right form completion, the new client is being inserted and the submission sends the data to the ledger.

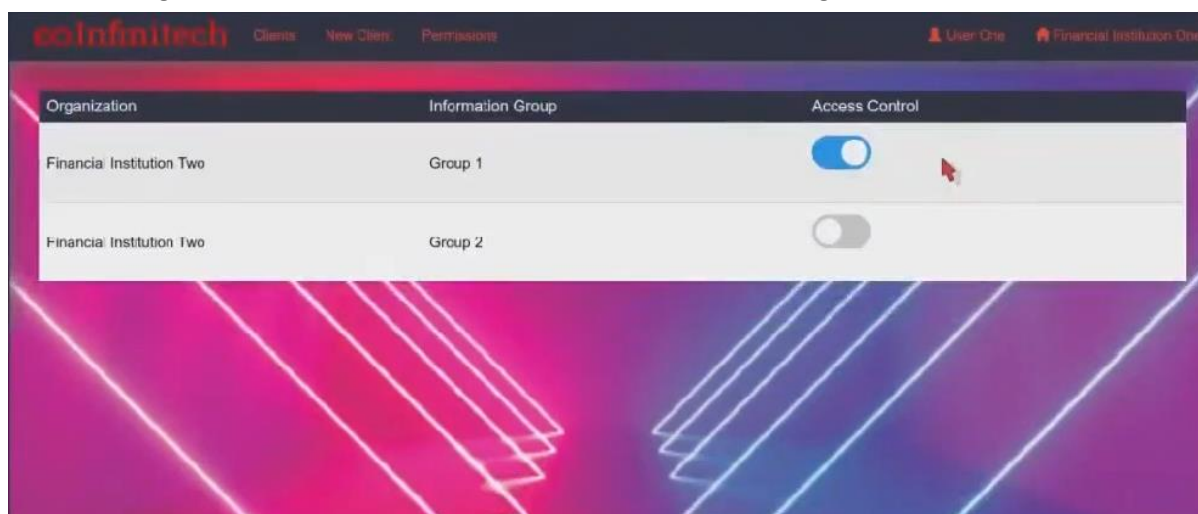


Figure 36: Permissions View

In Figure 36, the Permissions view displays the control access status of a financial institution. Every request for access from other financial institutions is depicted in this view while the financial institution user is able to permit or forbid access.

Video Link of Solution in INFINITECH Marketplace:

<https://marketplace.infinittech-h2020.eu/webinar/kyc-kyb-on-chain-data-governance>

3.3 Tokenization

Updates from D4.8:

The current section provides an updated short description of the work performed for the specific use case since the previous version. As this work is performed within the context of T4.4, the complete documentation is available in deliverables D4.11 and D4.14, which were delivered in M22, and D4.12 that will be delivered in M30.

3.3.1 Business need & Rationale

Tokenization is a disruptive technology as it allows ownership of any asset (e.g., services, bonds, national digital currencies, company shares, product ownerships, tickets, data, cloud machine hours, and licenses) to be represented digitally. Digitally represented assets, also known as virtual assets, can be easily globally marketed to end users, businesses, or agencies with very low transaction costs. Furthermore, these virtual assets can be programmed to be subject to business rules and regulations about their usage. In D4.10 we documented some of the most prominent use cases for token utilization in FinTech. In finance these include: Cryptocurrency/cross boarder payments; fundraising; letter of credit; credit risk scoring; debt issuance; digital Coin; fiat-backed token; physical asset-backed token; and securities tokens. Examples of potential use cases of tokens in the insurance sector include claim processing; multinational insurance; reinsurance/risk assessment; fraud detection; crowdfunded insurance; and cryptocurrency backed loans.

