Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.sciencedirect.com/journal/computers-in-industry

A blockchain-based framework for trusted quality data sharing towards zero-defect manufacturing

Mauro Isaja [a], Phu Nguyen [b,*], Arda Goknil [b], Sagar Sen [b], Erik Johannes Husom [b], Simeon Tverdal [b], Abhilash Anand [c], Yunman Jiang [c], Karl John Pedersen [c], Per Myrseth [c], Jørgen Stang [c], Harris Niavis [d], Simon Pfeifhofer [e], Patrick Lamplmair [e]

[a] *Engineering Ingegneria Informatica SpA, Italy*
[b] *SINTEF, Norway*
[c] *DNV, Norway*
[d] *Inlecom Innovation, Greece*
[e] *Tributech Solutions GmbH, Austria*

ARTICLE INFO

Keywords:
Quality Hallmark
Quality services
Zero-waste value chain
Trusted data sharing
Zero-defect manufacturing
ZDM

ABSTRACT

There is a current wave of a new generation of digital solutions based on intelligent systems, hybrid digital twins and AI-driven optimization tools to assure quality in smart factories. Such digital solutions heavily depend on quality-related information within the supply chain business ecosystem to drive zero-waste value chains. To empower zero-waste value chain strategies with meaningful, reliable, and trustful data, there must be a solution for end-to-end industrial data traceability, trust, and security across multiple process chains or even inter-organizational supply chains. In this paper, we first present Product, Process, and Data quality services to drive zero-waste value chain strategies. Following this, we present the Trusted Framework (TF), which is a key enabler for the secure and effective sharing of quality-related information within the supply chain business ecosystem, and thus for quality optimization actions towards zero-defect manufacturing. The TF specification includes the data model and format of the Process/Product/Data (PPD) Quality Hallmark, the OpenAPI exposed to factory system and a comprehensive Identity Management layer, for secure horizontal- and vertical quality data integration. The PPD hallmark and the TF already address some of the industrial needs to have a trusted approach to share quality data between the different stakeholders of the production chain to empower zero-waste value chain strategies.

1. Introduction

The transformation of data into economic value for the many supply chain participants in accordance with the societal concerns facing Europe is the primary challenge of the digitalization process. In line with the European Green Deal[1] as well as the UN's Sustainable Development Goals (SDGs),[2] manufacturing businesses must limit the utilization (and particularly waste) of resources within entire industrial ecosystems, while consistently offering higher quality goods of growing complexity at reduced costs (Colledani et al., 2014; Powell et al., 2022a). In-line data gathering solutions, data storage and communication standards, data analytics tools, and digital manufacturing technologies are just a few of the key enabling technologies that are starting to open up new possibilities for Zero Defect Manufacturing (ZDM) (Eger et al., 2018) and help organizations to get even closer to realizing the goal of zero defects. Given the advent of Industry 4.0 and the maturing of its digital technologies, the ZDM paradigm has advanced dramatically in recent years. ZDM strategies base their operations on the collection of production and quality data from diverse sources and on the integration of that data with data from various levels of the plant (Magnanini et al., 2020). However, one of the gaps as pointed out by a recent study (Powell et al., 2022b) is the lack of study on ZDM being advanced in order to go from zero-defect procedures to zero-waste value chain strategies. Among the challenges for data analysis for ZDM is the scattered data in different formats, from different end users, servers, and organizations, especially without the link between process data and product quality. Such greater values come from sharing data (Jernigan et al., 2016) between stakeholders in the ecosystems. One of the biggest challenges

* Correspondence to: SINTEF, Forskningsveien 1, 0373 Oslo, Norway.
E-mail address: phu.nguyen@sintef.no (P. Nguyen).

[1] https://ec.europa.eu/reform-support/what-we-do/green-transition_en

[2] https://sdgs.un.org/goals

https://doi.org/10.1016/j.compind.2023.103853
Received 15 July 2022; Received in revised form 28 November 2022; Accepted 4 January 2023
Available online 19 January 2023

is ensuring security and enabling trust for data sharing (Atluri et al., 2020). To realize the vision of ZDM in the data-driven era of industry 4.0, it is important to develop technologies that enable the exchange of trustworthy and traceable quality data between factories (of the same or other enterprises) in the product supply chain.

Blockchain-based solutions to ensure trusthworthy data, such as the trustworthiness of sensor observations are blooming (Casado-Vara et al., 2018; Dedeoglu et al., 2019; Ahmed et al., 2021). From the definition of IBM (2022), a blockchain is a shared and immutable ledger. Blockchain technology is usually used for recording transactions, tracking assets, and building trust. However, to the best of our knowledge, there exists no solution that systematically supports the exchange of trustworthy and traceable quality data of product, process, and data itself in the product supply chain. In this paper, we proposes the use of the Product – Process – Data (PPD) quality hallmark and the distributed ledger-based Trusted Framework (TF) to overcome the aforementioned problems. The quality hallmark is the basic information unit of quality assessment, which carries all the quality-related information about a well-identified object, e.g., Product – Process – Data. We give an overview of PPD quality services and detail on the data quality services as a representative type of services that share quality data using the TF. The TF is a concrete service infrastructure that meets the generic requirements of traceability, trust and security for the specific goal of collaborative PPD Quality Hallmark management, for secure horizontal- and vertical quality data integration.

The paper is structured as follows. Section 2 discusses some related work. We present the background concepts of Distributed Ledger Technology and blockchain in Section 3. Our introduction of the quality hallmarks is given in Section 4. Then, Section 5 describes the PPD quality services. Next, we introduce the whole Trusted Framework in Section 6. In Section 7, we show a proof-of-concept on how to publish QHD containing the quality of data collected by the process services, product services, or data services. Finally, we give our conclusions and our thoughts on future work in Section 9.

2. Related work

Although considerable research has been devoted to ensuring data trustworthiness or secure IoT data sharing, little attention has been paid to blockchain solutions for trusted quality data sharing in Industry 4.0 applications. Most existing works (e.g., Bai et al., 2019; Tang et al., 2019; Rahman et al., 2020; Bartol et al., 2020; Patel and Shrimali, 2021; Manogaran et al., 2021; Ma et al., 2020; Lopez and Farooq, 2020; Shafagh et al., 2017,?; Shen et al., 2019; Bajoudah et al., 2019; Kang et al., 2018; Akkaoui et al., 2020; Nguyen et al., 2021) employ blockchain solutions to address secure data sharing but not the sharing of quality-related information in the context of Industrial IoT (specifically manufacturing) for zero-defect manufacturing. For instance, Shafagh et al. (2017) propose a blockchain-based access control management to provide decentralized, resilient, and auditable access control management for secure IoT data sharing. These works mostly address energy (Bartol et al., 2020), transportation (Ma et al., 2020; Lopez and Farooq, 2020; Kang et al., 2018), healthcare (Shen et al., 2019; Akkaoui et al., 2020; Nguyen et al., 2021), agriculture (Rahman et al., 2020; Patel and Shrimali, 2021), and manufacturing (Bai et al., 2019; Manogaran et al., 2021) domains.

We have identified three papers (Casado-Vara et al., 2018; Dedeoglu et al., 2019; Ahmed et al., 2021) that employ the blockchain to ensure trusthworthy data, such as the trustworthiness of sensor observations, while the blockchain is also a storage medium. Casado-Vara et al. (2018) studied incidents where malicious data may lead to poor data quality. They presented a blockchain-based architecture to improve data security, with an edge computing layer executing a new algorithm using game theory for false data detection.

Dedeoglu et al. (2019) proposed another layered architecture to improve the end-to-end trust for a diverse range of blockchain-based IoT applications. The proposed architecture evaluates the trustworthiness of sensor observations at the data layer and adapts block verification at the blockchain layer through the proposed data trust and gateway reputation modules. The performance of the proposed architecture has been evaluated using a simulated indoor target localization scenario.

Ahmed et al. (2021) proposed a blockchain solution using a logistic traceability smart contract to assess the data quality of IoT data sources. They employed four quality dimensions (i.e., accuracy, completeness, consistency, and currentness) and proposed their corresponding measurement methods. They also proposed a data quality model specific to the logistic chain domain and a distributed traceability architecture.

The three approaches discussed above are representative for blockchain-based solutions like our TF. However, none so far has systematically supported the exchange of trustworthy and traceable quality data of product, process, and data itself (PPD quality hallmark) in the product supply chain.

2.1. Summary

Table 1 summarizes the differences between our work and related work based on a set of features necessary for trusted quality data sharing towards zero-defect manufacturing. For each related work in the table, the symbol '✓' indicates that the work provides the feature, the symbol '✗' indicates that it does not provide the feature. For instance, Manogaran et al. (2021) propose a blockchain-assisted secure data-sharing model for IoT-driven smart manufacturing. It does not address sharing quality-related information (quality hallmark) towards zero-defect manufacturing. Most approaches employ blockchain solutions to address secure IoT data sharing, but not data quality-related information towards zero-defect manufacturing in the context of Industrial IoT. A few works propose blockchain solutions to ensure trustworthy data in the context of data quality management, but none of them systematically address the exchange of data quality information. To the best of our knowledge, TF is the only approach that enables, with support for quality hallmarks, the secure sharing of data quality-related information in Industrial IIoT towards zero-defect manufacturing.

3. Distributed ledger technology and blockchain

One of the most revolutionary developments in the realm of information technologies, distributed ledger technology (DLT), has the potential to transform how people collaborate and organize in the workplace, society, and the economy (Sunyaev, 2020). Generally speaking, a distributed ledger is a "system of record" that is replicated and kept in-sync across all the nodes of a network, where all nodes are peers and no special administrative/coordination privileges and/or master copy of the ledger exist anywhere. In – the most mature and popular to date – the ledger itself is a time series consisting of immutable and timestamped records, each record being the outcome of one or more transactions. Such linear sequence of records can only be modified by appending new records after its end. So far, this architecture is no different than what underpins traditional databases: the transaction log. However, the distributed and – most importantly – decentralized nature of the ledger requires two additional steps to be taken: firstly, records/transactions are only accepted into the ledger if a consensus is reached among the peer nodes of the network about their "correctness" (more on this later); then, once accepted, their integrity – both in terms of data payload and of ordering – is forever protected by means of strong cryptographic algorithms, to the effect that any copy of the ledger is equally authoritative, regardless of the peer node holding it. The bottom line is that all transactions are guaranteed to be correct and, once committed to the ledger, cannot be revoked, repudiated or simply deleted. Moreover, as the whole system is decentralized, bending it to the will of a single participant – e.g., overriding the commonly-agreed business rules, changing the history of transactions or simply disrupting operations – would require such entity to control a significant number

Table 1
Summary and comparison of related work.

Studies	Blockchain solution	Support for Quality Hallmark (Trusted data quality sharing)	In the context of industrial IoT	Targeting manufacturing domain	Support for zero-defect manufacturing
Our work	✓	✓	✓	✓	✓
Casado-Vara et al. (2018)	✓	✗	✗	✗	✗
Dedeoglu et al. (2019)	✓	✗	✗	✗	✗
Ahmed et al. (2021)	✓	✗	✗	✗	✗
Bai et al. (2019)	✓	✗	✓	✗	✗
Tang et al. (2019)	✓	✗	✗	✗	✗
Rahman et al. (2020)	✓	✗	✓	✗	✗
Bartol et al. (2020)	✓	✗	✓	✗	✗
Patel and Shrimali (2021)	✓	✗	✓	✗	✗
Manogaran et al. (2021)	✓	✗	✓	✓	✗
Ma et al. (2020)	✓	✗	✗	✗	✗
Lopez and Farooq (2020)	✓	✗	✗	✗	✗
Shafagh et al. (2017)	✓	✗	✗	✗	✗
Shen et al. (2019)	✓	✗	✗	✗	✗
Bajoudah et al. (2019)	✓	✗	✗	✗	✗
Kang et al. (2018)	✓	✗	✗	✗	✗
Akkaoui et al. (2020)	✓	✗	✗	✗	✗
Nguyen et al. (2021)	✓	✗	✗	✗	✗

of its nodes, like a company's shareholder that gains full control by buying the majority of shares. This is unlikely to happen without the fact being noticed by other network participants.

