



D3.11 – Blockchain Traceability of Data v2

WP3 – BUILD: Manufacturing
Data Quality



Document Information

GRANT AGREEMENT NUMBER	958205	ACRONYM		i4Q
FULL TITLE	Industrial Data Services for Quality Control in Smart Manufacturing			
START DATE	01-01-2021	DURATION		36 months
PROJECT URL	https://www.i4q-project.eu/			
DELIVERABLE	D3.11 – Blockchain Traceability of Data v2			
WORK PACKAGE	WP3 – BUILD: Manufacturing Data Quality			
DATE OF DELIVERY	CONTRACTUAL	31-Dec-2022	ACTUAL	30-Dec-2022
NATURE	Report	DISSEMINATION LEVEL		Public
LEAD BENEFICIARY	IBM			
RESPONSIBLE AUTHOR	Yoav Tock, Benjamin Mandler (IBM)			
CONTRIBUTIONS FROM	8-BIBA			
TARGET AUDIENCE	i4Q Project partners, industrial community.			
DELIVERABLE CONTEXT/DEPENDENCIES	This document is D3.11. It is the second version of deliverable D3.3, due in month 24. Its relationship to other documents is as follows: - D3.3 Blockchain Traceability of Data - is the first version of deliverable D3.3. - D1.1 Project Vision Guide Document: vision of the i4Q^{BC} solution. - D1.4 Requirements Analysis and Functional Specification: requirements for the i4Q^{BC} solution. - D1.9 Requirements Analysis and Functional Specification v2: requirements for the i4Q^{BC} solution. - D2.7 i4Q Reference Architecture and Viewpoints Analysis v2: the role of the i4Q^{BC} solution in the reference architecture.			
EXTERNAL ANNEXES/SUPPORTING DOCUMENTS	None			
READING NOTES	None			

ABSTRACT

The Blockchain Traceability of Data (i4Q^{BC}) solution aims to enhance the level of trust that different solutions and components can place on data. It provides services of immobility and finality of data, serving as the source of truth, enabling trust in data by providing the possibility for full provenance and audit trail of data stored. This deliverable is the second part of the deliverables associated with Task T3.2. In the first deliverable (D3.3, 0), we mainly described Hyperledger Labs Orion, the blockchain database selected for the implementation of the i4Q Blockchain Infrastructure. In this deliverable, we describe the infrastructure and application built on top of Hyperledger Labs Orion, which is specifically geared towards increasing the quality and reliability of data in an industrial use case. The infrastructure we report upon tracks the changes made to the configuration of manufacturing machines, while using the advanced feature of the blockchain database. We thus gain the blockchain features of tamper-evidence, provenance, data lineage, authenticity, and non-repudiation. This deliverable describes the architecture, APIs, and implementation of said infrastructure, as well as some technical advances made to the Hyperledger Labs Orion server and SDKs.¹

¹ The abstract and some of the content in this document also appear in the i4Q project's web site, see: https://i4q.upv.es/3_i4Q_BC/index.html; by the same authors.

Document History

VERSION	ISSUE DATE	STAGE	DESCRIPTION	CONTRIBUTOR
0.1	07-Nov-2022	ToC	ToC Draft and first input	IBM
0.2	21-Nov-2022	Working Version	Input to all sections	IBM, BIBA
1.0	25-Nov-2022	1 st Draft	Available for Internal Review	IBM
1.1	30-Nov-2022	Internal Review	Internal Review	UPV, FACTOR
1.2	02-Dec-2022	2 nd Draft	Addressing comments from internal reviewers	IBM, BIBA
1.3	09-Dec-2022	Final Draft	Submission for final draft for quality check	IBM
1.4	30-Dec-2022	Final Document	Final quality check and issue of final document	CERTH

Disclaimer

Any dissemination of results reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.

Copyright message

© i4Q Consortium, 2022

This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both. Reproduction is authorised provided the source is acknowledged.



TABLE OF CONTENTS

Executive summary	8
Document structure	9
1. General Description	10
1.1 Overview	10
1.2 Blockchain DB Overview.....	10
1.3 Overview of Blockchain Technology for Machine Configuration Management	11
2. Technical Specifications.....	13
2.1 Machine Configuration and Configuration Management.....	13
2.2 High level architecture.....	14
2.3 APIs for Operators, Controllers & Auditors	17
2.3.1 APIs for Operators.....	17
2.3.1.1 List machines for which configurations can be initialized.....	17
2.3.1.2 Initialize configuration for a machine.....	18
2.3.1.3 Get a list of machines for which configuration can be changed.....	18
2.3.1.4 Get the current valid configuration for a machine	18
2.3.1.5 Propose request changes on machine configuration.....	19
2.3.1.6 Get a list of machines for which the change request of machine configuration is still pending.....	20
2.3.1.7 Get the pending configuration for a machine	20
2.3.2 APIs for Controllers	20
2.3.2.1 Get a list of machines for which there are pending change requests.....	20
2.3.2.2 Get the pending configuration for a machine	20
2.3.2.3 Evaluate and process change request of machine configuration	21
2.3.3 APIs for Auditors.....	21
2.3.3.1 Get a list of machines which are verifiable by a given auditor	21
2.3.3.2 Track the history of machine configurations	22
2.3.3.3 Verify the machine configuration in Blockchain	22
2.4 Deployment.....	22
2.5 Reference Architecture Diagram.....	27
2.6 Hyperledger Labs Orion Improvements.....	28
2.6.1 Java SDK: Building a database instance	29



2.6.2	Java SDK: Creating a user session	29
2.6.3	Java SDK: Creating and executing a transaction	30
2.6.4	Java SDK: Administration transactions.....	30
3.	Implementation Status	33
3.1	Current implementation	33
3.1.1	Solution features mapping with user requirements.....	33
3.1.2	The Machine Configuration Management solution	33
3.1.2.1	The frontend service.....	33
3.1.2.2	The machine configuration management service.....	34
3.1.3	The Hyperledger Orion client SDK.....	36
3.1.4	Demonstration	37
3.2	History	37
3.3	Next developments	38
4.	Conclusions	39
	Appendix I	40
	References.....	41

LIST OF FIGURES

Figure 1.	Three main elements of configuration management in the i4Q ^{BC} Solution	12
Figure 2.	Sample machine configuration.....	13
Figure 3.	Use Case Diagram - i4Q ^{BC} Machine Configuration Management.....	14
Figure 4.	High Level Solution Architecture - i4Q ^{BC} Machine Configuration Management	15
Figure 5.	Sample Sequence Diagram - Initialization of machine configuration.....	16
Figure 6.	Data model of response entry.....	17
Figure 7.	Data model of the machine configuration entity.....	19
Figure 8.	Sample deployment - i4Q ^{BC} Machine Configuration Management.....	23
Figure 9.	i4Q ^{RA} mapping with i4Q ^{BC}	28
Figure 10.	Orion Java SDK: Building a database instance	29
Figure 11.	Orion Java SDK: Creating a user session	29
Figure 12.	Orion Java SDK: Creating and executing a Data TX	30
Figure 13.	Orion Java SDK: database management operations.....	31
Figure 14.	Orion Java SDK: user management operations	31
Figure 15.	Orion Java SDK: cluster management operations	32
Figure 16.	Sample screenshot of the frontend service for operators.....	34
Figure 17.	Sample screenshot of the Swagger documentation	35
Figure 18.	Sample email notification on approval of change request for operators	35
Figure 19.	Sample Scenario for managing machine configurations.....	37



LIST OF TABLES

Table 1. Technical overview	36
-----------------------------------	----

ABBREVIATIONS/ACRONYMS

ACL	Access Control List
API	Application Programming Interface
BC	Blockchain
BCD	Blockchain Database
BFT	Byzantine Fault Tolerance
CA	Certification Authority
DB	Database
DBS	Database Management System
GA	Grant Agreement
GDP	General Data Protection Regulation
HLF	Hyperledger Fabric
IPR	Intellectual Property Rights
IT	Information Technology
JSON	JavaScript Object Notation
PDF	Portable Document Format
PoS	Proof of Stake
PoW	Proof of Work
REST	Representational State Transfer
SDK	Software Development Kit
SMT	Simple Mail Transfer Protocol
SQL	Structured Query Language
TX	Transaction
UC	Use Case



Executive summary

The purpose of this document is to report upon the machine-configuration tracking infrastructure that was built on the [i4Q](#) blockchain platform. The blockchain platform of the [i4Q](#) projects consists of the Hyperledger Orion blockchain database, which was presented and explained in the first deliverable of Task 3.2 - “Manufacturing Data Trustiness and Traceability”. That first deliverable (D3.3, [1]) described the design choices for selection of Hyperledger Labs Orion, its architecture, and APIs.

In this deliverable², we described the development of the machine-configuration tracking infrastructure. It takes advantage of the Blockchain (BC) technology, i.e., Hyperledger Orion blockchain database, to track the value changes of the machine configuration data and provide the capability of tracking data validation and verification as well as data auditing. It is based on microservice architecture and can be deployed as docker containers. It includes a frontend service providing web-based user interfaces, and a machine configuration management service to provide REST APIs. The machine configuration management service uses the Mongo DB and the Hyperledger Orion Server (Blockchain DB) for data management. It can be integrated with message brokers i.e., Kafka server, to exchange machine configuration data. This deliverable gives technical specifications as well as the current implementation and plans for further developments.