Whereas tokenization has been practiced in public blockchains for unregulated initial coin offerings during the last six years, its applications at the regulated institutional and enterprise level is just starting. There is an increasing interest from businesses to tokenize their assets since this will enable them to reach out to external markets easily and automate the assets trading. In particular, since (permissioned) Hyperledger Fabric is the choice of blockchain at the enterprise level, standardized ERC-20 and ERC-1155 contributed by INFINITECH project can accelerate tokenization adoption at the enterprise level. To this end, the INFINITECH marketplace can make use of ERC-20 and ERC-1155 token implementations as a way to represent its assets as virtual assets and market them to the outside world. Tokenization of INFINITECH assets in the marketplace could also act as a demonstrated model for other businesses. Finally, since blockchains store transactions, ERC-20 and ERC-1155 token transfer transaction data can also be retrieved through standard interfaces and analysed using big data analysis techniques for providing business intelligence, and hence, also contributing to BDVA efforts.

3.3.2 Description of the solution

Generally speaking, the work around tokens since M15 revolved around two parallel directions: a) extension to more elaborated standards; and b) building a privacy preserving federated machine learning framework in which tokens play the role of trading mechanism in a data marketplace. The first topic focused on the implementation of the ERC-1155 [22] standard led by BOUN and was reported in “D4.11 - Blockchain Tokenization and Smart Contracts – II”, while the work on the second topic was a jointly effort between FBK and IBM and was reported in “D4.14 - Encrypted Data Querying and Personal Data Market – II”.

3.3.2.1 ERC-1155 implementation

One of the most natural ways to extend our initial work on token standards implementations (reported in D4.10) was the development of an extended and more mature standard that includes also non-fungible tokens. This was strengthened during the workshop Blockchain Applications in Digital Finance (beyond cryptocurrencies) hosted by INFINITECH project and held in March 2021 in which such a requirement arose. To this end, the ERC-1155 standard, which is the most advanced

Ethereum standard, due to its capability of allowing mix of multiple fungible and non-fungible token types in a single contract, was implemented on top of Hyperledger Fabric. ERC-1155 generalizes fungible ERC 20 [23] and non-fungible ERC 721 [24] token standards by combining them under a single contract that offers multiple token types with multiple quantities.

The main artefacts from the ERC-1155 standard implementation are:

- The implementation of the six standard functions and the additional nine functions which provide a full set of functionalities and ensure avoidance of key collisions while preserving high rate of throughput.
- Code available at the GitLab repository of the project at: <https://gitlab.infinitech-h2020.eu/blockchain/erc1155-tokenization>. More importantly, the code has been reviewed and accepted to be part of the fabric-samples repository and is publicly available at: <https://github.com/hyperledger/fabric-samples/tree/main/token-erc-1155>
- A demonstrator (movie) showing all implemented flows available at the INFINITECH marketplace (see link below).

Video Link of ERC1155 Solution:

- <https://marketplace.infinitech-h2020.eu/webinar/erc1155-token-smart-contract-for-hyperledger>

3.3.2.2 A secure and trustworthy federated machine learning framework

Another future direction identified as a result of the work in tokens, was the collaboration of IBM with FBK towards a framework for federated machine learning with privacy-preserving and execution guarantees. A thorough design of this framework has been detailed in D4.14 “Encrypted Data Querying and Personal Data Market – II” and the team is currently working on the development of a minimal viable product (MVP) of the proposed framework. This MVP is the core of our future deliverable “D4.12 - Blockchain Tokenization and Smart Contracts – III” to be submitted in M30. Tokens play a vital role in such a framework as a means of trading learning models in a data marketplace.

4 The INFINITECH Blockchain Network

Updates from D4.8:

The particular section remained unchanged from the previous version. It presents the details of the designed and deployed blockchain network that supports the implementation of the blockchain applications presented in section 3.

The cornerstone of every blockchain deployment is the underlying blockchain network in which the immutable transaction ledger is maintained by a distributed network of peer nodes. As explained in Section 2, within the context of the INFINITECH project, the Hyperledger Fabric is exploited in order to provide the required permissioned blockchain network on which the distributed applications presented in Section 3 are deployed.

