Interoperable Distributed Ledger Technologies - I-DELTA

Deliverable

I-DELTA Use-Case Design: Actors, Actions, Technologies, and Protocols

Use-Case Name: All use cases

Use-Case Provider: All use cases

Editor: Dakik Software

Document properties

Distribution	Confidential
Version	0.1
Editor	Dakik Software
Authors/ Contributors	Partners from Turkey and Czechia
Pages	

ITEA 3 Project 18021

I-DELTA Use-Case Design: Actors, Actions, Technologies, and Protocols

Use-Case Name: Template Use Case

Use-Case Provider: Dakik Software

Editor: Dakik Software

Sub-document properties

Distribution	Confidential
Version	[01]
Editor	Kamer Kaya
Authors/ Contributors	Dakik Software
Pages	10

1. Introduction

This document presents a primitive proof of concept use case and its design regarding atomic swaps between two blockchains. In this PoC, two HyperLedger Fabric Blockchains are set up and connected to a NodeJS backend to simulate two unique users and a ReactJS frontend to provide a skeletal user interface that provides a simple UI to interact with the DLT.

1.1. Purpose and Audience of the Document

The purpose of the document is to dissect an I-DELTA use case to its functional primitives and provide the details of each primitive via a sequence of intra- and cross-chain operations. For each use case, a similar document will be prepared. The audience is all the consortium members contributing to the design and implementation of the I-DELTA platform.

1.2. Document Structure

Chapter 2 introduces the scope of the use-case, defines its actors and actions, and provides technical details. Chapter 3 presents a detailed view of each action of the use case. Chapter 4 is reserved for providing alternatives for the design and technologies used for the use case.

1.3. Terms and Definitions

Table 1: Terms and definitions

Term	Definition
DLT	Distributed Ledger Technology
PoC	Proof of Concepts
SDK	Software Development Kit
DID	Decentralized Identifier
UUID	Universally Unique Identifier
SC	Smart Contract
YFT	*Your First Term*
YST	*Your Second Term*

2. Scope

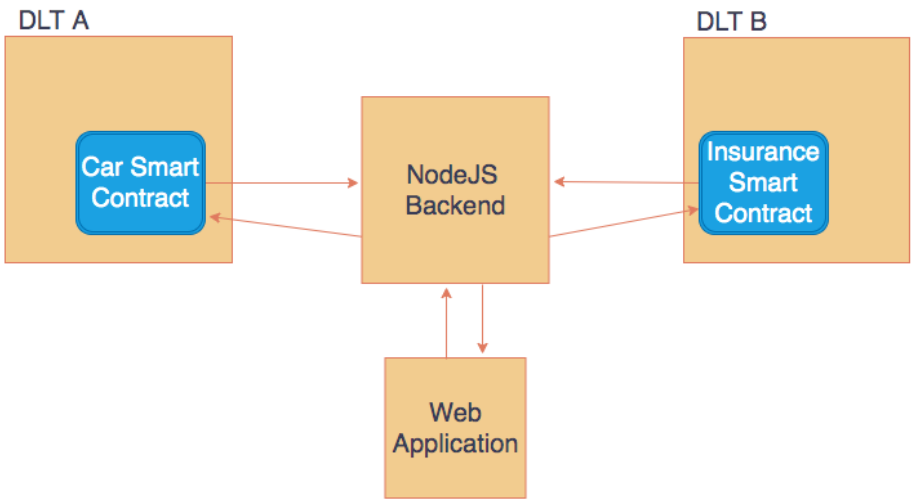
2.1. Business Context

This section presents a primitive PoC involving an atomic cross-chain operation between two DLTs. In this use case, there exist two contracts on two different DLTs. The contracts are deployed on two HyperLedger Fabric instances, the backend is expected to be implemented via NodeJS and the frontend uses ReactJS to provide a skeletal user interface that forms a simple UI to interact with these DLTs seamlessly.

In our use case, we have two smart contracts with each deployed on a different blockchain. The *Car Smart Contract* is responsible for creating new car assets with relevant properties and changing the ownerships of existing car assets on the chain. The *Car Insurance Smart Contract*, as the name implies, is responsible for creating insurances for the car assets created in blockchain A. The contract can create new insurances and change the insured's name via the business logic implemented within it.

In this document, a smart contract is considered as an agreement among the users in the form of computer code. They run on the blockchain, so they are stored on a distributed database and cannot be changed. The transactions that happen in a smart contract are processed by the blockchain, which means they can be sent automatically without a third party. The transactions only happen when the conditions in the agreement are met (i.e: the business logic defined) and there is no third party involved, so there are no issues with trust.

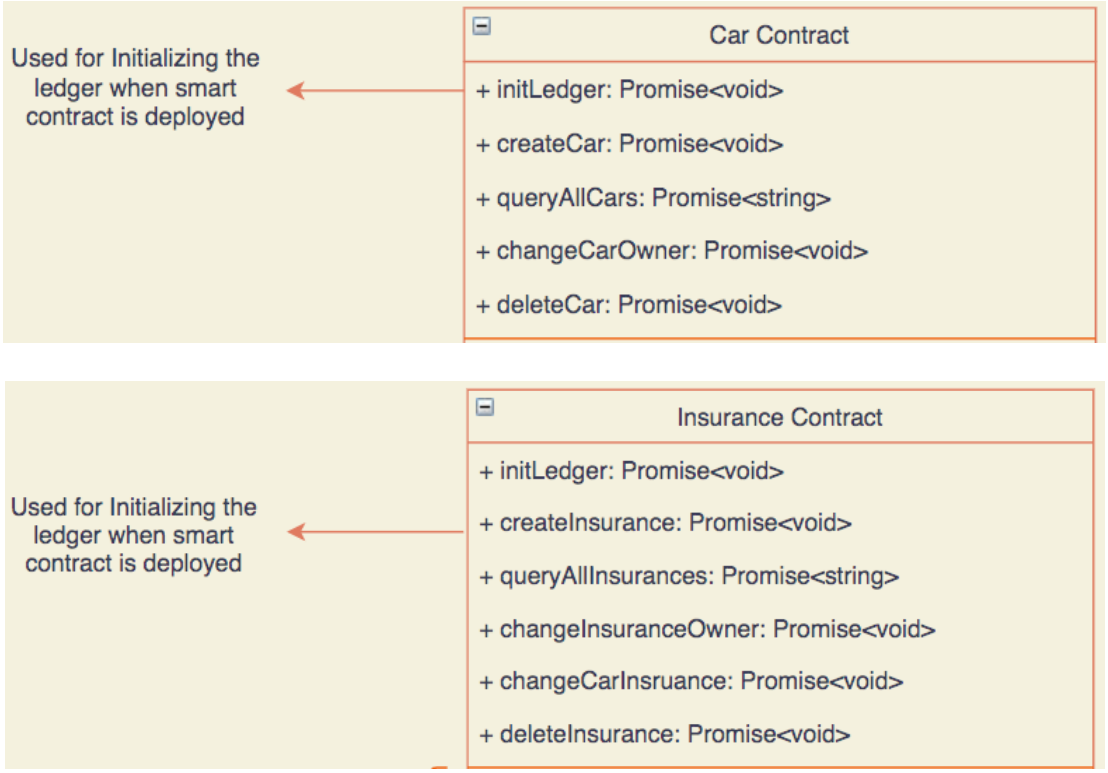
2.2. Technical Infrastructure



The figure above provides *an eagle-eye view* of the relationships between the different technologies being utilized by the use case. The arrows in the figure show all the possible flows of communication, information, money, transactions, etc.

As can be seen in the figure, the two DLTs are connected via a NodeJS backend server. The server itself interacts with a web application to obtain the necessary information to carry out transactions on the relevant blockchains via the deployed smart contracts.

As mentioned above, each DLT network has smart contracts deployed that are responsible for handling the business logic for the use case in question. The smart contracts are the entities the NodeJS backend interacts with using the fabric software development kit (SDK) provided by Hyperledger. The details of the smart contracts are given below.



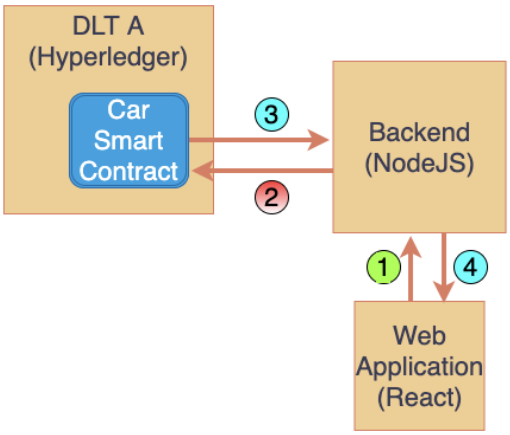
Although our PoC use-case does not, the actual use-cases can have other components such as *wallets*, *oracles,* etc. If this is the case they must be introduced in this section.

2.2.1. Use-case Actions

The smart contract class diagrams described above summarize the functions and (if there exist) their return types utilized in the backend to create a functioning first draft of our use case. We can think of this information containing the building blocks of the protocols (of the actions) the use-case will have. For this use-case, there exist 3 actions that can be performed:

1. Create Car
2. Create Insurance
3. Change Owner (Car and Insurance)

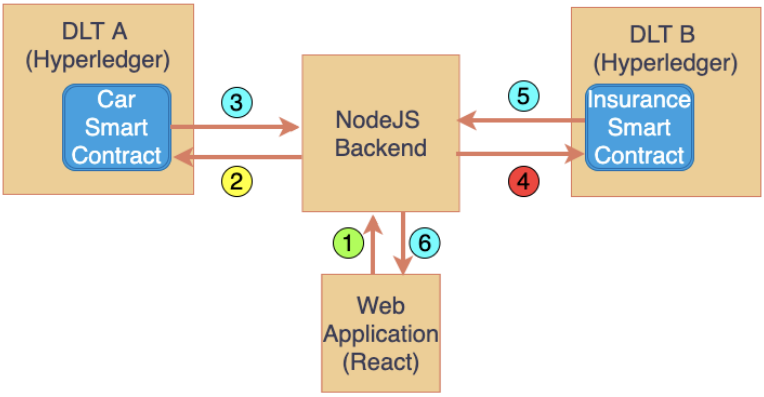
This section describes each of these actions in more detail with related diagrams and explanations.



Action 1: Create Car

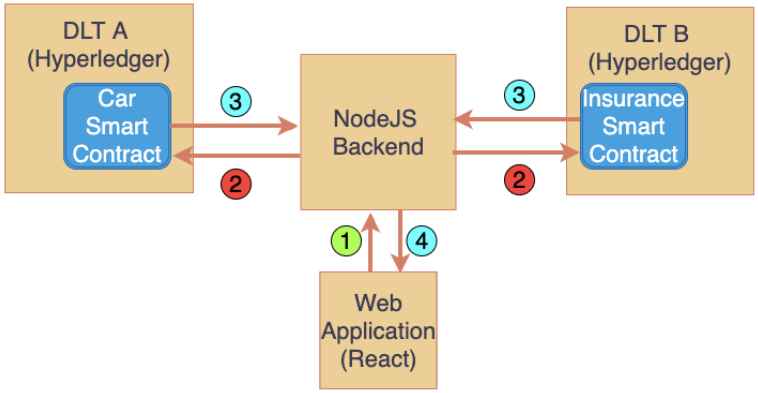
This action only interacts with the Car contract; the user creates a car asset with no insurance. The action is described by the diagram on the left.

The Car smart contract is living on DLT A (which is a Hyperledger instance as mentioned before). The initial request is created via a frontend (1: green), then a state change request is sent to the Car smart contract (2: red), the confirmation is obtained from the smart contract (3: blue), which is then forwarded to the frontend (4: blue).