The machine configuration infrastructure required a Java client SDK in order to use Hyperledger Labs Orion. We therefore developed a Java SDK for that purpose and published it to open source. The Java SDK is in its first release and is still receiving active development efforts.

² This document [i4Q D3.11 v2](#) is an update of v1 of D3.3., for this reason it contains some information of the 1st version together with the updates developed in this 2nd version.



Document structure

Section 1: Contains a general description of the **i4Q^{BC}**, providing an overview and the list of features. It is addressed to final users of the **i4Q** Solution.

Section 2: Contains the technical specifications of the **i4Q^{BC}**, providing an overview and its architecture diagram. It is addressed to software developers.

Section 3: Details the implementation status of the **i4Q^{BC}**, explaining the current status, next steps and summarizing the implementation history.

Section 4: Provides the conclusions.

APPENDIX I: Provides the PDF version of the **i4Q Blockchain Traceability of Data** web documentation, which can be accessed online at: http://i4q.upv.es/3_i4Q_BC/index.html



1. General Description

1.1 Overview

Blockchain (BC) technology enables data traceability, which is a capability provided by i4Q^{BC} Solution. Blockchain provides the underlying technology for ensuring immutability, finality, and provenance contributing to non-repudiation of the stored state. i4Q^{BC} provides primitives within or on top of the base blockchain infrastructure to provide these capabilities in a smart manufacturing environment, for storing and querying manufacturing related data, configuration changes decisions and AI accountability.

In a previous report (D3.3) we explored different blockchain approaches and decided to establish the blockchain infrastructure on the Hyperledger Labs Orion blockchain database³. Hyperledger Labs Orion is a centralised, trusted blockchain database that provides tamper-evidence, provenance, data lineage, authenticity, and non-repudiation through data centric Application Programming Interfaces. In said report we also described the architecture, features, APIs, and deployment flow of Orion. We also provided a client SDK⁴ in the Go programming language to accompany the server. In order to demonstrate the capabilities of Hyperledger Labs Orion in an industrial setting, we released a short demo of machine configuration tracking and management, implemented with the Go SDK⁵.

In this deliverable, we described the development of the machine-configuration tracking infrastructure. It takes advantage of the Blockchain (BC) technology, i.e., Hyperledger Orion blockchain database, to track the value changes of the machine configuration data and provide the capability of tracking data validation and verification as well as data auditing. It is based on microservice architecture and can be deployed as docker containers. It includes a frontend service providing web-based user interfaces, and a machine configuration management service to provide REST APIs. The machine configuration management service uses the Mongo DB and the Hyperledger Orion Server (Blockchain DB) for data management. It can be integrated with message brokers i.e., Kafka server, to exchange machine configuration data. The following sections in this deliverable give more details about the technical specifications as well as the current implementation and plans for further developments.

1.2 Blockchain DB Overview

Hyperledger Labs Orion (a.k.a. BCDB for BlockChain DataBase) is a key-value/document database that provide some blockchain properties such as tamper evident data structures, non-repudiation through digital signatures, and provenance queries. BCDB provides traditional DBMS features such as data-oriented APIs, transactions with serialisation isolation level, crypto-based authentication, confidentiality, and access control. BCDB is a replicated system, running across a cluster of servers which execute a consensus protocol. This protocol tries to provide high

³ Hyperledger Labs Orion blockchain DB: <https://github.com/hyperledger-labs/orion-server>

⁴ Hyperledger Labs Orion Go client SDK: <https://github.com/hyperledger-labs/orion-sdk-go>

⁵ Machine configuration demo, in Go: <https://github.com/orion-bcdb/i4Q-demo-go>



availability and fault tolerance while ensuring the consistency of the underlying ledger. BCDB does not support the decentralisation of trust, as it is governed by a single organization. Clients, however, do not necessarily have to trust each other, and establish trust with the blockchain mechanisms that BCDB provides. Moreover, BCDB does not need to be blindly trusted by the clients, since some of the crypto-based data structures that BCDB provides allow clients to verify that the BCDB system was not compromised. Therefore, BCDB is not a Decentralised Distributed Ledger but rather a Ledger DBMS. BCDB does not support smart contracts; those pieces of code are executed on the client side. However, a unique mechanism of multi-signature transactions allows users to engage in multiparty transactions, allowing multiple users to agree and endorse a transaction candidate.

Hyperledger Labs Orion is an open-source project maintained by IBM and is part of the Hyperledger Labs open-source community⁶, that is under the Linux Foundation⁷. The IBM team is striving to get Orion further accepted as a main-line Hyperledger project.

1.3 Overview of Blockchain Technology for Machine Configuration Management

Data lineage or data traceability is a topic that is becoming important in recent years. Although the concept of data lineage and data traceability are quite similar, Allen and Cuervo provide a summary of the differences between them: „Data lineage states where data is coming from, where it is going, and what transformations are applied to it as it flows through multiple processes. It helps understand the data life cycle. Data traceability is the actual exercise to track access, values, and changes to the data as they flow through their lineage. Data traceability can be used for data validation and verification as well as data auditing. In summary, data lineage is the documentation of the data life cycle, while data traceability is the process of evaluating that the data is following its life cycle as expected. ...” [2]. In the i4Q project, the solution is more related to the data traceability. It takes advantage of the Blockchain (BC) technology to track the values of data changes and provide the capability of tracking data validation and verification as well as data auditing. This deliverable takes the example of machine configuration management to demonstrate the capability i4Q^{BC} Solution.

By employing blockchain technology for configuration management, the i4Q^{BC} Solution helps secure management of machine configurations, which ensures: (i) immutability, non-repudiation, and auditability of the configuration update history; and (ii) integrity and consistency of stored configuration information. With the mentioned features, it can bring several benefits for companies, for example, (i) track, control and audit of configuration changes; (ii) help stakeholder (e.g. producers of machine) to check the compliance of machine usage i.e. machine configuration with laws and guidelines; (iii) audit and validate the machine configuration i.e. machine usage to

⁶ Hyperledger: <https://www.hyperledger.org/>

⁷ The Linux Foundation: <https://www.linuxfoundation.org/>

during maintenance and warranty cases (iv) linking to the working context data, such as the quality data of the produced product component, to find optimal machine configurations.

The i4Q^{BC} Solution provides mainly the following three main elements to facilitate the blockchain-technology-based configuration management, as shown in the Figure 1:

- Configuration Identification & Initialization. It creates a baseline for the machine configuration and supports users to initialize the configuration values for a machine.
- Configuration Control & Change Management. It includes the process of evaluating all change proposals on a machine configuration. It will lead to the approval or disapproval of a machine configuration.
- Configuration Verification & Audit. It provides an independent review of the historical of changes in the machine configuration. It allows auditors to verify that the machine configuration complies with guidelines.

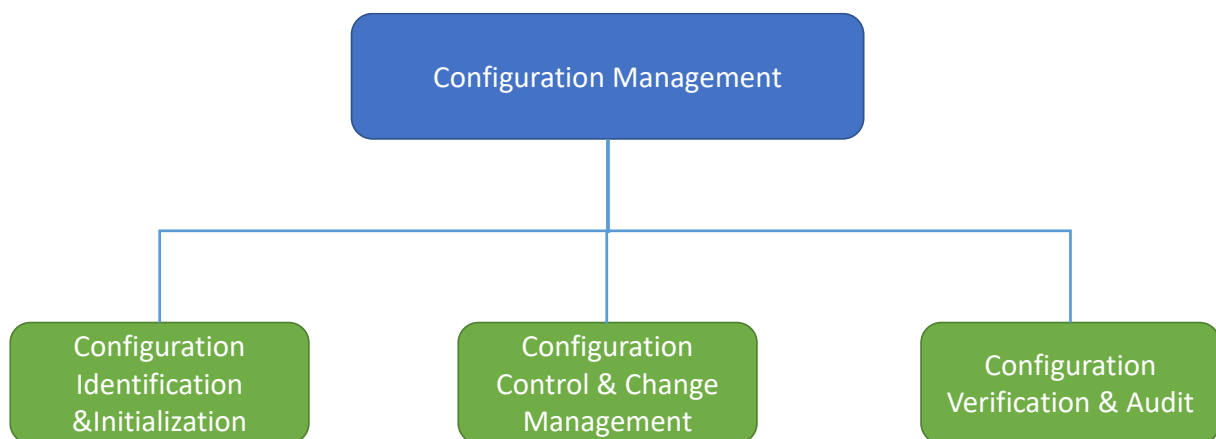


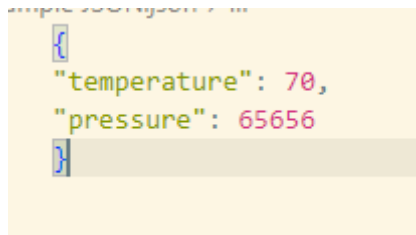
Figure 1. Three main elements of configuration management in the i4Q^{BC} Solution

2. Technical Specifications

In this section we describe the machine configuration infrastructure built on top of Hyperledger Labs Orion. In addition, we report upon technical improvements to the Hyperledger Labs Orion server and SDKs.