The main elements of the blockchain network are as follows [22] :

- **Peer nodes:** Peers are the nodes that are formulating the blockchain network, hosting the distributed ledger(s) and the smart contract(s) (chaincode), as well as other services of the network.
- **Organisations:** The peers are owned by different organisations that are contributing to the blockchain network as members of the network. Each peer has a digital identity in the form of a digital certificate issued by the Certificate Authority of each organisation.
- **Channel:** A channel refers to the isolated logical structure of a collection of peers which establish a sub-network. In a channel, only its members can see the particular transactions of the specified ledger and can have access to the deployed smart contract (chaincode) as well as all the data being transacted. Each channel is regulated by a specific channel policy as defined by the participants of the channel and is completely isolated as they cannot communicate with other channels. Furthermore, each channel contains all the configurations of communication between the peers along with the list of peers and which peers are endorsing, anchor and leader peers.
- **Ledger:** The distributed shared ledger where the current and historical state of the facts are maintained. Each peer has a copy of the ledger that they share through a channel.
- **Smart Contract (chaincode):** The trusted distributed applications that contain the business logic of the system. They are responsible for all the interactions with the ledger and can be invoked by all the external applications that have access to them.
- **Ordering service:** The purpose of the ordering service, deployed on a peer or a set of peers, is to formulate the transactions into blocks and ensure the delivery of the blocks to the peers of the channel. Furthermore, it guarantees that the transactions are included in the proper order and that all peers have the same updated ledger. The most common ordering services are Solo, Kafka, and Raft. Just like peers, ordering nodes belong to an organization. And similarly to what is done with peers, a separate Certificate Authority should be used for each organization.
- **Membership Services Provider (MSP):** The MSP acts as a Certificate Authority (CA) that issues and manages the certificates utilised in order to authenticate the identity and roles of the members of the network. As Fabric implements permissioned blockchain networks, all participants of the network must be authenticated in order to perform any kind of operation. Everything that interacts with a blockchain network, including peers, applications, administrators and orderers, acquires their organizational identity from their digital certificate and their Membership Service Provider (MSP) definition.

- **Network Policy:** The blockchain network is created and regulated by a network policy that is applied across the whole network with permissions that are determined prior to the network creation by the involved organisations.

To facilitate the implementation of the trusted distributed applications that were presented in Section 3, the consortium designed the suitable blockchain network that is capable of hosting all three applications that were developed in the context of Task 4.3, as well as the tokenization use case that is developed within the context of Task 4.4.

Figure 37 depicts the testbed topology of the Hyperledger Fabric network. In total, seven (7) nodes are set up in order to serve all the applications. Four (4) out of seven (7) nodes are used purely as the foundation of the blockchain network, hosting the ledger, the chaincodes and the peers together with the single orderer that is used, while the other three (3) are mainly hosting the external applications that interact with the blockchain network and they are also providing resources for the required certificate authorities.

In particular, each of the seven (7) nodes corresponds to a single VM, which interacts with the entire blockchain network N. Three (3) organizations (FO1, FO2, FO3) are created in total, with two peers for FO1 (P1, P4) and one peer for FO2 (P2) and FO3 (P3) respectively, while nodes 1, 2, 3 and 4 are hosting one peer each as well. In this topology of the network, four (4) channels (C1, C2, C3, C4) exist with their own copy of the ledger (L1, L2, L3, L4). While (P1, P2, P3) have been added in the first three channels and are able to deploy smart contracts inside the first three channels (S1, S2, S3), P4 has been added in the C4 channel and is able to deploy smart contracts inside the corresponding fourth channel (S4). Finally, in node 1, the single orderer (O) is included.

Regarding nodes 5, 6, and 7, they are mainly hosting the external applications as well as one Certificate Authority each (CA1 or CA2). Besides Certificate Authority CA1, node 5 contains the entire KYC/KYB external application, which is interacting with channels C1 and C2. Two distinct KYC/KYB applications were developed in order to operate with the two distinguished channels C1 and C2. In a similar manner, besides Certificate Authority CA2, node 6 is hosting the whole Tokenization external application, which is interacting with channel C3. Finally, the external application for the Consent Management is hosted in node 7, which includes CA1 and is interacting with channel C4.