One of the most well-known applications of DLT is blockchain technology, in which the ledger consists of "blocks" of transactions. In other words, blockchain is a network of decentralized, distributed blocks used to store information with digital signatures (Monrat et al., 2019). A Blockchain is perceived as an enabler of trust. Bitcoin, which was the first "killer application" for this technology, demonstrated the feasibility of secure peer-to-peer financial transactions over an insecure public network, without the need for trusted intermediaries or even any form of trust between parties. Ethereum then extended Bitcoin's baseline technology, creating the first programmable Blockchain: custom business rules could be enforced by a special kind of software program called smart contract, thus extending the reach of the system well beyond finance. This gave birth to the concept of decentralized application – also known in brief as Dapp or dApp – where the Blockchain network plays the role of a distributed computing environment that is owned and operated collectively by a community of users.

More recently, with the growth of DLT awareness in many more business communities, a new generation of platforms has emerged that aims at bringing decentralization to the enterprise world. As opposed to the original vision of a permissionless network of peer nodes (i.e., a network that any computing system may join anonymously, without being granted permission from a governance body), these new platforms only support permissioned configurations, thus restricting participation to a selected list of approved members. Although this approach limits the role of Blockchain as an enabler of trust, it does indeed provide some important advantages for enterprises — most notably, that they are still in control of the infrastructure they use for conducting business.

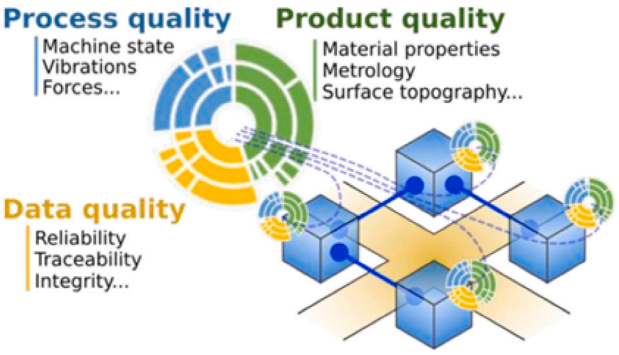


Fig. 1. Conceptual view of the PPD Quality Hallmark.

4. Quality Hallmarks

ZDM solutions need to assure the quality in smart factories in a holistic manner including process, product and data quality (Cassoli et al., 2022). The broad vision is to allow controlling the quality of a smart manufacturing environment in an end-to-end approach by means of a PPD Quality Hallmark stored in a distributed ledger. The PPD Quality Hallmark is the basic information unit of quality assessment; it carries all the quality-related information about a well-identified object (see Fig. 1):

- Process quality: information originating from the manufacturing process (e.g., derived from machine-level sensors).
- Product quality: information originating from the finished product or part (e.g., derived from metrology procedures).
- Data quality: meta-data about the accuracy and reliability of the raw data used for the assessment.

A very important point is that the Quality Hallmark does not contain, under any circumstances, raw data such as sensor readings or physical measurements. Raw data is stored locally at the plant – or in the cloud – and then analysed as part of a quality assessment process. The results of such assessment, which will take the form of KPI values, is what the Quality Hallmark is all about. Another important property of the Quality Hallmark is trustworthiness. This means that the information contained is certified to be authentic by some entity that takes full responsibility for it. Finally, Quality Hallmarks are public, at least within the boundaries of the business ecosystem they are created for. The concept here is that a Quality Hallmark is a public statement that, once made, cannot be modified or retracted. This implies not only that they should be made accessible to all the stakeholders on some sort of common infrastructure, but also that their format and data model should be standardized, in order to ensure interoperability between heterogeneous systems.

5. PPD quality data and data quality services

Product quality data and process quality data have been increasingly collected from manufacturing execution systems. For example, two types of quality information sources are currently available on the cylinder head production line:

- Dimensional controls performed by a CMM Zeiss machine. These controls are not directly online but by sampling on the rule one part per machine per shift. In this case, the parts are taken out of the line and controlled asynchronously to the production. The results of these tests are then stored in a database and part serial number are used as an access key to the database.
- Leakage controls are performed online by a specific machine. Result of these tests are stored on a local PC, also by using part serial number as access key.

Sharing information on **product quality** across the supply chain is commonplace. In fact, the workpiece is frequently delivered with the metrology reports. Additionally, users of machine tools frequently save the results of their internal quality control processes in case a later incident occurs. For example, in the aerospace industry, where investigations are ongoing following every significant incident and the issue might be linked to any manufacturing process carried out on the defective item, this condition is taken to an extreme. However, there is currently no accepted method for exchanging reliable information, and there is no consensus on how to demonstrate the sincerity of each party participating.

Process quality data can be used for controlling the quality of the manufacturing processes and equipment. Indeed, process quality data must also be shared, and/or to go hand-in-hand with the corresponding product data. The main difficulty on using dimensional controls results mentioned earlier is to synchronize them with process data.

As data is key, the quality of critical data for ZDM must be checked. The high quality and continuity of sensor data streams is fundamental to predict phenomena such as geometric deformations, surface roughness, excessive coolant use and imminent tool wear with adequate accuracy and appropriate timing. However, in practice, data acquired by a subset of all sensors can be of poor quality and unsuitable for prediction due to sensors faults stemming from environmental factors (Husom et al., 2022; Sen et al., 2022; Nguyen et al., 2022). Therefore, the meta-data about the quality of data used for the assessment must be also shared and trusted. In this section, we focus on presenting some **data quality** services as a representative type of services that share data quality metrics. Data quality metrics are the measurements used to assess data. They benchmark how beneficial and relevant data is and help differentiate between high-quality and low-quality data. In Section 5.1, we present the data quality metrics as data quality rules that can be verified using a data profiling service such as Great Expectations (GE). Then, in Section 5.2, another data quality service for Erroneous Data Repair is briefly presented. Such data quality metrics produced by data quality services can be shared. In the same way, the product quality data and process quality data can also be shared.

5.1. Great expectations

The data quality metrics are translated to data quality rules using the Great Expectations (GE) library. GE was chosen as it is a shared, open source standard for data quality, which is used by many organizations worldwide. It is a Python library that helps us to validate, document, and profile the data to ensure that it is of good quality and meets our expectation. An “Expectation” is a statement describing a verifiable property of data such as missing data, duplicate data, data between a given range, etc. GE provides several functions to evaluate the data from many different perspectives.

We have categorized GE into three categories as follows.

5.1.1. Standard great expectations

We have utilized the common Expectations from the Great Expectations’ built-in library. These are used to solve the commonly encountered data quality issues. The users cannot provide input for the parameters, all the input parameters for rules are fixed, except for “Value is in set” rule, which requires the user to provide a set of values and a value to be checked (for its presence in the set) as the inputs.

Some examples of GE standard rules:

- Column exist: Checks if the input column exists
- Column is datetime: Checks if the column has a valid datetime format
- Column is of type: Checks if a column is of a valid type

5.1.2. Custom great expectations

This category consists of custom Expectations, which were built using one of the four templates of Expectation subclasses mentioned in the Great Expectations website.[3] We created our own domain-specific logical Expectations that is required to verify the data obtained in the project. These rules under Custom GE give users flexibility to provide their own parameters.

Some examples of GE custom rules:

- Duplicate records: Checks for duplicate records in a column
- Missing records: Checks for missing records in a column
- Missing values: Checks for missing values in a column

5.1.3. General great expectations

We have also created some Expectations that allows the client to perform general tasks such as listing all the files in the folder, listing all the rules defined in a library, plotting a rule, monitoring API health, uploading a CSV file to the data folder and so on. These rules are similar to the standard GE but are slightly tweaked to perform the aforementioned tasks.

Some examples of General rules:

- Rules defined in library: List all the rules
- Rule definition: Provides the definition of a particular rule
- Run rule by name: Runs a rule based on the input rule name

GE is an example of a data profiling service. Next, we present a data quality service, which not only checking for erroneous data but also repairing them.

5.2. Erroneous data repair

Two ML-based data quality pipelines shown in Fig. 2 are unsupervised data validation pipeline and erroneous data repair respectively (Sen et al., 2022). The first data quality task is to detect and tag erroneous and missing values on the edge gateway in-motion data filtering. Our ML pipelines leverage recurrent patterns in Industrial

[3] https://greatexpectations.io/

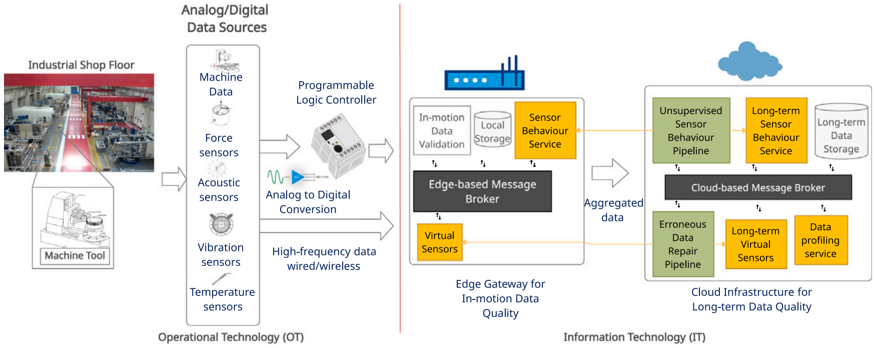

Analog/Digital Data Sources
Industrial Shop Floor
Machine Tool
Machine Data
Force sensors
Acoustic sensors
Vibration sensors
Temperature sensors
Programmable Logic Controller
Analog to Digital Conversion
High-frequency data wired/wireless
In-motion Data Validation
Local Storage
Sensor Behaviour Service
Edge-based Message Broker
Virtual Sensors
Edge Gateway for In-motion Data Quality
Aggregated data
Unsupervised Sensor Behaviour Pipeline
Long-term Sensor Behaviour Service
Long-term Data Storage
Cloud-based Message Broker
Erroneous Data Repair Pipeline
Long-term Virtual Sensors
Data profiling service
Cloud Infrastructure for Long-term Data Quality
Operational Technology (OT)
Information Technology (IT)

Fig. 2. A general architecture of ML-based data quality pipelines.

Internet of Things (IIoT) data to (i) generate virtual sensors repairing missing/corrupt data in one sensor based on high-quality data from other sensors (erroneous data repair pipeline) and (ii) determine events and deviations in an unsupervised manner (unsupervised data validation pipeline). Their services (ML models) are containerized and deployed on edge for real-time inference and the cloud for inference on historical data. Domain experts specify domain heuristics to profile data and generate quality metrics (data profiling service). These metrics help maintain high data quality standards for AI-based applications and auditing.

Our ML pipelines can detect sensor drift and repair erroneous data. However, human involvement and domain-specific knowledge about data quality can help detect quality issues that the pipelines cannot. For instance, temperature readings exceeding 120 degrees more than five percent of data points are invalid since several occurrences of such temperatures could damage machine components. Machine tool temperature and force measurements often have a linear correlation. It is crucial to quantify the limits of this correlation and ensure that nonlinear behaviour is detected before catastrophic consequences. Our architecture also adopts Great Expectations (GE), which aids in auditing data acquisition and persistence from industrial processes. Furthermore, the data quality reports from GE help create a data quality culture by making engineers confident of their data-driven decisions and the corresponding uncertainty estimates.

