



Deliverable D 3.1

B4CM deployed proof-of-concept: Technical report detailing the software produced, how it is applied in an industrial context, and presenting a case study of its use as a demonstration of applicability of the framework within the European rail sector

Project acronym:	B4CM
Starting date:	01/12/2018
Duration (in months):	48
Call (part) identifier:	H2020-S2RJU-OC-2018
Grant agreement no:	826156
Due date of deliverable:	Month 45
Actual submission date:	16/11/2023
Responsible/Author:	Rahma Alzahrani (UoB)
Dissemination level:	Public
Status:	Issued

Reviewed: Yes



This project has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No 826156.

Document history		
Revision	Date	Description
1	15 th October 2023	Initial release
2	16 th November 2023	<p>Updated release in line with reviewer’s comments, changes include:</p> <ul style="list-style-type: none"> <input type="checkbox"/> Page 8 – clarified description of the functionality of the smart gateway. <input type="checkbox"/> Page 9 – generalisation of implementation-specific blockchain terminology in steps 1 and 2 of the workflow. <input type="checkbox"/> Page 10 – expanded commentary on the relationship between the records on the chain and the fulfilment / verification of SLAs established in the data sharing agreement. <input type="checkbox"/> Page 12 – clarifications around data frequency and distribution of costs. <input type="checkbox"/> Pages 16-17 – figures 7 and 8 enlarged <input type="checkbox"/> Pages 17-18 – additional figure and clarification on data streamed by the sensor simulator / appropriate formats during the walkthrough added to section 7.3 <input type="checkbox"/> Page 38 – comment acknowledging the potential of EVM based solutions for increased portability added to the conclusions alongside other recommendations in this area.

Report contributors		
Name	Beneficiary Short Name	Details of contribution
Rahma Alzahrani	UoB	Lead author. Development of reported framework, use cases, and media (screenshots, recordings etc.)
John Easton	UoB	Contributing author.
Simon Herko	IB	Review

Disclaimer

The information in this document is provided “as is”, and no guarantee or warranty is given that the information is fit for any particular purpose. The content of this document reflects only the author’s view – the Joint Undertaking is not responsible for any use that may be made of the information it contains. The users use the information at their sole risk and liability.

The content of this publication does not reflect the official opinion of the Europe’s Rail Joint Undertaking (EU-Rail JU). Responsibility for the information and views expressed in the paper lies entirely with the author(s).



Table of Contents

1	Executive Summary	1
2	Abbreviations and acronyms	2
3	Background.....	3
4	Objective/Aim.....	4
5	Use Cases.....	5
5.1	Switch and Point Machine Monitoring System (Infrastructure Monitoring Train)	5
5.2	Railway Track Monitoring Using Onboard Inertial Measurements (Train Monitoring Infrastructure).....	6
6	Workflow in Deployed Software	8
6.1	Additional Deployment Considerations	11
7	Deployment Context / Environment	13
7.1	Setup of Development Environment.....	13
7.2	Sequence of Data Exchanges.....	15
7.3	IoT sensor simulator	15
8	Deployed Proof of Concept	19
8.1	System Architecture	19
8.1.1	Transactional Flow	21
8.1.2	Claims Management, Escrow and the Distribution of Costs	27
8.1.3	Payments.....	28
8.2	Mapping of Roles to Entities in the Framework.....	30
8.3	Walkthrough of Core Functionality of Proof of Concept.....	31
8.3.1	Administration.....	31
8.3.2	Journeys	32
8.3.3	Data Offers	32
8.3.4	Escrow	35
9	Conclusions.....	38
10	References.....	39

1 Executive Summary

The pursuit of higher quality services in the railway sector is a continuous process, and the availability in recent years of affordable, reliable, digitally enabled additions to traditionally mechanical-based infrastructure systems has provided a fruitful avenue for advancement. Remote Condition Monitoring (RCM) systems are one example of a tool that has been widely deployed to improve the standards of maintenance, reliability, and safety across the rail network. Such systems offer particular benefits at the traditional boundaries of responsibility within the industry (e.g. the interface between the infrastructure and rolling stock) where complex physical interactions may make the cause to defects difficult to determine. Although this type of cross-interface monitoring of assets may be the most technically practical solution to many industry-wide problems, commercially they can prove complex as the business paying to install, maintain, and operate the sensing device is not the party benefitting from the data collected. As a result, it can be hard to generate business cases for the purchase, installation and operation of cross-interface monitoring systems that would have recognised industry-wide benefits.

The aim of this deliverable is to present the B4CM proof of concept demonstrator.

The document begins by reintroducing the use cases to be modelled, as these have been updated since they were first reported. It then moves on to discuss the workflow through the software as deployed, before providing details of the operational environment (dependencies etc.) in which the proof of concept runs. Finally, it provides walkthroughs of the proof of concept as both screenshots and linked videos, enabling the reader to evaluate the proof of concept for themselves.

Moving forwards, the B4CM team recommend that:

- Where future proof of concept platforms of this type are planned, they make the greatest use possible of virtual environments such as Docker containers – this should minimise the changes needed in migration across physical platforms and ease distribution.

2 Abbreviations and acronyms

Abbreviation / Acronym	Description
B4CM	Blockchains for Condition Monitoring
BCRRE	Birmingham Centre for Railway Research and Education
CSS	Cascading Style Sheet
GNSS	Global Navigation Satellite System
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IMU	Inertial Measurement Unit
IoT	Internet of Things
MEMS	Micro-electromechanical Systems
RCM	Remote Condition Monitoring
SLA	Service Level Agreement
STS	Spring Tool Suite

3 Background

Over the past decade there has been a significant level of investment throughout Europe in the digitalisation of the rail network. This includes the installation of sensors on the infrastructure and vehicles, the deployment of next generation traffic management systems that allow real-time management of the system, and the provision of mobile applications for passengers and staff. Despite the wealth of new data provided by these systems, the railways are still struggling in their aspiration to be an information-led industry due to a lack of traceability of information usage, and the commercial barriers between stakeholders.

Blockchains are a disruptive technology that have the potential to accelerate the development of rail as the primary medium-distance carrier within the wider multi-modal transportation system. Directly funded by the rail industry via the EU Shift2Rail Joint Undertaking, the B4CM project will identify key use cases for the technology within the railways, deliver a blockchain-based testbed that enables the benefits of the technology to be formally evaluated, and demonstrate the value of blockchains in the attribution of data costs across organisational boundaries within the European rail sector.

The overall aim of the B4CM project is to develop and deliver a blockchain-based testbed for the attribution of data costs across organisational boundaries, and to demonstrate the operation of the framework and in the context of the European Rail Industry, enabling future developers to extend the tools produced based on a known working configuration.

B4CM has the following research and training objectives:

Objective 1: To identify and develop use cases that support the application of blockchain in the railway sector;

Objective 2: To develop an implementable blockchain framework for the attribution of data costs in systems crossing organisational boundaries;

Objective 3: To evaluate mechanisms for the incorporation of the developed blockchain framework into the financial processes of the European rail sector;

Objective 4: To develop a testbed, demonstrating the operation of the framework in the context of rail sector, enabling future developers to extend the tools produced based on a known working configuration;

Objective 5: To disseminate the findings of the project and the lessons learned to influence best practice in innovation and technology uptake in a key and evolving field within the European rail sector;

Objective 6: To support the development of a researcher in gaining a PhD and thus generating a skilled specialist valuable to the European rail sector.

This document, reporting the B4CM proof of concept, is written primarily in response to Objective 4 of the B4CM project.



4 Objective/Aim

This document forms part of Work Package 3 of the B4CM project and has been prepared to report the deployment of the framework developed by the B4CM project team as part of a representative industry data exchange scenario. To that end, two use cases are discussed, one involving the monitoring of axle bearings using statically mounted sensors on the track, and a second that makes use of sensors mounted on-board a passenger vehicle to monitor track condition. Data drawn from both systems is used to prove the concept of managing data exchanges on this type through a distributed ledger.

Although originally devised under the Shift2Rail programme and intended to contribute to TD 4.6 of that activity, the content of this document can also be seen as a direct contribution to the objectives of FP3 (IAM4RAIL) under the current Europe's Rail programme, focussing on the holistic and integrated asset management for Europe's rail system.

5 Use Cases

Two use cases have been chosen in order to test the model against them and both are developed projects by the Birmingham Centre for Railway Research and Education (BCRRE), located at the University of Birmingham. The first use case uses multiple sensors attached to the track and used to collect data to monitor the condition of axle journal bearing remotely[1]. The second use case uses sensors attached to the train that are used to collect data whenever the train is passing on the track. Then, the collected data will be processed to monitor the track health over time[2]. Each use case will be discussed in more detail in the following subsections.

5.1 Switch and Point Machine Monitoring System (Infrastructure Monitoring Train)

Switches and crossings in the tracks have been recognized to be the main cause of points failures and motor solidity problems. Recent in-service monitoring has shown that the swing-nose crossing of points 2076 on the down line at Stratford was a subject to some of the highest impact forces on the Network Rail High-Speed managed infrastructure. Therefore, there is a need to analyse the dynamic forces established between the wheel and rail, monitoring components to predict the type and cause of failure. For this purpose, a remote condition monitoring solution has been developed that involves several sensors and measurement devices mounted around the swing-nose crossing and associated point operating equipment. The remote monitoring system has two parts. The first part is mounted at the swing-nose of 2076S crossing and the other part is at the tip as shown in Figure 1 which depicts the overall system parts. In general, 17 sensors are in use and divided as follows:

- 10 Micro-electromechanical Systems (MEMS) Accelerometer sensors are used to measure the vertical movements and vibrations of the bears across the swing-nose. There will be 2 accelerometers on the bearers to support the swing-nose point machine and 8 on the other end of the swing-nose crossing bearers, as shown in Figure 1;
- 3 Piezoelectric accelerometers, 2x to monitor the wheel-rail impacts at and 1x to record the point machine vibrations;
- Ruggedized free-field microphone to measure the sound level and extract audio signatures; and
- 3 Miniature AC current clamps to measure point machine 3 phase currents.

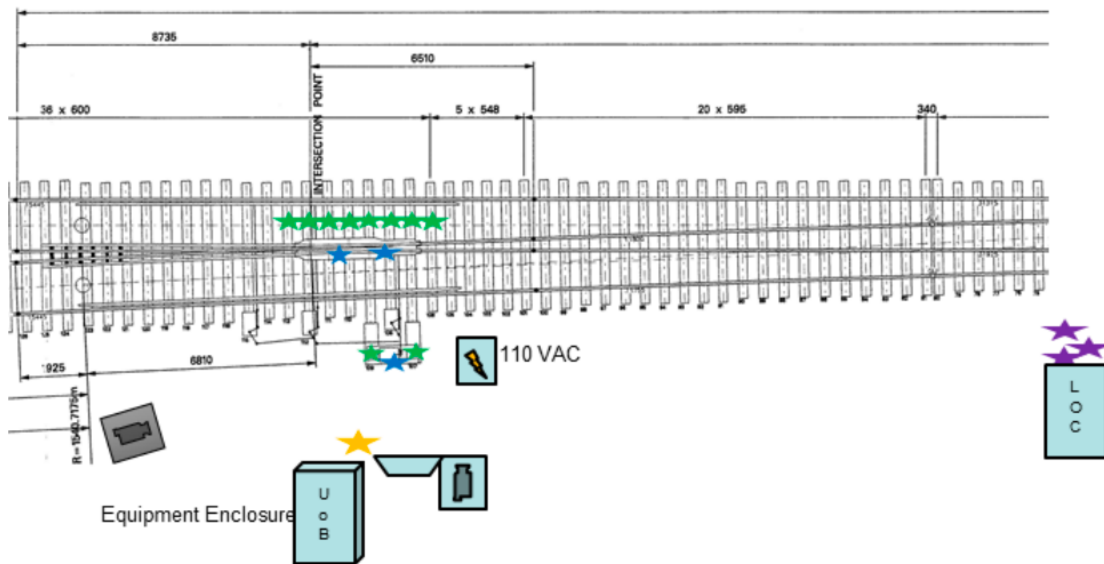


Figure 1: Schematic of monitoring system used as data provider

5.2 Railway Track Monitoring Using Onboard Inertial Measurements (Train Monitoring Infrastructure)

The track's shape may change over time from the geometry that was originally intended. Track degradation is a phenomenon that can happen for a variety of causes; e.g.

- Following the construction of the track-bed and/or as a result of the regular use of railroad vehicles on the track, ballast settlement may have changed; or
- Changing in weather conditions and extreme temperature may lead to unwanted infrastructure movement. Frost heave effect in [3] is an example of how water between the ballast particles may freeze and cause the ballast layer to expand which results in track movement.

Uneven settlement of ballast leads to support the short sections of track more than sections then creating vertical track profile with variations and decreasing the ride quality. Therefore, this vertical profile is one of the essential parameters to evaluate the track degradation. In addition, measuring distance is also important in track condition monitoring to evaluate and estimate the track health and locate faults if any. There are interesting publications that investigated how to use the IMU sensor to monitor track irregularities by fitting this sensor on the axle box and/or the bogie of several vehicles [2], [4]. A team in the University of Birmingham has proposed a track measurement system that consists of a compact IMU to be attached to the in-service vehicle. This system gathers daily measurements to be instantly ready to be compared with historical measurement recordings, enabling for quick awareness of any track degradation. This proposed system can be installed on any in-service vehicles to provide the needed measurements for large segments of railway networks and cut the costs of operating dedicated and separated measurement vehicles. The proposed system used IMU sensor which is a MEMS-

based sensor that can measure rotational velocities and accelerations at a constant rate over time through the embedded 3-axis of accelerometers and gyroscopes that form a six degree of freedom model. It receives GNSS and tachograph data when available, stores onto a local SD card, and/or transmits data to an on-board PC. This recorded inertial data from the BCRRE IMU system will be processed into meaningful track condition information. The published result of this project has shown the significance of track geometry and its measurement. It also illustrates how bogie mounted and onboard IMUs can be utilized to identify concerns with ride comfort and track geometry degradation[2]. Sensors are fitted on an in-service class 377 train in three different positions as shown in Figure 2.