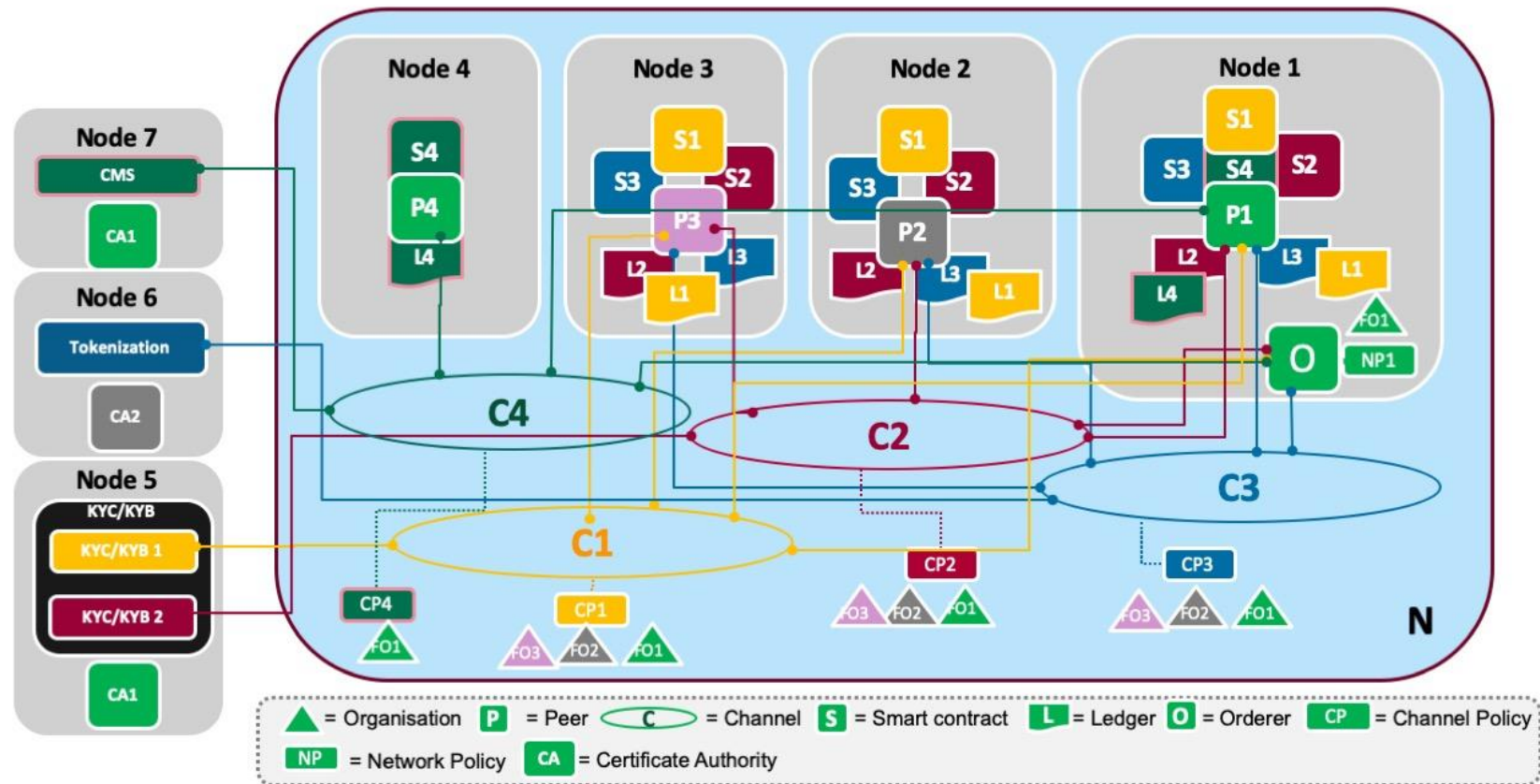


Figure 37: The final INFINITECH Blockchain network

Regarding the node interactions, the deployment and the network topology, the following table illustrates the number of nodes, the hosted services and the overall hardware resources needed per node.