6. The trusted framework

As explained in Section 4, supply chain actors will capture all quality information of interest into PPD Quality Hallmark (QH) records and then share them with all the stakeholders, thus supporting the collective effort towards ZDM. The existence of QHs, however, does not solve the problem of how such information units can be securely exchanged within the supply chain's business ecosystem. The obvious solution is some sort of common ICT infrastructure, requiring a decentralized approach to interoperability and trust. In other words, such an infrastructure must be a non-hierarchical distributed system where all the individual nodes operate as peers (i.e., without anyone having more authority over the others), each node is owned by a stakeholder and all stakeholders are equally represented. Moreover, trustworthiness must stem from decentralization (trust-by-design), rather than from software- and human-based safeguards that are usually adopted to improve security in centralized systems. The module that enables the secure exchange of trustworthy QH records is called Trusted Framework (TF, Fig. 3). The TF includes the data model and format of the Process/Product/Data (PPD) Quality Hallmark, the OpenAPI exposed to factory system, trusted nodes, personal nodes, their internal components and, most importantly, the processes supporting QH and identity management.

6.1. How the TF supporting ZDM based on its permissioned Blockchain platform

The TF is positioned as the link connecting, on the one side, systems that collect raw data on the shopfloor and turn them into Quality Hallmarks and, on the other, software tools supporting ZDM decision-making. Fig. 4 depicts the flow of quality-related data, showing how raw data go through incremental refining and aggregation in order to feed the ZDM optimization loop. Three horizontal "swim lanes", which corresponding to the three facets of the problem: Process, Product and Data, deal with the first stages of collection, transformation, and analysis. The three lanes then converge into a single one, creating a unified view of quality that is captured as certified Quality Hallmark Document (QHD), which will be explained in some detail later on and published on the TF. Finally, QHs are used as a trustworthy input to quality optimization tasks.

It is worth noting that the TF's concept of "certification" is focused on accountability rather than truth: there is no way for the TF to check that a QH contains information that is true and exact with respect to the real-world phenomena it describes; however, the TF can and does ensure the integrity, reliability and traceability (including timestamping) of the QHs that are published. To achieve these goals, the TF relies on Distributed Ledger Technology (DLT), or more precisely the Blockchain variant of DLT. Given the context and the requirements of the TF, a permissioned Blockchain platform has been chosen as the backbone of its implementation. We detail the main components of the TF in the following subsections.

6.2. Trusted nodes

Each company of a supply chain's business ecosystem is then expected to run at least one physical node, called Trusted Node (tNode), of the TF's network, which will thus consist of a number of fully autonomous tNodes that maintain their own synchronized replica of shared data. Node-to-node communication within the network happens by means of peer-to-peer protocols (Fig. 5).

A tNode is defined as a secure computing environment deployed in a specific scope of the TF network and under the control of one well-identified company. Security, scope, control and identity are the four keywords of importance in this context, because any company that runs a tNode must be held accountable and must be "caged" within the boundaries of its scope (for more details on what a scope is in the TF context, see the explanation of "Clusters" below). The general idea is that a company will be allowed – by the other companies in the same business ecosystem – to operate a tNode if the company is known and trusted, to the effect that trust in the company will extend to the node, and the node will then be able to effectively participate in a peer-to-peer collaborative process. On the other hand, if the node will later become untrusted because of some incorrect behaviour that

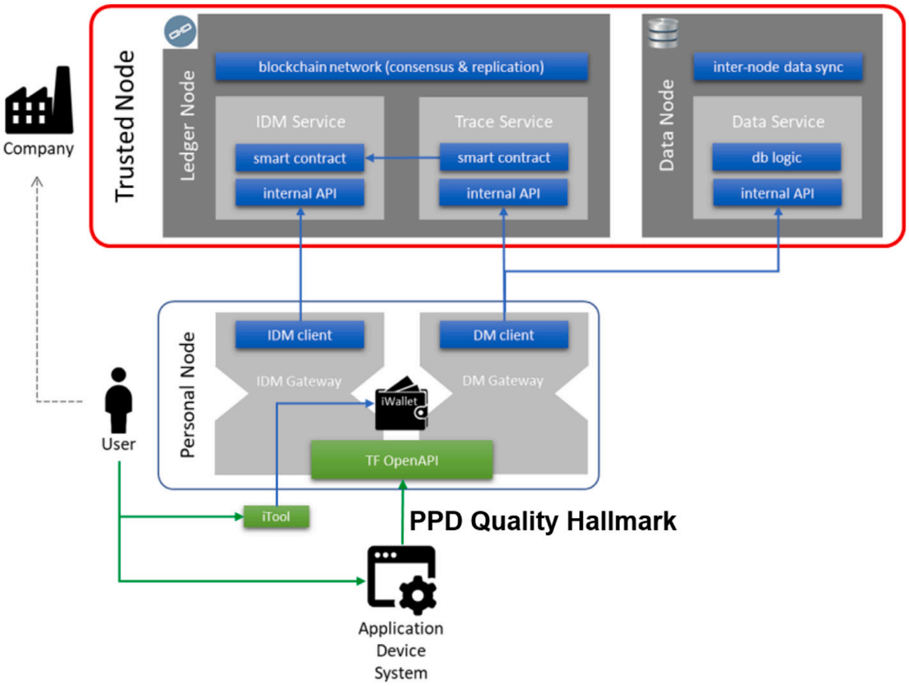


Fig. 3. Components of the Trusted Framework.

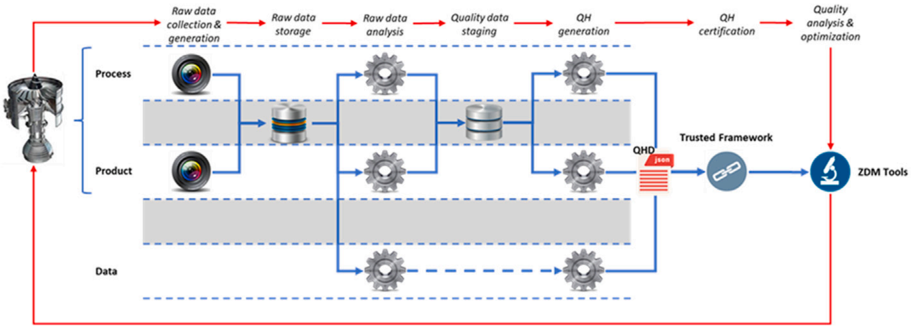


Fig. 4. The ZDM optimization loop.

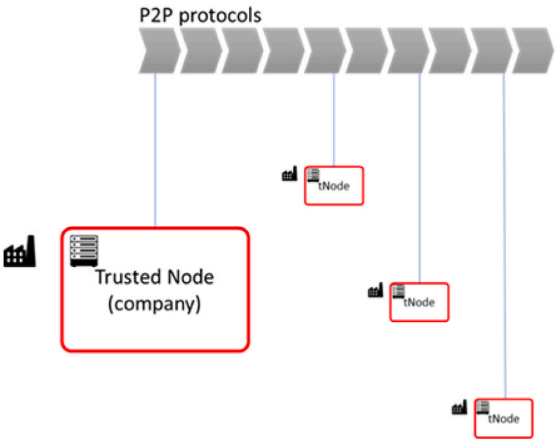


Fig. 5. Trusted Nodes of the TF.

is detected – e.g., technical malfunction, intentional violation of rules, security breach – such lack of trust will be reflected on the company.

Basically, every tNode represents its owner within the TF. This is the general principle that underpins "permissioned" Blockchains: the system as a whole can tolerate a number of "bad" nodes, provided a sufficient number of "good" ones is still online, but must also be able to exclude bad nodes as long as they are not trusted any more. From this, it may appear that "Trusted Node" is just synonymous of "Blockchain peer node". Actually, it is not so: the TF must deal with persistent entities – i.e., the QHDs – that have no predefined limits in terms of size and number. In other words, the TF should be able to manage any volume of shared data. This requirement cannot be met by a pure Blockchain system, because of the structural weakness of the technology when it comes to storage efficiency.

To overcome this constraint, the tNode contains two separate components that work in parallel to get the job done: the Ledger Node and the Data Node. The Ledger Node is indeed a Blockchain peer node that hosts and runs smart contracts: the Identity Management (IDM) Service and the Trace Service. The Data Node is a separate but fully integrated environment where a non-Blockchain distributed storage sub-system, the Data Service, is executed. The Trace Service and the Data Service are complementary systems that cooperate to a common goal: ensuring that QHDs are published, within the boundaries of the intended scope, in a secure and trustworthy way. The Data Service is responsible for storing the actual QHD object (which is replicated on

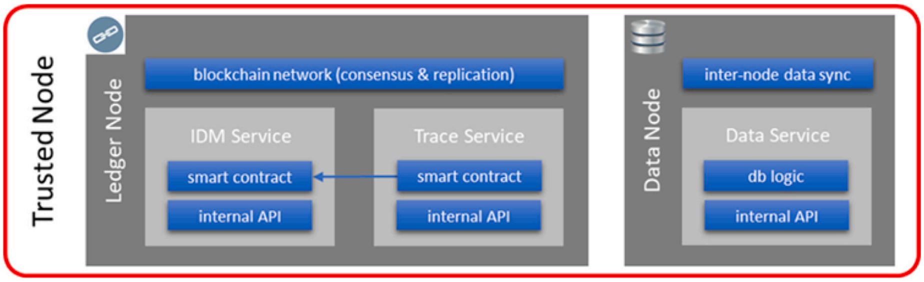


Fig. 6. Internal structure of a Trusted Node.

multiple nodes for redundancy), while the Trace Service writes on the Blockchain a record that holds a digital fingerprint of the QHD object and a reference to its storage location (plus additional meta-data). This way, the tNode delivers the best of two worlds: robust and virtually unlimited storage provided by the Data Service and decentralized trustworthiness provided by the Trace Service. On the contrary, the IDM Service is a Blockchain smart contract that is tightly integrated with the Trace Service one: the functionality it provides is the registration and authentication of TF users — more specifically, of the Trace Service's API. The relationship between these three components of the tNode is shown in Fig. 6 below.

Having described, at a very high level, the role and internals of the tNode, we now take a step back and have a look at the general picture of the TF, again — this time, with enough background knowledge to better understand the key issues of its deployment.

6.3. Users, companies and clusters

In this section, we explain two important concepts that shape the TF architecture: actors and scopes. Actors have little need for explanation, because it is a standard term that describes active entities that interact with a system. In our case, we only have two kinds of actors: Users and Companies. Companies are straightforward: they are legal entities, their identity is known to the system and they hold some digital credentials to provide proof of their identity in online interactions. Typically, the authentication of a Company's identity is done by the TF in order to check if that Company can legitimately operate a tNode. Users are not much different, although they correspond to a physical entity: a human being, a smart machine or tool, an IIoT device, etc. The most significant difference, however, is that every User must "belong" to a Company. Such parent–child relationship, that puts Companies and Users in a hierarchical relationship, is a key feature of TF's digital identity management system. Human Users are typically employees of the Company, while machines and devices are the Company's assets. The bottom line is that Users, whatever they are in the real world, always operate under the responsibility of the parent Company. Scopes are a more abstract concept than Users and Companies, because they do not (normally) correspond to any real-world entity and have no "owner" holding matching credentials. A scope defines, as implied by the name, the boundaries of a subset – or partition – of a TF network. It is used to limit the visibility of data to some specific Companies and their Users. The idea is that multiple scopes may exist in a given TF system, thus creating separate private data spaces. The exact RA term for a scope is Cluster, because it is used to cluster multiple Companies into a single data space – although the same Company may belong to more than one Cluster. This scheme of things mimics real business: manufacturing companies may participate in more than one supply chain (which is a rather informal and dynamic aggregate, not a legal entity), but data shared within any supply chain is not to be leaked outside of it. So, in the TF context, a Cluster can be seen as the equivalent of a supply chain ecosystem, although the same architecture can support other business models. To put Users, Companies and Clusters in the right perspective, Fig. 7 provides a practical example of a TF deployment for three Clusters, each corresponding to a separate supply chain. The same Company (the fictional "ACME") is involved in two supply chains ("A" and "B"). In order to avoid visual clutter, only three Users per Company are shown in the first Cluster, and the other two Clusters are depicted in a simplified style. In most real scenarios, each Company will include tens or even hundreds of Users, while Clusters will be composed of several Companies.