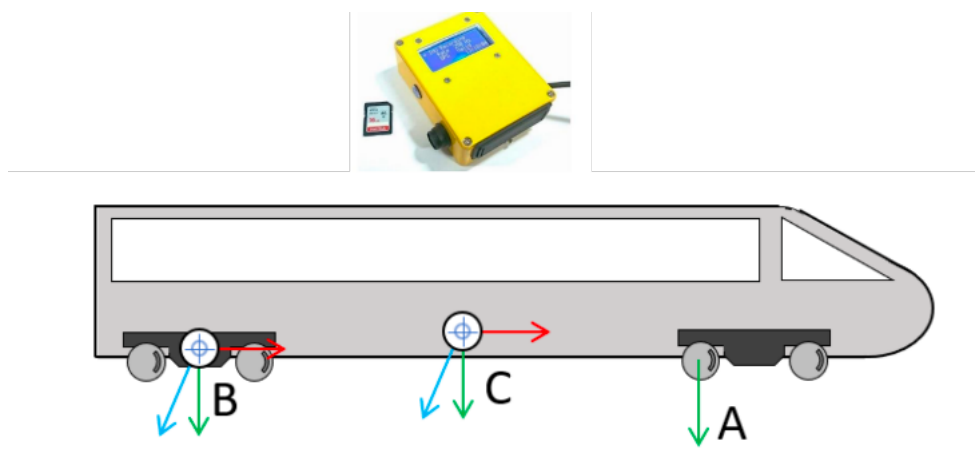


Figure 2: IMU/sensor mounting locations on vehicle

Specifically, the sensor groupings used in the deployed system are:

- Axlebox IMU (A_IMU);
- Bogie and cab GPS (B_GPS , C_GPS); and
- Bogie and Cab IMU (B_IMU , C_IMU).

6 Workflow in Deployed Software

The flow of connection between the end user and the device to obtain generated data is shown in Figure 3 which presents the main parts of the developed system. The roles of individual components / actors are as follows:

- **Users:** Users of the system consists of administrators, data customer, and device owner/ data providers. The customers and providers will trade the data generated by devices under the control of the blockchain which will work as a main coordinator and immutable log of all the data sharing processes based on the business logic that is defined in the smart contracts. The system's administrator is responsible for maintaining and monitoring the blockchain and smart gateway program.
- **Blockchain:** It is the main part of the system that runs the business logic through several installed smart contracts.
- **Smart gateway:** Typical IoT devices are energy-constrained and possess minimal onboard processing power, making them ill-suited to the rapid performance of cryptographic functions. As a result, it is inappropriate for IoT devices themselves to form peers within a blockchain network. The smart gateway acts as a bridge between the IoT devices (external to the blockchain network), and one of the blockchain peers (within the network), essentially providing a client that enables the IoT devices to communicate with the network without having to perform the intensive maths required to validate blocks.
- **IoT devices:** IoT devices generate new resources (usually asset monitoring data) to be recorded on the ledger. Whenever the device is generating a new resource, it will be sent to the blockchain via the client services exposed by the smart gateway.
- **IoT server:** many services shall be provided through the IoT server such as interacting with the smart gateway, gathering the generated data from all sensors, hashing data and appending hashes to the blockchain, storing data to the database, and processing all kind of commands to perform operations on sensors. There are many communication protocols that can be applied by the local bridge to connect devices to the server such as Bluetooth, ZigBee, WiFi, and 2G/3G/4G cellular.
- **Storage:** Two different kinds of storage will be used in the system. The first one resides in the blockchain network which is a comprehensive and immutable log of all transactions, trading operations, and data hashes. The second can be a hardware storage like a hard disk or software storage like DB and will be used to store the raw data collected by the sensors and devices.

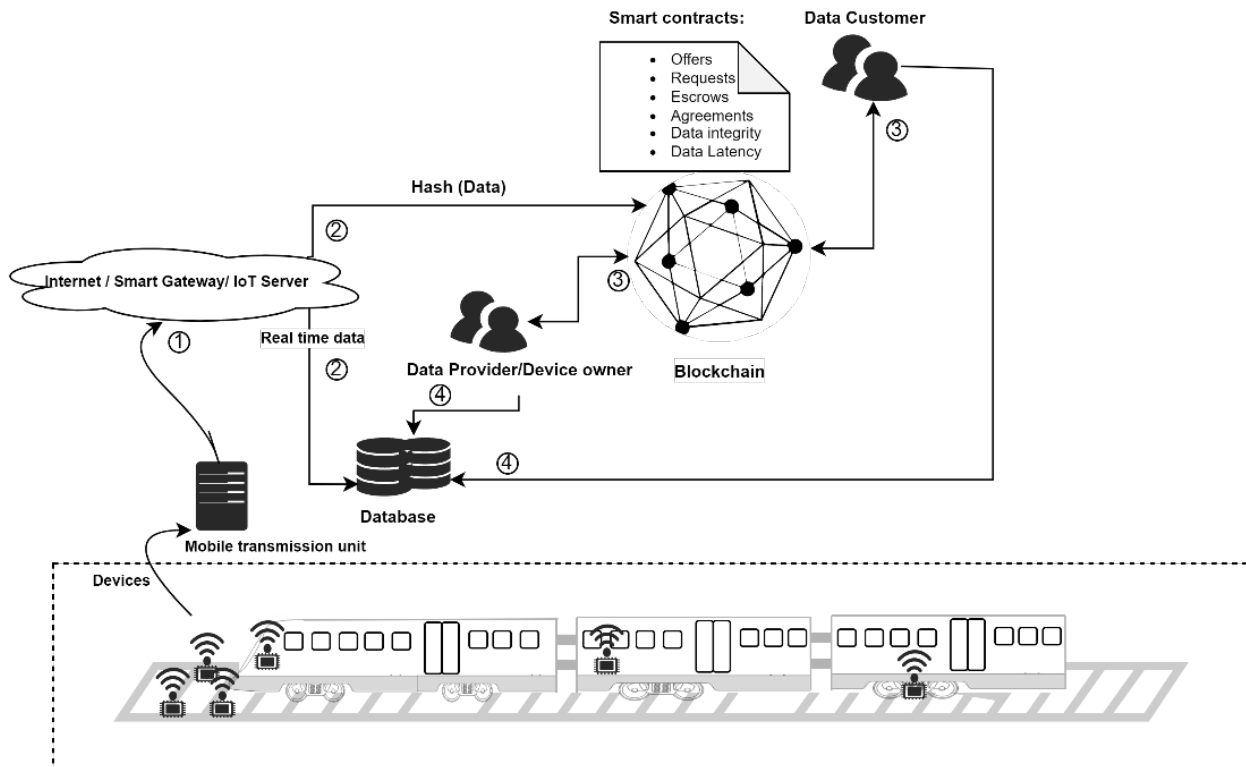


Figure 3: Conceptual view of deployed framework within wider monitoring ecosystem

As shown in Figure 4, the workflow of the whole system principally comprises several steps to achieve the ultimate goal of trading data and reaching fair cost attribution between different parties. Some modifications have been made to the initial suggested framework in [5] to explore in detail how this process should adapt and what the interfaces would be in case of real-time monitoring and direct data exchanges in IoT environment.

- **First step:** Initialization of the blockchain network and installation of the code for any smart contracts on all peers. These essential processes enable the core functionality of the framework, and are performed by nodes with administration privileges on the network. In order to ensure the network can be initialized in a smooth and repeatable way, core configuration information should be scripted (e.g. using YAML) allowing choices such as the consensus algorithm to be used, and the identities of critical peers to be persisted if required. Once the nodes are running and configured, the remaining network start-up processes can be completed, including establishing communication between groups of peers (known as channels in Hyperledger), and instantiated the smart contracts.
- **Second step:** Before users are ready to submit the transaction to the blockchain according to defined business logic in a smart contract, they are required to enrol with a certificate authority to obtain their identity, a certificate, that contains the public & private keys used to lodge transactions.

- **Third step:** Users are now ready to submit transaction proposals to the blockchain network for consuming services defined in the smart contracts. The data providers will share which sensors they are offering to give access to. The customers shall, based on available offers, send a request determining the sensors they are interested in and the duration of access. This request won't be considered until the escrow is initiated and the payment is processed.
- **Fourth step:** The payment process is initiated by sending the request as mentioned in the previous step. The consumer shall pay to the escrow account to initiate the agreement. The escrow is fully monitored by the fabric blockchain and has known accounts for all parties.
- **Fifth step:** The blockchain will notify the provider about the request and wait for the provider's response to consider and lock the escrow. The provider may take one of the following actions:
 - Processing the deposit payment and this shall be considered as an acceptance and the consumer will be acknowledged and the agreement will be activated consequently.
 - If the provider rejects the request the escrow should pay the consumer back.
 - Neglecting the request until the date of starting time is missed, this will be considered as indirect rejection.
- **Fifth step (continued):** If the provider processes his payment, the escrow will be locked and the following steps will take place. Otherwise, the escrow will be released and a payment back will be processed to the consumer leading to a communication termination between the customer and provider.
- **Sixth step:** The consumer shall gain access to the sensor according to the generated agreement and all generated data will be shared with the consumer as long as the agreement is active. The date and time are included in the agreement to avoid any reuse of the same agreement to access the sensor in the future. Once the data transmission is completed or the agreement has expired, the sensor shall stop sharing its data with the consumer while sending the raw data to the provider and the hash value to the blockchain.
- **Seventh step:** For claims processing and cost distribution, the system provides for the creation of Service Level Agreements (SLA) allowing the consumer to verify that the timely provision and integrity of the shared data conformed with the requirements / data transmission schedule that was stated in the original sharing agreement. Details of the sharing agreements themselves, including participating parties, start and end periods, the details of the data products included, and the escrow mechanisms are all lodged on the blockchain to enable this verification to take place. Full details of the SLAs supported can be found in sections 6 and 7 of B4CM project deliverable D2.1.

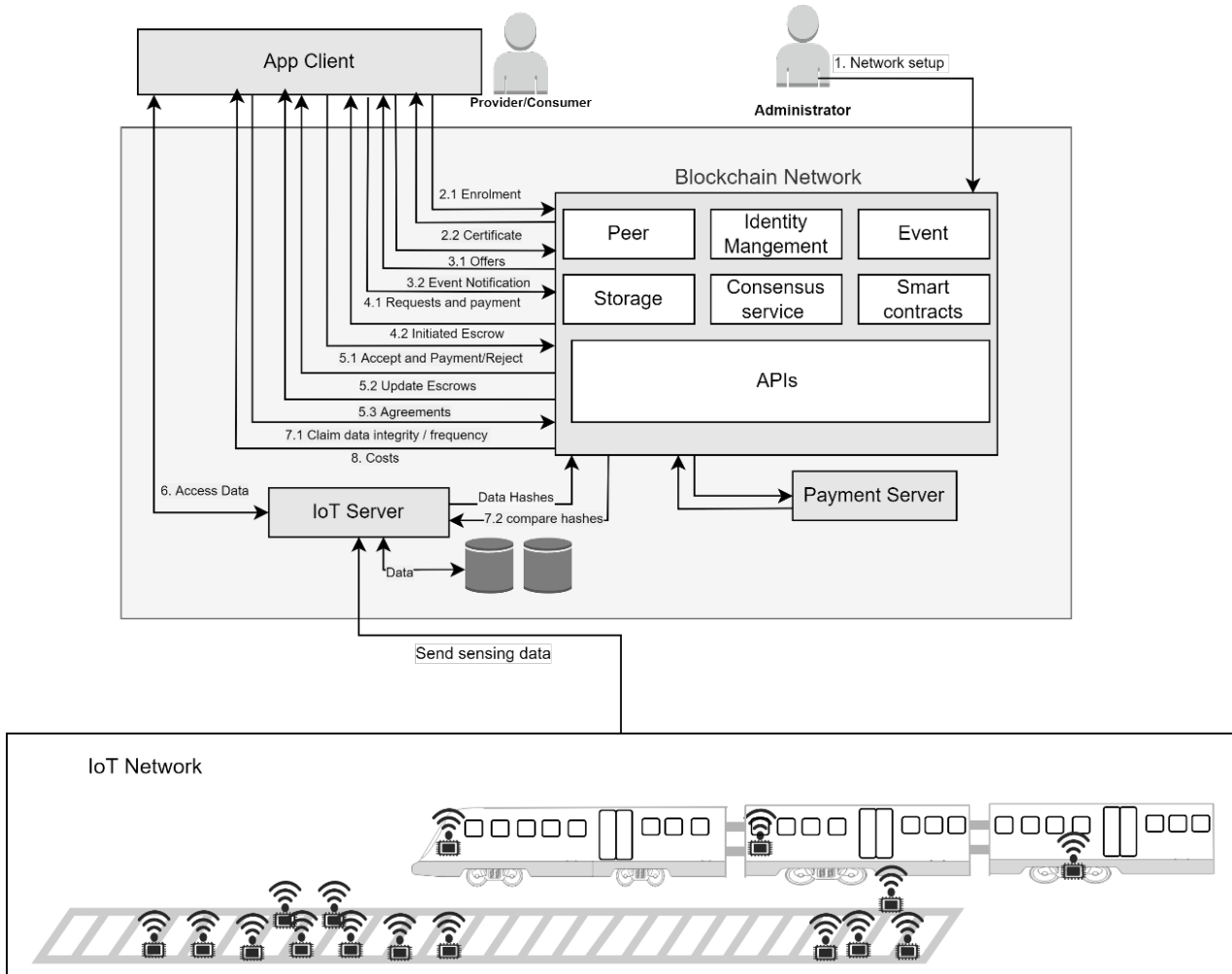


Figure 4: Workflow through the platform

6.1 Additional Deployment Considerations

Data Integrity

The data generated from the sensors should be hashed to avoid and detect any data manipulation. In both use cases, the consumer has direct access to the sensors and will gain the same data the provider is gaining from the same sensors. When the server is hashing this data before attaching it to the blockchain, this could help to trace and allocate any presence of data manipulation when occurs due to external or internal penetration by referencing the saved hash.