2.1 Machine Configuration and Configuration Management

Industry is shifting from mass production to mass customization. The manufacturing industry goes beyond engineering and manufacturing of standard products to the production of customized product. ‘Mass Customization’ can be considered “the ‘holy grail’ of manufacturing in which products made are perfectly attuned to customers’ wants and needs” [3] . As a result, in manufacturing, the configuration values of machines would often be added, updated, or removed, according to the working context of the machine. For example, in processing different materials or products, the configuration values of a machine could be different. According to the working contexts of a machine, engineers would select or set configuration values for the machine. A sample machine configuration is shown in Figure 2. However, as the configuration becomes complex and working context of the machine changes, it is easy to lead to the problems of, for example, disorganized and non-optimal configuration. Configuration management “helps engineering teams build robust and stable systems through the use of tools that automatically manage and monitor updates to configuration data.” [4].



```

{
  "temperature": 70,
  "pressure": 65656
}

```

Figure 2. Sample machine configuration

The ISO standard, **ISO 10007**, “Quality management – Guidelines for configuration management” provides guidance on the use of configuration management within an organization [5]. According to the ISO standard:

- A configuration is an “interrelated functional and physical characteristics of a product or service as described in the configuration information.”
- a *configuration item* is an “entity within a configuration that satisfies an end use function”.
- Configuration management is a “management activity that applies technical and administrative direction over the life cycle of a product and service, its configuration identification and status, and related product and service configuration information.”

The configuration management “secures that product configurations and project documentation are transparent and traceable at any time during the engineering process of a complex system.” [6]. Following the overview and features described, Figure 3 gives a use case diagram of the machine configuration management in the i4Q^{BC} Solution. It describes the main stakeholders and scenarios that the i4Q^{BC} Solution will support.

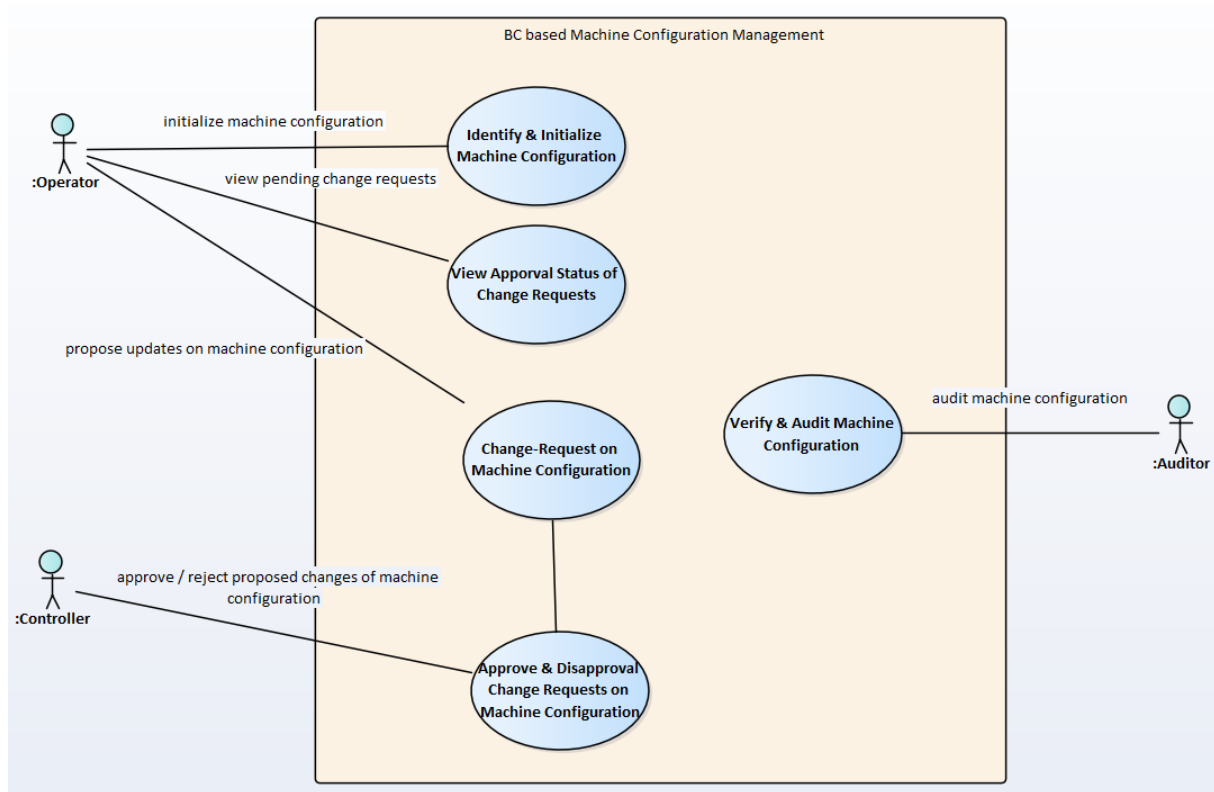


Figure 3. Use Case Diagram - i4Q^{BC} Machine Configuration Management

As shown in the use case diagram, it involves mainly three types of actors:

- Operator. A user who plans and identify the machines and machine configurations that need to be managed with blockchain technology. According to the plan, the user can then initialize the and request updates on machine configurations.
- Controller. A user who evaluates the update request on machine configurations. According to the evaluation, a change request can be approved or rejected.
- Auditor. A user who verifies and audit the historical of machine configurations.

In addition, it requires IT administrators to manage the system, for example, add new machines or users into the system.

2.2 High level architecture

Figure 4 shows the high-level solution architecture of the i4Q^{BC} Machine Configuration Management. It is based on microservice architecture and includes mainly six services & components: the frontend service, the machine configuration management service, the Mongo DB, the Hyperledger Orion Server (Blockchain DB), the Kafka server, and a SMTP server.

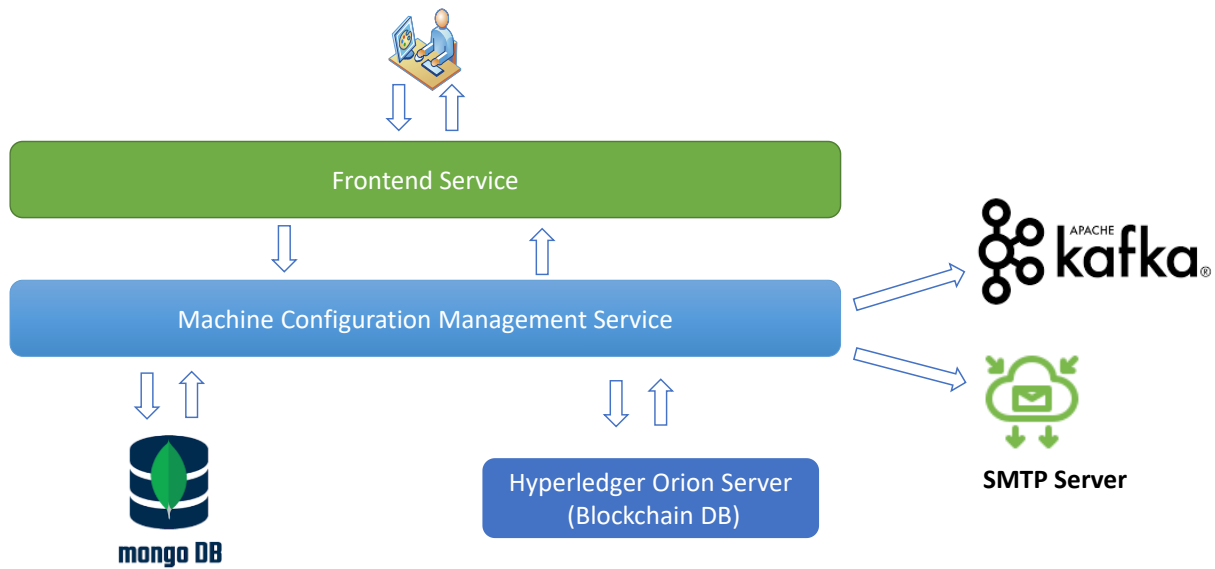


Figure 4. High Level Solution Architecture - i4Q^{BC} Machine Configuration Management

The frontend service provides web-based user interfaces to facilitate users to work with the configuration management system. The frontend service communicates with the machine configuration management service to handle the business logics behind. The machine configuration management service is one of the core services in the solution. It provides functionalities and APIs to manage the machine configurations. The machine configuration service uses a NoSQL database, i.e., Mongo DB, to manage all system relevant data, such as users, machines, and relevant configurations. Meanwhile, it keeps the history of machine configurations into the Blockchain database i.e., Hyperledger Orion Server. This ensures the immutability of the machine configurations and enable users to verify and audit the history records of machine configurations. In addition, the machine configuration service uses a SMTP server to send email notifications to users in case of different events, for example, when a change request of machine configuration is approved by a controller. These kind of event data, such as, change request of machine configuration, are also sent to defined Kafka topics. It allows other stakeholder and components, which have interest on the machine configurations, can consume the data.

Figure 5 provides a sample sequence diagram. It illustrates how the six components interact with each other when an operator initializes a machine configuration with the i4Q^{BC} Solution.