Action 2: Create Insurance

This action interacts with both contracts. The first one, the Car smart contract is accessed to get/read the ownership information (2: yellow with reply 3: blue), and the Insurance smart contract is accessed for a state change (4: red with reply 5: blue) to create new insurance for the same car.



Action 3: Change Owner (Car & Insurance)

This action interacts with both contracts; the Car and Insurance smart contracts are accessed (2: red) to change the car ownership information and the owner info on the insurance, respectively. Both of these operations must happen at the same time for consistency purposes. If there is an error for one, the other operation must also be cancelled. The implementation must also guarantee that there are no deadlocks or race conditions that can damage the integrity and consistency of the data. A simple UI mock-up is given below for this operation.

2.3. Logical View

2.3.1. System Decomposition – Design phase

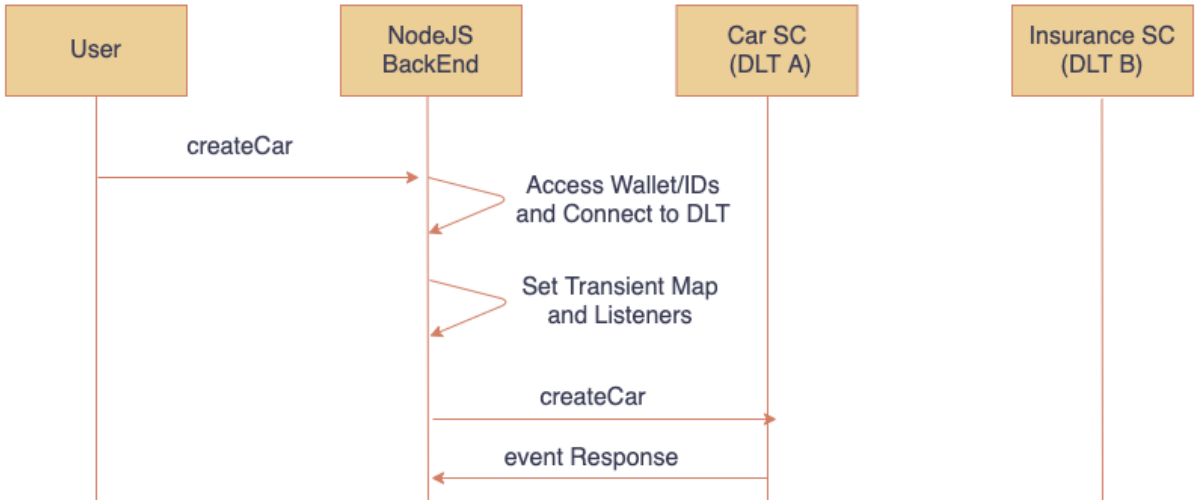
2.3.1.1. Overview and Vision

This section of the document shows sequence diagram(s) designed to illustrate the aforementioned actions in the current use case. Each sequence diagram will be shown separately with accompanying explanations describing the diagram. We will first use the first action (create Car) of the PoC. The next diagram corresponds to a successful transaction taking place where the owners in both chains are successfully changed. The last diagram represents the situation where things can go wrong and the transaction fails.

Although we do not explicitly do this here, the safety and liveness properties of the transaction per action must be discussed by using the details of the protocol and the technical details of the DLTs.

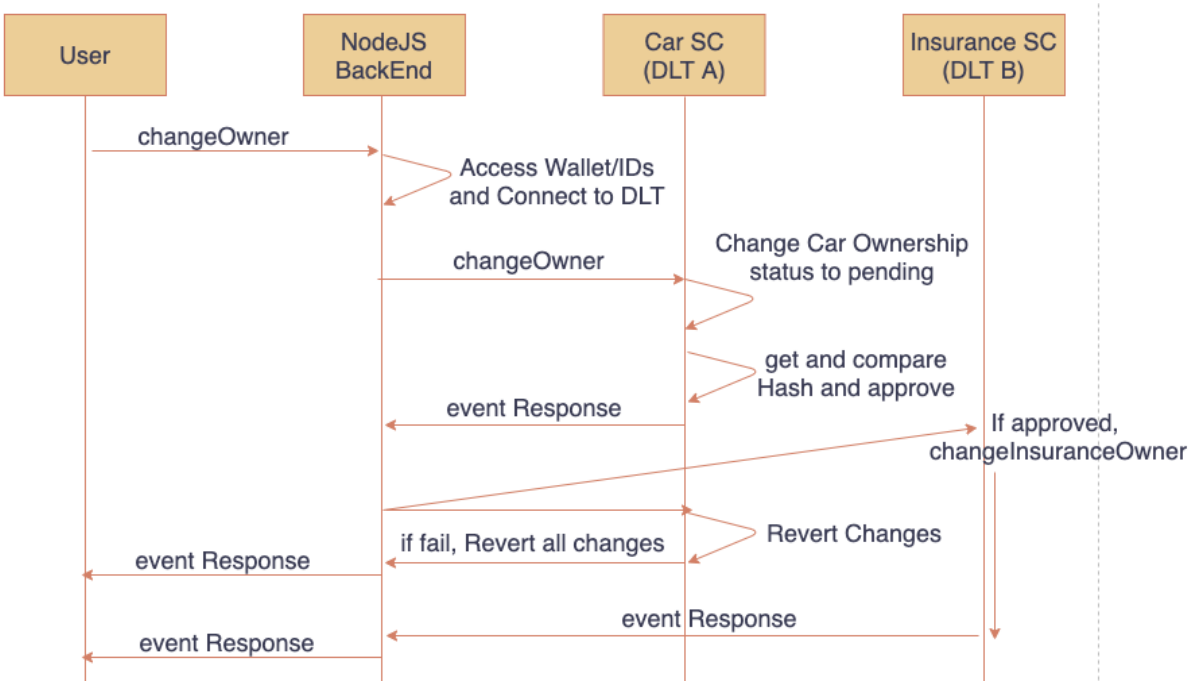
2.3.1.2. Runtime View

2.3.1.2.1. Runtime View of Successful Create Car Transaction



The first diagram represents a perfect scenario where the Create Car action initiates and terminates successfully. As the diagram shows, only the Car smart contract on DLT A is accessed and the other DLT, B, does not contribute to the action. The details given in the diagram, e.g., set transient map and listeners, depending on the DLT, as well as the libraries and tools used for the backend. As the sample diagram illustrates, in our PoC, the NodeJS backend accesses the IDs stored inside the wallet to allow interaction with the DLT. For our application, the Hyperledger fabric does not allow NodeJS to interact with the blockchain network without the use of generated wallet IDs and so it is paramount to register IDs into the wallet for further functionality. After the IDs have been retrieved from the wallet, we now have permission to interact with the DLT.

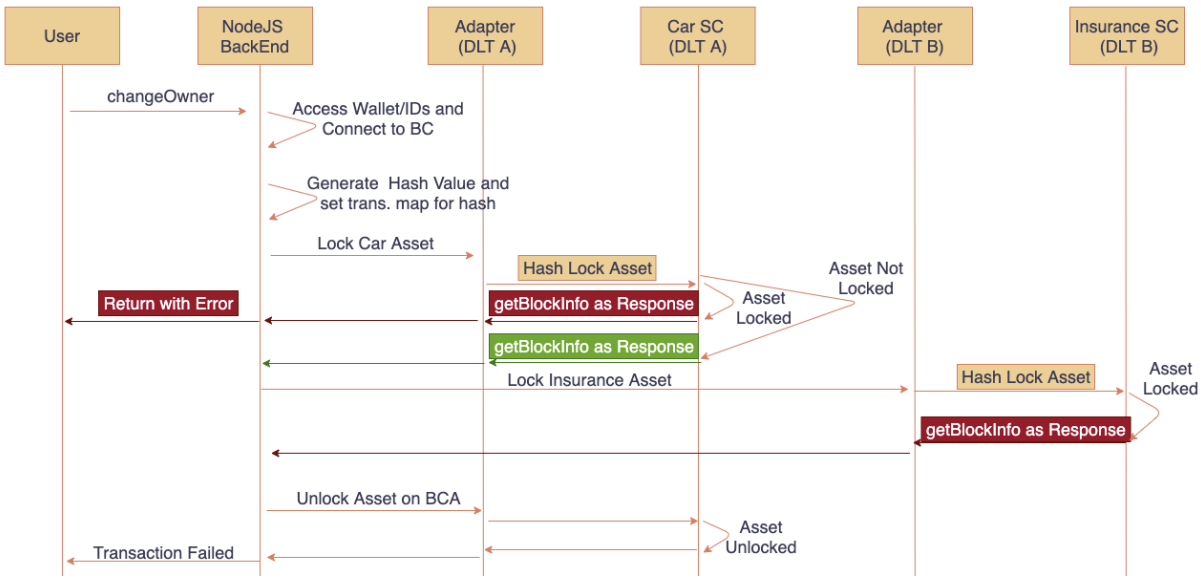
2.3.1.2.2. Runtime View of Successful Change Owner Transaction



The diagram above shows the sequences of transactions to realize the change owner action. For this PoC, the protocol is using a hash-locking mechanism to guarantee atomicity of the action but other solutions from the literature can also be leveraged. Note that the diagram only shows the smart contracts but not the adapters/connectors, e.g., additional actors required to implement the action.

For this implementation, we take the advantage of Hyperledger fabric's private data functionality where data can be stored privately on the ledger but not committed. We take advantage of a "map" data structure as the name implies to pass on private data to the DLT. In this specific implementation, the private data consists of a randomly generated *secret* that is used to *hash-lock* the contract as well as the timestamp of when the map is generated. The timestamp is used to implement the *time-lock* functionality of a *hash-time-lock* contract. Once both assets on both chains are locked, the transaction continues. The backend server promptly reveals the secret that can unlock the *hash-lock* and the transaction proceeds as expected.

2.3.1.2.3. Runtime View of Unsuccessful Change Owner Transaction



Compared to the previous one, the diagram also identifies the adapters that will be used hence can be considered a better, more detailed view of the action. During the entire ChangeOwner transaction, there are a bunch of potential vulnerabilities that can occur and can cause a deadlock or instant failure in transactions. For our design and protocol selection, since we are communicating with the blockchain via the Hyperledger Fabric SDK, it is highly probable that in the case of two simultaneous users, one may lock the asset on DLT A while the other locks the asset on DLT B simply due to lag or network issues. In such a case, a deadlock will occur where no one can access the asset any longer. It is also possible that due to some unforeseen issues, the asset may remain locked for prolonged periods. It is also of paramount importance that in the event of a failure, the state of the asset on both remains as it was.

To combat such grievances, a robust protocol implementation is needed that itself is future-proof and provides the necessary functionality, protection, and usability to encompass differing use cases. The above sequence diagram represents the necessary actions taken during such errors. In the case of transaction failure, the NodeJS backend calls on the necessary functions in the adapters and smart contracts to restore the chain state to its original setting. The protocol also checks for the timestamp (stored as private data) if the asset is already locked to check if the asset has been locked for more than some predefined duration. If so, the asset is unlocked on both chains. Using such functions to lock/unlock assets based on the error handling the implementation, the protocol can display barebones working functionality.

2.3.2. Future Plans and Possible Changes

Since we are at the protocol design state, potential changes are possible. If there are such changes you can state them here.

3. Specific Platform Needs, Architectural Decisions, and Alternatives

In this section, first, specific platform needs must be described. Then the architectural decisions and remarks can be discussed. The technology stack such as the runtime environment and the programming language will be described to understand the technical requirements of the I-DELTA platform.