Data Frequency

Any latency in sending the data to the consumer can be proven against the pre-scheduled arrangement that has been added to the agreement smart contract. Once the consumer has an issue with any latency in receiving the data, the appended hash value time stamp will be

compared to the data generation schedule. There is a delay margin that should be agreed on the forehead in order to maintain the transmission delay as sensors are subject to latency due to delays in data processing and transmission of the data via a mobile network. It should be noted at this stage that in order to avoid large, variable delays associated with the preparation / commitment of large quantities of sensor data to the blockchain, raw sensor data itself is not directly lodged on the chain; instead, a much smaller hash, enabling verification of the data once delivered, is added to the chain with the data itself lodged in a more traditional data store. In this way latency due to writing information to the blockchain can be controlled and minimised.

Cost Distribution

Cost attribution between the parties is handled through a combination of the claim, escrow, and data sharing agreement smart contracts. The cost attribution process is automatically executed by the blockchain when a data sharing agreement either expires or is revoked by one of the parties involved. Escrow payments associated with the agreement (those set out between the data producer and consumer when accepting / confirming the initial data offer) are calculated first; next the claims smart contract is used to ensure no penalty clauses are to be applied, and adjustments to the final amount of the transaction made if needed, before payment is transferred between the consumer and producer of the data. In the case of recurring transactions, such as may result from schedule-driven data sharing in response to a timetabled vehicle traversing a section of line each day, cost distribution takes place at the conclusion of each “cycle” of the schedule. If we assume that there are scheduled journeys published to all participants in the network, the consumer is able to expect the number of transactions based on this published timetable by the time of sending the request. The frequency of capturing data depends on the number of train journeys on each day. Therefore, publishing the journey timetable will help determine how many transactions will take place between the provider and the consumer. Based on the number of transactions that already occurred, the payment will be calculated.

Historical Data

This is not handled within an IoT environment as this is not considered real-time processing and no IoT device has the ability to store massive amounts of historical data. Therefore, requests for historical data will require the system include a historian database / archival store to be available.

7 Deployment Context / Environment

7.1 Setup of Development Environment

The introduced platform consists of three main parts, the Blockchain Network, The IoT devices simulator, and the Client App. In our implementation, the sensors have been simulated to emulate the data-gathering process and how will be stored in the database. Accordingly, the hash value of the generated data will be stored on the blockchain at the same time. The development stack for the implementation of the Blockchain network is illustrated in Table 1. The operating system used for development is Ubuntu Linux 18.04.4 LTS while the processor is Intel Core i7-8650@1.90 GHz along with 15.5 GB of memory. The running environment is provided through docker engine of version 19.03.8 while the docker images and containers are configured and integrated by the docker-composer of version 1.23.2. The open-source blockchain framework Hyperledger Fabric (V 2.2) is used and to use the software development kit (SDK) to develop fabric-network, the Node (v14.15.4) is installed. For implementing the smart contracts, Golang is used. To hold the current state values of the logs, Couch DB is used to enable complex queries. In Table 2, the development tools used to implement the customized IoT platform are listed. The used IDE is Spring Tool Suite (STS) which is a free Eclipse-based development environment that is a JAVA-based platform highly useful for developing web applications to build local customized IoT platforms. The communication between device simulators and the server will be through Redis while the communication between the device server and blockchain network shall be using HTTP. For data, we used data samples generated from the real devices to feed the simulators to verify the system performance. The app provides an intuitive interface through which the client is able to simulate how the device is generating data and connected through the gateway server (Redis) to the blockchain, see Figure 5.

Table 1: Development environment (blockchain)

Component	Description
CPU	Intel Core i7-8650 @ 1.90 GHz
Memory	15.5GB
Operating System	Ubuntu Linux 18.04.4 LTS
Docker Engine	Version 19.03.8
Docker Compose	Version 1.23.2
Node	Version 14.15.4
Hyperledger Fabric	Version 2.2
IDE	Visual Studio Code
DBMS	Couch DB, MongoDB
Programming Language	Node.js, GoLang

Table 2: Development environment (IoT simulator)

Component	Description
Storage server / Gateway	Redis
Memory	15.5GB
Operating System	Ubuntu Linux 18.04.4 LTS
IDE	STS
Transmission Protocol	HTTP
Programming Language	JAVA

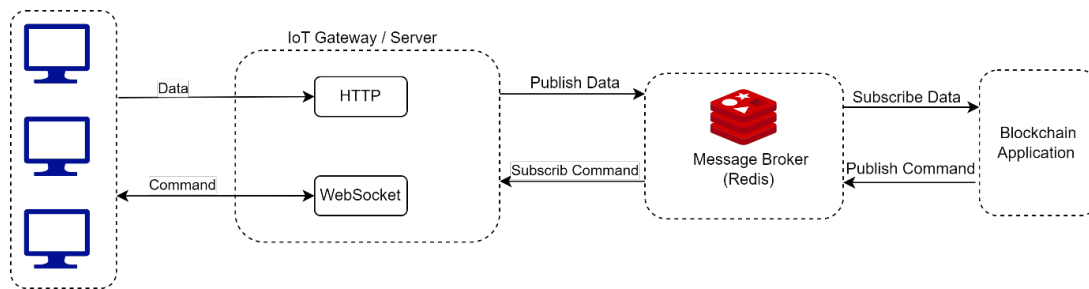


Figure 5: Integration of IoT simulator with blockchain network via Redis

For the client app, a web app which is divided into backend and frontend is developed using tools listed in Table 3. The backend part represents the server that is translating communication protocol and routing the requests from the web application to the blockchain and vice versa. For the frontend, HTML, CSS, JavaScript are used to develop customized graphical user interfaces to enable interaction with the REST server to invoke relevant APIs to submit proper transactions to the blockchain by building HTTP requests.

Table 3: Development environment (frontend)

Component	Description
Browser	Chrome, Firefox
Memory	15.5GB
Operating System	Ubuntu Linux 18.04.4 LTS
IDE	Angular CLI V11.2.11
Transmission Protocol	HTTP
Programming Language	Node.js, HTML, CSS, JavaScript

7.2 Sequence of Data Exchanges

The execution sequence of how the developed simulator interacts with the blockchain platform is described in Figure 6; this assumes that an agreement is already in place between the producer and consumer of the data that is based around a set of published offers of the scheduled journey in which the start time and end time of data sensing are stated. The sensor through the simulator in our application will register data via HTTP request to the IoT server using the POST method with the device's unique identifier (Device_id) to connect the device with a broker. Data communication is via the MQTT protocol, and makes use of a topic called "device_data" to flag data to the network. Then the server will publish the sensing data to the broker (Redis) which in turn submits this data to the subscribed blockchain nodes. On the blockchain side, the hash value of the data shall be calculated and recorded to the filesystem and the state database shall be updated with the newly added hash value while the raw data shall be stored to the external storage. The blockchain will also emit the notification to the clients via WebSockets based on the agreement which is still active to notify the consumers of the generated data.

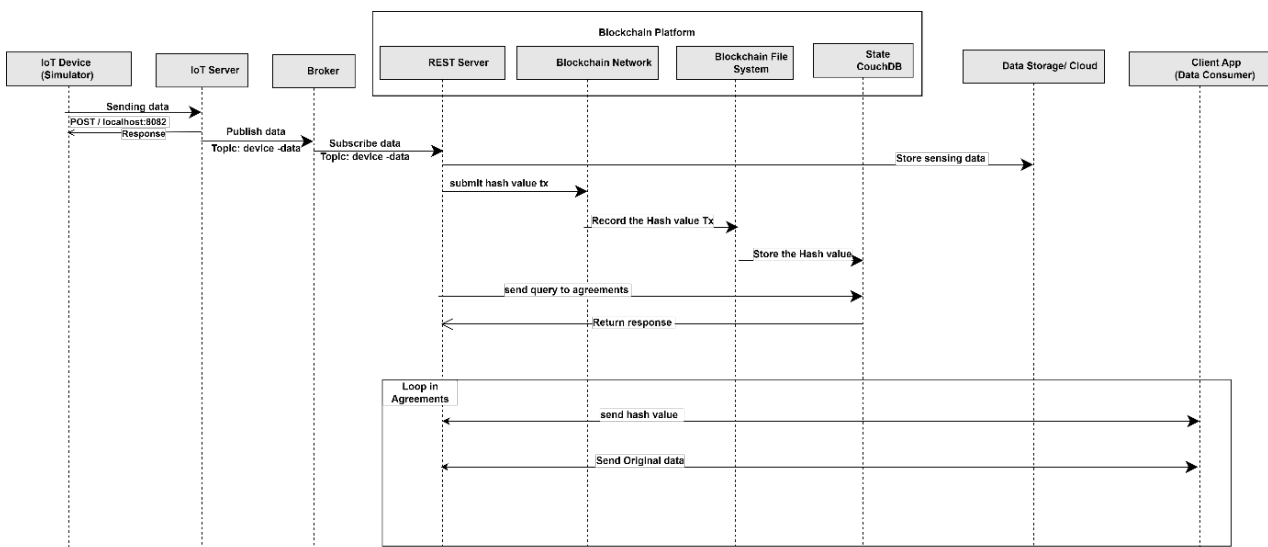


Figure 6: Sequence diagram illustrating data exchanges between sensors and the blockchain

7.3 IoT sensor simulator

In the first use case, there are 17 different sensors collaborating together to fetch different measurements which are processed later to estimate the condition of the axle journal bearing. At first, the back-end simulators which represent the 17 sensors that are mounted on the rail track to fetch the condition data from the axle journal will be set to send data based on the scheduled time (as recorded at collection but with an offset for the starting time of the simulation) as shown in Figure 7. The data will be transmitted to the server to be then published to the broker.

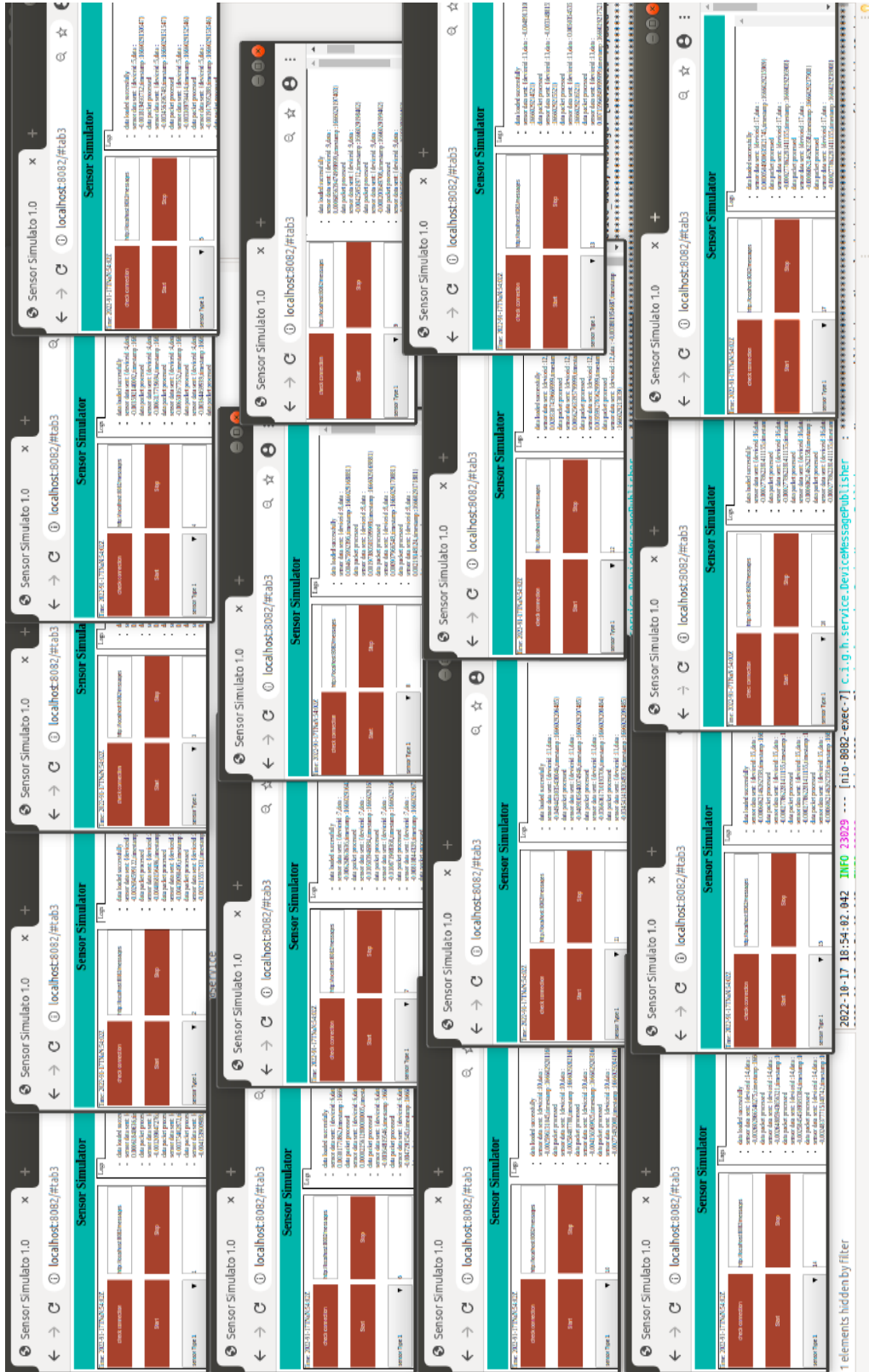


Figure 7: Simulated sensors in the first use case

The second use case is using different number of sensors and different data structures for each sensor, see Figure 8. The data stream will trigger the Redis broker to send the produced stream to the subscribers, in our case the blockchain, to calculate the hash value of the data and to store data in the database of the authenticated users. The developed simulator communicates to the blockchain through the MQTT protocol, all device nodes should listen to the channel and then subscribe or publish message/data on specific topic; namely “device_data”. At the client’s side, the blockchain node is supposed to listen to the same channel and subscribe to the same topic to be able to receive messages when generated from devices. The triggered messages from sensors that further communicate with Redis broker and are then fetched by the blockchain will be redirected to users according to the agreements that have been built between consumers and providers as mentioned before. Queries are written within the smart contract in the blockchain and by them, the data will be easily extracted from the data storage.