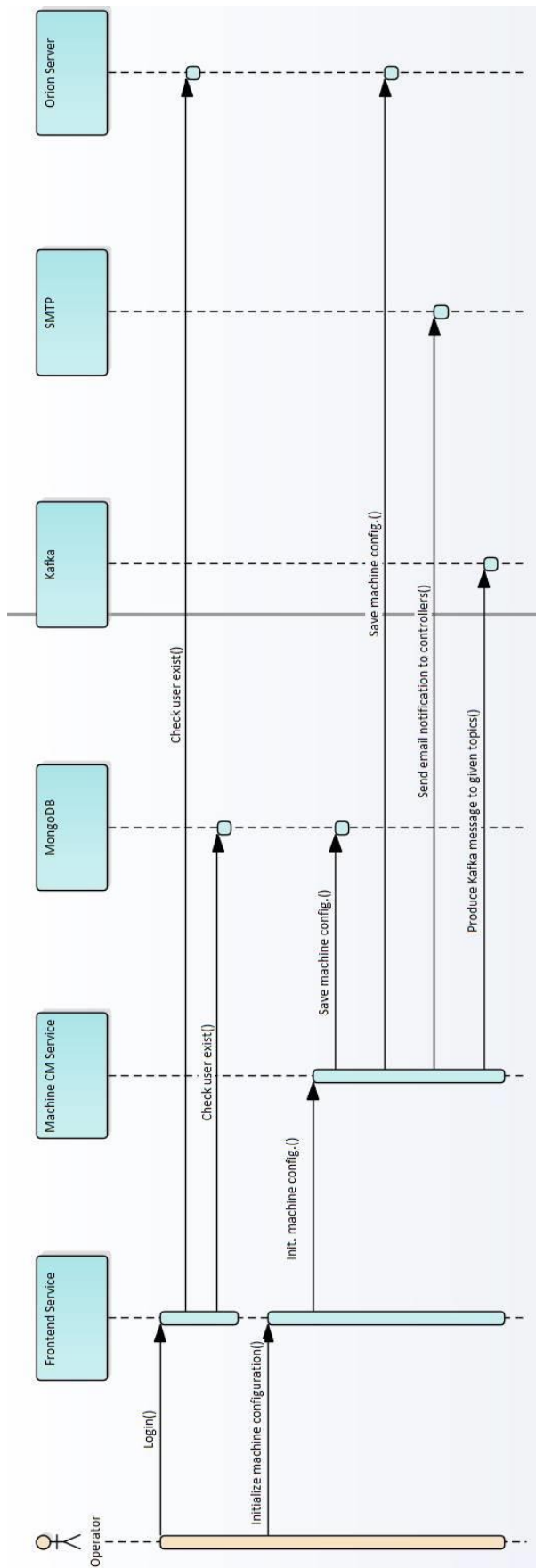


Figure 5. Sample Sequence Diagram - Initialization of machine configuration

2.3 APIs for Operators, Controllers & Auditors

Besides the provided user interfaces in the frontend service, REST APIs are provided to enable users or other component to interact with the i4Q^{BC} Machine Configuration Management. The machine configuration management service provides most of the REST APIs for users i.e., operators, controllers, and auditors. Before he communicates with the machine configuration management service with REST APIs, a user should first login and be authenticated with username i.e., email and password. Afterwards, the user can use an authentication token to issue a HTTP request. For the communication with the Blockchain DB i.e., Orion server cluster, the machine configuration management service manages the cryptographic material for each authorized user and will use the cryptographic material to add a digital signature and communicate with a client Software Development Kit (SDK) for JAVA. The following sub-sections provides short descriptions of the main APIs for operators, controllers, and auditors.

2.3.1 APIs for Operators

2.3.1.1 *List machines for which configurations can be initialized*

Operator calls this REST API, when he/she want to get a list of machines for which he/she is responsible for the initialization of the machine configurations.

To call the API, issue:

```
GET /machines/listMachineWithNoConfigs

Parameters

Authorization Authorization token

username Email of the operator
```

In case of success, it returns a list of machines. Figure 6 describes the data model of the response entity.

Example Value	Model
<pre> { auditors: [string] configs: [...] controllers: [string] current_valid_config_id: string machine_id: string model_name: string operators: [string] }</pre>	<pre> Machine { auditors: [string] configs: [...] controllers: [string] current_valid_config_id: string machine_id: string model_name: string operators: [string] }</pre>

Figure 6. Data model of response entry

2.3.1.2 Initialize configuration for a machine

This REST API is called, when an operator wants to initialize configuration for a machine. To call the API, issue:

PUT	<code>/machines/initMachineConfig /{operatorEmail}/{machineID}</code>
<i>Parameters</i>	
Authorization	Authorization token
operatorEmail	Email of the operator
machineID	Machine identifier
configValue	Configuration values in JSON format

This API will produce Kafka message to defined topics to inform that a machine configuration has been initialized. Meanwhile, it will manage the machine configuration in the Blockchain DB i.e., Orion server cluster.

2.3.1.3 Get a list of machines for which configuration can be changed

This REST API is called, when an operator wants to get a list of machines, for which the operator can propose changes on machine configuration. Before The machine configuration can be changed by the operator, only when the following pre-conditions are met: 1) the operator has permission to propose changes; and 2) the machine has a valid configuration, and there are no pending change requests of the machine configuration. To call the API, issue:

GET	<code>/machines/listMachineWithNoConfigs</code>
<i>Parameters</i>	
Authorization	Authorization token
username	Email of the operator

2.3.1.4 Get the current valid configuration for a machine

This REST API is called, when an operator needs to get the current valid configuration for a machine. To call the API, issue:

GET	<code>/machines/getCurrentMachineConfig /{machineID}</code>
<i>Parameters</i>	
Authorization	Authorization token
machineID	Machine identifier

It returns a machine configuration entity, from which the configuration values in JSON can be retrieved. Figure 7 describes the data model of the machine configuration entity.

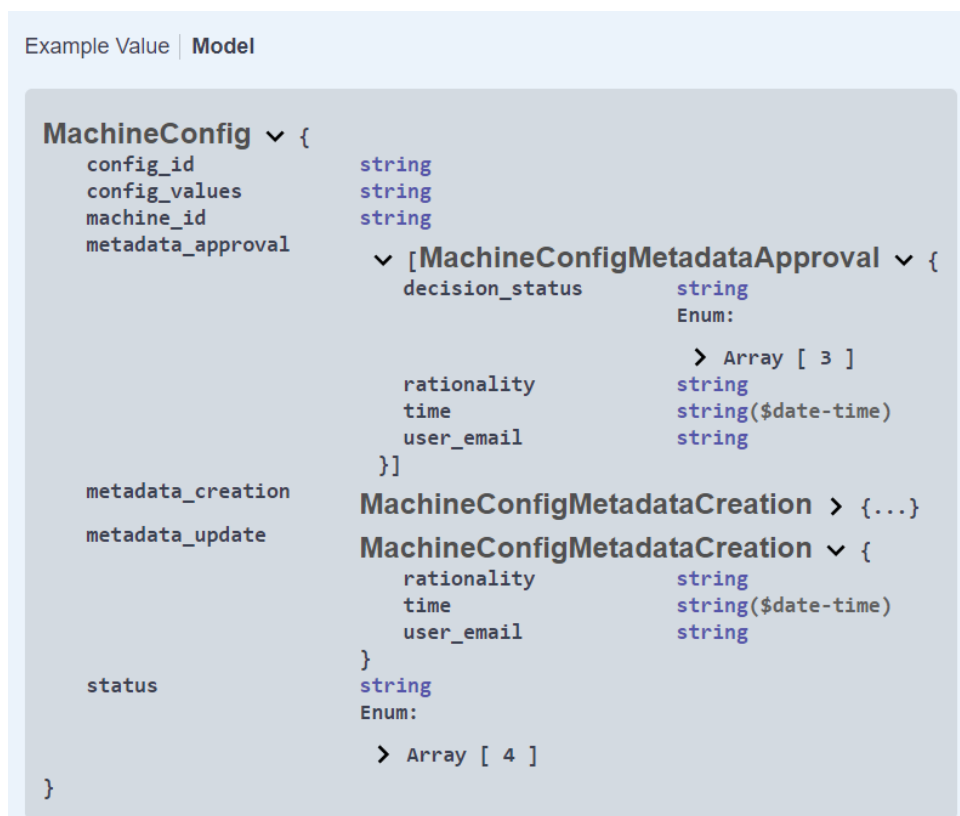


Figure 7. Data model of the machine configuration entity.

2.3.1.5 Propose request changes on machine configuration

This REST API is called, when operator want to propose changes on machine configuration for a machine. To call the API, issue:

PUT	/machines/requestUpdateMachineConfig/{operatorEmail}/{machineID}
<i>Parameters</i>	
Authorization	Authorization token
operatorEmail	Email of the operator
machineID	Machine identifier
configValue	Configuration values in JSON formate

This API will produce Kafka message to defined topics to inform that there is a change request on the machine configuration. Meanwhile, it will send email notifications to all controllers to inform them about the change request.