Topic	**Decision/Selection/Rationale/Alternative**
Hyperledger	...
React JS	...
Node.JS	...

3.1. Runtime environment and programming language

This section contains the technical specifications of the implementation environment.

4. Other Comments and References to State of the Art

You can alter the structure of the document in the way that best suits your use case. If you have other details to add, please use this section. If you believe there exists a protocol, algorithm, etc. we need to follow through the implementation of the platform, also add them here with an explanation of (1) *what does it propose* and (2) *why it is necessary and important for I-DELTA and also for your use case*.

(e.g.:

[1] Peter Robinson and Raghavendra Ramesh. *General Purpose Atomic Cross-chain Transactions*, 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), 10.1109/ICBC51069.2021.9461132, May 2021

The General Purpose Atomic Cross-chain Transaction protocol allows composable programming across multiple DLTs. It allows for inter-contract and inter-blockchain function calls that are both synchronous and atomic: if one part fails, the whole call execution tree of function calls is rolled back. The protocol operates on existing Ethereum blockchains without modification. It works for both public permissioned and consortium blockchains. Additionally, the protocol is expected to work across heterogeneous blockchains other than Ethereum.

Since I-DELTA is interested in connecting heterogeneous DLTs, public and private ones, the protocol may be of use for some of the use-cases in case some form of atomicity is required.

)

I-DELTA Use-Case Design: Actors, Actions, Technologies, and Protocols

Use-Case Name: Digitalization of Legal Agendas

Use-Case Provider: EXPECT-IT, Selfcon Systems, K&P Partners

Editor: Pavel Furch

Sub-document properties

Distribution	Confidential
Version	[01]
Editor	Pavel Furch
Authors/ Contributors	Michal Batko, Karel Slavíček, Pavel Furch
Pages	7

1 Introduction

Our application scenario covers voting for both citizens of a city and special use at general meetings using digital technologies based on DLT. The voting process is inherently regulated by a number of rules, some of them legally required to be verified by a notary record. A notary functions as a 3rd party arbiter validating votes and decisions made by legitimate voters.

Important decisions might include public polls for city hall decisions or in case of companies change of a management, structure or distribution of profit. electronically. Using the GPC framework, embedded devices and distributed ledger technology, an external arbiter is no longer required. Notary is substituted by a DLT register receiving data from embedded devices.

A wearable or easy to transport HW module which will perform authorization of the legal act and save the transaction record into DLT is an integral part of our use case. The reason for this approach is to make utilization of DLT transparent to non IT specialists. For end users like lawyers who are not professionals in IT it is necessary to provide an easy to use solution, not requiring knowledge of used technology like DLT, digital signing, encryption algorithms, etc. The HW module will provide some authentication mechanisms like fingerprint scanner or keypad for input of PIN, push-buttons for voting, secure storage of PKI keys and communication interface.

1.1 Purpose and Audience of the Document

The purpose of the document is to dissect an I-DELTA use case to its functional primitives and provide the details of each primitive via a sequence of intra- and cross-chain operations. For each use case, a similar document will be prepared. The audience is all the consortium members contributing to the design and implementation of the I-DELTA platform.

1.2 Document Structure

Chapter 2 introduces the scope of the use-case, defines its actors and actions, and provides technical details. Chapter 3 will present a detailed view of each action of the use case (in works).

1.3 Terms and Definitions

Table 1: Terms and definitions

Term	Definition
DLT	Distributed Ledger Technology
PoC	Proof of Concepts
SDK	Software Development Kit

2 Scope

2.1 Business Context

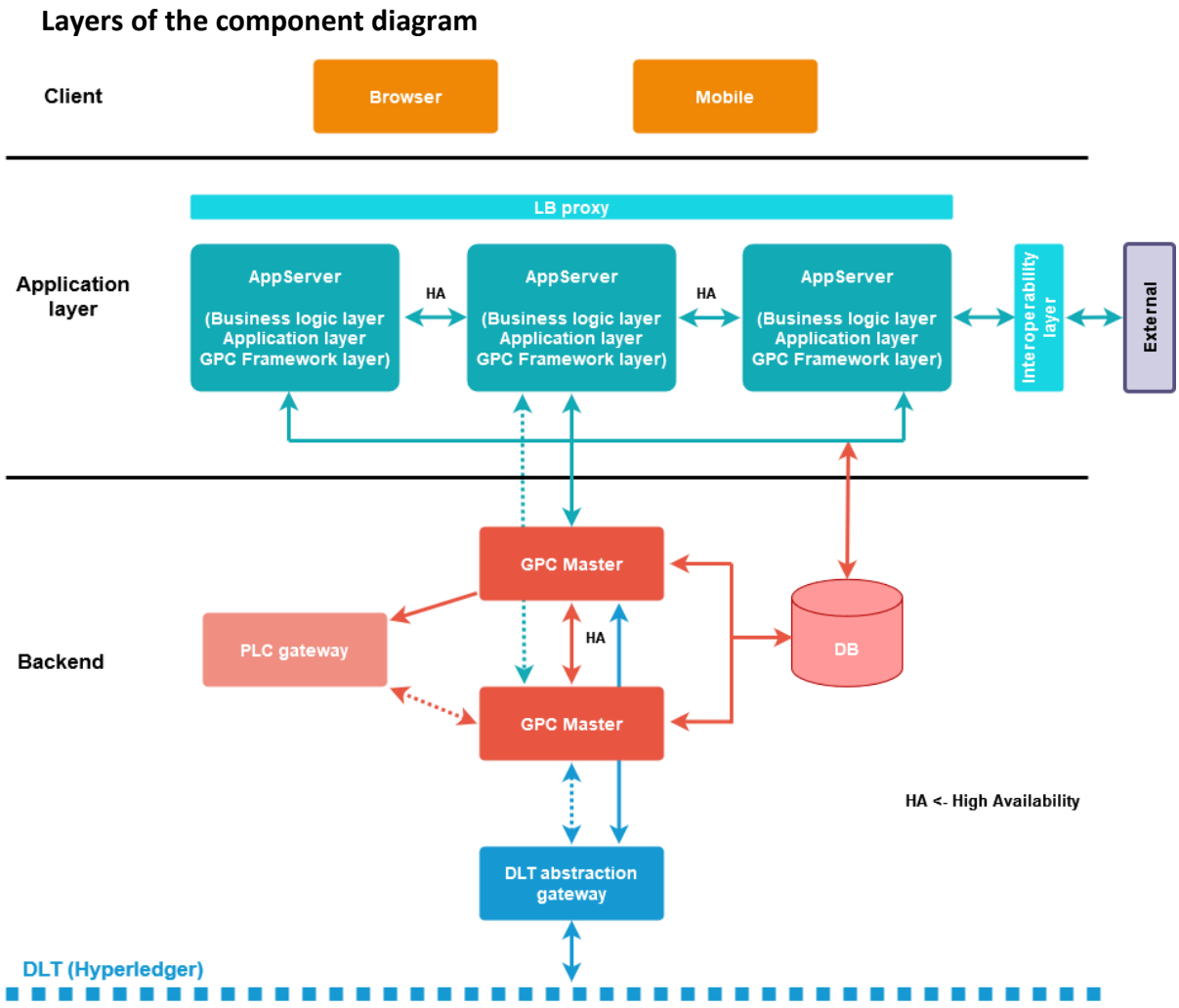
Our application scenario covers voting at company general meetings using digital technologies based on DLT.

As it is the highest body in companies and organizations the general meetings are regulated by a number of rules, some of them legally required to be verified by a notary record. A notary functions as a 3rd party arbiter validating votes and decisions made by general meeting members. Important decisions might include changes of management, structure or distribution of profit in a company.

We aim to develop a technology to hold voting electronically. Using the GPC framework, embedded devices and distributed ledger technology, an external arbiter is no longer required. Notary is substituted by a DLT register receiving data from embedded devices.

2.2 Technical Infrastructure

The individual layers of this platform will have precisely defined and publicly known interfaces so that for a specific application it will be possible to combine the solution developed by us with third-party components if it is advantageous for the end user, e.g. due to the availability of special devices or better technical parameters of some components.



End User Service Layer / Business Logic

The entire solution is built from the perspective of design and operation of services for end users, using the (technical) resources of the lower layers. At this level, there are means for designing end-user services and their corresponding service catalogs, means for problem decomposition, defining agendas and how to execute them, defining the necessary data and how to collect/calculate it, defining control mechanisms, KPIs, etc.

Operationally, there is the status of the platform in terms of the currently offered / available services, overviews of current requirements and their fulfillment, identification of problems of the platform impacting on service availability, or the fulfillment of quality assurance arrangements / availability of services, etc.

Application layer

Individual sub-applications, functionalities and support services operated within the platform or offered by external providers / service providers with which the platform is connected, etc.

Middleware

At this level, the operational management of the entire IoT ecosystem takes place. It includes the means for managing the central and end infrastructure, integration with native and external applications and functionalities, monitoring the status of sub-components, implementation of event management processes, incident management, problem management and change management (or Service Operation in general).

Key functionalities at this level are mainly adding/removing/replacing individual platform components, i.e. devices, applications, service providers, etc.
There is also support for technologies including distributed systems, cloud computing, edge computing, temporal and network relational databases and computations/operators over a generic data model.

In the area of security, support for encryption, data anonymization and pseudonymization, authentication, authorization, security model and role-based access control (RBAC). Resources for support for security functions will also be implemented at the layers to the extent necessary for lower layers.

Communication and interconnection layer

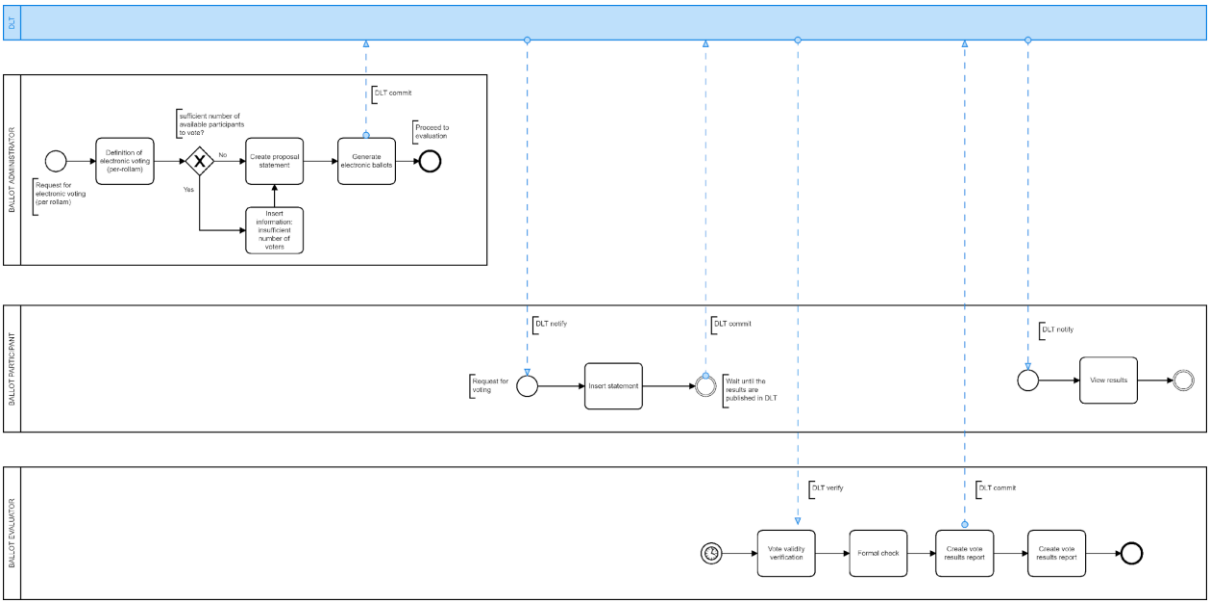
This layer contains both hardware and software resources and is used to ensure mutual communication (whether permanent, intermittent, on-demand, etc.) between the elements that make up the IoT ecosystem.