6.4. Personal nodes and the TF OpenAPI

The Personal Node (pNode) is the "front end" to the TF: each pNode installation is exclusively dedicated to a single owner and acts as the gateway to a specific tNode; the same tNode, on the other hand, can be reached by any number of pNodes. Fig. 8 illustrates this concept in a schematic way.

The pNode impersonates a specific user during any interaction with the tNode. In order to do this, it embeds an Identity Wallet where user credentials are stored. Most importantly, the pNode exposes a REST-over-HTTP service endpoint that provides the TF's OpenAPI: all the low-level APIs exposed by the tNode are available through the OpenAPI – only in a simpler, operation-oriented fashion that hides much of their technical complexities, like proprietary communication and authentication protocols. In the coming paragraphs we provide simplified documentation of the OpenAPI.

6.4.1. OpenAPI call: publish a QH

Allows the caller to publish a new Quality Hallmark by uploading a QHD. The QHD will be shared within the scope the caller belongs to.

```
method: POST
path: (endpoint address)/[domain]/tf/v1.0/qhs
content type: application/json
body: (a JSON literal with the following structure)
  {
    ‘‘pwd’’: ‘‘(p1)’’,
    ‘‘cid’’: ‘‘(p2)’’,
    ‘‘qhd’’: ‘‘(p3)’’
  }
```

Listing 1: HTTP request

JSON literal parameters:

- p1: the password that unlocks the caller's iWallet
- p2: the identity that the Personal Node must impersonate when calling the Trusted Node
- p3: the QHD to be published, as a nested JSON literal

```
status code: 201 Created
body: (unique ID assigned by the TF to the published QHD
    )
```

Listing 2: HTTP response on success

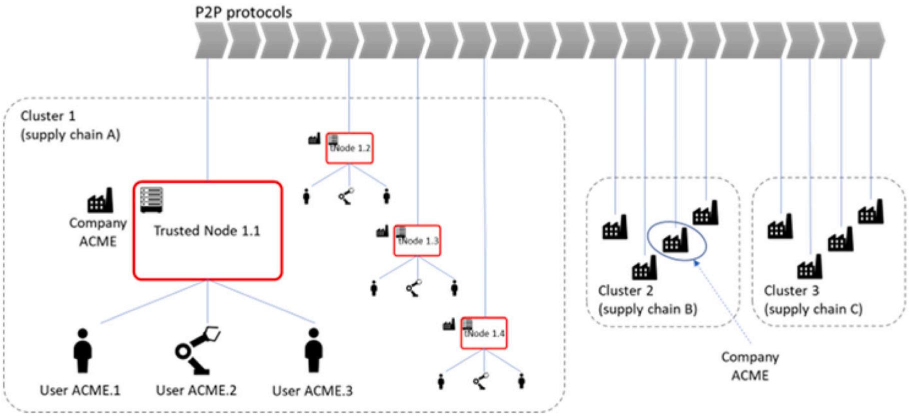


Fig. 7. Example of TF deployment with three Clusters.

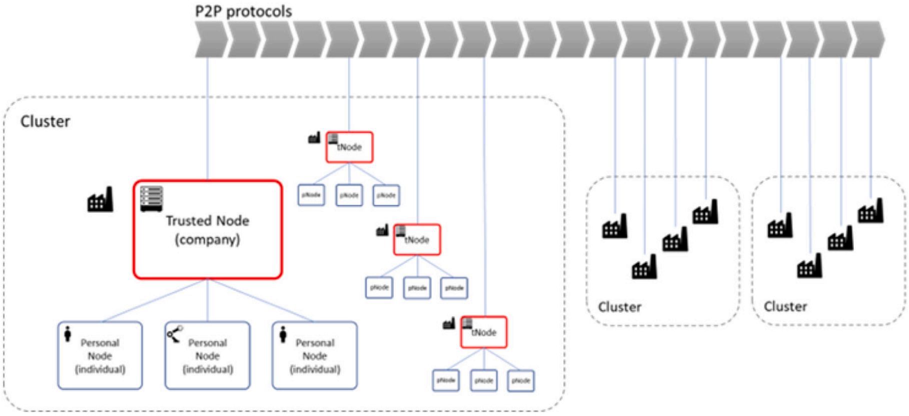


Fig. 8. Personal Nodes vs. Trusted Nodes.

6.4.2. *OpenAPI call: retrieve a single QH*

Allows the caller to retrieve the QHD of a published Quality Hallmark of which the ID is known. Scope restrictions apply: the caller will not be given access to QHDs that are shared outside of the scope the caller belongs to. Moreover, if the target QHD is found have been altered after publishing, access will be denied.

```
method: GET
path: (endpoint address)/[domain]/tf/v1.0/qhs/(p0)?pwd=(
    p1)&cid=(p2)
```

Listing 3: HTTP request

Path parameter:

- p0: the ID of the QHD to be retrieved

Query string parameters:

- p1: the password that unlocks the caller's iWallet
- p2: the identity that the Personal Node must impersonate when calling the Trusted Node

```
status code: 200 Ok
body: (the retrieved QHD, as a JSON literal)
```

Listing 4: HTTP response on success

6.4.3. *OpenAPI call: retrieve a set of QH matching some criteria*

Allows the caller to search the TF for all QHs that match some condition, which can be specified by providing a target value for one or more fields of the QHD header (see below for more details on the QHD structure). Scope restrictions apply: the caller will not be given access to QHDs that are shared outside of the scope the caller belongs to. Moreover, any QHD that matches the conditions but is found to have been altered after publishing will not be included in the results.

```
method: GET
path: (endpoint address)/[domain]/tf/v1.0/qhs?
pwd=(p1)&cid=(p2)[&o=(p3)][&a=(p4)][&m=(p5)][&s=(p6)][&
    tf=(p7)][&tt=(p8)]
```

Listing 5: HTTP request

Query string parameters:

- p1: the password that unlocks the caller's iWallet
- p2: the identity that the Personal Node must impersonate when calling the Trusted Node
- p3: identifier of the QHD owner – exact match with the owner header field's value
- p4: name of the target asset – exact match with the asset header field's value
- p5: name of the assessment model – exact match with the model header field's value
- p6: name of the assessment subject – exact match with the subject header field's value
- p7: from date/time – lower than or equal to the timeref header field's value
- p8: till date/time – higher than or equal to the timeref header field's value

```
status code: 200 Ok
body: (a possibly empty JSON array of QHDs, each as a
    JSON literal)
```

Listing 6: HTTP response on success

6.5. The Quality Hallmark Document

A Quality Hallmark is a set of quality indicators that materialize in the TF's decentralized infrastructure as a Quality Hallmark Document. Every QHD is issued by a well-identified entity and shared, within the scope of a collaborative business process, as a self-certified statement – i.e., a non-repudiable declaration made at a known point in time. There are five main aspects of a Quality Hallmark that are captured in a QHD:

- Every Quality Hallmark is linked to a specific assessment model – i.e., the notion of what is measured and how. This can be seen as a "type" or "class" of quality assessments, because it defines the quality indicators to be used in every individual assessment that belongs to that type. In the QHD, such notion is expressed as a unique name. Although mandatory, the model's name has no particular meaning for the TF infrastructure: assessment models are just a convention that is freely established between the stakeholders of an assessment process. However, the model implies – by virtue of the same convention – the data model and format of the QHD section containing quality indicators: all stakeholders are assumed to be able to properly parse and understand the contents of QHDs of any type that is known to them.
- Every Quality Hallmark is about one single assessment subject: the process, product or data set which is the target of that assessment. QHDs always include a pointer to their target, using a syntax that is defined by the same stakeholders' convention that underpins the assessment model (see the previous point).
- Quality Hallmarks are typically the result of an iterative process: an assessment of a given type that is repeated multiple times on the same subject, in order to observe the evolution over time of quality indicators. A single QHD can then be considered as one data point of a time series. To support this view, QHD carry a time reference, which represents the conventional point in time the assessment is referred to in the context of the overall quality assessment process. (It is worth noting that the model name, the subject identifier and the assessment's time reference represent, together, the "natural key" of Quality Hallmarks).
- A Quality Hallmark is always produced by a well-identified entity, the assessment owner, who is held accountable for the measurements/analysis reported there. To track this relationship, the QHD always includes a reference to its owner. Thanks to the TF's embedded IDM, such reference is trustworthy and can be easily resolved into the corresponding real-world identity.
- Sometimes, knowing the "subject" of a Quality Hallmark is not enough. For example, a periodic quality assessment of a given type (the model) may be scheduled for a production line (the subject), and each assessment result will be tagged with its time reference, but yet it might be interesting to know exactly which product item was used as the sample: in these cases, a reference to the specific asset being measured may also be included in the QHD, although this is not mandatory.

Concretely, a QHD is a digital document in JSON format. The TF specifies how a valid QHD instance is constructed, but still leaves plenty of room for case-by-case customization. Every QHD must mandatorily start with a standard header section that sets the Quality Hallmark's context – i.e., the four main aspects outlined above. This section has a fixed data model and format, and is subject to extensive validity checks on submission. The header is followed by a body section containing the actual information payload, which consists of Quality Hallmark indicators. The data model and format of this section must conform to the assessment model declared in the header, which means that they are totally discretional: every quality assessment process is free to define its own set of indicators and a standard way of representing them. However, there is one general constraint that applies to the QHD body: it must be a well-formed JSON string. This rule enables JSONPath filter expressions to be effectively run on the contents of the TF repository, without the TF having any notion of their meaning. The skeletal structure of the QHD is provided below:

```
{
  ''qhd-header'': {
    ...
    (standard header fields, including a reference to
        the data model used   for the body)
    ...
  },
  ''qhd-body'': {
    ...
    (custom body fields, in accordance to the data model
        declared in the header)
    ...
  }
}
```

Listing 7: QHD Header

6.5.1. QHD header

The fields in the header, together with the validity rules that apply to them, are listed below. To understand how some validity checks are done, it is important to consider that QHDs can only be submitted by users that own a currently valid identity registered in the TF's IDM subsystem, and that their identity claim is checked at runtime through a secure challenge-response protocol.

- owner – Link to the entity (typically, an organization) that is accountable for the assessment represented by the QHD. The link is expressed as a Decentralized Identifier (DID)[1] that points to a record in the TF's identity registry. (e.g., did:[domain]-id:1K31KZXjcochXpRhjH9g5MxFFTHPi2zEXb). On submission of a new QHD, the TF will check the identity of the OpenAPI caller against the identity of the assessment owner that is declared in the document header: the submission is accepted only if the two identities are the same or if the caller is related to the owner as a subordinate entity.
- asset – Name of the asset which is the target of the assessment. This information is optional, and can be entirely omitted if not required. Moreover, the format of this value can be freely defined by the stakeholders, the only requirement being that it can be correctly interpreted by everyone. For example, the asset may be a product item that is identified in the context of the assessment (i.e., model + subject) by its serial number.
- model – Name of the assessment model the QHD conforms to. The name needs to be a globally unique identifier. More specifically, it must be a URL with the following syntax: [domain]-qhd://domain-name/model-name[/version-number], where the "domain-name" component is a registered DNS name that is owned by the stakeholder who plays the role of "authority" for that namespace, "model-name" is an arbitrary string and "version-number" (optional) is used when a given model goes through revisions over time.
- subject – Identifier of the phenomenon under observation. The format of this value depends entirely on the nature of the phenomenon and on how this can be unambiguously identified in its context. For example, it may be a production process which is assessed periodically, so that its identifier will be composed by the product's SKU plus the name of the production line. Target phenomena that are entirely digital, like data sets, will need some proper addressing scheme as well. Whatever the

case, it is strongly advised that subject identifiers are structured as an array of name–value pairs, with a double colon as the name–value separator and a semicolon as the array separator; e.g., line::XYZ;sku::AB66.
- timeref – Point in time the assessment refers to. It is a timestamp that follows the ISO 8601-1:2019 standard, is expressed as UTC time and has one-second resolution (e.g., 2021-04-05T14:30:15Z).