2.3.1.6 Get a list of machines for which the change request of machine configuration is still pending

This REST API is called, when an operator wants to get a list of machines, for which the change request of machine configuration is still pending. The operator can afterwards check the approval status of the changes request for these machines. To call the API, issue:

GET /machines/listMachineWithPendingConfig

Parameters

Authorization Authorization token

username Email of the operator

2.3.1.7 Get the pending configuration for a machine

This REST API is called, when operator want to check the approval status of the changes request of machine configuration for a machine. To call the API, issue:

GET /machines/getPendingMachineConfig/{machineID}

Parameters

Authorization Authorization token

machineID Machine identifier

It returns a machine configuration entity, from which the approval status of the change request of machine configuration can be retrieved. It will list the approval decision made by each controller.

2.3.2 APIs for Controllers

2.3.2.1 Get a list of machines for which there are pending change requests

This REST API is called, when controller want to get list of machines for which there are pending change requests. It will return only machines that the given controller is responsible and has permission to approve & reject the change requests. To call the API, issue:

GET /machines/listMachineWithValidAndToBeApprovedConfigs

Parameters

Authorization Authorization token

username Email of the controller

2.3.2.2 Get the pending configuration for a machine

This REST API is called, when controller want to evaluate the changes request of machine configuration for a machine. To call the API, issue:



GET	<code>/machines/getPendingMachineConfig/{machineID}</code>
<i>Parameters</i>	
Authorization	Authorization token
machineID	Machine identifier

It returns a machine configuration entity, from which the change request of machine configuration can be retrieved.

2.3.2.3 Evaluate and process change request of machine configuration

This REST API is called, when controller want to evaluate and process the change request of machine configuration. To call the API, issue:

PUT	<code>/machines/processUpdateRequestOnMachineConfig/{controllerEmail}/{machineID}</code>
<i>Parameters</i>	
Authorization	Authorization token
controllerEmail	Email of the controller
machineID	Machine identifier
decision	Approval decision for the change request. It can be approved or rejected.

Only when all controllers have approved the machine configuration, then the configuration is marked as approved. In case of a rejection by one of the assigned controllers, the change request will be marked as rejected. The decision by controller will be sent per email to notify relevant operators. It will produce a Kafka message to inform others about the decisions. In addition, the approved machine configuration will be kept and managed in the Orion Server cluster.

2.3.3 APIs for Auditors

2.3.3.1 Get a list of machines which are verifiable by a given auditor

This REST API is called, when auditor needs to track the history of machine configurations and want to audit the machine configurations. It returns a list of machines, which the given auditor has permission to check. To call the API, issue:

GET	<code>/machines/listAuditableMachines</code>
<i>Parameters</i>	
Authorization	Authorization token
username	Email of the auditor

2.3.3.2 Track the history of machine configurations

This REST API helps auditors to track the full history of the machine configurations for a machine. It returns a list of machine configurations entities covering all historical actions performed by operators and controllers. For example, who at which time proposes change request of machine configuration for the machine, and who approved or rejected the change request at which time. To call the API, issue:

GET	<code>/machines/trackMachineConfigs</code>
<i>Parameters</i>	
Authorization	Authorization token
machineID	Machine identifier

2.3.3.3 Verify the machine configuration in Blockchain

This REST API uses the records in the Blockchain DB i.e., Orion Server Cluster to verify the latest machine configuration of a given machine. To call the API, issue:

GET	<code>/machines/verifyLatestMachineConfigInBC</code>
<i>Parameters</i>	
Authorization	Authorization token
machineID	Machine identifier

2.4 Deployment

The deployment is based on Docker. That means, each microservice will be transformed into a Docker container and uploaded to a Docker registry. The configuration of a microservice uses environmental variables to enable the customization of a microservice to a new IT-infrastructure. Figure 8 gives a sample deployment of the **i4Q^{BC}** Machine Configuration Management. It currently uses BIBA server for the deployment. In later stages, the deployment can be switched to a **i4Q** server, and uses the message broker i.e., Kafka provided by **i4Q** to share the information about machine configurations.

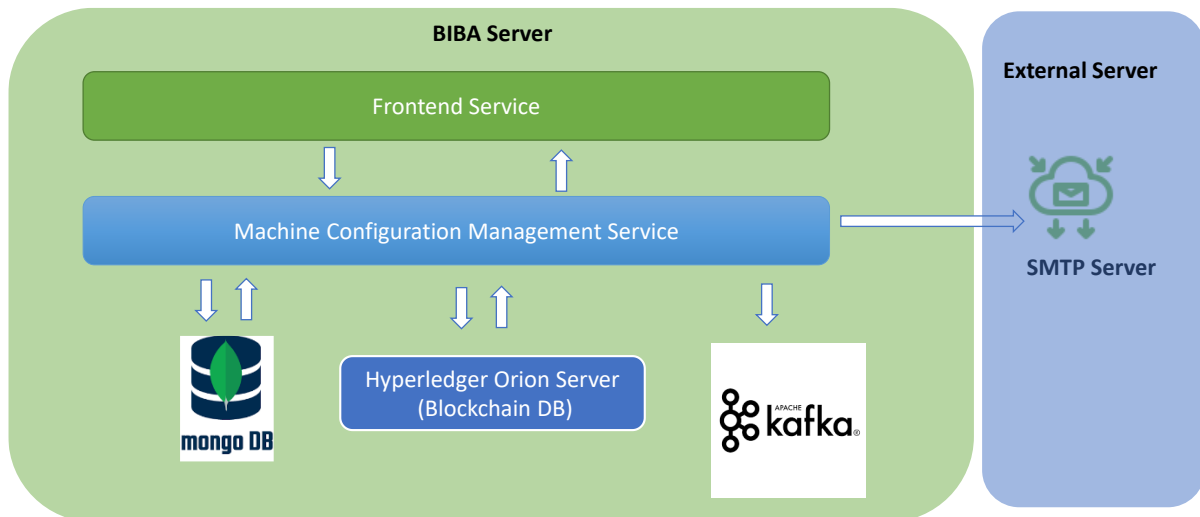


Figure 8. Sample deployment - i4Q^{BC} Machine Configuration Management

For that deployment a Docker-compose file is created summarising the microservice and the values of the needed environmental variables.

```
version: '2'

services:

#####

#### mc-management-service ####

#####

mc-management-service:

  image: reg.biba.uni-bremen.de/i4q/mc-management-service:latest

  environment:

    - backendHost=i4q-mc.ikap.biba.uni-bremen.de/mc-management-service
    - mongo_url=...
    - email_enable_notification=true
    - kafka_enable_notification=true
    - orion_server_enable=true
    - orion_crypto_folder=/etc/orion-server/crypto
    - orion_srv_endpoint=http://i4q-orion-server:6001
```

- *orion_srv_id=orion-server1*
- *machine_config_db=machine*
- *mc_platform_url=https://i4q-mc.ikap.biba.uni-bremen.de/*
- *kafka_topic=bc-mc-management*
- *kafka_host=i4q-kafka:9092*
- *adminToken=...*
- *orionAdminToken=...*
- *userToken=...*
- *notification_mail_password=...*

volumes:

- *./orion-server/crypto:/etc/orion-server/crypto*

ports:

- *"8713:8080"*

networks:

- *infra*

#####

i4q-mc-management-db:

image: mongo:4.0

environment:

- *MONGO_INITDB_ROOT_USERNAME=...*
- *MONGO_INITDB_ROOT_PASSWORD=...*

restart: on-failure

ports:

- *27014:27017*

volumes:

- *i4q_mongodbd_data_server:/data/db*

networks:

- *infra*

```
#####

#### i4q-mc-frontend ##

#####

i4q-mc-frontend:

  image: reg.biba.uni-bremen.de/i4q/mc-frontend:latest

  ports:

    - "8714:80"

  networks:

    - infra

#####

#### i4q-orion-server ##

#####

i4q-orion-server:

  image: orionbcdb/orion-server:latest

  volumes:

    - ./orion-server/crypto:/etc/orion-server/crypto
    - ./orion-server/config:/etc/orion-server/config
    - ./orion-server/ledger:/var/orion-server/ledger

  ports:

    - "6001:6001"
    - "7050:7050"

  networks:

    - infra

i4q-zookeeper:
```

image: confluentinc/cp-zookeeper:latest

environment:

ZOOKEEPER_CLIENT_PORT: 2181

ZOOKEEPER_TICK_TIME: 2000

restart: always

volumes:

- i4q_zk_data:/var/lib/zookeeper/data

- i4q_zk_log:/var/lib/zookeeper/log

ports:

- 22181:2181

networks:

- infra

i4q-kafka:

image: confluentinc/cp-kafka:latest

depends_on:

- i4q-zookeeper

ports:

- 29092:29092

restart: always

environment:

KAFKA_BROKER_ID: 1

KAFKA_ZOOKEEPER_CONNECT: i4q-zookeeper:2181

KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://i4q-kafka:9092,PLAINTEXT_HOST://agilevt-dev.ikap.biba.uni-bremen.de:29092

KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT

KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT

KAFKA_OFFSETS_TOPIC_REPLICATION_FACTOR: 1

volumes:

```
- i4q_kafka_data:/var/lib/kafka/data

networks:

- infra

volumes:

i4q_mongodbddata_server:

i4q_kafka_data:

i4q_zk_data:

i4q_zk_log:

networks:

infra:

external:

name: i4q_infra_dev_default
```

2.5 Reference Architecture Diagram

Figure 9 presents the mapping of the *i4Q^{BC}* solution in the reference architecture of *i4Q* (see D2.7 0). The Orion database itself is most likely to be deployed in the platform tier, either on premises or in the cloud, as a service. The frontend service and machine configuration management service in the machine configuration infrastructure can also be deployed with docker images in the platform tier, as micro-services. The machine configuration management service shall connect with the “Data Brokering and storage” or Kafka servers to allow various other components to access the machine configurations as well as various events e.g., changes on the machine configurations. For example, the “Data Analytics and Services” may be able to link the machine configuration data with product quality data to find optimal machine configurations. It can also be connected with the Edge Tier to enable the flow of machine configuration from/to the machine tools.