Brokerage layer – PLC Gateway

This layer contains concentrators for connecting several hardware modules (sensors, actuators, display devices, etc.). Individual brokers are globally addressable throughout the ecosystem and it is the smallest directly queryable unit offering certain (service catalog defined) services. In addition to collecting data and executing defined commands over connected end-elements, devices in this layer can provide other, more complex services towards the IoT ecosystem – typically Edge Computing support (e.g. for input aggregation, data preprocessing, etc.), temporary data storage and caching, custom control logic in case of unavailability of parent control nodes (either due to failure or planned in areas with non-guaranteed permanent network connectivity).

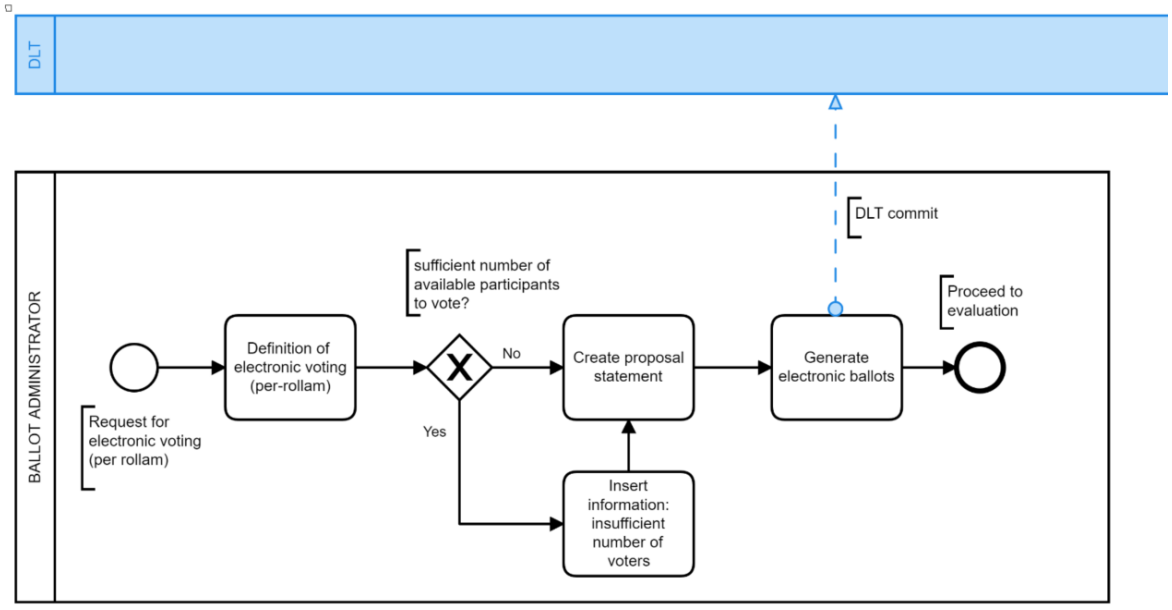
2.2.1 Use-case Actions

Complete overview of the voting



The diagram reveals administration, participant and evaluator workflow, all communicating through commits written to and received from DLT (Hyperledger Fabric in our case). Let's have a closer look at each step below.

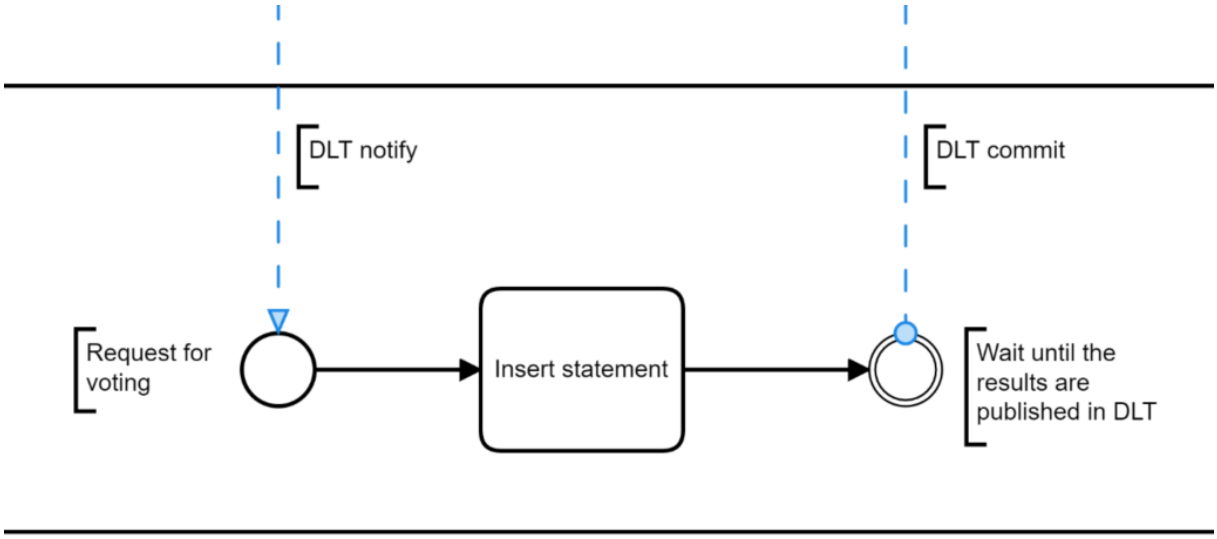
1. Ballot administrator workflow

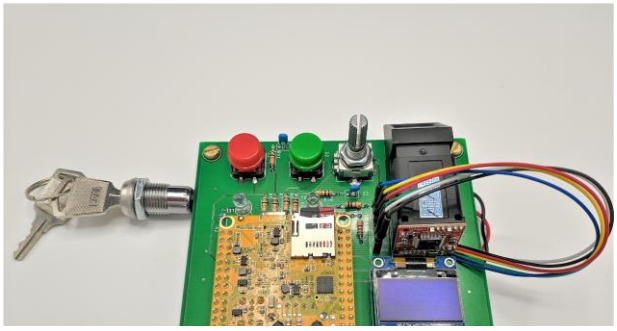


By signing into the application a ballot administrator creates a request for remote voting (per rollam). For the subsequent step the user sets properties of the voting (definition of the voting). These properties include how big the majority of voters must be in order to have a valid result, voting period, description, additional conditions for valid vote etc. If the number of available participants is sufficient a proposal statement with generated electronic ballots are created. Ballots are then committed into our Hyperledger Fabric, making it available for eligible voters for a defined time period.

2. Ballot participant workflow

The voter (end-user of our application) receives notification for the voting. Now he can insert his vote through either an application or a dedicated voting device, which we have developed for cases when the voting device is required to remain at an office or public place. Both of these choices possess authentication tools. The vote is committed into DLT. As soon

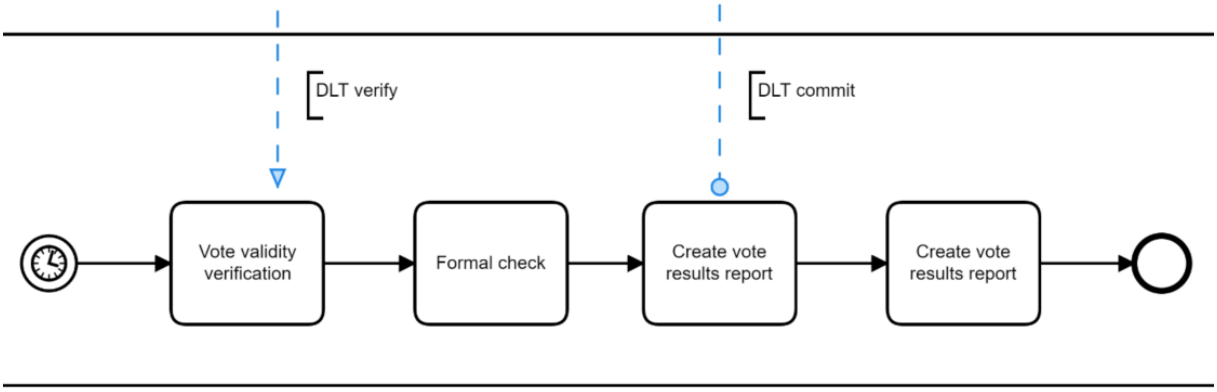




as the voting results are available (processed in the evaluator layer) the user receives a notification.

3.

4. Evaluator workflow



When the time period defined at the beginning (definition of electronic voting) passes the received results are put under validity verification process and formal check. If all data met the required conditions a result is reported to both users and the administrator.

2.3 Logical View

2.3.1 System Decomposition – Design phase

2.3.1.1 Overview and Vision

This section of the document shows sequence diagram(s) designed to illustrate the aforementioned actions in the current use case. Each sequence diagram will be shown separately with accompanying explanations describing the diagram.

3 Specific Platform Needs, Architectural Decisions, and Alternatives

Our limited experience with the Hyperledger Fabric leads us to an idea that a more robust implementation is needed to contain the user identification (used for our voting workflow). Our proposition is therefore creation of a common ledger containing user/citizen identities used among all use cases. With the ledger at hand we will be able to write and read the user data through commits but we advise that the ledger itself should be set by the DLT/interoperability providers among our partners.

I-DELTA Use-Case Design: Actors, Actions, Technologies, and Protocols

Use-Case Name: P2P Marketplace

Use-Case Provider: KoçSistem

Editor: Aylin Yorulmaz

Sub-Document properties

Distribution	Confidential
Version	0.1
Editor	Aylin Yorulmaz
Authors/ Contributors	Koç Sistem
Pages	5

1. Introduction

This document presents a primitive proof of concept use case and its design regarding blockchain based P2P trading of energy. In this PoC, two HyperLedger Fabric Blockchains are set up and connected to a NodeJS backend to simulate two unique users and a ReactJS frontend to provide a skeletal user interface that provides a simple UI to interact with the DLT.

1.1. Purpose and Audience of the Document

The purpose of the document is to dissect an I-DELTA use case to its functional primitives and provide the details of each primitive via a sequence of intra- and cross-chain operations. For each use case, a similar document will be prepared. The audience is all the consortium members contributing to the design and implementation of the I-DELTA platform.

1.2. Document Structure

Chapter 2 introduces the scope of the use-case, defines its actors and actions, and provides technical details. Chapter 3 presents a detailed view of each action of the use case. Chapter 4 is reserved for providing alternatives for the design and technologies used for the use case.

1.3. Terms and Definitions

Table 1: Terms and definitions

Term	Definition
DLT	Distributed Ledger Technology
PoC	Proof of Concepts
SDK	Software Development Kit
DID	Decentralized Identifier
UUID	Universally Unique Identifier
SC	Smart Contract

2. Scope

2.1. Business Context

Blockchain is one of the most important technologies to emerge in recent years, with many experts believing it will change our world in the next two decades as much as the internet has over the last five. Although it is early in its development, firms pursuing blockchain technology include IBM, Microsoft, KT, JPMorgan Chase, Nasdaq, Foxconn, Visa, Mastercard, and shipping giant Maersk. The applications for blockchain technology seem endless. While the first obvious ones are financial: international payments, remittances, complex financial products, and cryptocurrency, it can also solve problems and create new opportunities in healthcare, defence, supply chain management, luxury goods, government and voting, and other industries.