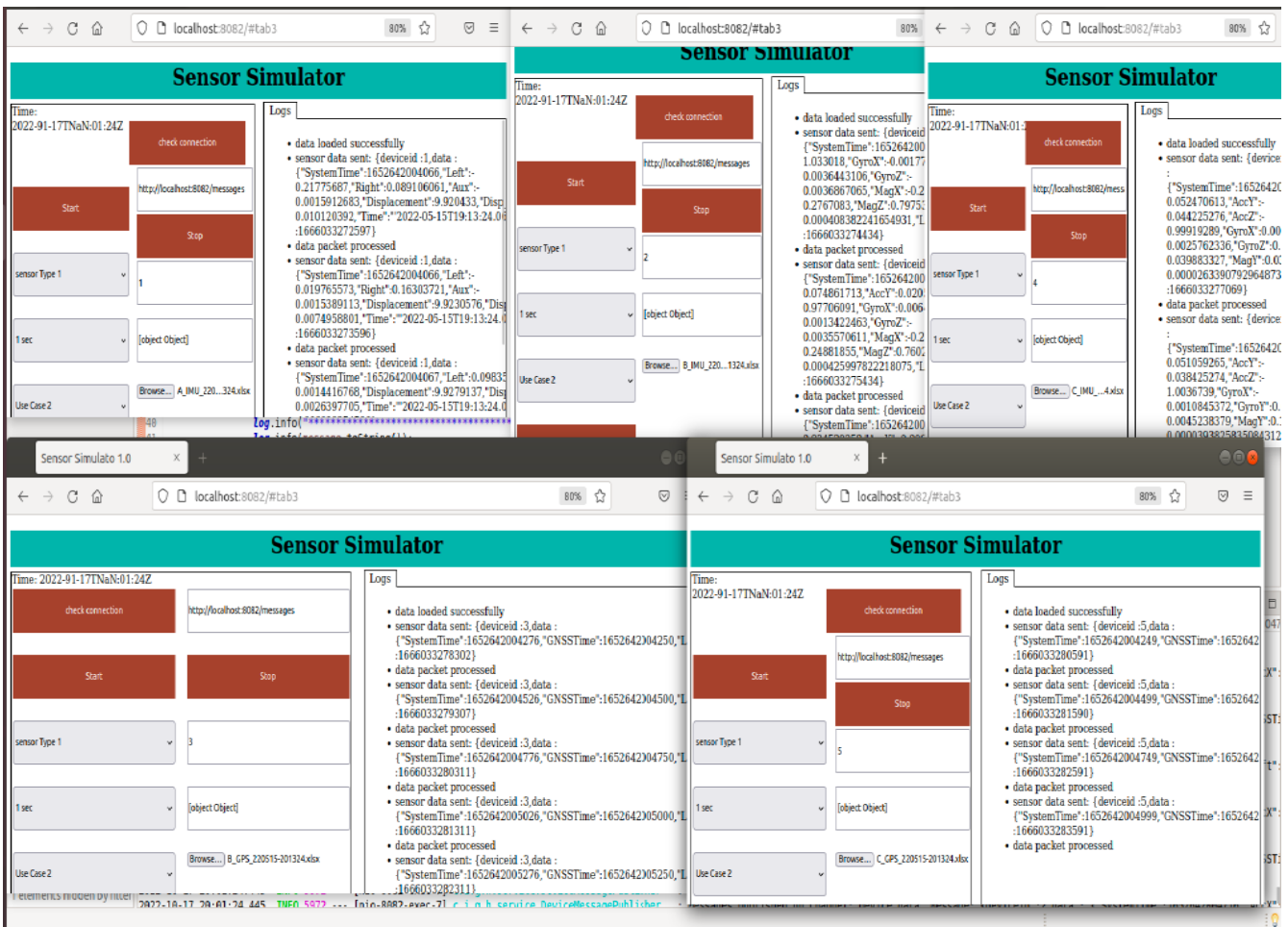


Figure 8: Sensors simulated in the second use case

As reported in section 5, the data used in the demonstrator is drawn from representative use cases within the rail industry. In the case of the walkthrough video linked in section 8.3 of this document, that is an inertial measurement system designed to be mounted on an in-service

vehicle. While the complete system uses a number of sensors mounted in different areas of the vehicle, Figure 9 shows the simulation / replay of a single sensor node (containing several individual sensors) from this use case to enable a clearer view of the form of the data within the system (see right hand panel). In transit, and in common with many telemetry systems around the industry, this data is exchanged using a proprietary XML format that is custom to the deployed system. The simulator is streaming an unpacked version of that xml stored in a comma separated Excel file for ease of use, but it would be a trivial exercise to extend this to include, for example, the raw xml. In industrial systems a number of initiatives have proposed specific IoT data exchange formats that could be used to push data to the framework. The Horizon 2020 In2Rail project[6], for example, proposed the use of a data format based on the Open Geospatial Consortium’s sensorML for use in combination with railML to allow sensor data originating from rail assets / representing asset status to be exchanged with respect to known locations on the infrastructure. In order to deliver similar functionality to XML models of this type with less overheads, many IoT sensors working in closed constraints make use of JSON as a representation of choice; a semi-structured system of essentially key-value pairs, JSON is intended to be “self-describing” to human readers, but without the overhead and structure of XML.

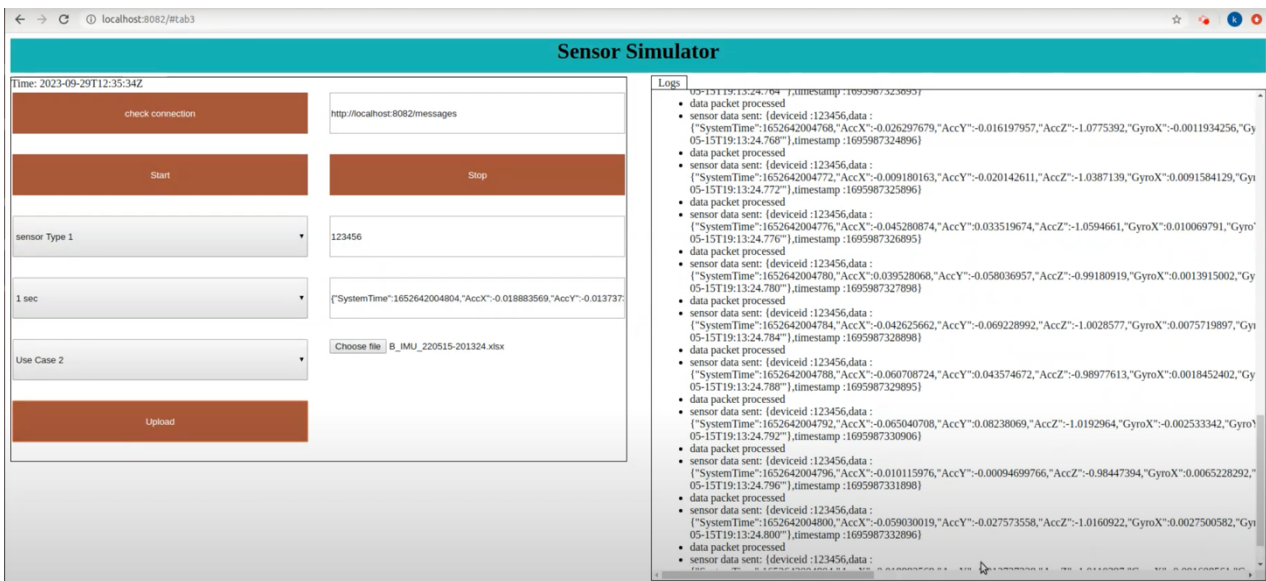


Figure 9: Single sensor simulation enabling a clearer view of the data streamed.

8 Deployed Proof of Concept

This developed project stands as an innovative blockchain-based platform constructed on the foundation of Hyperledger Fabric. At its core, the platform employs smart contracts scripted within chaincode. This main endeavour is driven by a fundamental mission: to effectively combat data integrity concerns by establishing a secure and transparent ecosystem. Its focal point revolves around the exchange of sensor data linked to railway assets to support the remote condition monitoring process. The underpinning power of blockchain technology lies at the heart of this project. By harnessing its capabilities, a fortified infrastructure is erected. This, in turn, empowers data providers to vendor the generated data from specific sensors for specific train journeys with confidence. Meanwhile, prospective consumers are afforded the opportunity to requisition and acquire sensor data, resulting in an unassailable data marketplace characterized by reliability and immutability. Executing this mission involves a dynamic synergy of technologies. Notably, Node.js and MongoDB assume pivotal roles, harmonizing seamlessly with the blockchain framework. The strategic marriage of these elements yields an agile mechanism for data operations. It ensures the efficient storage, retrieval, and processing of data, ultimately culminating in a solution celebrated for its resilience and scalability. Furthermore, the project's ingenuity extends to its incorporation of an Internet of Things (IoT) simulator. Through this integration, the platform orchestrates the reception of real-time data from sensors. This infusion of live data enhances the authenticity and pertinence of the information exchanged within the ecosystem. As a result, the "Data Management" project exemplifies a holistic approach that amalgamates cutting-edge technologies to reimagine data exchange with newfound security and transparency.

8.1 System Architecture

Figure 10 and Figure 11 illustrate the overall system architecture and the business processes architecture respectively.