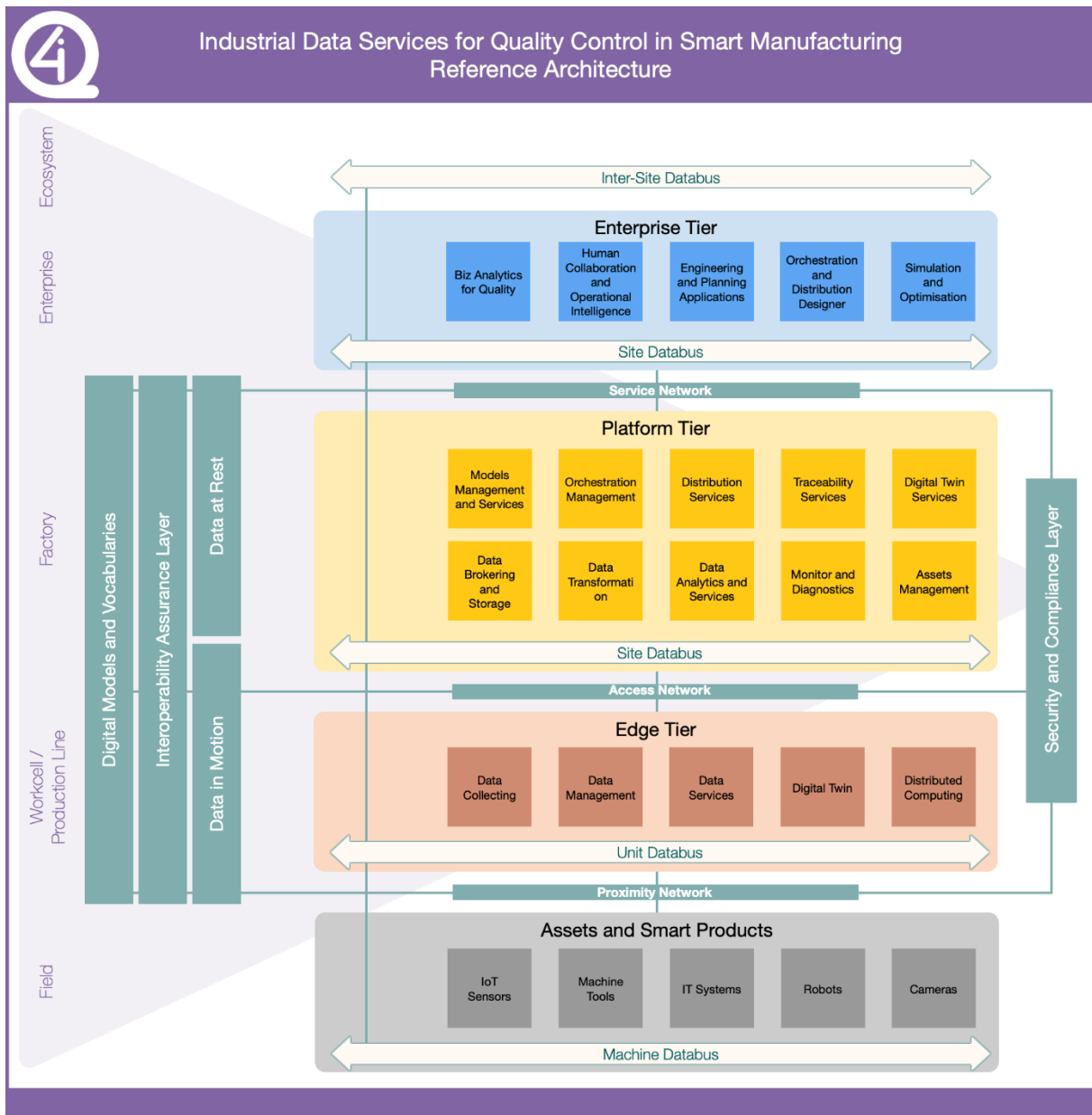


Figure 9. i4Q^{RA} mapping with i4Q^{BC}

2.6 Hyperledger Labs Orion Improvements

The Hyperledger Labs Orion server saw continued development since the last deliverable. The main items worked at were performance improvements, bug fixes, and some changes in wire format to support SDKs in different languages other the existing Go SDK. In particular, we developed a Java SDK in order to support the development of the machine configuration infrastructure reported above. These improvements resulted in a new version to the server (v0.2.6), and the Java SDK (v0.1.0).

The server code base is as open source available in:

<https://github.com/hyperledger-labs/orion-server>

and the documentation is available here:

<http://labs.hyperledger.org/orion-server/>

The server's architecture, features and APIs were covered in depth in the previous deliverable (D3.3, 0).

The Java SDK was published to open source as well, and is available here:

<https://github.com/orion-bcdb/orion-sdk-java>

The API of the Java SDK largely follows the design principles behind the Go SDK⁸. Here is a brief description of the Java API. The full documentation can be found in the Javadoc attached to the source code⁹. In addition, the `test` folder contains multiple examples.

2.6.1 Java SDK: Building a database instance

The database object `DB` contains the endpoints of the database cluster replicas as well as additional connection-oriented configuration parameters, which are all included in the `ConnectionConfig` object.

```
Replica r = new Replica("node1", "http://127.0.0.1:6001");
ConnectionConfig cConfig = new ConnectionConfig(new Replica[] { r }, null);
DB db = new DB(cConfig);
```

Figure 10. Orion Java SDK: Building a database instance

The object `DB` object is used in order to create a user session to the specified cluster.

2.6.2 Java SDK: Creating a user session

The `Session` object encapsulated the credentials of a particular user as well as additional connection-oriented parameters (e.g., time-out intervals). To create a session, user credentials (private key & certificate) are loaded onto a `UserConfig` object, which is then used to create the sessions configuration object `SessionConfig`.

```
String adminCertFilePath = "src/test/resources/crypto/admin/admin.pem";
String adminKeyFilePath = "src/test/resources/crypto/admin/admin.key";

UserConfig user = new UserConfig("admin", adminCertFilePath, adminKeyFilePath);
SessionConfig sessionConfig = new SessionConfig(user, Duration.ofSeconds(5), Duration.ofSeconds(5));
DBSession dbSession = db.session(sessionConfig);
```

Figure 11. Orion Java SDK: Creating a user session

⁸ See: <https://github.com/hyperledger-labs/orion-sdk-go>

⁹ See: <https://github.com/orion-bcdb/orion-sdk-java>



When a session is created, it connects to the cluster for the first time, fetching the most updated cluster configuration information. For example, it will fetch all the cluster's endpoint addresses even if the database instance was created using a stale set of endpoints.

The ``Session`` object allows the user to start a transaction context.

2.6.3 Java SDK: Creating and executing a transaction

Orion provides four types of transactions: data, user-administration, database-administration, and cluster-administration. Data transactions are useful to regular users manipulating data, whereas the other three types are for administrators. To create a data transaction context ``DataTxContext`` one invokes the ``createDataTx`` method. This object essentially build the read-set and write-set that are later submitted to the server. This object allows the user to read data from the cluster and write data locally to the write set. When the user has finished building the transaction, he calls ``Commit``.

```
DataTxContext tx = session.createDataTx();  
ValueWithMetadata v = tx.get("bdb", "key1");  
var value = v.getValue().toStringUtf8();  
value = value + "appendedvalue";  
tx.put("bdb", "key1", value.getBytes(), null);  
var receipt = tx.commit(true);
```

Figure 12. Orion Java SDK: Creating and executing a Data TX

That call marshals the read-set and write-set into payload bytes, signs the payload using the user's credentials in the session, and finally submits the signed transaction envelope to the server. The submission can be either synchronous or asynchronous.

When the server receives a TX envelope, it will validate the user's signature, validate the transaction's read-set versus the current state-DB, and finally commit the TX. With a synchronous invocation, the server responds with a "receipt", which contains the TX location in the ledger and its validation status, all signed by the server.

2.6.4 Java SDK: Administration transactions

The three types of administration transactions allow admin users to manage databases, users, and the cluster.