Table 20: INFINITECH Blockchain network details

Node Name / Service	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7
Orderer	Yes						
Peer	Yes	Yes	Yes	Yes			
CA					Yes	Yes	Yes
KYC/KYB					Yes		
Tokenization						Yes	
CMS							Yes
GDPR							
Hardware Resources	2 vCPUs / 4GB RAM / 120 GB Disk	2 vCPUs / 4GB RAM / 100 GB Disk	2 vCPUs / 4GB RAM / 100 GB Disk	2 vCPUs / 4GB RAM / 100 GB Disk	4 vCPUs / 8GB RAM / 220 GB Disk	4 vCPUs / 8GB RAM / 220 GB Disk	4 vCPUs / 8GB RAM / 220 GB Disk
Total Hardware Resources	18 vCPUs, 40 GB RAM, 1.10 TB Disk storage						

5 Baseline Technologies and Tools

Updates from D4.8:

- Keycloak has been added as the key technology for user management and authorisation operations.

The development of the presented applications is based on a set of carefully selected technologies and tools that will facilitate the development teams to deliver the described functionalities and use cases. Towards this end, the consortium decided to exploit a set of well-established technologies and tools that include open-source software, libraries and frameworks which will enable the smooth and effortless implementation, as well as integration, of the designed use cases. The criteria during the selection process were their level of maturity, their applicability to the designed use cases and the compatibility of each selected technology and tool with the rest ones.

The following table presents the list of core technologies and tools that will be leveraged during the implementation phase of the presented applications.

Table 21: Baseline Technologies and Tools

Software Name	Short description	Version
Hyperledger Fabric	The open source enterprise-grade permissioned distributed ledger technology (DLT) platform.	2.2.0
Fabric - CA	The Hyperledger Fabric CA is a Certificate Authority (CA) for Hyperledger Fabric that will be used to authorise the various components and users.	1.4.8
Vault	Vault provides high-level policy management, secret leasing, audit logging, and automatic revocation to protect sensitive information.	1.5.3
Consul	The Consul storage backend is used to retain Vault’s data in Consul’s key-value store.	1.8.0
PostgreSQL	PostgreSQL is the RDBMS that will be utilised to store the certificates information by the Fabric CA.	12.0
Keycloak	Keycloak is utilised for managing the authorization aspects of the CMS application. It provides robust authorization and holistic user account management lifecycle operations.	15.0.2

6 Conclusions

The purpose of the deliverable herewith, entitled “D4.9 - “Permissioned Blockchain for Finance and Insurance - III” was to provide the final documentation of the efforts undertaken within the context of T4.3 “Distributed Ledger Technologies for Decentralized Data Sharing” of WP4. To this end, the deliverable included the updated documentation that supplemented the information documented in deliverable D4.8 with regard to the complete design specifications of the produced blockchain applications, the comprehensive implementation details of these applications and a presentation of the user interface and implemented functionalities, while also providing the specifications of the INFINITECH blockchain network that supports their operations and the technologies that were leveraged for their implementation and deployment.

In details, the deliverable presented the reported the results of the in-depth analysis of the key characteristics and offerings of the blockchain technology and how these were leveraged in order to set the role of the blockchain technology within the INFINITECH RA. It should be noted that the results remained unchanged from the previous iteration of the deliverable and they were reported for coherency reasons.

Furthermore, the deliverable presented the rationale behind the implementation of the blockchain applications, namely the namely (i) the Consent Management and (ii) the Know-Your-Customer (KYC) / Know-Your-Business (KYB), describing the business need they are effectively covering. It provided the final and complete design specifications of these blockchain applications and how these were translated into use cases and consequently sequence diagrams. In addition to this, the deliverable presented the complete implementation details of the delivered fully functional blockchain applications with the complete documentation of the various implemented services and their functions, as well as an overview of their source code structure. Finally, the deliverable presented their implemented core offerings and features with a walkthrough through the user’s perspective. Both blockchain applications are available to all financial and insurance sectors’ stakeholders through the INFINITECH Marketplace.

The deliverable documented the final design details of the INFINITECH blockchain network by presenting the designed network topology, elaborating on in the nodes included and their roles as well as the interactions between them. The INFINITECH blockchain network is composed of seven nodes, four of which are utilised as the blockchain network (peers), while three of them serve as the hosting environment for the developed external applications with the blockchain network and the certificate authorities. The designed blockchain applications are supported with the registration of three distinct organisations which interact through four individual channels in which three distinct smart contracts (chaincode) are deployed.