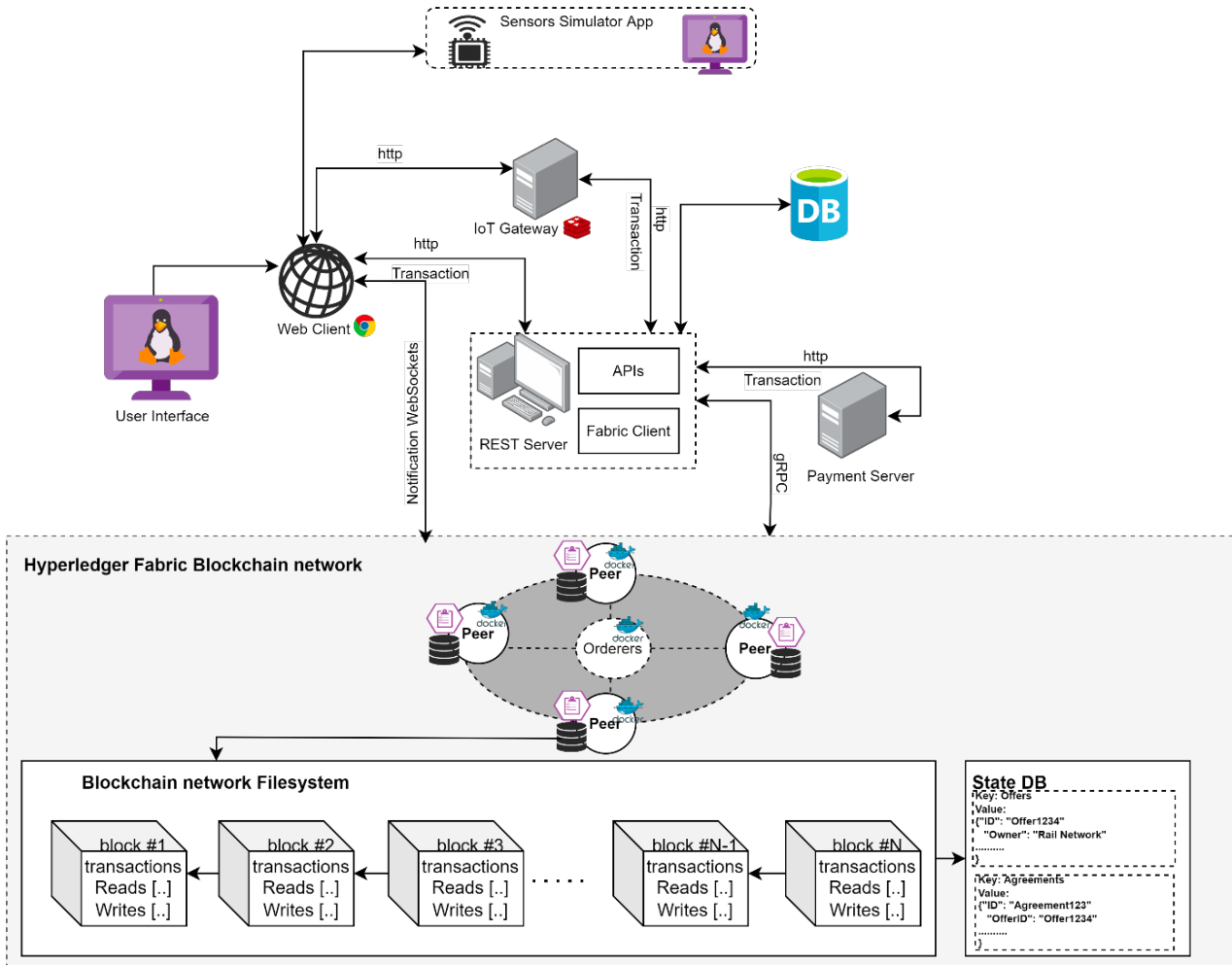


Figure 10: System architecture for proof of concept

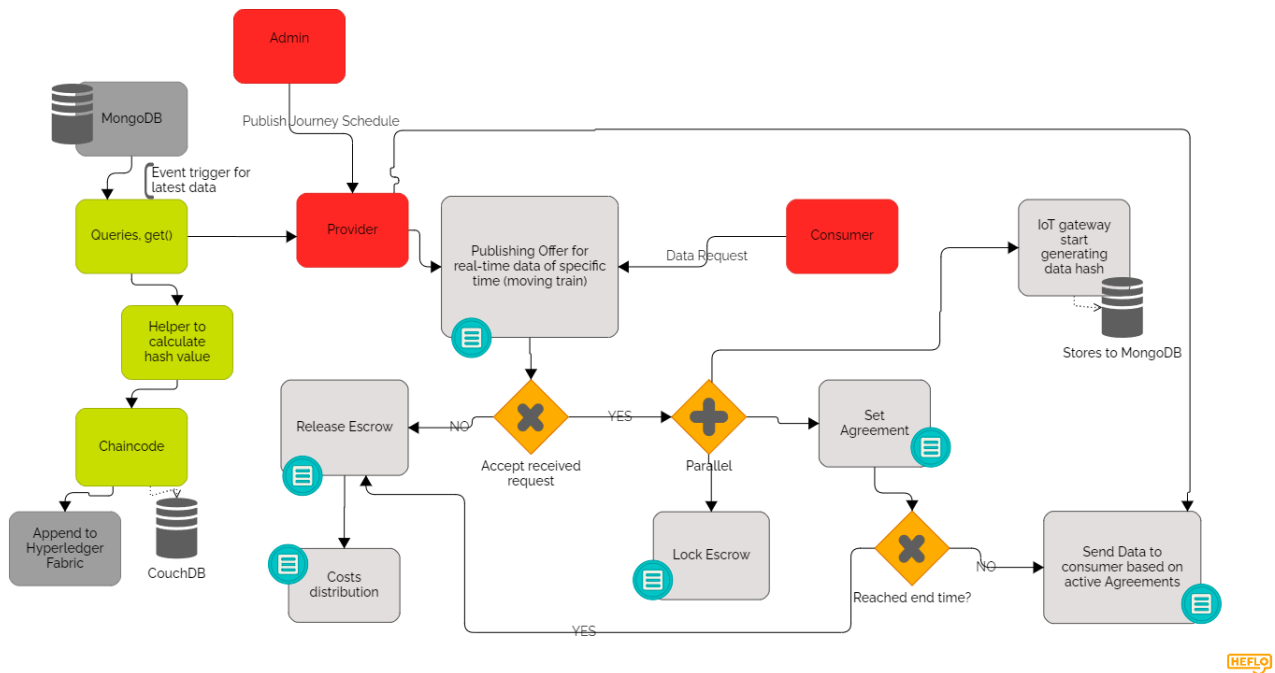
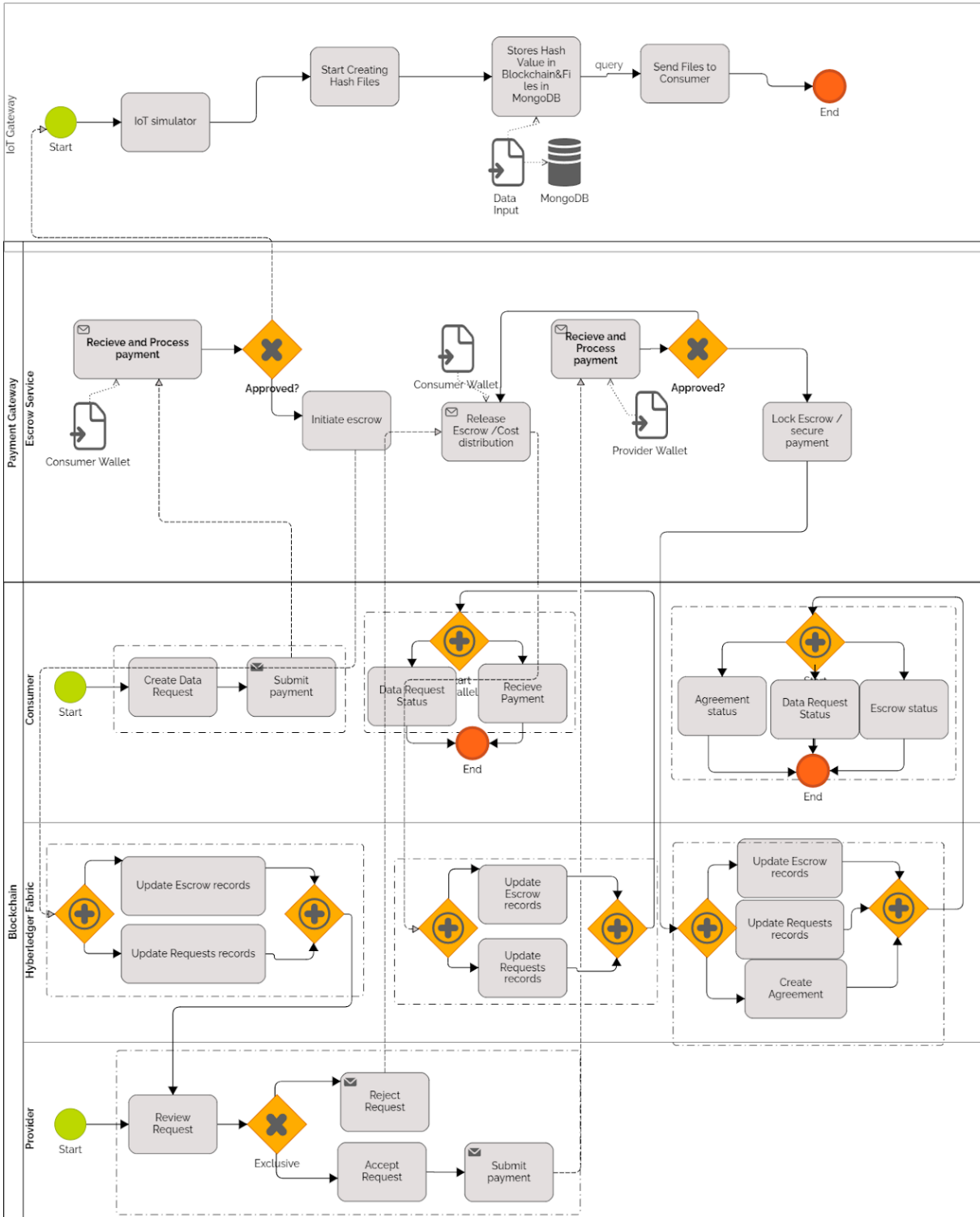


Figure 11: Business process flow for proof of concept

Within the framework, the blockchain provides a resilient, decentralised tamper-proof ledger of data transactions. This is realised as a private, permissioned blockchain network through Hyperledger Fabric. Management of the smart contracts is inherently provided within the network.

8.1.1 Transactional Flow

The flow of a transaction through the proof of concept is as shown in Figure 12.



HEFLO

Figure 12: Flow through a transaction

The flow includes the following steps:

1. Schedule generation for services - Administrators possess the capability to formulate journey schedules, outlining the precise time, date, and location of forthcoming events. This establishes a well-organised chronology for the process of sharing data between providers and consumers.
2. Initiation of data offer by provider – Data providers initiate the data-sharing process by initiating offers in alignment with predefined journey schedules. They specify pricing, data file formats, and other pertinent particulars, facilitating consumer engagement.
3. Offer inquiry from consumer – Consumers are afforded the flexibility to explore and initiate inquiries regarding offers presented by providers. By selecting suitable offers, they express their intention to acquire specific sets of data, thereby commencing the negotiation phase.
4. Data transmission and escrow – Upon offer acceptance, providers securely transmit data files generated from IoT devices to consumers. To ensure transparent financial transactions, an escrow mechanism is established. Payments are securely held in escrow and are released upon the expiration of the offer's end date, fostering trust between involved parties.
5. Generation of agreements – Agreements are automatically generated subsequent to offer acceptance. These agreements encapsulate the terms governing the exchange of data, encompassing payment arrangements, data file specifications, and other pertinent elements. They serve as legally binding records, affirming the validity of the transaction.

Detailed representations of the flows of sensor (commodity) data, the offer / escrow / agreement process, and the determination of costs are shown in Figure 13, Figure 14 and Figure 15 respectively.