Database management transactions are used to create, delete, and query the existence of a database table.

```
/**
 * DBsTxContext defines the methods available on a database administration
 * transaction.
 *
 */
interface DBsTxContext extends TxContext {

    * createDB creates new database along with index definition for the query.[]
    void createDB(String dbName, HashMap<String, IndexAttributeType> index) throws Exception;

    * deleteDB deletes database.[]
    void deleteDB(String dbName) throws Exception;

    * exists checks whenever database is already created.[]
    boolean exists(String dbName) throws Exception;

    * getDBIndex returns the index definition associated with the given database.[]
    HashMap<String, IndexAttributeType> getDBIndex(String dbName) throws Exception;
}
```

Figure 13. Orion Java SDK: database management operations

User management transactions are used to add, update, delete, and get the credentials of users.

```
/**
 * UsersTxContext transaction context to operate with user management related
 * transactions.
 *
 */
interface UsersTxContext extends TxContext {

    * PutUser introduce new user into database.[]
    void putUser(User user, AccessControl acl) throws Exception;

    * GetUser obtain user's record from database.[]
    User getUser(String userID) throws Exception;

    * RemoveUser delete existing user from the database.[]
    void removeUser(String userID) throws Exception;
}
```

Figure 14. Orion Java SDK: user management operations

Cluster configuration transactions are used to add or remove cluster nodes, manage the parameters of consensus, manage certificate authorities, and admin users.

```
/**
 * ConfigTxContext transaction context to operate with cluster configuration
 * management related transactions.
 *
 */
public interface ConfigTxContext extends TxContext {

    * getClusterConfig returns the current cluster config.[]
    ClusterConfig getClusterConfig() throws Exception;

    * setClusterConfig sets a new cluster config object.[]
    void setClusterConfig(ClusterConfig newConfig) throws Exception;
}
```

Figure 15. Orion Java SDK: cluster management operations

Admin transactions can be executed by admin users only and are recorded in the ledger like any other transaction. This makes admin actions and config changes traceable just like any other piece of data.

3. Implementation Status

3.1 Current implementation

The current implementation includes three components:

- A machine configuration management service within the Machine Configuration Management solution
- A frontend service within the Machine Configuration Management solution
- A Hyperledger Orion client SDK in JAVA.

They are built on based on different technologies, such as, non-SQL database, Kafka, and Angular, and Orion client SDK in JAVA. The frontend service and machine configuration management service are available currently within a Gitlab repository within BIBA and can be shared in future. The Hyperledger Orion client SDK in JAVA is open source in GitHub repository.

3.1.1 Solution features mapping with user requirements

The Deliverable 3.3 describes how the **i4Q^{BC}** Solution address the requirements defined for the **i4Q-BC** solution. The machine configuration management solution takes advantages of the developed Blockchain database i.e., Orion server, to show the achievement of:

- enhance the level of trust in the platform by employing a blockchain based data service. The blockchain features presented by Orion are immutability, tamper evidence, authenticity, and non-repudiation. The machine configuration management solution uses the Orion server to manage the machine configurations and uses the immutable records in the Orion server for verification and enhance the trust in the managed machine configurations.
- support data traceability in the data that flows directly to the blockchain. The Orion server exposing data centric REST APIs, which allow the user to submit TXs and execute queries. In addition, it has provenance API for exploring data lineage. The machine configuration management service uses Orion server and MongoDB to manages all transactions about the initialization of machine configurations, as well as the change requests and approval decisions of all controllers. This allows auditors to track and verify all changes on the machine configurations.

Thus, machine configuration management service with the usage of the Hyperledger Orion can serve as a single point of truth, preserving provenance and supporting non-repudiation for the machine configurations.

3.1.2 The Machine Configuration Management solution

3.1.2.1 The frontend service

The frontend service is implemented based on a TypeScript-based free and open-source web application framework, Angular. The source code is current available in a Gitlab repository within BIBA server. The docker image of the frontend service is published in a docker registry within BIBA server. The material, including source codes and docker images etc., will be shared in future.

Meanwhile, it has been deployed as a docker container within a BIBA server and is accessible under <https://i4q-mc.ikap.biba.uni-bremen.de/>. Figure 16 gives a sample screenshot of the frontend service for operators. The usability and layout of the frontend service is still in improvement.

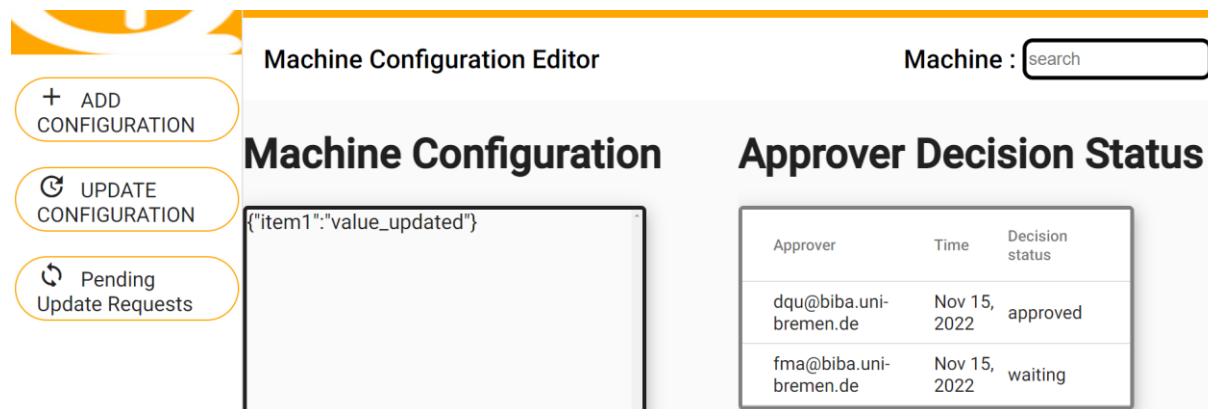


Figure 16. Sample screenshot of the frontend service for operators

3.1.2.2 The machine configuration management service

The machine configuration management service is implemented with JAVA. Similar as the frontend service, the source codes and the docker image is currently available within repositories within BIBA server. And these materials can be shared in a future time. Meanwhile, it has also been deployed as a docker container within a BIBA server and is accessible under <https://i4q-mc.ikap.biba.uni-bremen.de/mc-management-service>.

In addition, the REST APIs are documented with Swagger documentation, and is accessible under: <https://i4q-mc.ikap.biba.uni-bremen.de/mc-management-service/swagger-ui/index.html#>.

Besides the REST APIs introduced in the last chapter, it has many additional REST APIs to support, for example, the administration of users for the machine configuration management solution, and databases in the Blockchain database i.e., Orion Server. The swagger documentation allows users to try and understand the APIs before real usage in their own applications. Figure 17 provides a sample screenshot of the swagger documentation. Figure 18 gives a sample email notification on approval of change request of machine configurations for operators.



Entities API ^{0.1}

[Base URL: i4q-mc.ikap.biba.uni-bremen.de/mc-management-service/]
<https://i4q-mc.ikap.biba.uni-bremen.de/mc-management-service/api-docs>

This API provides access to Entity data.

admin-controller Admin Controller

machine-controller Machine Controller

GET /machines/getCurrentMachineConfig/{machineID} getCurrentMachineConfig

GET /machines/getPendingMachineConfig/{machineID} getPendingMachineConfig

PUT /machines/initMachineConfig/{operatorEmail}/{machineID} initMachineConfig

GET /machines/listAuditableMachines listAuditableMachines

GET /machines/listMachineWithNoConfigs listMachineWithNoConfigs

GET /machines/listMachineWithPendingConfig listMachineWithPendingConfig

GET /machines/listMachineWithValidAndNoPendingConfigs listMachineWithValidAndNoPendingConfigs

GET /machines/listMachineWithValidAndToBeApprovedConfigs listMachineWithValidAndToBeApprovedConfigs

PUT /machines/processUpdateRequestOnMachineConfig/{controllerEmail}/{machineID} processUpdateRequestOnMachineConfig

PUT /machines/requestUpdateMachineConfig/{operatorEmail}/{machineID} requestUpdateMachineConfig

GET /machines/trackMachineConfigs trackMachineConfigs

GET /machines/verifyLatestMachineConfigInBC verifyLatestMachineConfigInBC

orion-admin-controller Orion Admin Controller

Figure 17. Sample screenshot of the Swagger documentation

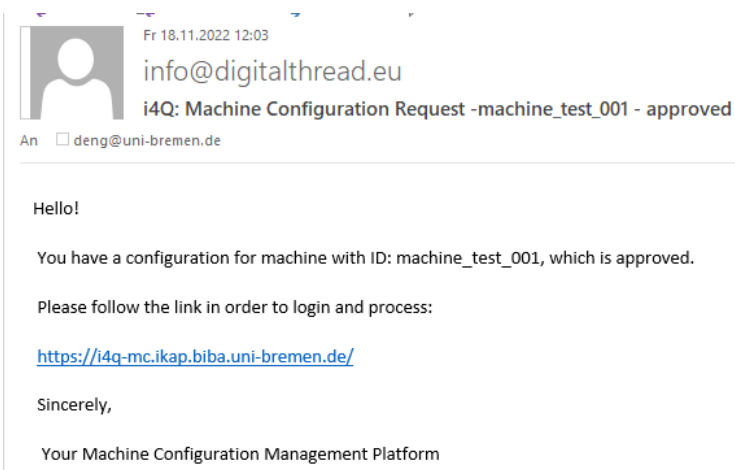


Figure 18. Sample email notification on approval of change request for operators



Table 1 gives a brief technical overview of the technical stake of the machine configuration management service.