What is blockchain: A blockchain is a single version of the truth made possible by an immutable and secure time-stamped ledger, copies of which are held by multiple parties.

Why it matters: It shifts trust in business from an institution or entity to software and could someday spell the demise of many traditional companies. It also promises to make tradeable many assets that are illiquid today, enable our devices and gadgets to become consumers, and bring trust to many areas of business, eliminating fraud and counterfeiting in the process.

How it works: Cryptography secures the data and new transactions are linked to previous ones, making it near-impossible to change older records without having to change subsequent ones. And because multiple 'nodes' (computers) run the network, one would need to gain control of more than half of them in order to make changes.

Why it's disruptive: At the very least, it promises to make firms' back-end operations more efficient and cheaper, but down the line, it could replace middleman companies altogether.

Business opportunities: New services and products will pop up in areas such as creating and trading assets, tracking provenance, managing supply chains, managing identities, and in providing ancillary services to the software itself. According to Deloitte's 2019 Global Blockchain Survey, 53 percent of respondents said that blockchain technology has become a critical priority for their organizations and 83 percent of those surveyed believe there are compelling use cases for the technology in the enterprise.

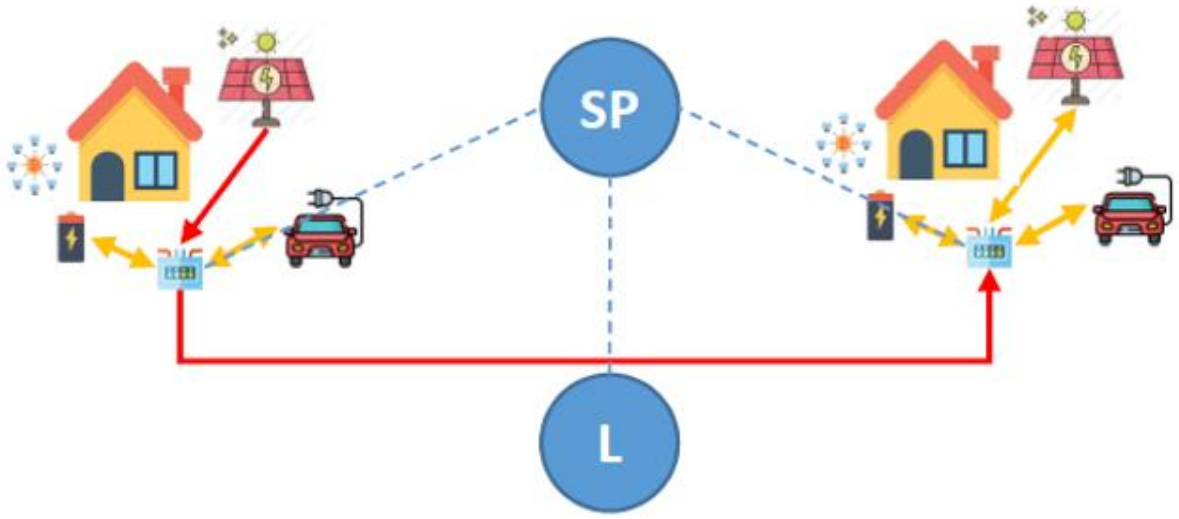
Although some executives might fear software replacing their role or their companies, even email hasn't killed snail mail. The technology does promise to change existing market share, but companies can avoid becoming obsolete by seizing upon new opportunities. In fact, blockchain technology will enable companies to offer services that previously were impossible without it.

2.1.1. Use-case Actions

The smart contract class diagrams described above summarize the functions and (if there exist) their return types utilized in the backend to create a functioning first draft of our use case. We can think of this information containing the building blocks of the protocols (of the actions) the use-case will have. For this use-case, there exist 3 actions that can be performed:

1. Selling and buying order

2. Settlement



This section describes each of these actions in more detail with related tables and explanations.

Actors: SP: Service Provider, L: Local Distributor, Prosumer_1, Consumer_1

Scenarios: Prosumer_1 has solar panels on his rooftop. He buys a metering device. He downloads the mobile app that lets him join his local micro-grid community. The app is enabled by his Participant Id authentication. He earns a currency in simulation for selling his power to Consumer_1 in the same grid on the local market.

Consumer_1 downloads the app because he wants to buy power from Prosumer_1 in the same grid. He sets his budget for local renewable energy in the app and pays in a currency.

SP creates the rules of the marketplace (inside micro-grid), is paid to run settlements between prosumers and consumers, and allows local value added services into the marketplace.

Usage scenario	Selling and Buying Order Placement
Description	Scenario that belongs to mutual contract depending on match of purchase and sell order in the market place
Actors	• Panel owner • Battery
Assumptions	• Actors must exist in the system and defined • Battery must be on and working • Battery must be connected to consumer end points • Battery must have sufficient energy stored • Buy Order – Sell Order N-N relation (selling transctions may be completed as a whole or partially) • Date must be in format dd MM YYYY HH:MM , transfer is subject to within one hour of release date

Steps	2. Through which asset selling is going to take place is chosen (in this case battery) 3. Order date and amount and validity duration is entered 4. Selling amount option is selected 5. System must allow more than one buying order associated with selling order 6. Buyer can select amount of purchase option or buying as a bulk in the same order
Required Fields	● Selling order amount ● Order validity period ● Date of record entry ● price ● Prosumer identity
Potential Risks and Challenges	● Panel failure ● Connection failure ● Smart meter problems ● Grid maintenance

Usage scenario	Settlement
Description	Settlement based on monthly production, sell and consumption
Actors	● Consumer ● Prosumer ● Market Operator ● DSO – Distribution System Operator
Assumptions	Actors must be registered and identified in the marketplace
Steps	Settlement takes place: For consumers; Amount of transaction * commission rate platform provider Amount of energy consumed * commission rate marketplace operator and DSO
Required Fields	None, backend

3. Specific Platform Needs, Architectural Decisions, and Alternatives

In this section, first, specific platform needs must be described. Then the architectural decisions and remarks can be discussed. The technology stack such as the runtime environment and the programming language will be described to understand the technical requirements of the I-DELTA platform.

Hyperledger Fabric

- 1.4 or higher

To run Hyperledger Composer and Hyperledger Fabric, at least 4Gb of memory recommended (single organization and single peer)

The following are prerequisites for installing the required development tools:

- Operating Systems: Ubuntu Linux 16.04 / 18LTS (both 64-bit), or Mac OS 10.12
- Docker Engine: Version 17.03 or higher
- Docker-Compose: Version 1.8 or higher
- Node: 8.9 or higher (note version 9 is not supported)
- npm: v5.x

IBM Cloud Kubernetes Cluster

At least two organization and two peer for each organization, IBM Cloud Kubernetes Cluster requirements are **4 VCPUs 16GB RAM, 3 worker nodes with 3 zones.**

IBM Blockchain Platform

Code Editor

- VSCode

Off Chain Data Storage

Postgre

4. Other Comments and References to State of the Art

A smart grid serves several purposes and the movement from traditional electric grids to smart grids is driven by multiple factors, including the deregulation of the energy market, evolutions in metering, changes on the level of electricity production, decentralization (distributed energy), the advent of the involved 'prosumer', changing regulations, the rise of microgeneration and (isolated) microgrids, renewable energy mandates with more energy sources and new points where and purposes for which electricity is needed (e.g. electrical vehicle charging points). However, solutions for Smart Grids have also raised various security problems which will be a growing concern. Physical attacks, cyber-attacks or natural disasters are major notable forms of threats to smart grid deployment which could lead to infrastructural failure, customer privacy breach, blackouts, energy theft and/ or loss.

Microgrids are comprised of a diverse set of technologies and resources

In addition, as technological advancements come to life, renewable energy resources have become possible, such as rooftop solar panels, small biogas plants etc. Thus, the traditional energy consumers are deeply involved in the energy system and market – they become energy prosumers (i.e., producers and consumers). Furthermore, the prosumers want to know and trace their energy generation and consumption in detail. Decentralized provision and consumption of energy will revolutionize the energy system and market. Hence, the energy system of the future will be decentralized and based on renewable energies. This puts forward new challenges in stabilizing the energy transmission and distribution system and satisfying the needs of users.

Real-time control and supervision play an important role in the smart energy grids management and operation at medium and low voltage levels. Lately, due to the rapid growth in the deployment of Distributed Energy Prosumers (DEPs) the smart grid management problems can no longer be efficiently addressed using centralized approaches, thus, the need for visionary decentralized approaches and architectures is widely recognized. The development of Internet of Things (IoT) smart metering devices together with the prospect of renewable energy integration has increased the level of adoption of decentralized energy networks where, due to the lack of grid-scale energy storage capacity, electrical energy must be used as it is generated. However, the integration of renewable energy has added a level of uncertainty due to the intermittent and unpredictable nature of its generation. Variations in energy production, either surplus or deficit, may threaten the security of energy supply, leading to energy components overload and culminating with power outages or service disruptions. A better approach is the demand side management aiming at matching the energy demand with the production by motivating DEPs to shed or shift their energy demand to deal with peak load periods. (Blockchain Based Decentralized Management of Demand Response Programs in Smart Energy Grids: Claudia Pop, Tudor Cioara, Marcel Antal, Ionut Anghel, Ioan Salomie and Massimo Bertoncini)

As more and more energies are produced by households or small, private companies, both the energy distribution networks and the big, central energy producing plants are to be affected. How to maintain and optimize the grid stability is a big challenge to grid operators. Since customers are deeply involved in the energy system, new mechanism and models needs to be introduced to operate the grid in order to reduce the energy waste. New demands from

energy consumers will emerge. For example, consumers may also want to use the green energy and be sure what energy they are using. How to keep the energy system transparent to the customers is of great importance. Blockchain is a distributed ledger technology, providing distributed trust, anonymity, data integrity and availability. Blockchain technology can benefit the energy system in two aspects fundamentally. One is it is distributed system in nature. The other is Blockchain has intrinsic security mechanisms by design. Hence, Blockchain poses at this point as a promising technology for meeting the requirements of future energy system, and a solution to state-of-the-art problems of modern smart grids which cannot be solved by existing mature technology solutions. (A Blockchain-based Architecture for Stable and Trustworthy Smart Grid, August 2019, Yuhong Li, Rahim Rahmani, Nicolas Fouassier, Peik Stenlund, Kun Ouyang)

Towards resilient micro grid, we must fill several technology gaps with national exploitations by showcasing blockchain enabled peer to peer electricity trading mechanism. The conventional architecture of electricity markets is hierarchical, dependent on the centralized generation, inflexible and with a limited pool of bidders with a few sellers and buyers, and restricted scalability. This structure is not efficient by means of restricted demand response management options because of centralized control algorithms and less adaptability to distributed generation integration. There are still risk factors to overcome until a flexible market is fully operational:

- It is a new territory in all aspects without any legal or compliance precedents to follow, which poses a serious problem for IOT manufacturers and services providers.
- Lack of grid-scale energy storage capacity
- Integration of renewable energy adds a level of uncertainty due to the intermittent and unpredictable nature of its generation
- Variations in energy production, either surplus or deficit, may threaten the security of energy supply leading to energy components overload and resulting power outages or service disruptions.
- Large gap between IOT data transfer speeds and blockchain processing times

Review of Existing Peer-to-Peer Energy Trading Projects in Europe

There are already several projects and trails on P2P energy trading carried out worldwide shortcoming issues (GAP analysis):