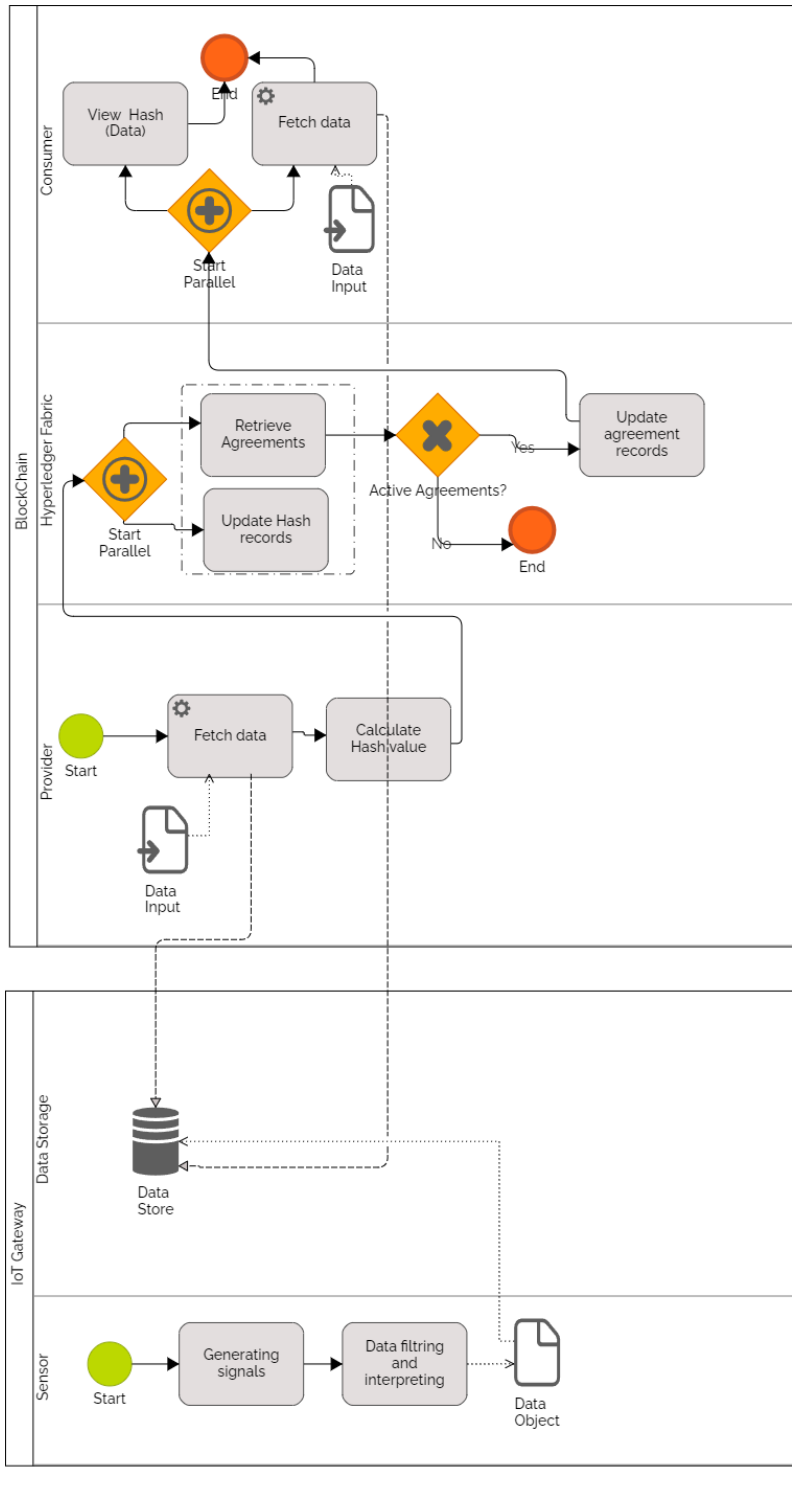


Figure 13: Data flow for commodity (IoT sensor) data within the proof of concept

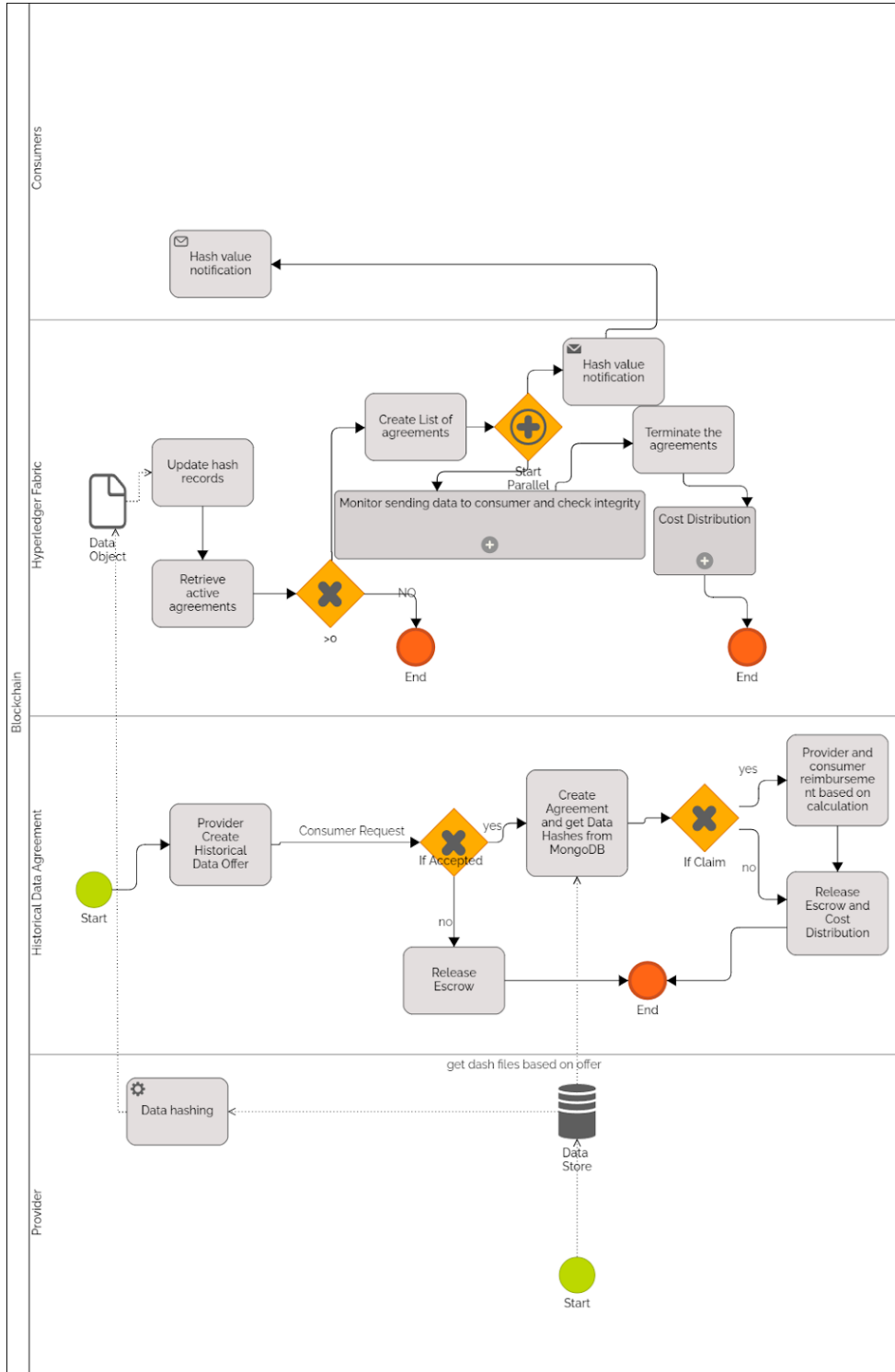


Figure 14: flow of the offer / escrow / agreement process within the proof of concept

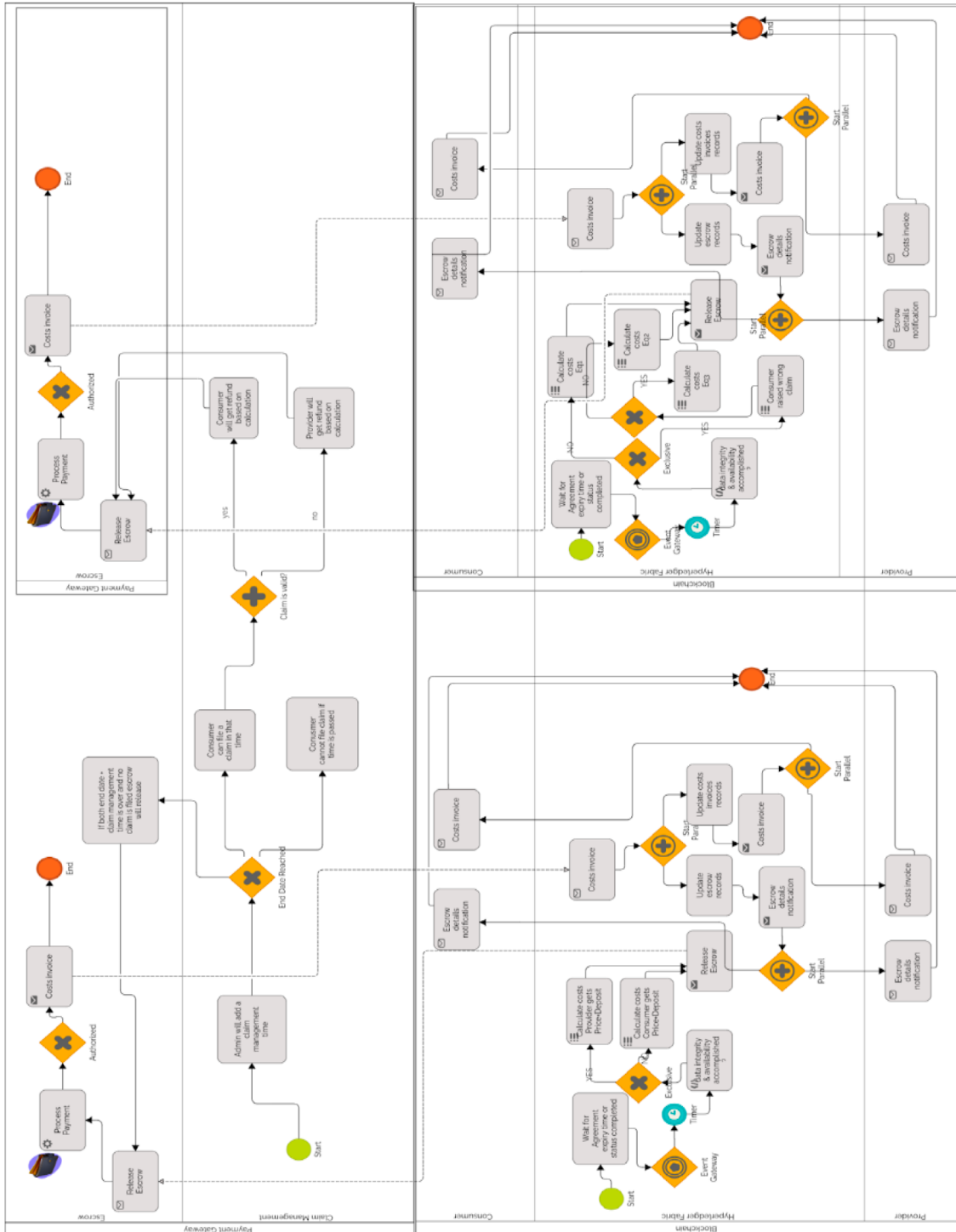


Figure 15: Flow of cost determination within the proof of concept

8.1.2 Claims Management, Escrow and the Distribution of Costs

As described in the accounting model in Deliverable 1, the B4CM framework includes mechanisms for exchanging parties to report problems with the transaction, and these impact on all payment workflows. Within the proof of concept that process is captured as follows (depending on the state of the transaction):

- **Claim Management and Escrow Release** – Once the agreement’s end date is met, the claim management process begins its countdown. The administrator retains the flexibility to adjust the claim management time as per platform requirements. Following the expiration of both the end date and the claim management time, the escrow linked with the offer is released.
- **Final Distribution of Costs** – The conclusive allocation of costs is initiated upon the successful fulfilment of the offer. This distribution meticulously documents all financial interactions between the provider and the consumer.
- **Absence of Consumer Concerns** – In cases where the consumer does not raise any concerns regarding the data file or the offer, and the end date, along with the claim management time, has passed, the escrow is unblocked. The consumer is refunded their deposit, and the provider receives their deposit alongside the consumer’s payment.
- **Activation of the Claim Management Timer** – Upon the conclusion of the end date, the claim management timer springs into action. During this span, consumers can evaluate the received data file for any discrepancies or issues.
- **Consumer’s Claim and Associated Consequences** – If the consumer identifies discrepancies or omissions within the data file during the claim management duration, they possess the right to lodge a claim. Penalties are imposed on the provider if the data file proves incomplete or inaccurate. Subsequently, the provider’s deposit is reimbursed to the consumer, along with their own deposit and the payment received.
- **Verification of Data Integrity** – The platform meticulously cross-examines the hash values of the data file against both the blockchain and the MongoDB database. In the event of disparities, the provider is subjected to penalties. However, if the hash values correspond, the claim process proceeds systematically.
- **Unsubstantiated Penalty Claim by the Consumer** – Should the consumer raise a claim without presenting substantial proof of issues within the data file, and if the data hash values are verified as accurate, the consumer bears the brunt of a penalty. Consequently, the provider reclaims both the consumer’s deposit and their own, while the platform duly updates the cost distribution record.
- **Revocation of Agreements** – While the end date is not yet reached, both the provider and the consumer reserve the option to annul the agreement. A penalty is imposed upon the party initiating the revocation post acceptance.
- **Agreement Revocation by the Provider** – In the scenario where the provider opts to revoke the agreement after acceptance, the consumer regains their deposit, and the provider retrieves their own deposit. The consumer also receives their payment.

- **Agreement Revocation by the Consumer** – Should the consumer opt to revoke the agreement after acceptance, the provider acquires both the consumer’s deposit and their own. While the consumer receives their payment, their deposit remains unrecovered.
- **Revocation Beyond the End Date** – Revocation of the agreement is rendered infeasible for either party once the end date has lapsed.
- **Resolution of Revocation and Distribution** – The entirety of the details associated with agreement revocations is comprehensively documented within the cost distribution record. Serving as a comprehensive summary of financial transactions, penalties, and reimbursements shared between the provider and the consumer, this record encapsulates the complexity of the cost distribution process. It encompasses claim management, penalties, refunds, revocation scenarios, and meticulous recording of pertinent particulars within the cost distribution record.

8.1.3 Payments

The B4CM proof of concept makes use of a third-party payment system development harness to show how micropayments integrate with data exchange transactions. This is shown in Figure 16 (payment processing) and Figure 17 (integration of payment with workflow). When a consumer initiates a payment, the payment gateway initiates a hosted checkout page. This page presents an intuitive interface, allowing the consumer to select their preferred payment method. If the consumer opts to pay via a card, they will be prompted to input essential card details, including the card number, expiry date, and CVV code.

Upon inputting the card details, the payment gateway undertakes a validation process to verify the precision and legitimacy of the provided information. If the card details are accurate and the transaction goes through successfully, the payment gateway presents a message confirming the successful transaction, signifying that the payment has been processed seamlessly.

Following the display of the success message, the payment gateway redirects the consumer back to the merchant’s website, signalling the successful completion of the product purchase. At this juncture, the consumer gains access to the purchased product or service on the merchant’s website.