To recap, the QHD header must follow the format of the example provided below, which includes some fake values (whitespace and line terminators added for better clarity).

```
‘‘qhd-header’’: {
‘‘owner’’: ‘‘did:[domain]-id:1K31KZXjcochXpRhjH9g5
    MxFFTHPi2zEXb’’,
‘‘asset’’: ‘‘order=12345’’,
‘‘model’’: ‘‘[domain]-qhd://zdm.com/mymod/3’’,
‘‘subject’’: ‘‘line::XYZ;sku::AB66’’,
‘‘timeref’’: ‘‘2021-04-05T14:30:15Z’’
}
```

Listing 8: QHD Header

6.5.2. QHD body

When it comes to the QHD body, things get simpler – at least from the normative perspective. Basically, the main payload of the QHD can be any well-formed JSON literal that goes under the name of qhd-body. The implementation of the TF decentralized infrastructure may also pose some constraint on the maximum size of the QHD as a whole (header + body), but this aspect is not considered in this specification. The following is a very short example of a hypothetical QHD body:

```
‘‘qhd-body’’: {
  ‘‘program_name’’: ‘‘...’’,
  ‘‘start_time’’: ‘‘...’’,
  ‘‘end_time’’: ‘‘...’’,
  ‘‘machine_dimensional_quality_control’’: ‘‘...’’,
  ‘‘process_roughing’’: {
    ‘‘touch_probe_measurement’’: {
      ‘‘IND_diameter_1’’: ‘‘...’’,
      ‘‘IND_height_1’’: ‘‘...’’
    },
    ‘‘geometric_corrections’’: ‘‘...’’,
    ‘‘subprocess_first_roughing’’: {
      ‘‘IND_max_vibration’’ : ‘‘...’’,
      ‘‘IND_min_vibration’’ : ‘‘...’’
    },
    ‘‘subprocess_final_roughing’’: {
      ‘‘IND_max_vibration’’ : ‘‘...’’,
      ‘‘IND_min_vibration’’ : ‘‘...’’
    }
  }
}
```

Listing 9: QHD Body

Although fictional, this example provides the reader with several hints on how the JSON language is used to construct a well-formed QHD body. More formally, the QHD body is a JSON object literal that is named qhd-body and contains any number of properties (i.e., name–value pairs). These are subject to some rules:

- Names are lowercase alphanumeric strings without any whitespace.
- Names that identify a quality indicator must start with the “IND_” prefix.
- Values can be any of the following:
  - a string;
  - a number (decimal separator is the “.” character, no other characters are allowed);
  - the true or false keywords (to represent a boolean);
  - the null keyword (to denote an undefined value);
  - only for properties that are not a quality indicator: a JSON object literal (a nested object).

Although this is implied by the rules listed above, it is worth pointing out that array values are not supported in the QHD body. As previously mentioned, the body’s structure is supposed to match a pattern that is identified by the model field of the QHD header. This rule however is not enforced by the TF infrastructure, so that well-formed but faulty QHDs can be successfully published. It is the responsibility of applications to check the validity of any QHD they retrieve from the TF, without assuming their formal correctness.

6.5.3. QHD envelope and seal

While the QHD described in the previous sections is the logical information unit that is managed in the TF, its physical storage is more complex than that. Firstly, every published QHD is composed of two physical records:

- the QHD envelope, which contains the QHD and is replicated on all the TF Data Nodes that belong to the same visibility scope the publisher belongs to;
- the QHD seal, which is replicated on all the TF Ledger Nodes, regardless of scope.

The QHD envelope is just what its name implies: a wrapper around the actual QHD. The role of the envelope is to add one single piece of information that is not part of the QHD but is still required for its management: the unique ID assigned by the TF to the Quality Hallmark when it is published. The same ID is also included in the QHD seal, so that the two records are linked in a one-to-one relationship. The name of the QHD seal is also self-explanatory: it is a “seal of authenticity” that guarantees the provenance, timing and integrity of a QHD, independently by its storage location. Its data model consists of six fields:

- the QHD ID, assigned by the TF;
- the publishing timestamp, assigned by the TF;
- the identity of the publisher (i.e., the user who made the API call), verified by the TF;
- the name of the visibility scope, which is an attribute assigned to the Personal Node (at deployment time) from which the publishing API call originated;
- the hash value of the QHD;
- the name of the hashing algorithm used to calculate the hash value.

The last two fields can be used, together, to check the integrity of the QHD represented by the seal: after feeding the named hashing algorithm with the QHD’s bytes, the output must coincide with the stored value. The integrity of the seal record itself is guaranteed by the use of a blockchain for its storage. The seal does not contain any sensitive information, and thus it is stored as clear text (i.e., not encrypted) without any access constraints. Below we provide a concrete example of QHD seal. In this case, the value in HASHVAL is calculated by feeding the entire content of the QHD (header + body) identified in ID as the input to the hashing function named in HASHALG (i.e., SHA256).

```
ID       123e4567-e89b-12d3-a456-426614174001
TS       2021-04-10T01:25:10.345Z
USER     1K31KZXjcochXpRhjH9g5MxFFTHPi2zEXb
SCOPE    MYSCOPE
HASHVAL  ba7816bf8f01cfea414140de5dae2223b00361a396177a9
    cb410ff61f20015ad
HASHALG  SHA256
```

Listing 10: A concrete example of QHD seal

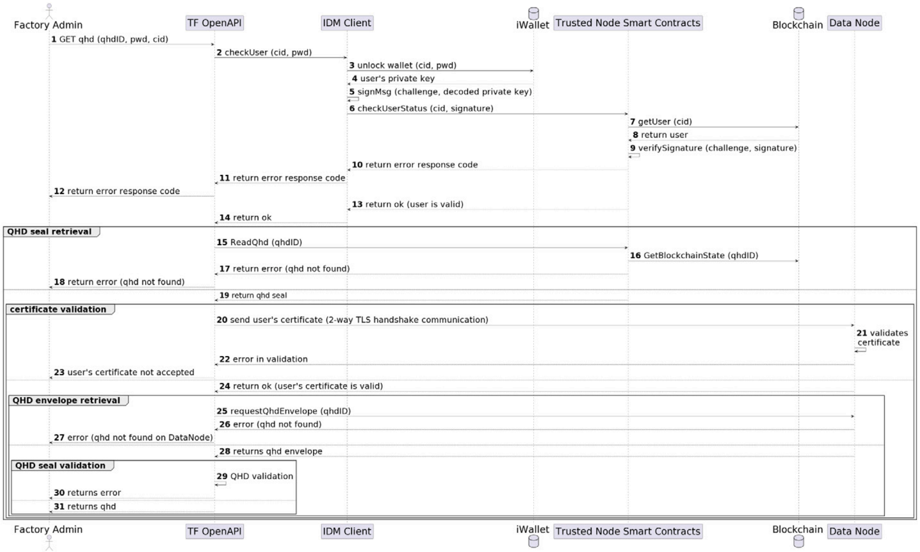

Factory Admin
TF OpenAPI
IDM Client
iWallet
Trusted Node Smart Contracts
Blockchain
Data Node
1 GET qhd (qhdID, pwd, cid)
2 checkUser (cid, pwd)
3 unlock wallet (cid, pwd)
4 user's private key
5 signMsg (challenge, decoded private key)
6 checkUserStatus (cid, signature)
7 getUser (cid)
8 return user
9 verifySignature (challenge, signature)
10 return error response code
11 return error response code
12 return error response code
13 return ok (user is valid)
14 return ok
QHD seal retrieval
15 ReadQhd (qhdID)
16 GetBlockchainState (qhdID)
17 return error (qhd not found)
18 return error (qhd not found)
19 return qhd seal
certificate validation
20 send user's certificate (2-way TLS handshake communication)
21 validates certificate
22 error in validation
23 user's certificate not accepted
24 return ok (user's certificate is valid)
QHD envelope retrieval
25 requestQhdEnvelope (qhdID)
26 error (qhd not found)
27 error (qhd not found on DataNode)
28 returns qhd envelope
QHD seal validation
29 QHD validation
30 returns error
31 returns qhd
Factory Admin
TF OpenAPI
IDM Client
iWallet
Trusted Node Smart Contracts
Blockchain
Data Node

Fig. 9. The sequence flow of a QHD request to the TF OpenAPI.

Every time a previously-published QHD is requested by a user through the OpenAPI, the two records – the envelope and the seal – are selected, the QHD is extracted from the envelope and then compared to the hash value stored in the seal: the QHD is handed back to the user only in case of a positive match, which means that the document was not altered after being publishing.

6.6. The QHD request in the TF

This section describes the complete flow and the corresponding sequence diagram of the request to retrieve a single QH (see Fig. 9), also described in Section 6.4.2, as an example to help the reader understand the end-to-end interactions between the internal components of the TF.

```
method: GET
path: (endpoint address)/[domain]/tf/v1.0/qhs/(p0)?pwd=(
    p1)&cid=(p2)
```

Listing 11: HTTP request

- User request. The TF OpenAPI allows an authorized caller to retrieve a published QH by sending a GET request and passing the unique id of the QH (step 1).
- User authentication. The authentication of the user is handled by the IDM Gateway of the Personal Node (steps 2–9) which employs the iWallet and the IDM Smart Contracts to check the status of the user invoking the request.
- Get the QHD seal. Upon the successful user authentication, the TF OpenAPI invokes the Trace Service smart contract in the Ledger Node to receive the QHD seal by its unique id (steps 13–16).
- Get QHD envelope from the Data Node. The TF OpenAPI communicates with the Data Node in order to retrieve the specified QHD envelope using a fully encrypted 2-way TLS connection with authentication on both the server and the client side (steps 20–21). Consequently, and in the case of a successful authentication, the TF OpenAPI sends a GET HTTP request to the Data Node Service: /QualityHallmarkData/id (step 25) and receives the QHD envelope (step 28).
- Validate QHD envelope against the QHD seal stored in the blockchain. At this step, the validation of the integrity of the QHD envelope takes place through the validation of its hash (step 29). The output of the hash of the data retrieved from the Data Node must coincide with the hash value stored in the QHD seal returned from the blockchain. In case of a mismatch, an error is returned to the TF user through an HTTP response error code and the process is terminated (step 30). If the two hashes do match as expected, then the TF OpenAPI returns to the TF user the JSON literal containing the QHD (step 31).

7. QHD and TF in action

In this section, we show a proof-of-concept on how to publish QHD containing the quality of data collected by the process services, product services, or data services. We focus on the data services and an orchestration framework to publish QHD to the TF. The structure of the orchestrator is shown in Fig. 10. The orchestrator will call real-time streaming data from manufacturing data systems (e.g., SAVVY data systems[4])/datalakes and store the data in a CSV file. Data Quality as a Service (DQaaS) can be accessed through REST API and the orchestrator will pull the DQ results for the real-time streaming data. Other data services such as Erroneous Data Repair presented in Section 5 can also be called by the orchestrator for the DQ results. The QHD header part will be sent to the TF's OpenAPI by the orchestrator. The orchestrator is built on Jupyter Notebook. Relevant statistics such as API sessions can be shown in a dashboard with DASH visualization. The DQaaS page displays lists the data quality rules available on the left of the screen, as shown in Fig. 11. The rules are a selection of generic rules direct from the Great Expectations Python library, in addition to tailored rules. We now show examples of running DQaaS rules that generate QHD to be published in the TF.

For example, Fig. 12 shows how to check for Missing Records with the parameters are described in Section 5. This will provide a response and the result provided in Fig. 13 shows that there are 3259 rows in the input data (which contains a total of 3528 rows) where the column Axis_FeedRate_actual has missing records. The first 10 records containing missing records are shown in the "partial_unexpected_list" in the respond content (Fig. 13).