Item	Value
Nature	Spring Service based on Spring 2.7.5
System requirements	Microsoft Windows, Linux
Programming Language	Java 17
Development Tools	Eclipse EE
Additional Libraries	JAXB, Spring-Kafka, Spring Data mongoDB, Lombok, Spring Web, Java Mail Sender, Orion client SDK Java, etc.
Databases	MongoDB, Blockchain DB (i.e., Orion Server Cluster)
I/O formats	REST using JSON for I/O, Kafka using JSON, Machine configuration managed in Orion Server in JSON

Table 1. Technical overview

The Orion Java SDK is still in an “alpha” release and is under active development. The machine configuration management service has dependencies on the SDK and will be improved accordingly when more features & APIs are provided the Orion Java SDK.

3.1.3 The Hyperledger Orion client SDK

The Hyperledger Orion SDK-Java implements a client SDK that exposes a rich and easy to use transactional database API. It is available as open source:

<https://github.com/orion-bcdb/orion-sdk-java>

The SDK Java project is operated with Gradle. To build a JAR file use:

```
./gradlew jar
```

The SDK is in an “alpha” release and is under active development. It now supports data, user administration, and database administration transactions. In the next few months, the IBM team will work to add to the Java SDK the features that already exist in the Go SDK¹⁰.

The client SDK can be used to easily integrate the Orion DB into any solution. It can be used in solutions that reside in the platform tier and/or in the edge tier of i4Q.

In addition to the work on the Orion Java SDK, the Orion Server¹¹ continued to be updated and maintained, and version v0.2.6 was released, with bug fixes, performance improvements, and compatibility with Java SDK.

¹⁰ See: <https://github.com/hyperledger-labs/orion-sdk-go>

¹¹ <https://github.com/hyperledger-labs/orion-server>

3.1.4 Demonstration

Based on the discussions with pilot partners, we have derived a sample application scenario to demonstrate the capabilities of Hyperledger Orion based machine configuration management solution. It supports the management of machine configurations in the blockchain database.

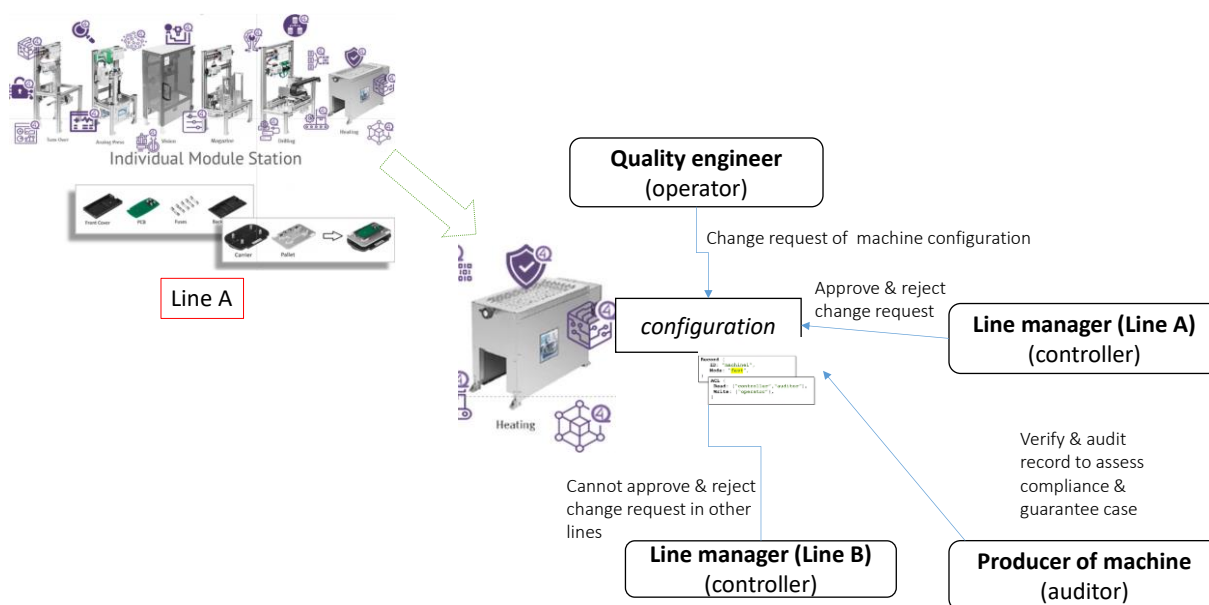


Figure 19. Sample Scenario for managing machine configurations

Figure 19 illustrates the sample scenario. It takes a machine from the production line for demonstration of the machine configuration management. In this scenario, the quality engineer, line manager, and producers of machine can play, respectively, the role operator, controller, and auditor. A quality manager (operator) proposes a change to the configuration of the machine. A line manager (controller) checks if the configuration is valid and should be applied to the machine and make decisions on approval or dis-approval. In case the configuration is applicable, then the change request of configuration is approved and saved in the Blockchain (Orion server). Later, the users, such as, machine producer or production manager, takes the role as auditor to review & audit the historical changes of the machine configuration. They can, for example, use the records to check the compliance of the machine usage, or link the machine configuration data with product quality data or warranty & maintenance cases to help data-driven decision making. Meanwhile, there are various permission controls, and only users with appropriate permission can approval change requests or audit history machine configurations. The demonstration is available under: <https://i4q-mc.ikap.biba.uni-bremen.de/>

3.2 History

Version	Release date	New features
Orion server v0.1.0	Jun. 24, 2021	API ready, single node
Orion server v0.2.3	Apr. 5, 2022	Cluster deployment, including docker

Version	Release date	New features
Orion server v0.2.6	Sep. 22, 2022	Bug fixes, performance, compatibility with Java SDK
Orion Java SDK v0.1.0	Nov. 20, 2022	Data, User, & DB-Admin APIs ready
Machine configuration management service v0.1.0	Nov. 20, 2022	REST APIs ready, with integration of Orion Java SDK v0.1.0
Frontend service v0.1.0	Nov. 25, 2022	Web-based user interfaces with functions for operators, controllers, and auditors

3.3 Next developments

As indicated above, the Orion Java SDK is in an “alpha” release and is under active development. It now supports data, user administration, and database administration transactions. In the next few months, the IBM team will work to add to the Java SDK the features that already exist in the Go SDK. That includes:

- Data transactions with multi-signature support
- Config transactions
- Query support – JSON queries and Range queries
- Provenance queries

As the Java SDK continues to be developed, the machine configuration management service will be improved to include new features provided by the Java SDK. The frontend service has necessary functions for operators, controllers, and auditors, but currently without good usability. The user interfaces are still under improvement.

In addition, the **i4Q^{BC}** solution will be continuously developed to cover pilot specific requirements, especially the generic pilot (T6.7). It will be adapted, integrated, validated, and demonstrate in WP6.

4. Conclusions

This focus of this deliverable is to take the example of machine configuration management to demonstrate the capability i4Q^{BC} Solution. It describes the development of the machine-configuration tracking infrastructure. The machine-configuration tracking infrastructure takes advantage of the Blockchain (BC) technology, i.e., Hyperledger Orion blockchain database, to track the value changes of the machine configuration data and provide the capability of tracking data validation and verification as well as data auditing. It demonstrates how operators, controllers, and auditors can use the i4Q^{BC} Solution to manage and track the machine configuration data. The machine-configuration tracking infrastructure provides web-based user interfaces and REST APIs, as well as the integration of message brokers i.e., Kafka server to facilitate the interaction with human and other components.

In order to enable the development of the machine configuration infrastructure, A Java client SDK was developed for the Hyperledger Labs Orion blockchain database.

As next step, the i4Q^{BC} Solution will be continuously developed and improved to handle pilot specific requirements. It will be further integrated, validated, and demonstrated in the frame of WP6.



Appendix I

The **i4Q Blockchain Traceability of Data** web documentation, which can be accessed online at:
http://i4q.upv.es/3_i4O_BC/index.html

References

- [1] i4Q deliverable D3.3 – Blockchain Traceability of Data.
- [2] Allen, Mark; Cervo, Dalton (2015): Data Quality Management. In : Multi-Domain Master Data Management: Elsevier, pp. 131–160.
- [3] Caddy, I.; Helou, M.; Callan, J. (2002): From Mass Production to Mass Customization: Impact on Integrated Supply Chains. In Claus Rautenstrauch, Ralph Seelmann-Eggebert, Klaus Turowski (Eds.): Moving into Mass Customization. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 59–70.
- [4] IAN BUCHANAN: Configuration management. How configuration management helps engineering teams build robust and stable systems. Available online at <https://www.atlassian.com/microservices/microservices-architecture/configuration-management#:~:text=Read%20more-,Configuration%20management%20helps%20engineering%20teams%20build%20robust%20and%20stable%20systems,granularity%20of%20size%20and%20complexity.,> checked on 11/23/2022.
- [5] ISO, 2017-03: ISO 10007:2017. Available online at <https://www.iso.org/standard/70400.html>, checked on 11/23/2022.
- [6] Müller, Patrick (2013): Configuration Management – A Core Competence for Successful through-life Systems Engineering and Engineering Services. In Procedia CIRP 11, pp. 187–192. DOI: 10.1016/j.procir.2013.07.032.
- [7] i4Q deliverable D2.7 – i4Q Reference Architecture and Viewpoints Analysis v2.