Product/ Project	Origin	Begin	Objectives	Network Size	P2P Layers	Outcomes	Shortcoming	Research Focus
Piclo	UK	2014	P2P energy trading platform from suppliers perspective	National	Business	A P2P energy trading platform	Ignore introducing this model to smaller scale local energy markets	Match generation and consumption, visual analytics
Vandebron	NL	2014	P2P energy trading platform from suppliers perspective	National	Business	A P2P energy trading platform	No discussion on control system	Link consumer and generator, balance market
PeerEnergyCloud	DE	2012	Cloud-based P2P Energy Trading Platform, Smart Home	Microgrids	Energy Network, ICT	Cloud-based platform for smart homes	No discussion on control system	focused on the ICT technologies suitable for local P2P energy markets
Smart Watts	DE	2011	Optimizing energy supply via ICT	Regional	Energy Network, ICT	A smart meter gateway as interface to Internet of energy	No discussion on control system	focused on the ICT technologies suitable for local P2P energy markets
Yeloha, Mosaic	US	2015	Solar sharing network for lower energy bills.	Regional	Business	Terminated due to funding issues	No discussion on local markets	Interested in solar power
SonnenCommunity	DE	2015	P2P energy trading with storage system	National	Energy Network, Business	A P2P energy trading platform (online)	No discussion on local markets	storage of the surplus energy in a virtual energy pool
Lichtblick Swarm Energy	DE	2010	IT platform for energy markets and customers	National	Energy Network, ICT	Plenty of services provided by the energy supplier	No discussion on local markets	Platform combines products and services for both residential and business customers
TransActive Grid > Exergy (LO3Energy)	US	2015	P2P energy trading within Microgrids using Blockchain	Grid-connected Microgrids	Energy Network, Control, ICT, Business	Automatic energy trading platform within Microgrids	Communication before exchange was ignored	Ethereum blockchain, local P2P energy market in Microgrids
Electron	UK	2016	Energy metering and billing platform using Blockchain	N/A	Energy Network, ICT, Business	Not started yet	Not started yet	Ethereum blockchain, advanced billing platform

Piclo: The meter data, generator pricing and consumer preference information are used to match electricity demand and supply every half hour. Generators have control and visibility over who buys electricity from them. Consumers can select and prioritize from which generators to buy electricity. Piclo matches generation and consumption according to preferences and locality, providing customers with data visualizations and analytics. Good Energy provides contracts, meter data, billing, award-winning customer service, and balances the marketplace.

Vandebron: Vandebron is an online platform in Netherland where energy consumers can buy electricity directly from independent producers, such as farmers with wind turbines in their fields. Very similarly toPiclo, it acts as an energy supplier who links consumers and generators and balances the whole market.

PeerEnergyCloud: It developed cloud-based technologies for a local electronic trading platform for dealing with local excessive production. It was established in order to investigate innovative recording and forecasting procedures for device specific electricity consumption, to establish a virtual marketplace for power trading and to develop value added services within a Microgrid.

Smart Watts: It proposes new approaches for optimizing energy supply using modern information and communication technologies (ICT), and these ICTs are developed and tested. It exploits the optimization potential of ICT to achieve greater cost-effectiveness and security of supply.

Yeloha and Masaic: They allow interested consumers, such as apartment owners and others who do not own solar systems, to pay for a portion of the solar energy generated by the host's solar system. The subscribers get a reduction on their utility bills, so that in total, they save money, even if they move. They are like Piclo and Vandebron, but more interested in solar power than other renewables.

sonnenCommunity is developed by sonnenBatterie, which is a storage manufacturer in Germany. It is a community of sonnenBatterie owners who can share self-produced energy with others. As a result, there is no need for a conventional energy supplier anymore. With a sonnenBatterie and a photovoltaic system, members can completely cover their own energy needs on sunny days – often even generating a surplus. This surplus is not fed into the conventional power grid, but into a virtual energy pool that serves other members in times when they cannot produce enough energy due to bad weather. A central software links up

and monitors all sonnenCommunity members - while balancing energy supply and demand. This idea is very similar to Piclo's and Vandebron's, but sonnenCommunity obviously highlights the importance of storage system.

Swarm Energy is a set of services provided by energy supplier Lichtblick. Swarm Conductor, which is one part of Swarm Energy services, is a unique IT platform in the energy market. On the platform, the processes of an increasingly complex world of energy to customer-friendly products and services for residential and business customers are combined. Customers' local power plants and storage are optimized. Swarm Energy allows a meaningful interaction of distributed and renewable energy sources.

TransActive Grid is a community energy market, and a combination of software and hardware that enables members to buy and sell energy from each other securely and automatically, using smart contracts and the blockchain. The current prototype uses the Ethereum blockchain. Located in Brooklyn, New York City, consumers can choose where to buy renewables from. Home energy producers can sell their surplus to their neighbors, and communities can keep energy resources local, reducing dissipation and increasing micro and macro grid efficiency.

Electron is a revolutionary new platform for gas and electricity metering and billing systems, which is still under development. It will open the way for exciting and innovative consumer energy services. It is a completely secure, transparent, decentralized platform that runs on a blockchain and provides a provably honest metering, billing and switching service using Smart Contracts and the power of Distributed Consensus. The platform will be open source and operate for the benefit of all users. It will not be owned or controlled by suppliers or brokers.

Finally, both **Exergy** and **Electron** introduced the blockchain technology into energy sector to simplify the metering and billing system in the energy markets. However, TransActive Grid is more interested in developing a local P2P energy market in Microgrids, while Electron is targeted only at an advanced billing platform for energy suppliers.

In **TenneT's** pilot project with sonnen eServices, a group of residential batteries has been made available to help balance wind energy intermittency during periods of network congestion, when other generators may not be able to contribute to balancing. A blockchain-based interface will enable TenneT to view the status of flexible resources, to dispatch resources, and to maintain a record of the batteries' contributions to grid balancing.

I-DELTA Use-Case Design: Actors, Actions, Technologies, and Protocols

Use-Case Name: P2P Marketplace

Use-Case Provider: T2

Editor: Kamer Kaya, Öge Bozyiğit

Sub-Document properties

Distribution	Confidential
Version	0.1
Editor	Kamer Kaya, Öge Bozyiğit
Authors/ Contributors	T2, Erste Software, Dakik Software
Pages	13

1. Introduction

1.1. Purpose and Audience of the Document

The purpose of the document is to dissect the loyalty I-DELTA use case to its functional primitives and provide the details of each primitive via a sequence of intra- and cross-chain operations. The aim of this document is to provide a detailed design view for the implementers of the loyalty use case of the I-DELTA system. It will also provide insight regarding the technical operations of the use case to those in the I-DELTA ecosystem who aim to integrate their own use case implementations with the loyalty scenario.

1.2. Document Structure

Chapter 2 introduces the scope of the use-case, defines its actors and actions, and provides technical details. Chapter 3 presents a detailed view of each action of the use case. Chapter 4 is reserved for providing alternatives for the design and technologies used for the use case.

1.3. Terms, Definitions and Abbreviations

Table 1: Actors/Terms and definitions

Term	Definition
DID Admin	Responsible for carrying out administrative tasks related to DID management (e.g.: DID creation, revocation, sharing, setting/revoking credentials)
System Admin	Responsible for authorizing the integration of company DLTs with the whole system as well as other system-wide administrative tasks (e.g.: the ones related to the message bus connecting blockchain networks).
Company Admin	Responsible for creating benefits, adding/removing company employees, and transferring token to employees/users
Employees	Earn/spend loyalty tokens by buying benefits via the platform.
Loyalty Tokens	Digital currency provided by companies to their employees to be used to claim benefits on the platform
Message-bus	Software platform that allows exchange or protocols and communication between one (or more) block chain networks

Message-bus Adapter	A software module that connects the company DLT with the system.
Hyperledger Fabric	Permissioned blockchain platform that offers protocols, smart contracts, standards and other key blockchain elements. Each company DLT is a Fabric instance.
Hyperledger Indy	A permissioned blockchain platform that offers protocols and smart contracts to manage decentralized IDs. The DID is ledger is an instance of Indy.
Hyperledger Aries	Infrastructure for peer-to-peer network and blockchain-rooted interactions within platforms that offer key management and secret management systems.
Smart Contract	An agreement among the users running on a DLT in the form of computer code.
RabbitMQ	RabbitMQ is an open source message broker software. It accepts messages from producers, and delivers them to consumers
Employee dApp	A web/mobile application employees use to check their balances, buy, transfer and swap benefits.

Table 2: Abbreviations

Term	Definition
DLT	Distributed Ledger Technology
PoC	Proof of Concepts
SDK	Software Development Kit
DID	Decentralized Identifier
UUID	Universally Unique Identifier
SC	Smart Contract
RMQ	RabbitMQ

DA	DID Admin
SA	System Admin
CA	Company Admin
LT	Loyalty Token
MB	Message Bus

2. Scope

2.1. Business Context

In the current state, companies give extra benefits to their employees in addition to their regular salary. These benefits create tax advantages and increase the loyalty of employees. On the other hand, some of the benefits given to employees are not utilized and are wasted. Companies want to maximize the effectiveness of the benefits of their employees without wasting unnecessary funds.

The aim of the Loyalty business scenario is to provide a marketplace that lives on the blockchain and allows users to purchase loyalty benefits from a benefit pool driven by more than one company. The companies themselves may have their own blockchain to manage their employee identities and their company currency, and thus it is the goal of I-DELTA to provide the technical infrastructure to manage and operate an interoperable cross-chain loyalty benefit pool across multiple companies. Registering companies and employees, creating and transferring benefits, managing wallets, and synchronization between the blockchains will be conducted through this technical infrastructure.

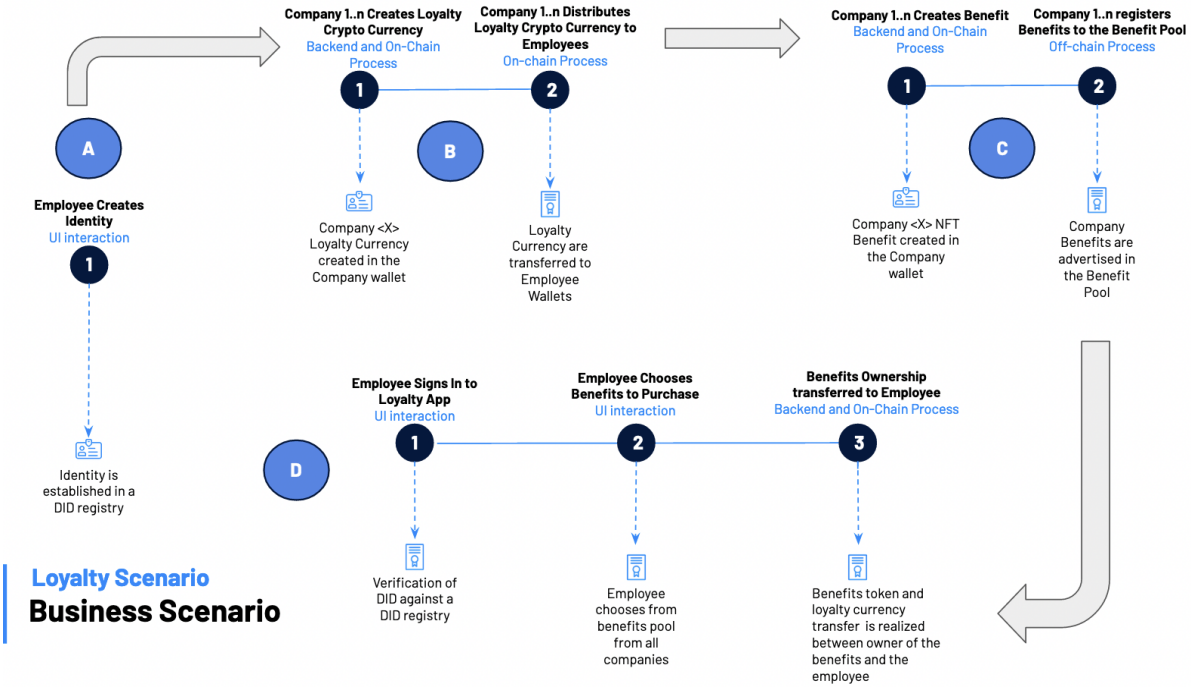


Figure 1

Using I-DELTA, companies will aim to maximize the purchase of their own company-generated benefits while also increasing the satisfaction of their employees by giving them access to a much larger benefit pool with a wide selection of benefits.