The rule "QualityHallmarkDoc" is developed to publish a Quality Hallmark document. The rule requires three inputs: a valid QHD in JSON format, a password (pwd) and a user identity id (cid) as shown in Fig. 14. QHD specifications are documented in Section 6. The QHD can either be manually typed in the JSON format or could be copied from the response body (after running a required rule) and pasted in the

[4] https://www.savvydatasystems.com/es/inicio

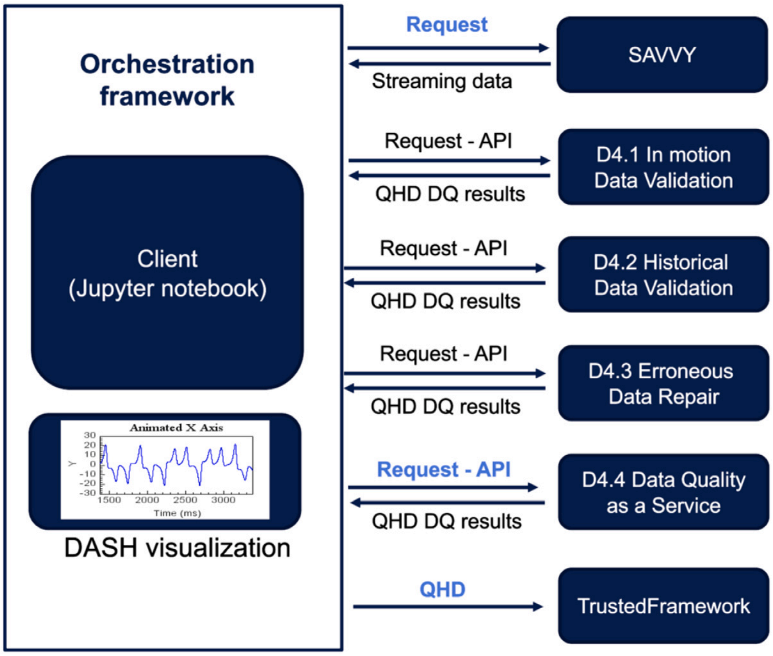
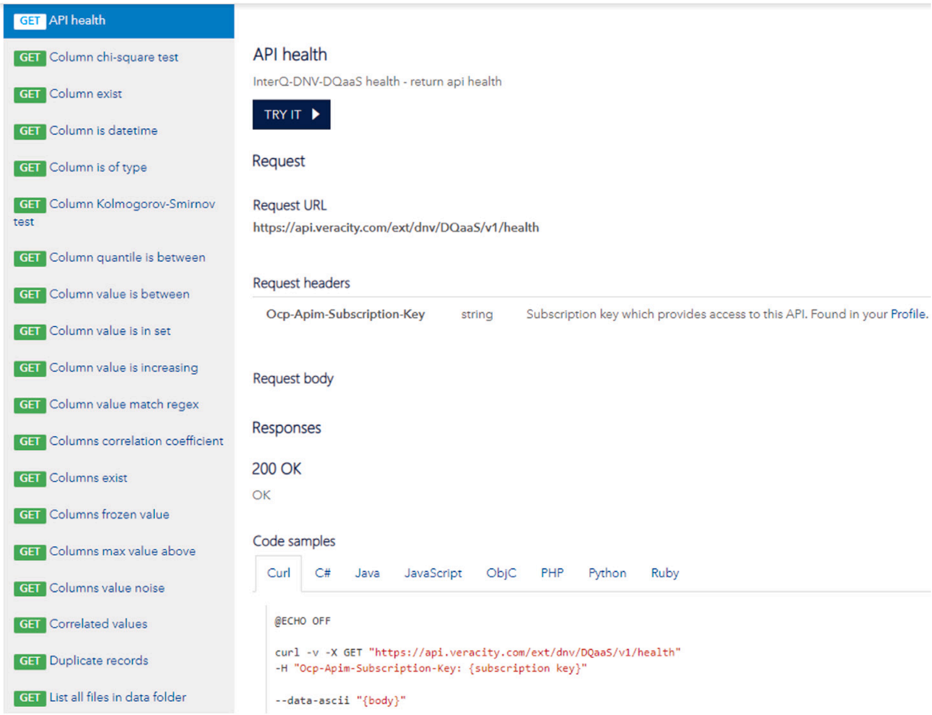


Fig. 10. Orchestration framework.


GET API health
GET Column chi-square test
GET Column exist
GET Column is datetime
GET Column is of type
GET Column Kolmogorov-Smirnov test
GET Column quantile is between
GET Column value is between
GET Column value is in set
GET Column value is increasing
GET Column value match regex
GET Columns correlation coefficient
GET Columns exist
GET Columns frozen value
GET Columns max value above
GET Columns value noise
GET Correlated values
GET Duplicate records
GET List all files in data folder
TRY IT
Curl
C#
Java
JavaScript
ObjC
PHP
Python
Ruby

API health

InterQ-DNV-DQaaS health - return api health

Request

Request URL
https://api.veracity.com/ext/dnv/DQaaS/v1/health

Request headers

Ocp-Apim-Subscription-Key	string	Subscription key which provides access to this API. Found in your Profile.

Request body

Responses

200 OK
OK

Code samples

```
@ECHO OFF

curl -v -X GET "https://api.veracity.com/ext/dnv/DQaaS/v1/health"
-H "Ocp-Apim-Subscription-Key: {subscription key}"

--data-ascii "{body}"
```

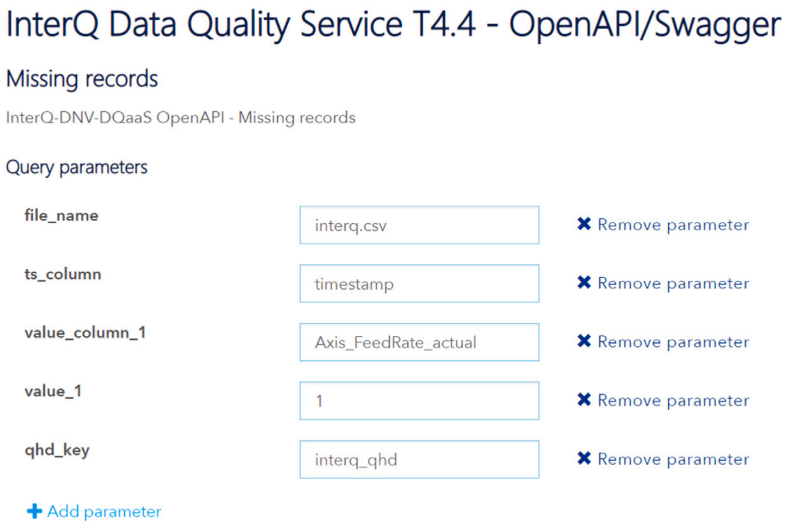
Fig. 11. Overview of data quality rules.

provided text box. The other two required parameters are the encrypted Identity Wallet (iWallet) decryption password and the ID of the identity to be impersonated. As presented in Section 6, their values must be provided by the caller on every call, so all OpenAPI operations define these two additional, mandatory parameters:

- pwd=p The password that unlocks the iWallet installed on the Personal Node
- cid=p The ID of the user identity to be used for impersonation

7.1. Great expectation standard rules

Example of a standard GE is shown in Fig. 15. We will discuss the “ColumnIsOfType” rule where the rule checks the datatype of the input column. When the “Try it out” button is pressed, we get the output as show in Fig. 16. The output is shown under the “Response body” heading in JSON format and “Response headers” lists the summary of the data which includes the connection type, content-length, content-type, timestamp of execution and the server’s name. The column that



InterQ Data Quality Service T4.4 - OpenAPI/Swagger

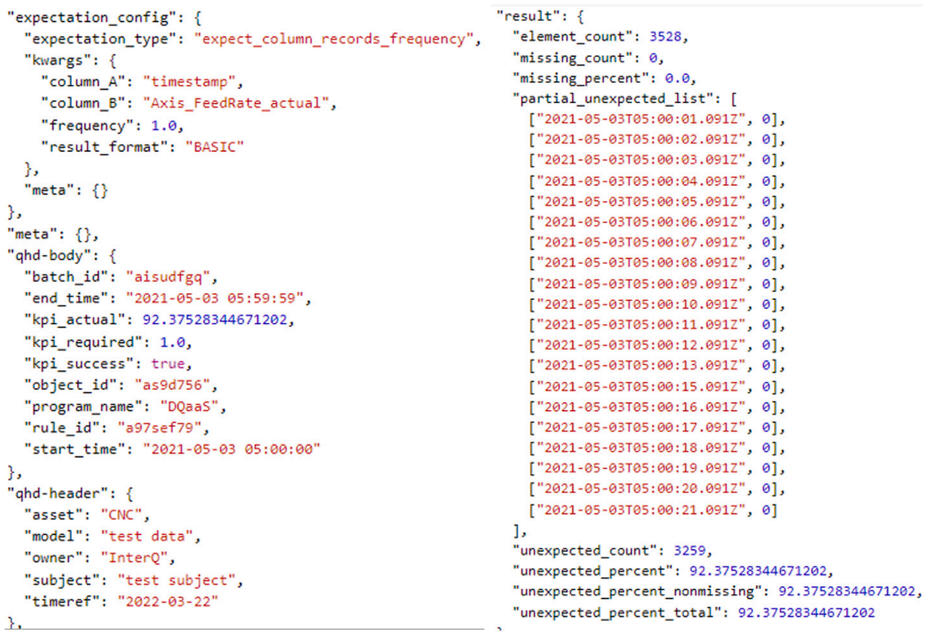
Missing records

InterQ-DNV-DQaaS OpenAPI - Missing records

Query parameters

file_name: interq.csv
ts_column: timestamp
value_column_1: Axis_FeedRate_actual
value_1: 1
qhd_key: interq_qhd

Fig. 12. Description of parameter for Missing records.



```
"expectation_config": {
  "expectation_type": "expect_column_records_frequency",
  "kwargs": {
    "column_A": "timestamp",
    "column_B": "Axis_FeedRate_actual",
    "frequency": 1.0,
    "result_format": "BASIC"
  },
  "meta": {}
},
"meta": {},
"qhd-body": {
  "batch_id": "aisudfgq",
  "end_time": "2021-05-03 05:59:59",
  "kpi_actual": 92.37528344671202,
  "kpi_required": 1.0,
  "kpi_success": true,
  "object_id": "as9d756",
  "program_name": "DQaaS",
  "rule_id": "a97sef79",
  "start_time": "2021-05-03 05:00:00"
},
"qhd-header": {
  "asset": "CNC",
  "model": "test data",
  "owner": "InterQ",
  "subject": "test subject",
  "timeref": "2022-03-22"
},
```

```
"result": {
  "element_count": 3528,
  "missing_count": 0,
  "missing_percent": 0.0,
  "partial_unexpected_list": [
    ["2021-05-03T05:00:01.091Z", 0],
    ["2021-05-03T05:00:02.091Z", 0],
    ["2021-05-03T05:00:03.091Z", 0],
    ["2021-05-03T05:00:04.091Z", 0],
    ["2021-05-03T05:00:05.091Z", 0],
    ["2021-05-03T05:00:06.091Z", 0],
    ["2021-05-03T05:00:07.091Z", 0],
    ["2021-05-03T05:00:08.091Z", 0],
    ["2021-05-03T05:00:09.091Z", 0],
    ["2021-05-03T05:00:10.091Z", 0],
    ["2021-05-03T05:00:11.091Z", 0],
    ["2021-05-03T05:00:12.091Z", 0],
    ["2021-05-03T05:00:13.091Z", 0],
    ["2021-05-03T05:00:15.091Z", 0],
    ["2021-05-03T05:00:16.091Z", 0],
    ["2021-05-03T05:00:17.091Z", 0],
    ["2021-05-03T05:00:18.091Z", 0],
    ["2021-05-03T05:00:19.091Z", 0],
    ["2021-05-03T05:00:20.091Z", 0],
    ["2021-05-03T05:00:21.091Z", 0]
  ],
  "unexpected_count": 3259,
  "unexpected_percent": 92.37528344671202,
  "unexpected_percent_nonmissing": 92.37528344671202,
  "unexpected_percent_total": 92.37528344671202
```

Fig. 13. Output of Missing Records rule.

is used as the input is “Axis_FeedRate_actual” and the result shows that the input data is of type “int64”. Both the input and output are indicated as red colour boxes in Fig. 16. The yellow colour box indicate the quality hallmark document header. The information contained in the “qhd-header” is stored in the blockchain and the information contained in the “qhd-body” is stored in the file storage.