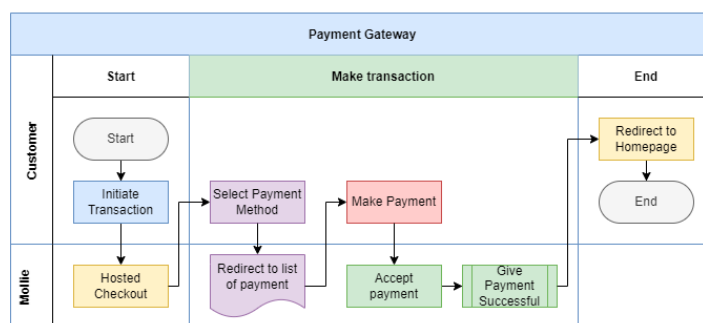


Figure 16: Transaction fulfilment within third party payment system

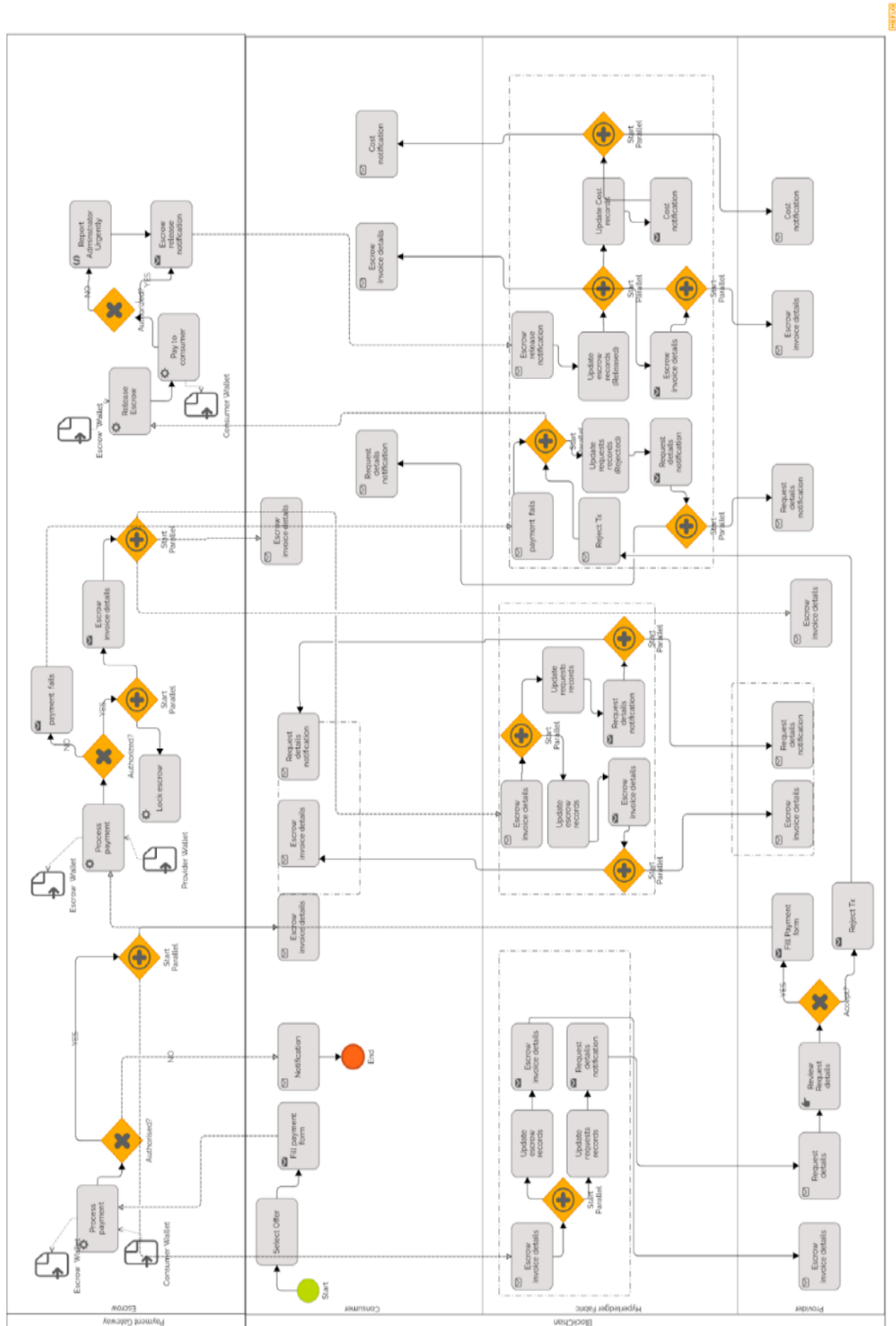


Figure 17: Payment process workflow with framework

8.2 Mapping of Roles to Entities in the Framework

This proof of concept employs the organizational structure of Hyperledger Fabric, with three distinct roles: provider, consumer, and admin, denoted as org1, org2, and org3, respectively. The core objective of this blockchain-driven project is to facilitate the secure and transparent exchange of sensor data associated with train journeys. Each role is endowed with specific functionalities and privileges. These are shown in Figure 18.

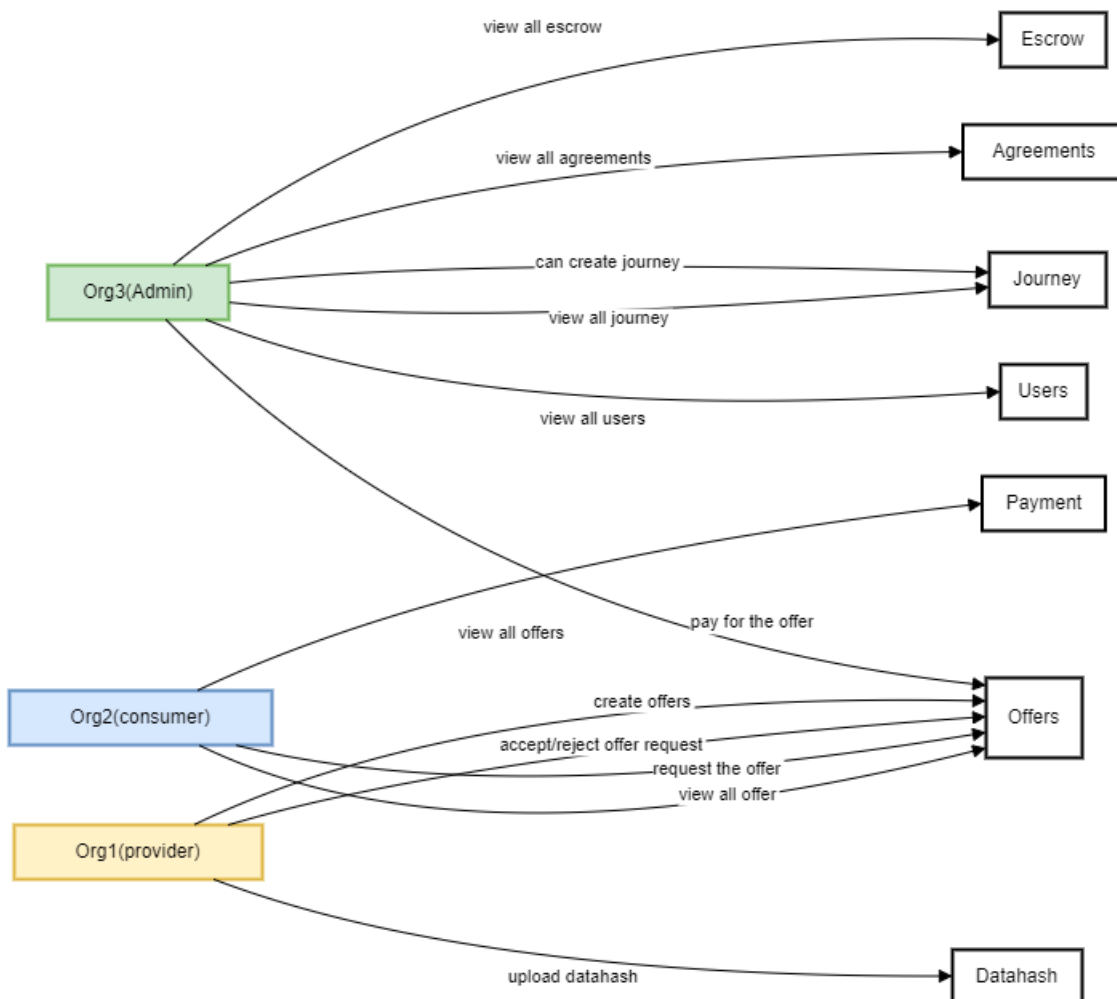


Figure 18: Mapping of user roles to Hyperledger entities

Provider (ORG1)

- Possesses the capability to generate offers concerning journey schedules, outlining the sensor data from the train to be encompassed in the offer through an IoT gateway.
- Establishes the price for the offer and presents it to potential consumers.

Consumer (ORG2)

- Can access available offers on the platform.
- Holds the choice to request offers from providers based on their preferences and needs.
- Once interested in an offer, consumers can make payments for the stipulated offer price.
- Following successful payment, consumers receive the sensor data from the chosen offer.

Admin (ORG3)

- Possesses administrative privileges within the system.
- Assumes responsibility for crafting and updating journey schedules.
- Possesses the ability to view all available offers on the platform.

8.3 Walkthrough of Core Functionality of Proof of Concept

This section walks through the core functionality of the proof of concept using screenshots; while appropriate given the format of the document, this isn't ideal as a demonstration and so the project team invites the reader to review two short videos available at:

- <https://youtu.be/0Du9G1q9YIc> (part 1 - an introduction to use cases)
- <https://youtu.be/4zxwuRMODEo> (part 2 - proof of concept demo)

8.3.1 Administration

Users with Admin roles within the framework have authority to manage all aspects of the configuration of the proof of concept, including provider and consumer role assignments within organisations participating within the network, and modification of stored credentials such as banking information (also available to the owning organisations).

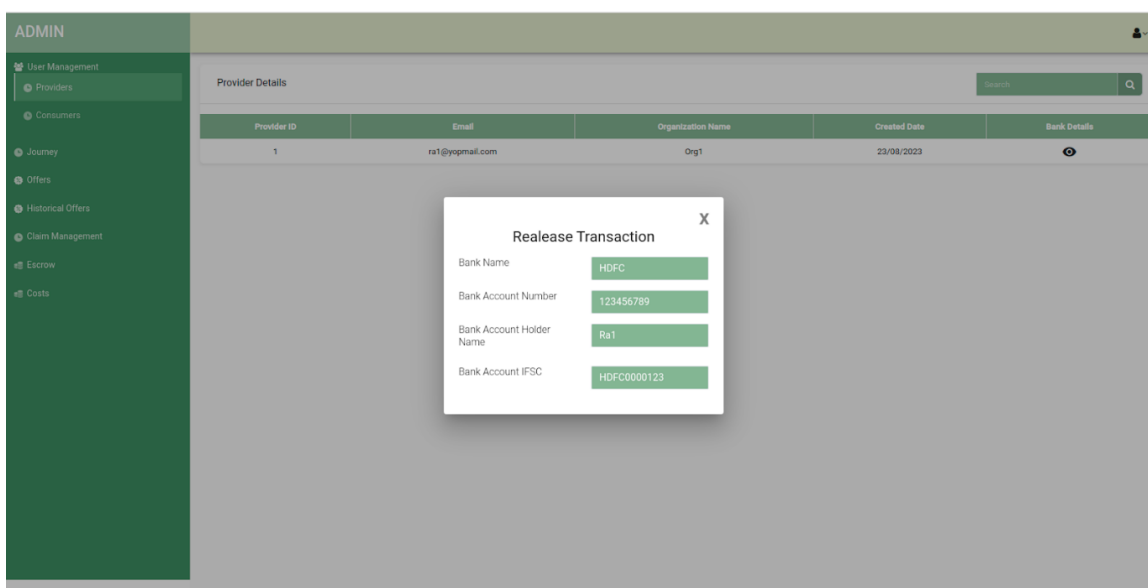


Figure 19: Management of financial credentials for an organisation (admin view)

8.3.2 Journeys

Journeys allow data from mobile platforms such as vehicles to be structured within the system. They can be overseen by users with an admin role within the system, and once created can persist for extended periods of time.

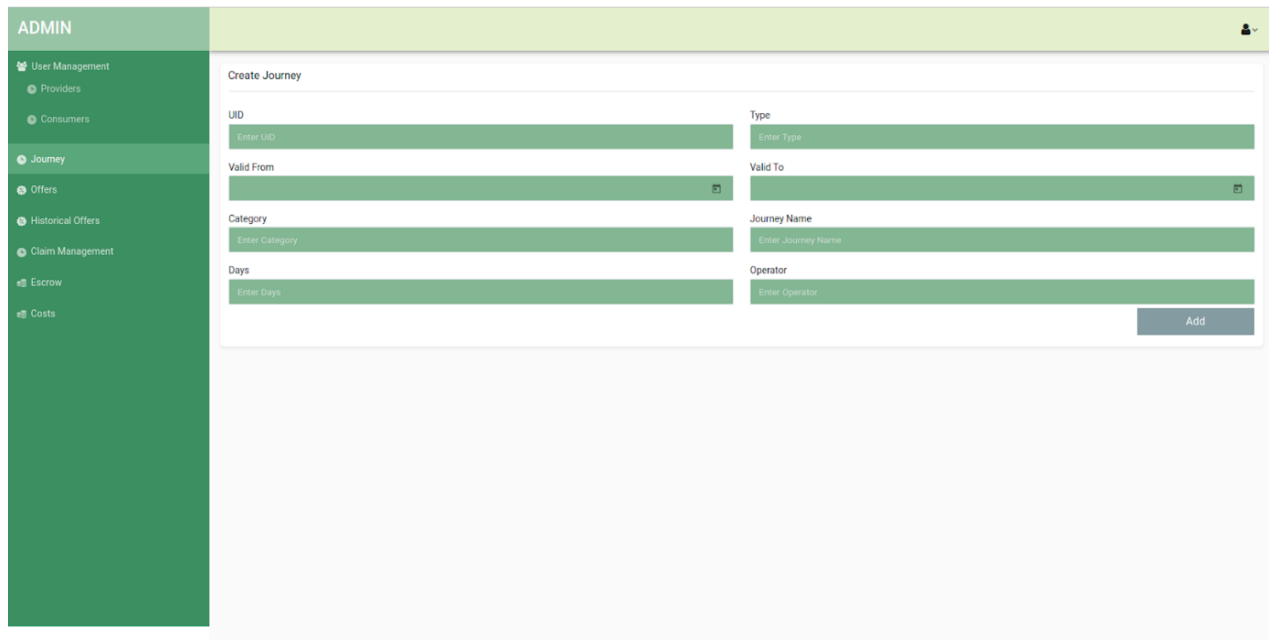


Figure 20: Journey creation (admin view)

8.3.3 Data Offers

Data provided for sale by providers is logged on the blockchain in the form of a Data Offer. Consumers and admin users (Figure 21) are able to review offered data and, in the case of consumers, instigate transactions against them. Providers can initiate the data sharing process by creating offer requests based on predefined journey schedules and other relevant details, making it easier for consumers to engage (Figure 22). When creating an offer within the proof of concept, the sensor simulator takes requests for the start time and end time of the offer in order to schedule it. Subsequently, it replays sensor data in accordance with the scheduled timeframe (Figure 23). In addition to reviewing live offers, providers need to review incoming requests for their data (against offers) and verify the details. If the details are accurate, they can accept the request; otherwise, they have the option to reject it (Figure 24). If an offer is accepted, it signifies the creation of an agreement, and subsequently, the payment gateway page will be presented to facilitate the completion of the payment for the specific agreement. Additionally, the data hash will be added to the blockchain, streamlining the process for the provider to access and download the associated file. In the event that an offer is rejected, the system will initiate a refund for the entirety of the consumer's deposit and fees. The agreement will then be closed.