2.2. Technical Infrastructure

The loyalty use-case employs blockchain frame-works launched by the Linux Foundation. Hyperledger Fabric, Indy and Aries are the principal pillars in our infrastructure that provide the necessary functionality for our use case. Each technological framework will be briefly described individually and how they compliment each other and enrich the use-case scenario.

Hyperledger Fabric:

Hyperledger fabric is an open source, modular blockchain framework developed by the linux foundation and is a well established enterprise grade blockchain platform. It provides a permissioned, modular architecture which allows for a “plug and play” functionality which allows for easy reuse of existing features and ready-made integration of various modules. The modular architecture of Hyperledger Fabric separates the transaction processing workflow into three different stages: smart contracts called chaincode that comprise the distributed logic processing and agreement of the system, transaction ordering, and transaction validation and commitment.

Hyperledger Indy:

Hyperledger Indy provides tools, libraries, and reusable components for providing digital identities rooted on blockchains or other distributed ledgers so that they are interoperable across administrative domains, applications, and any other silo. Indy is interoperable with other blockchains or can be used standalone powering the decentralization of identity.

Hyperledger Aries:

Hyperledger Aries provides a shared, reusable, interoperable tool kit designed for initiatives and solutions focused on creating, transmitting and storing verifiable digital credentials. It is infrastructure for blockchain-rooted, peer-to-peer interactions. This project consumes the cryptographic support provided by Hyperledger Ursa, to provide secure secret management and decentralized key management functionality.

How it comes together:

Hyperledger indy is used as the universal ledger for storing any and all benefits created by the companies that are part of the platform. One of the key challenges pertaining to this use case is to allow employees of all companies to partake in the purchase of all benefits regardless of the company that has created it. Although it can be achieved using a fabric network directly, complexity arises when atomicity must be guaranteed in the system. A complex protocol which can allow fabric networks of different companies to call their smart contracts must be generated which guarantees i) atomicity ii) privacy. Each company has their own fabric network and would not be willing to allow any access that may compromise their private intellectual properties. Thus to overcome this, a central service becomes a necessity.

However, centralization goes against the very nature of blockchain technology and so by utilizing hyperledger indy we are able to globalize the market place and significantly reduce the complexity of any transactions between employees of different companies. By utilizing indy and providing each benefit its own DID, we are able to track the benefit at all times and significantly reduce the complexity in providing protocols that ensure atomicity.

All communication with the indy ledger is done via aries agents that provide endpoints for interacting with the ledger and allows for storing of verifiable credentials that provide proof of ownership of benefits. Furthermore, aries support multi-tenancy which allows for the company to provide and administer sub-wallets that can be created and/or removed with significant ease. This allows our solution to be scalable due to the fact that we will not be managing multiple wallets on a per-employee level but rather a single super-wallet (belonging to the company).

The technical infrastructure is summarized in the following diagram:

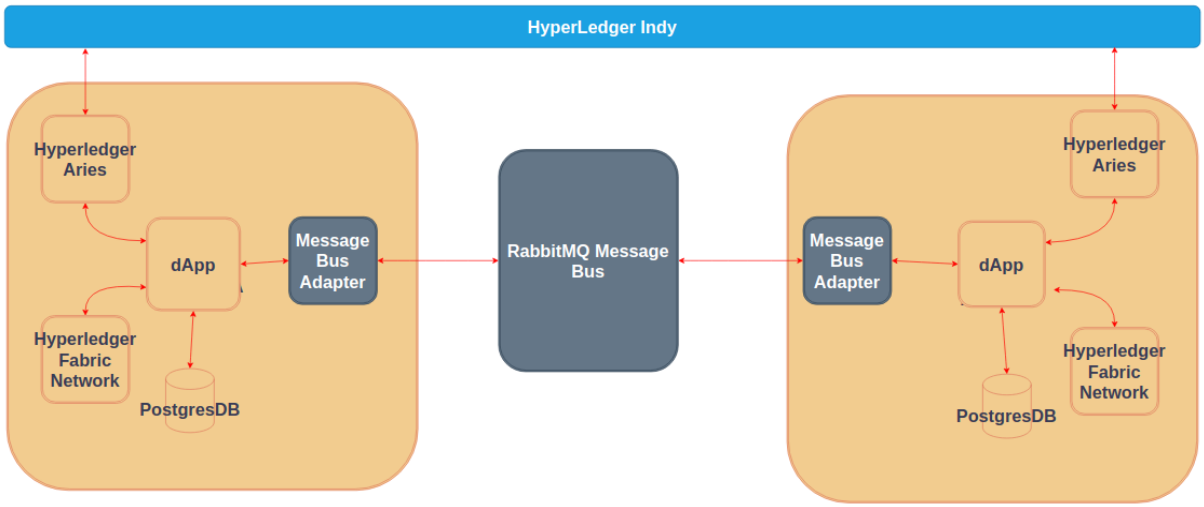


Figure 2

Use-Case Scenarios:

Buy Benefit

PreReq: Company and Employee are Registered

Scenario:

- Employee A (from Company A) wants to buy some Benefit A.

Steps:

1) Employee A browses the market-place and chooses to buy some benefit A.
    a) Benefits can either belong to the company of the employee, another company or another employee from the same or different company.

2) System check to ensure that employee A (buyer) has enough tokens to partake in the transaction
    a) API call to fabric network that checks and compares if the buyer (employee A) can perform the transaction.

3) API call returns the appropriate message regarding feasibility of the transaction.

4) Assuming employee A has enough tokens to buy the benefit, they initiate the transaction by starting the buying process.
    a) Schema and credential definition ids are needed to ensure the buyer is able to provide the appropriate attributes needed for change of ownership.

5) Holder (Employee A) connects to the issuer

6) The issuer provides connects to indy to update ownership status of the benefit

7) The indy ledger return a response indicating whether or not the ownership status has changed and the benefit DiD

8) The Issuer returns a verifiable credential to the holder

9) The Holder can see the their issued VCs on the dApp

10) The token balance of the issuer is updated to reflect the changes.

11) User returned a response indicating a successful transaction had occurred.

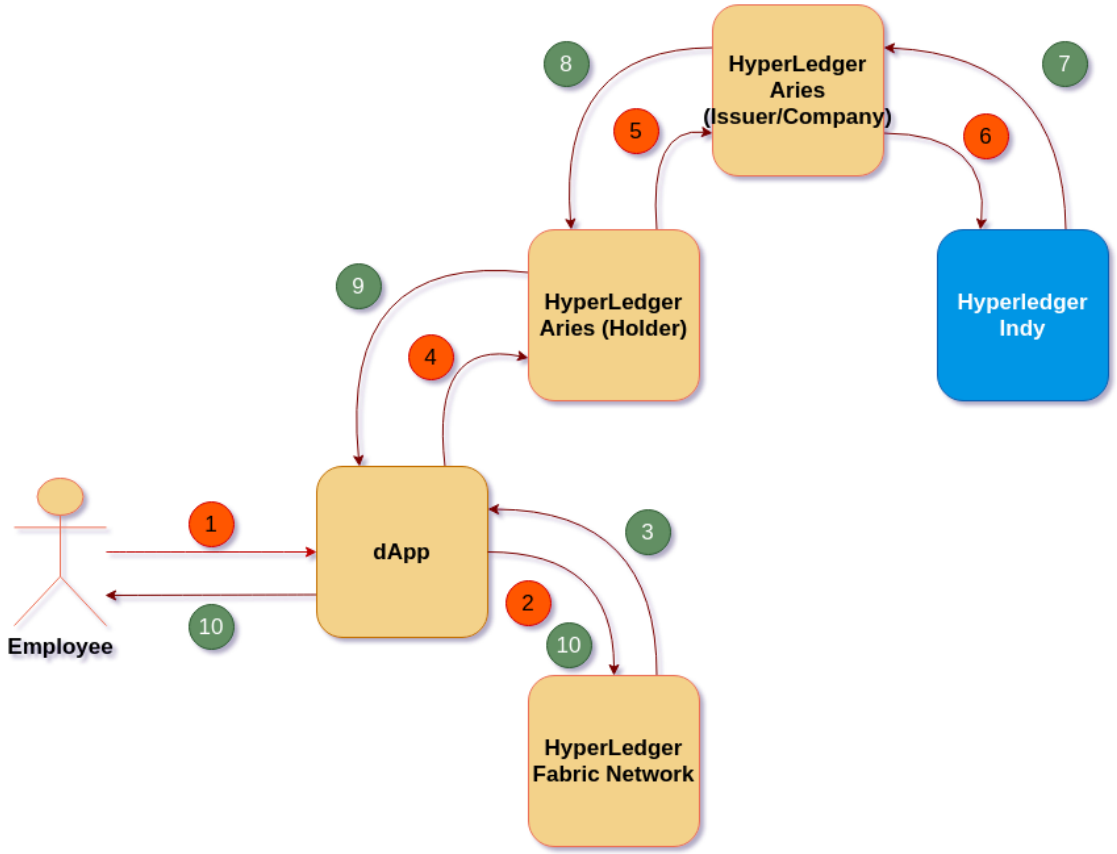

HyperLedger Aries (Issuer/Company)
HyperLedger Aries (Holder)
Hyperledger Indy
dApp
Employee
HyperLedger Fabric Network
1
2
3
4
5
6
7
8
9
10
10

Figure 3

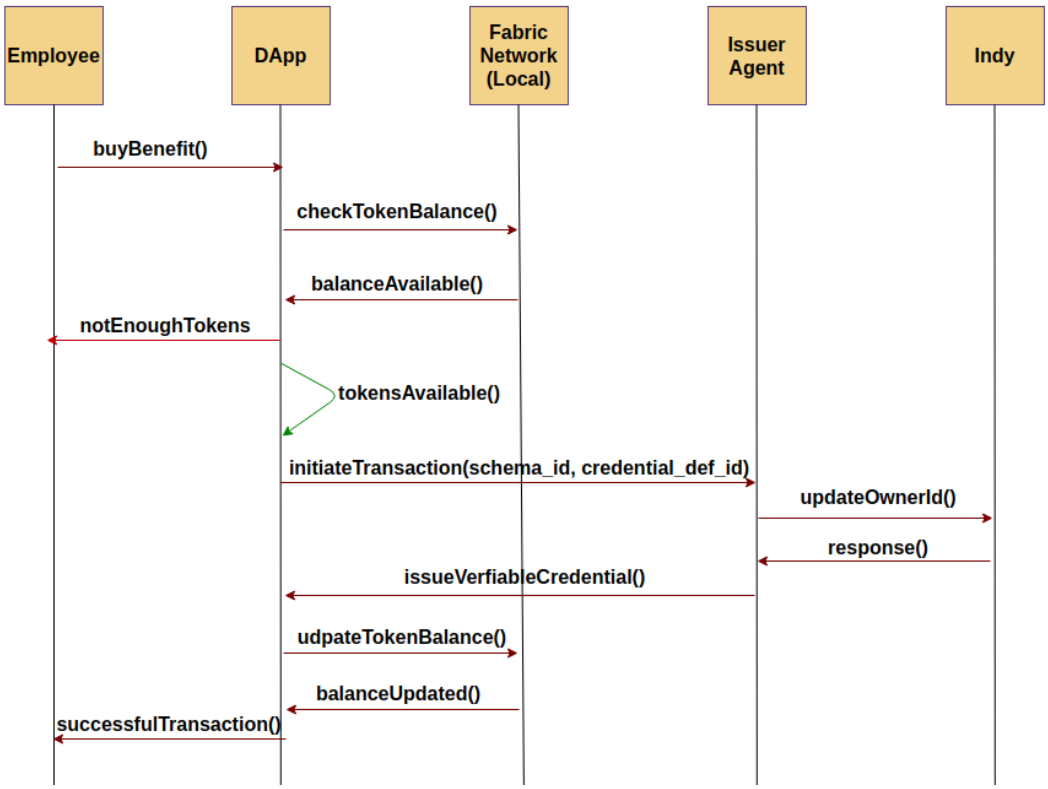


Figure 4

In an ideal scenario, the employee becomes the owner of the benefit they have purchased and their token balance is updated or incase of failure, the entire transaction is canceled and the original state of the actors involved is restored; meaning the token balance (if deducted) is restored and the benefit is restored on the marketplace. In the loyalty use-case scenario, there can be an event where two employees belonging to the same or different companies may try to purchase the same benefit simultaneously. In such an event, it is entirely possible that one of the parties involved may end-up worse off. A possible worst-case scenario is that one employee becomes the owner of the benefit but does not have their token balance reduced while the other has their token balance reduced but is not given ownership of the benefit. In this scenario the employee is worse off than they were before initiating the transaction.