Finally, the deliverable provided the final list of baseline technologies and tools that were utilised for both the deployment of the INFINITECH blockchain network, as well as the development of the designed blockchain applications. The list was enhanced with an additional technology, namely Keycloak, that supplements the already existing technologies towards the implementation of the designed blockchain-enabled solutions.

The presented blockchain applications constitute the final fully functional release of the blockchain applications in accordance with the INFINITECH Description of Action. In this final release, all the required optimisations and improvements were implemented and integrated. The current deliverable constitutes the final report of Task 4.3 and concludes the activities of the specific task.

Table 22: Conclusions (TASK Objectives with Deliverable achievements)

Objectives	Comment
<i>Exploit the key characteristics and offerings of the blockchain technology</i>	The proposed solutions successfully exploit the key characteristics and offerings of the blockchain technology to effectively address the needs of the financial and insurance sectors with trusted, immutable, decentralised and transparent solutions in the form of blockchain applications.

<i>Provide the blockchain infrastructure which supports the requirements of the financial and insurance sectors</i>	The designed and delivered permissioned blockchain network provides the basis for the implementation of secure and decentralised blockchain applications which can be leveraged to provide innovative services, products and offerings.
<i>Design and deliver the required solutions that address core business cases of the financial and insurance sectors</i>	The delivered solutions (Consent Management, KYC/KYB) address core business use cases of the financial and insurance sectors exploiting the benefits of the permissioned blockchain technology with a set of trusted distributed blockchain applications deployed on robust and secure permissioned blockchain network.

Table 23: Conclusions – (map TASK KPI with Deliverable achievements)

KPI	Comment
<i>Design and deliver a permissioned blockchain network</i>	<p><i>Target Value = 1</i></p> <p>The specific KPI is successfully achieved with the presented solution, namely the INFINITECH Blockchain network, which was designed and documented in detail in the current deliverable.</p> <p>The designed blockchain network consists of seven nodes (four peers and three external application hosts), three organisations, four channels, four copies of the ledger and four deployed smart contracts.</p>
<i>Development of permissioned blockchain solutions</i>	<p><i>Target Value >=2</i></p> <p>The specific KPI is successfully achieved with the delivered permissioned blockchain applications, namely the Consent Management and the Know-Your-Customer/Know-Your-Business that effectively address two core business use cases of the financial and insurance sectors. The final fully functional versions of the aforementioned blockchain applications are delivered within the current deliverable.</p>

Appendix A: Literature

- [1] D. Yaga, P. Mell, N. Roby and K. Scarfone, Blockchain technology overview, National Institute of Standards and Technology, 2018.
- [2] P. Treleaven, R. Gendal Brown and D. Yang, “Blockchain Technology in Finance,” *Computer*, vol. 50, no. 9, pp. 14-17, 2017.
- [3] M. Niranjnamurthy, B. N. Nithya and S. Jagannatha, “Analysis of Blockchain technology: pros, cons and SWOT,” *Cluster Computing*, vol. 22, no. S6, pp. 14743-14757, 2018.
- [4] M. Casey, J. Crane, G. Gensler, S. Johnson, N. Narula and C. A. Wyplosz, The impact of blockchain technology on finance, Geneva: International Center for Monetary and Banking Studies (ICMB), 2018.
- [5] “Hyperledger Fabric – Hyperledger,” 2020. [Online]. Available: <https://www.hyperledger.org/use/fabric>. [Accessed 5 September 2020].
- [6] “Introduction — hyperledger-fabricdocs master documentation,” Hyperledger, 2020. [Online]. Available: <https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatis.html>. [Accessed 03 September 2020].
- [7] K. Tuononen, “The impact of PSD2 directive on the financial services industry.,” 2019.
- [8] E. Commission, “European Commission,” 2015. [Online]. Available: https://ec.europa.eu/info/law/payment-services-psd-2-directive-eu-2015-2366_en. [Accessed 10 November 2021].
- [9] Deloitte, “Open banking - Privacy at the epicentre.,” Deloitte, 2018.
- [10] KPMG, “KPMG,” 2021. [Online]. Available: <https://home.kpmg/ph/en/home/insights/2019/07/open-banking-opens-opportunities-for-greater-value.html>. [Accessed 15 November 2021].
- [11] E. Union, “Official Journal of the European Union,” 2016. [Online]. Available: <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN>. [Accessed 05 November 2021].
- [12] E. & Y. G. Limited., “How banks can balance GDPR and PSD2,” 2019. [Online]. Available: https://www.ey.com/en_lu/banking-capital-markets/how-banks-can-balance-gdpr-and-psd2. [Accessed 14 November 2021].
- [13] KPMG, “Open banking opens opportunities for greater customer,” KPMG, 2021.
- [14] E. B. Association, “B2B Data Sharing: Digital Consent Management as a Driver for Data Opportunities,” Euro Banking Association, 2018.
- [15] “Consent Receipt Specification – Kantara Initiative,” Kantara Initiative, 2020. [Online]. Available: <https://kantarainitiative.org/download/7902/>. [Accessed 1 September 2020].
- [16] “Supporting Material - Archived Groups - Kantara Initiative,” Kantarainitiative.org, 2020. [Online]. Available: <https://kantarainitiative.org/confluence/display/archive/1+-+Supporting+Material>. [Accessed 2 September 2020].
- [17] A. Polyviou, P. Velanas and J. Soldatos, “Blockchain Technology: Financial Sector Applications Beyond Cryptocurrencies,” *Proceedings*, vol. 28, no. 1, p. 7, 2019.