ADMIN 👤

User Management

- Providers
- Consumers
- Journey
- Offers**
- Historical Offers
- Claim Management
- Escrow
- Costs

Offer Detail Search

Offer ID	Validity	Data Owner	Equipment	Sensor	Processing Level	Price	Deposit	Departure Date	Arrival Date
OFFER_DN28086962023FV05B	True	ra1@yopmail.com	E1	123456	L1	5	4	28/08/2023 16:42	28/08/2023 16:43
OFFER_H2280851S20237R3L6	True	ra1@yopmail.com	E6	123456	L6	6	6	28/08/2023 17:21	28/08/2023 17:22
OFFER_H5280891V2023770IG	True	ra1@yopmail.com	E3	123456	L3	6	6	28/08/2023 17:07	28/08/2023 17:08
OFFER_HB280830J2023T98VF	True	ra1@yopmail.com	E5	123456	L5	5	5	28/08/2023 17:19	28/08/2023 17:20
OFFER_KG2808TSN20237HCV8	True	ra1@yopmail.com	E2	123456	L2	2	1	28/08/2023 16:44	28/08/2023 16:45

Figure 21: A set of active data offers lodged within the system (admin user view)

ra1@yopmail.com Provider 👤

Provider

New Offer

Offer ID: OFFER_HH280837B2023OMUJC

Select Journey ID: U1

Departure Time: 2023-08-28 11:24

Arrival Time: 2023-08-28 11:25

Validity: True False

Data Owner: ra1@yopmail.com

Equipments: E1

Sensor ID: 123456

Processing Level: L1

Price: £ 5

Deposit: £ 3

Add

Figure 22: Creation of a data offer (provider view)

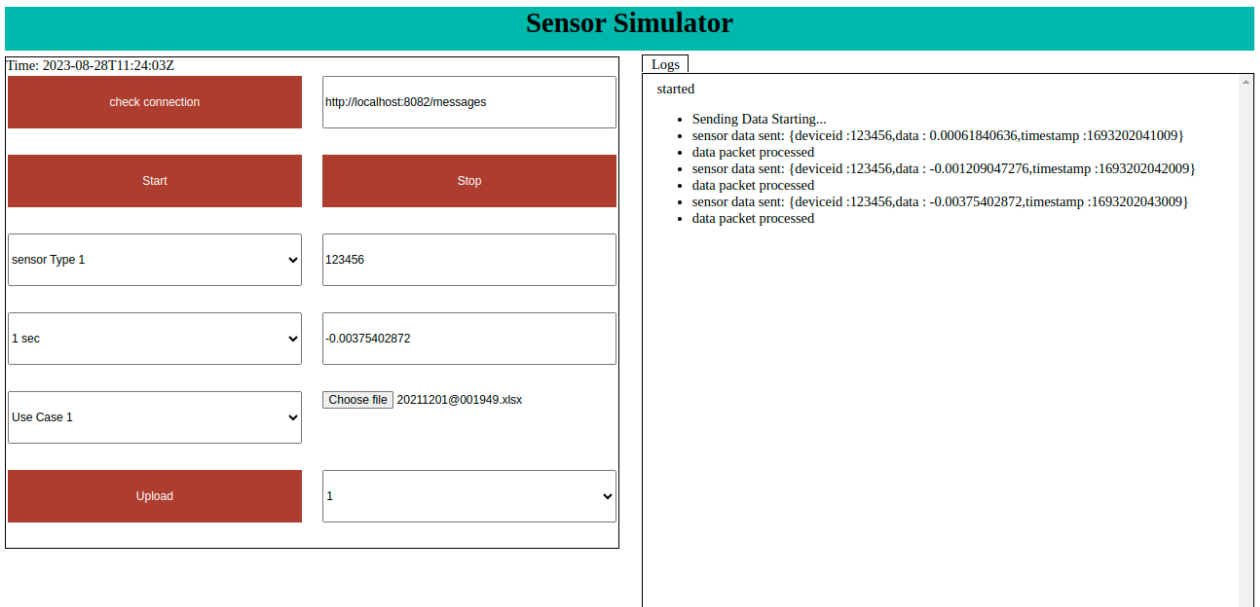


Figure 23: Replay of data by a simulated sensor

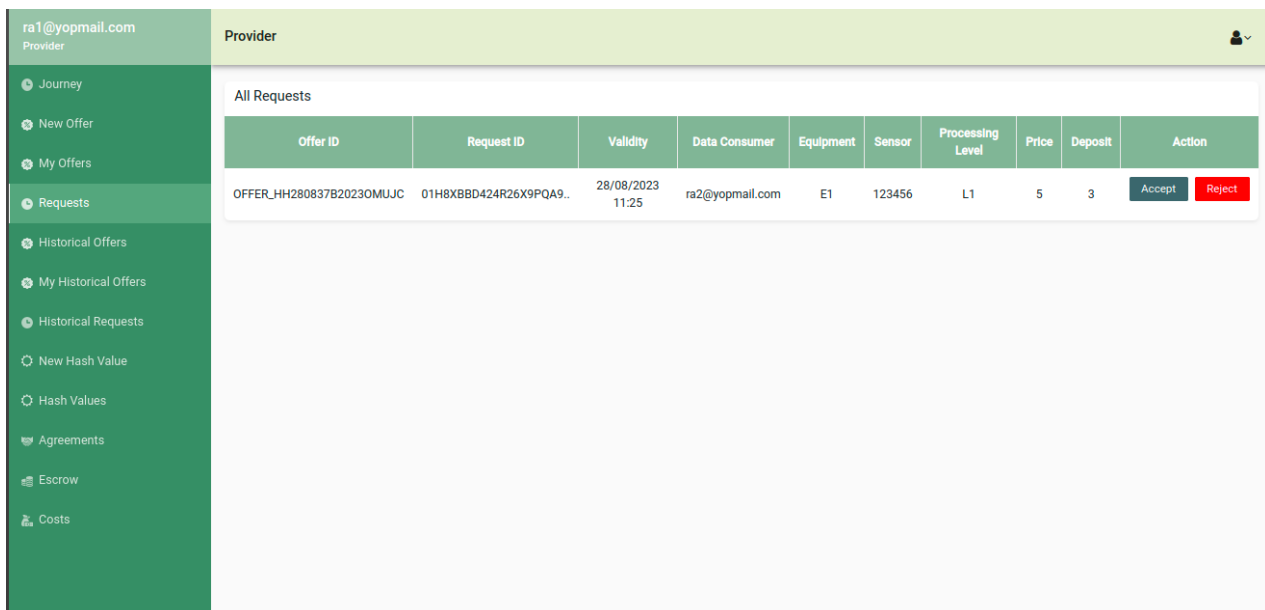


Figure 24: Provider view of requests in response to a data offer

Historical offers refer to offers for which normal offer agreements have been successfully established between consumers and providers. In other words, an offer becomes historical once an agreement has been reached between a consumer and a provider. Providers have the option of creating historical offers, essentially reusing data offers already issued and for which data exists within the framework with modified parameters such as cost (Figure 25). Payments etc. are handled in the same way as for new offers (Figure 28).

ra1@yopmail.com
Provider
Provider

- Journey
- New Offer
- My Offers
- Requests
- Historical Offers**
- My Historical Offers
- Historical Requests
- New Hash Value
- Hash Values
- Agreements
- Escrow
- Costs

Historical Offer

Sensor ID	123456	Offer ID	OFFER_SI2808S7U2023PSTC1
Departure Date	2023-08-28 09:00	Arrival Date	2023-08-28 15:15
Select Journey ID	U1	Select Multiple Offer ID	<input checked="" type="checkbox"/> OFFER_HH280837B20230MUJC <input checked="" type="checkbox"/> OFFER_AJ2808QU72023VET9N
Validity	<input checked="" type="checkbox"/> True <input type="checkbox"/> False	ra1@yopmail.com	
Equipments	EH1	Processing Level	LH1
Price	£ 4	Total Price	£ 8
Deposit	£ 2		

Figure 25: Historical offer interface (provider view)

8.3.4 Escrow

Admin users have an overview of all the Escrow processes lodged in the blockchain. This access enables administrators to monitor and oversee the financial interactions occurring between parties, ensuring transparency and accountability throughout the process (Figure 26).

ADMIN
ADMIN

- User Management
- Providers
- Consumers
- Journey
- Offers
- Historical Offers
- Claim Management
- Escrow**
- Costs

Escrow Details

Escrow ID	Offer ID	Offer Request ID	Provider Deposit	Consumer Deposit	Payment	Released
XGHCRCFKMJVVPQZ1M32WQ001H8X	OFFER_DN2808G962023FV05B	01H8XXGHCRCFKMJVVPQZ1M32WQ0	4	4	5	false
Y4H7MBANYHQ369W668MB601H8X	OFFER_KG2808T5N20237HCV8	01H8XY4H7MBANYHQ369W668MB6	1	1	2	false
ZCX0HBCY72DESSQYVYZB01H8X	OFFER_H5280891V202377OIG	01H8ZCX0HBCY72DESSQYVYZB	6	6	6	false
ZK5NQCZY41NX9V04YPMWF01H8X	OFFER_HB280830J2023T98VF	01H8XZK5NQCZY41NX9V04YPMWF	5	5	5	false
ZM0SWCNCNFVNMGMXEANGK01H8X	OFFER_H2280851S20237R3L6	01H8XZM0SWCNCNFVNMGMXEANGK	6	6	6	false

Figure 26: Active Escrow processes (admin view)

When the escrow is released, either at the end date of the agreement or upon offer rejection, a detailed "Cost Distribution" record is generated. This record provides a comprehensive breakdown of how the funds are distributed between the consumer and the provider. It accurately reflects the payments made by each party involved in the transaction. This informative record is accessible to administrators, allowing them to review and verify the financial distribution of the transaction (Figure 27).

ADMIN										
Costs										
ID	Agreement ID	Provider	Consumer	Provider Reimbursement	Consumer Refund	Provider Bank Details	Consumer Bank Details	Completed At	Action	
01H8XZYHH57C2ZZ1X4QBAGRNO7	QZY41NX9V04YPMWF01H8XZK5NC	ra1@yopmail.com	ra2@yopmail.com	10	5			2023-08-28 17:25	Release	
01H8Y0JP22KWCCMJQ17SW01Y1M	K9Z8Y7JFV03BZYFN01H8Y09BHK	ra1@yopmail.com	ra2@yopmail.com	4	1			2023-08-28 17:36	Release	
CNNFVNMGMXEANGK01H8XZM0SW	N/A	ra1@yopmail.com	ra2@yopmail.com	0	12			Invalid date	Release	

Figure 27: Record of "costs" (admin view)

Payment for OFFER SI2808S7U2023PSTC1

Test profile

€2.00

Note: this is a testmode payment.

Card number

4543 4740 0224 9996 VISA

Card holder

Rahul Baghel

Expiry date

10 / 25

CVV

123 ?

Pay >

Payment secured and provided by **mollie**

Figure 28: Payment gateway test harness used within the proof of concept

9 Conclusions

Objective 4 of the B4CM project was to develop a testbed application that could demonstrate the operation of the framework in the context of the rail sector, enabling future developers to extend the tools produced by the project.

As the team has shown in this document, B4CM has been largely successful in delivering that aim; a proof of concept has been produced that shows, for two representative use cases, how the framework can be deployed to handle data exchanges from those systems and provide the data to consumers through a set of published data offers and responses. The transactions are logged on the distributed ledger, providing an immutable audit chain that will enable fair attribution of costs over time, and the Escrow mechanism provides for some protection of data producers and consumers in the event of fraudulent behaviour by one or other party.

The portability of the proof of concept is an issue, and one that will need to be addressed if the technology is to achieve widespread adoption within the industry. The experiences of the project team have shown that while it is possible to migrate frameworks built on top of blockchain platforms between machines, a substantive reconfiguration effort is needed, and that requires an level of knowledge of the workings of the blockchain that some who wishes to experiment with the technology is unlikely to possess. Designs compatible with virtualization technology offered by commercial blockchain providers, such as the Ethereum Virtual Machine, offer one possible solution to this problem but are framed within the specific (Ethereum) ecosystem rather than Hyperledger as employed by the B4CM team. Nonetheless, the team have made the proof of concept available via the project's GitHub, and it is hoped that it will prove useful in the future.

Moving forwards, the B4CM team recommend that:

- Where future proof of concept platforms of this type are planned, they make the greatest use possible of virtual environments such as Docker containers – this should minimise the changes needed in migration across physical platforms and ease distribution.
- That portability of solution should be formally considered as a requirement in future blockchain frameworks of the type developed by B4CM.

10 References

- [1] M. Entezami, C. Roberts, P. Weston, E. Stewart, A. Amini, and M. Papaalias, “Perspectives on railway axle bearing condition monitoring,” *Proc Inst Mech Eng F J Rail Rapid Transit*, vol. 234, no. 1, pp. 17–31, Jan. 2020, doi: 10.1177/0954409719831822.
- [2] A. P. Gonzalo *et al.*, “Observations on track evolution from onboard inertial measurements,” in *2022 UKACC 13th International Conference on Control (CONTROL)*, IEEE, Apr. 2022, pp. 18–23. doi: 10.1109/Control55989.2022.9781456.
- [3] S. Akagawa, M. Hori, and J. Sugawara, “Frost Heaving in Ballast Railway Tracks,” *Procedia Eng*, vol. 189, pp. 547–553, 2017, doi: 10.1016/j.proeng.2017.05.087.
- [4] C. Liu, D. Thompson, M. J. Griffin, and M. Entezami, “Effect of train speed and track geometry on the ride comfort in high-speed railways based on ISO 2631-1,” *Proc Inst Mech Eng F J Rail Rapid Transit*, vol. 234, no. 7, pp. 765–778, Aug. 2020, doi: 10.1177/0954409719868050.
- [5] R. Alzahrani, C. Jones, and J. Easton, “D.1.1 B4CM software framework: An implemented software framework and supporting documentation,” <https://projects.shift2rail.org/download.aspx?id=de5c5940-7438-4a89-9073-1236d061fdd8>. Jan. 13, 2022.
- [6] In2Rail Consortium, “IN2RAIL D9.1 - Asset Status Representation,” 2016. [Online]. Available: <http://in2rail.eu/download.aspx?id=2760daae-5741-4c5a-bf28-416876058bea>