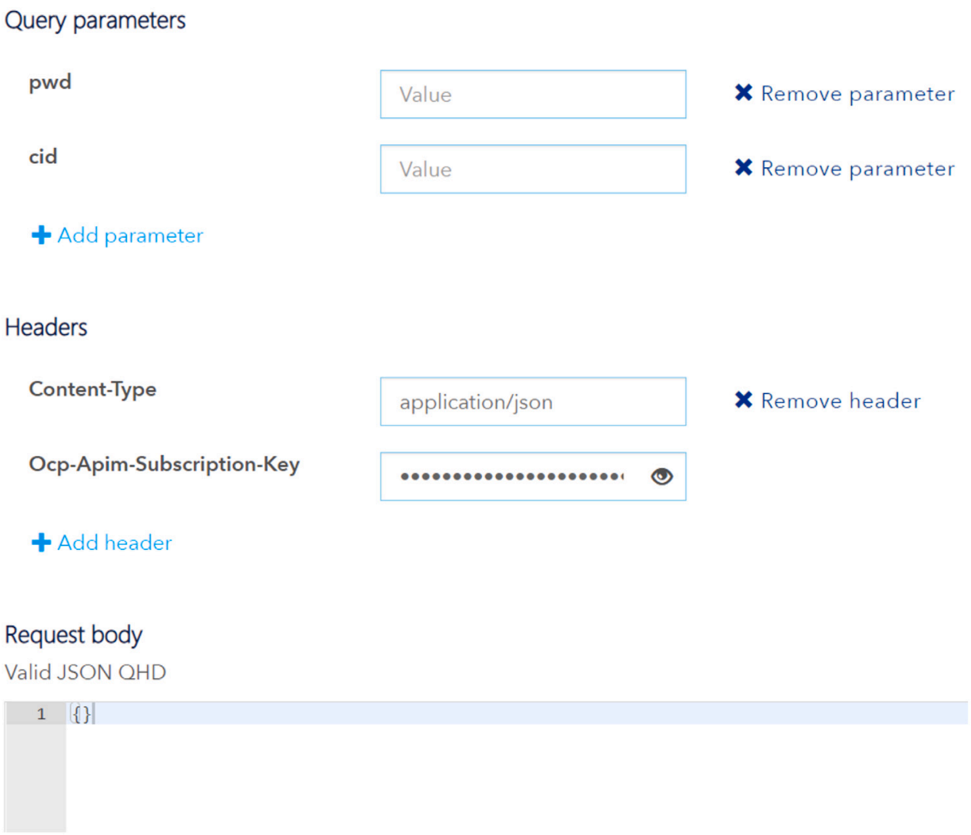
7.2. Great expectation custom rules

We have chosen the “DuplicateRecords” rule as an example of Custom GE rule and when the “Try it out” button on the right-hand side is pressed, the window expands, and default parameters are provided in the text boxes (refer Fig. 17). The “value_1” parameter is required only for certain rules where a threshold value is required. In our case we do not require a threshold value. This rule only requires “timestamp” as the input. Once the execute button is pressed, we get the output for the default parameters provided as the input (which is shown in Fig. 18).

The output is in JSON format which also contains the quality hallmark definition. The output screen shows a partial expected list which contains the duplicate records of the timestamp column, and total number of duplicate records in the timestamp column (which is indicated by “unexpected_count”: 3 in our case). Additionally, the percentage of duplicate records is displayed. These are indicated as red colour boxes in the figure below.

7.3. QHDKey

A system generated QHD-key can be obtained for a stakeholder’s own asset and model. The required parameters can be entered as shown in Fig. 19. The “owner” parameter requires the name of the owner who is accountable for the quality assessment measurements/analysis reported. The “asset” parameter requires the name of the asset (for example, a product item or a machine) that will undergo the quality assessment. The “model” parameter requires the name of a specific assessment model that QHD conforms to (that is, the notion of what is measured and how it is measured). It can be inferred as a “type” or “class” of quality assessments because it defines the quality indicators



Query parameters

pwd | Value | ✖ Remove parameter

cid | Value | ✖ Remove parameter

✚ Add parameter

Headers

Content-Type | application/json | ✖ Remove header

Ocp-Apim-Subscription-Key | •••••••••••••••••••••

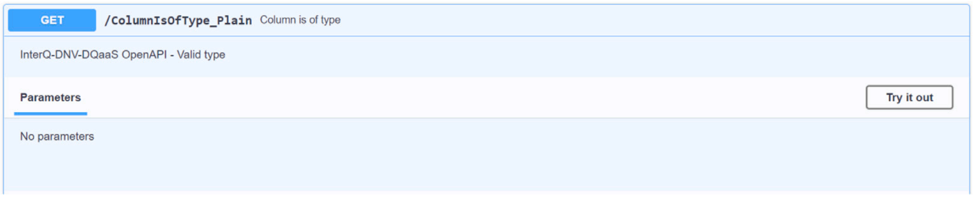
✚ Add header

Request body

Valid JSON QHD

```
1 {}
```

Fig. 14. Submit QHD in Veracity API gateway.



GET /ColumnIsOfType_Plain Column is of type

InterQ-DNV-DQaaS OpenAPI - Valid type

Parameters | Try it out

No parameters

Fig. 15. ColumnIsOfType Rule.

to be used in every individual assessment that belongs to that type. The “subject” parameter requires the name of the target subject that will undergo the quality assessment. The target subject could be a product, process, or a dataset.

After the parameters have been entered, the service will generate a “qhd_key”, which is indicated by a red coloured box in Fig. 20. The “qhd_key” allows DQaaS to identify the QHD containing the asset, model, owner, subject and time reference. The provided “qhd_key” in the “qhd_key” parameter needs to manually pasted in any rule to run. With the QHD key, users could reuse the same information stored in the key, and do not need to enter the owner, asset, model, and subject data each time when running the rules.

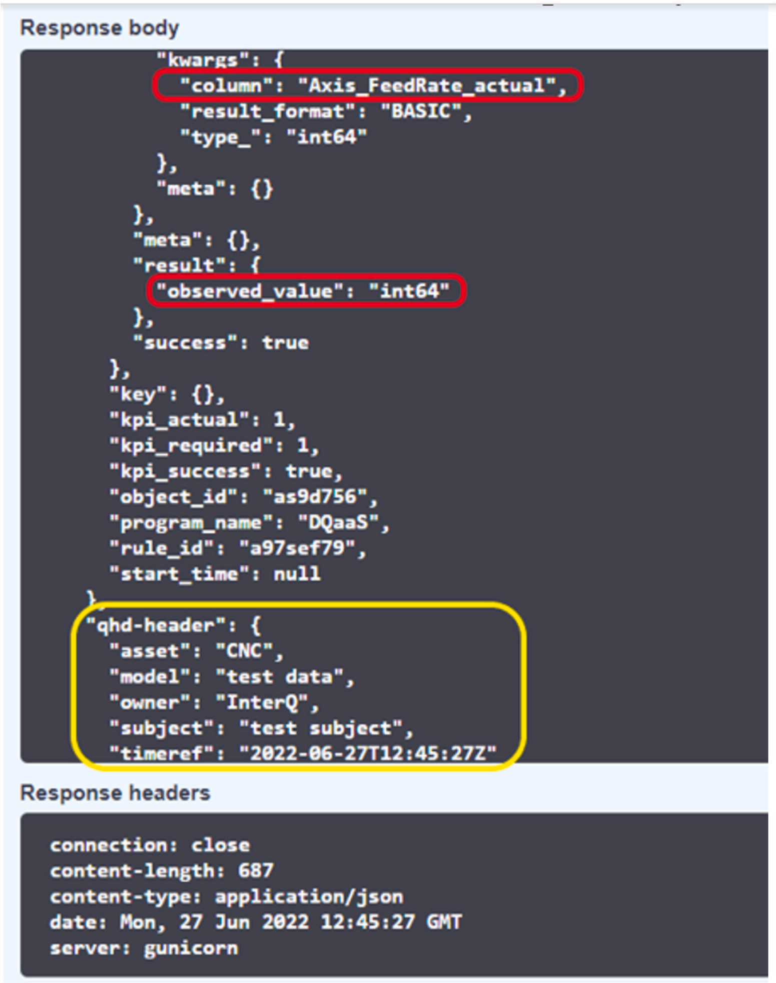
Finally, the fourth rule is “QualityHallmarkDoc” that supports the user with publishing the QHD to the TF’s blockchain network. The required parameters are shown in Fig. 21. The “qhd_json” parameter requires a valid QHD in JSON format. The QHD can either be manually typed in the JSON format or could be copied from the response body (after running a required rule) and pasted in the provided text box. The “pwd” parameter requires a password that unlocks the iWallet installed on the Personal Node. The “cid” parameter requires an ID of the identity of the user to be impersonated. As presented in Section 6, their values must be provided by the caller on every call, so all OpenAPI operations define these two additional, mandatory parameters.

In summary, we have demonstrated the data quality services and how they can publish the QHD to the TF’s blockchain network. The QHD can be queried by different stakeholders, who may want to check the quality metrics of the data being shared.

8. Discussion

The TF impacts the enterprise of zero-defect manufacturing in a number of dimensions as follows:

Traceability: Traceability is the ability to track quality hallmark documents (QHD) of process, product, and data quality (PPD) across the manufacturing life-cycle of one or more products manufactured by one or more companies. The TF allows controlled access to PPD information as QHD in JSON format on an immutable distributed ledger technology (DLT) with unique identifiers and time stamps. The TF is implemented



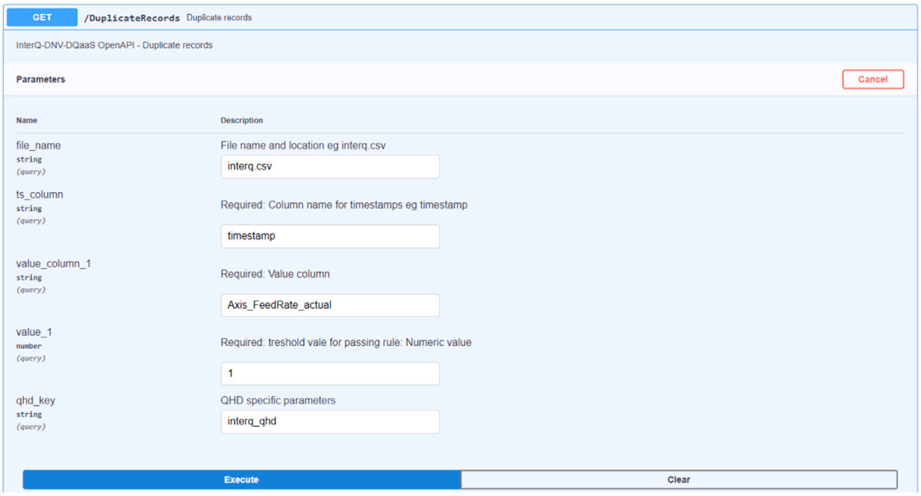
Response body

```
      "kwargs": {
        "column": "Axis_FeedRate_actual",
        "result_format": "BASIC",
        "type_": "int64"
      },
      "meta": {}
    },
    "meta": {},
    "result": {
      "observed_value": "int64"
    },
    "success": true
  },
  "key": {},
  "kpi_actual": 1,
  "kpi_required": 1,
  "kpi_success": true,
  "object_id": "as9d756",
  "program_name": "DQaaS",
  "rule_id": "a97sef79",
  "start_time": null
}
"qhd-header": {
  "asset": "CNC",
  "model": "test data",
  "owner": "InterQ",
  "subject": "test subject",
  "timeref": "2022-06-27T12:45:27Z"
```

Response headers

```
connection: close
content-length: 687
content-type: application/json
date: Mon, 27 Jun 2022 12:45:27 GMT
server: gunicorn
```

Fig. 16. Output of ColumnIsOfType rule.