- [18] N. Kapsoulis, A. Psychas, G. Palaiokrassas, A. Marinakis, A. Litke and T. Varvarigou, “Know Your Customer (KYC) Implementation with Smart Contracts on a Privacy-Oriented Decentralized Architecture,” *Future Internet*, vol. 12, no. 2, p. 41, 2020.
- [19] K. I. P. L. D. V. C. J. P. K. A. Bhaskaran, “Double-blind consent-driven data sharing on blockchain,” in *2018 IEEE International Conference on Cloud Engineering (IC2E)*, 2018.
- [20] I. Karagiannis, K. Mavrogiannis, J. Soldatos, D. Drakoulis, E. Troiano and A. Polyviou, “Blockchain Based Sharing of Security Information for Critical Infrastructures of the Finance Sector,” *Computer Security Lecture Notes in Computer Science, Computer Security. IOSEC 2019, MSTEC 2019, FINSEC 2019*, vol. 11981, pp. 226-241, 2020.
- [21] R. S. M. S. W. M. & S. R. Norvill, “Blockchain for the Simplification and Automation of KYC Result Sharing,” in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2019.
- [22] A. C. P. C. J. T. E. B. a. R. S. W. Radomski, “ERC-1155 multi token standard,” 2015. [Online]. Available: <https://eips.ethereum.org/EIPS/eip-1155>. [Accessed 20 November 2021].
- [23] F. V. a. V. Buterin, “ERC-20 token standard,” 2015. [Online]. Available: <https://eips.ethereum.org/EIPS/eip-20>. [Accessed 19 November 2021].
- [24] D. S. E. E. a. N. S. W. Entriken, “EIP-721: ERC-721 non-fungible token standard,” 2018. [Online]. Available: <https://eips.ethereum.org/EIPS/eip-721>. [Accessed 15 November 2021].
- [25] “Blockchain basics: Hyperledger Fabric,” IBM Developer, 2020. [Online]. Available: <https://developer.ibm.com/technologies/blockchain/articles/blockchain-basics-hyperledger-fabric/>. [Accessed 29 August 2020].
- [26] “EU data protection rules,” European Commission, 2020. [Online]. Available: https://ec.europa.eu/info/law/law-topic/data-protection/eu-data-protection-rules_en. [Accessed 27 August 2020].
- [27] D. Deuber, B. Magri and S. A. K. Thyagarajan, “Redactable Blockchain in the Permissionless Setting,” *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 124-138, 2019.
- [28] G. A. Stevens, B. Magri, D. Venturi and E. Andrade, “Redactable Blockchain – or – Rewriting History in Bitcoin and Friends,” *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 111-126, 2017.