To combat this grievance, the protocol must support conditions where multiple users are attempting to purchase the same benefit and ensure that one and only one user is given ownership and that only their token balance is updated. Hyperledger Indy does not support smart contracts that can be called upon to revert changes hence the protocol developed must allow for the system to reverse into its original state in the event of a failed transaction.

Create Token

PreReq: Company is Registered

Scenario:

- Company A creates token in Company blockchain

Steps:

1) Company Administrator triggers create token action to create tokens in Company wallet. (Figure 5)

Transfer Token (within Company)

PreReq: Company has created tokens in Company Wallet

Scenario:

- Company A transfers tokens to employees in Company A

Steps:

1) Company Administrator triggers transfer token action to transfer tokens from Company wallet to wallet of $Employee_1$ - $Employee_N$. (Figure 5)

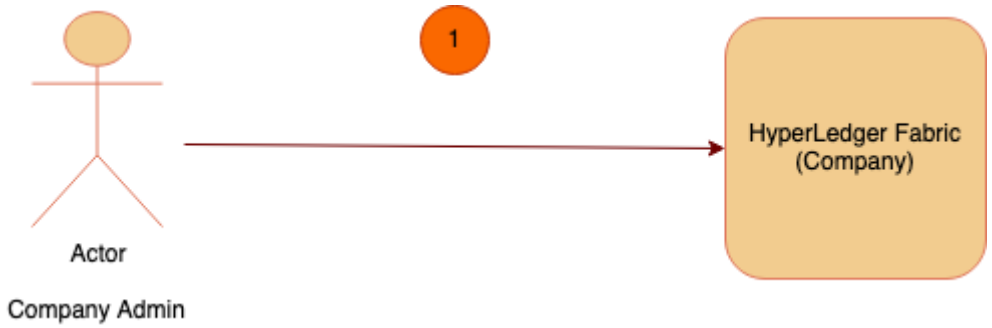


Figure 5

Create Benefit

PreReq: Company is Registered

Scenario:

- Company A creates a Benefit and stores it in the Indy ledger

Steps:

1) Company Administrator creates a benefit with its identifier and other attributes in the company wallet on the Indy ledger (Figure 6)

Update Benefit

PreReq: Company is Registered

Scenario:

- Company A owns some Benefit A and wants to update it.

Steps:

2) Company Administrator updates the attributes of the benefit stored on the indy ledger (Figure 6)

Remove Benefit

PreReq: Company is Registered

Scenario:

- Company A owns some Benefit A and wants to remove it.

Steps:

1) Company Administrator removes the benefit stored on the indy ledger (Figure 6)

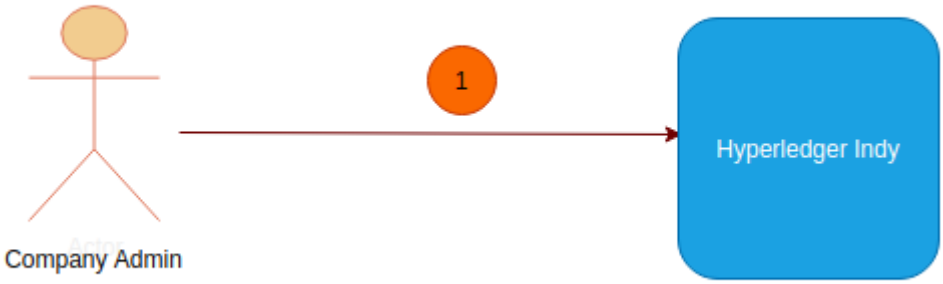


Figure 6

Register Company:

Scenario: A new company must be added to the idelta platform
Prerequisite: The company has an existing blockchain that can be integrated/connected with the message bus

Steps:

1) The Company Administrator provides the DID administrator with custom controller that contains the business logic the company will follow when interacting with the indy ledger via an aries agent

2) DID administrator launches an Aries agent with credentials provided by the company (Port Mapping and API Endpoints) and the controller which connects to indy
    a) DID administrator has access to the genesis file provided by indy that is used by the aries agent to connect and call endpoints on the indy ledger.

3) Once the aries agent has been set, the company administrator sets up wallets for each employee that will act as their DID and verifiable credential storage.

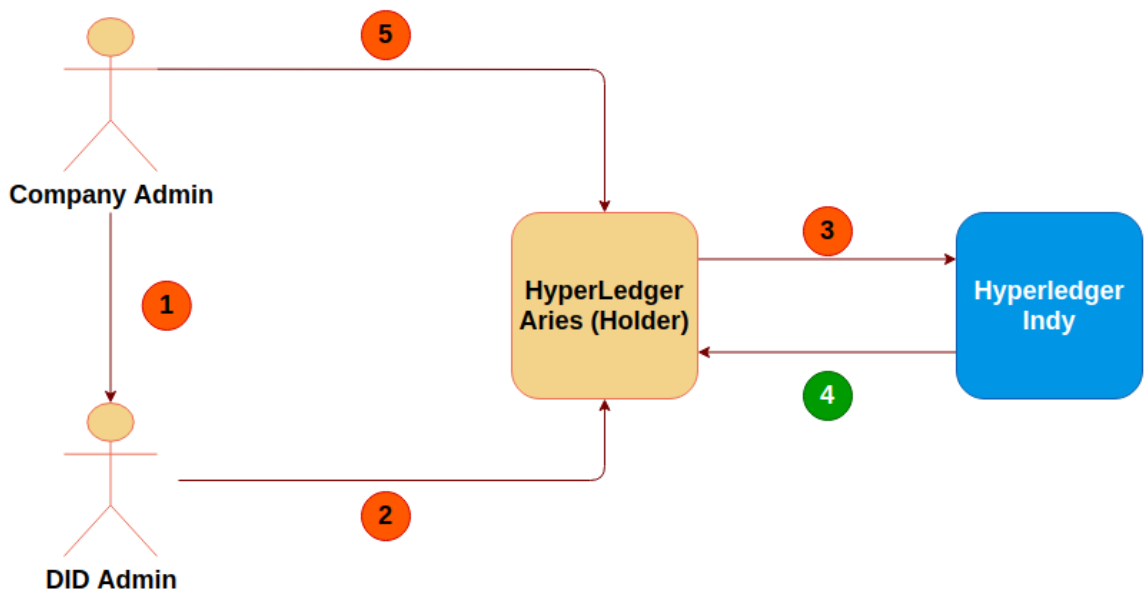


Figure 7

Remove Company:

Scenario: A company must be removed from the idelta platform

Steps:

1) The company administrator removes all corresponding VCs and wallets of employee agents stored on indy by calling the appropriate endpoint.

a) The company administrator calls the revocation registry and updates the tails files to remove all issued VCs for all employees.

2) The company administrator removes all corresponding benefits offered by said company from the marketplace and the ledger
   a) The company administrator calls the revocation registry and updates the tails files to remove all issued VCs for all employees.

3) The DID administrator removes the company agent (holder)

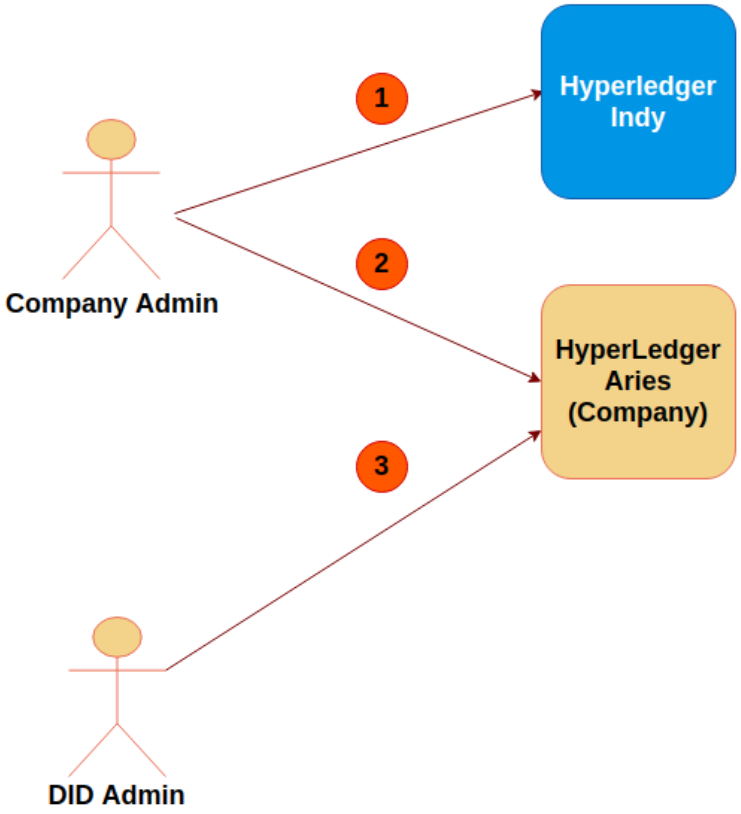


Figure 8

Add Employee

PreReq: Company is registered on the DID blockchain

Scenario:

- Company A wants to add an employee to the system

Steps:

1) The company administrator, through the agent, sets up a wallet for the employee that will act as their DID and verifiable credential storage (Figure 9).

Remove Employee

PreReq: Employee is registered on the blockchain

Scenario:

- Company A wants to remove an employee from the system

Steps:

1) The company administrator calls the revocation registry for the specific employee to be removed (Figure 9)

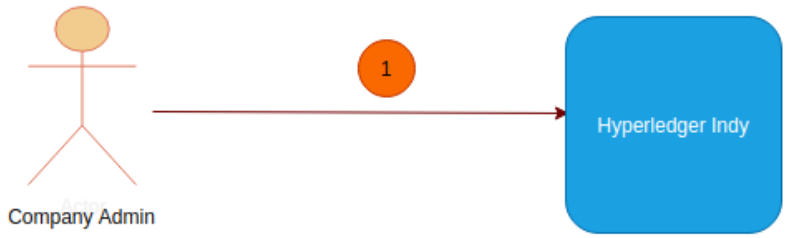


Figure 9