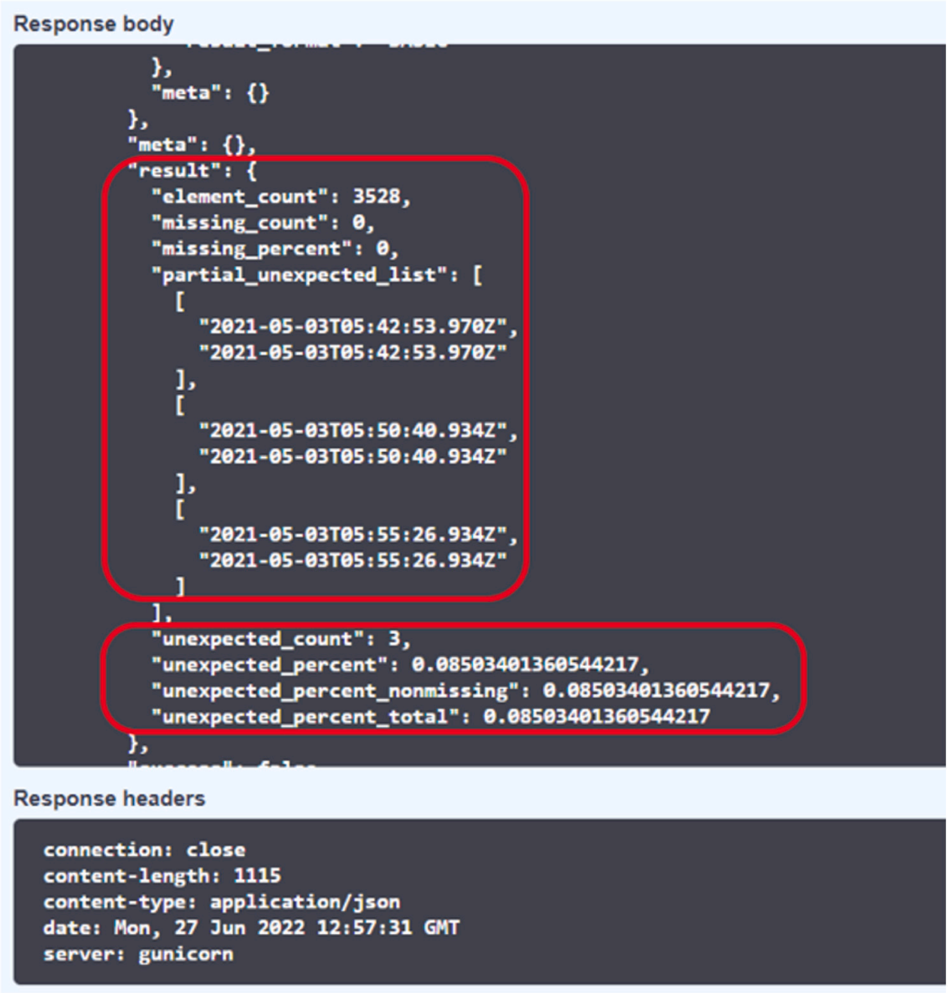
GET
/DuplicateRecords Duplicate records
InterQ-DNV-DQaaS OpenAPI - Duplicate records
Parameters
Cancel
Execute
Clear

Name	Description
file_name string *(query)*	File name and location eg interq.csv interq.csv
ts_column string *(query)*	Required: Column name for timestamps eg timestamp timestamp
value_column_1 string *(query)*	Required: Value column Axis_FeedRate_actual
value_1 number *(query)*	Required: treshold vale for passing rule: Numeric value 1
qhd_key string *(query)*	QHD specific parameters interq_qhd

Fig. 17. Default parameters for DuplicateRecords rule.

using a permissioned open-source blockchain framework called Hyperledger Fabric[5] that can be queried using the default LevelDB that stores chaincode data as simple key–value pairs and only supports key, key range, and composite key queries. The TF also supports CouchDB as an alternate state database that allows us to issue rich queries against data values rather than the keys in the QHD JSON documents. These query engines allow us to trace PPD information in QHDs based on timestamps and hierarchical unique identifiers that interlink process,

[5] https://www.hyperledger.org/use/fabric



```
          "result_format": "BASIC"
        },
        "meta": {}
      },
      "meta": {},
      "result": {
        "element_count": 3528,
        "missing_count": 0,
        "missing_percent": 0,
        "partial_unexpected_list": [
          [
            "2021-05-03T05:42:53.970Z",
            "2021-05-03T05:42:53.970Z"
          ],
          [
            "2021-05-03T05:50:40.934Z",
            "2021-05-03T05:50:40.934Z"
          ],
          [
            "2021-05-03T05:55:26.934Z",
            "2021-05-03T05:55:26.934Z"
          ]
        ],
        "unexpected_count": 3,
        "unexpected_percent": 0.08503401360544217,
        "unexpected_percent_nonmissing": 0.08503401360544217,
        "unexpected_percent_total": 0.08503401360544217
      },
      "success": false
```

```
connection: close
content-length: 1115
content-type: application/json
date: Mon, 27 Jun 2022 12:57:31 GMT
server: gunicorn
```

Fig. 18. Output of DuplicateRecords rule.

product, and data quality that are from the same manufacturing process. This hierarchical identity management system in TF is critical to the traceability of related quality hallmarks.

Accountability: In manufacturing, accountability is defined as all employees are to be held accountable to the same standard or range of performance throughout the facility. Fewer humans operate highly automated factories and are responsible for the quality of the process, data acquisition, and most importantly product quality. The TF enables us to trace quality hallmarks in time and in space (station in the production line) to produce an accurate representation of quality in a plant. Metrics derived from these quality hallmarks hold all the factory workers accountable for the current state of production.

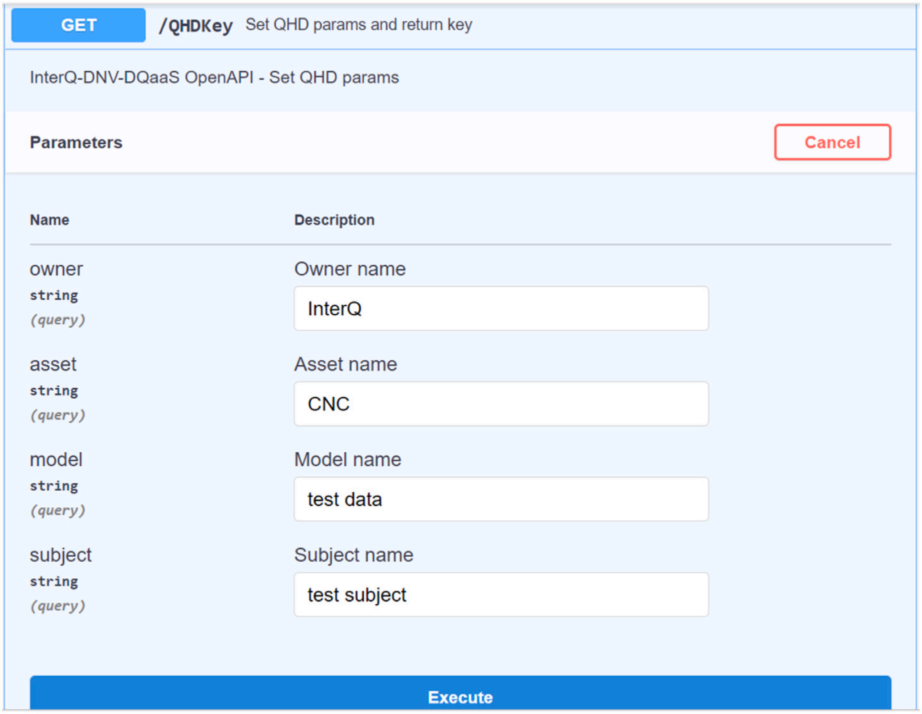
Trust: Blockchain promotes transparency and trust within manufacturing companies and users by providing relatively unfettered access to their records in a distributed ledger. This greatly helps cutting time and effort made on endless to-ing and fro-ing. The TF provides a distributed ledger that is replicated across all companies and users. Anyone with access can query the ledger to identify the root cause for both poor or good quality across a complex network of stakeholders. This type of transparency enhances trust and can then be used to negotiate clearer contracts and direct budget to improve parts of a production line that need to improve.

Certifiability: A quality manufacturing certification is a certification that manufacturing companies can receive, typically through third parties, that certifies that the company is keeping up with the industry's quality standards. For instance, Clause 9 of the ISO 9001:2015 standard requires a manufacturing company to measure and analyse processes and record results in order to demonstrate it conforms to the requirements of the ISO 9001 standard, ensure it applies all aspects of its Quality Management System, and support continual improvement in quality management across the company. The quality hallmarks recorded on the distributed ledger greatly eases the process of certification from a third-party certification agency.

Standardization: Standardization is a framework of agreements to which all relevant parties in an industry or organization must adhere to ensure that all processes associated with the creation of a good or performance of a service are performed within set guidelines. The concepts in the TF bring new contributions to several standards. For instance, data quality hallmarks (e.g. metrics on completeness, accuracy, timeliness) in the TF contributes to the ISO8000 standard on data quality. Furthermore, DLT in the TF contributes to the GS1 standards for end-to-end traceability and the ISO9000 standard for quality management.

Quality Culture: The term quality culture refers to the goal of a company and its members to permanently ensure and sustainably develop quality. Real-time feedback and visualization of quality metrics in the TF on a daily/regular basis can promote a quality culture in a company where factory workers are made aware of PPD quality and how they are associated to each other. It is increasingly a common practice now to present quality metrics on screens to inform factory workers.

Fault Diagnosis: Fault diagnosis is pinpointing one or more root causes of problems, to the point where corrective action can be taken. The chain of events recorded in the TF can be used to verify if PPD



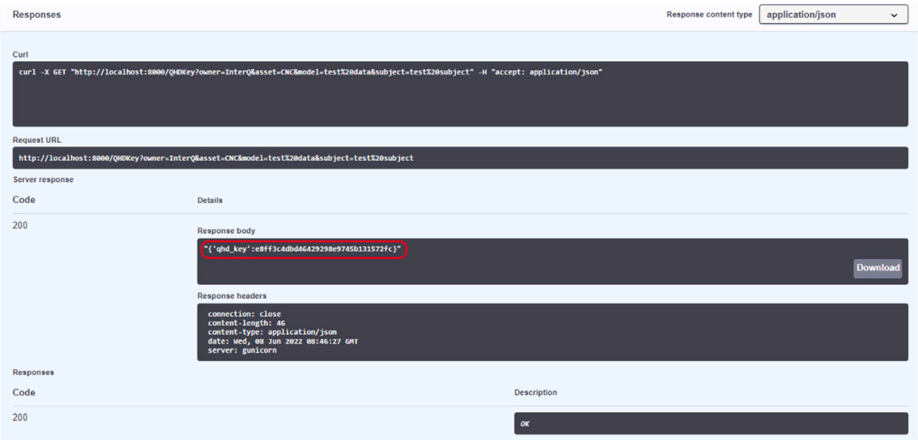
/QHDKey Set QHD params and return key

InterQ-DNV-DQaaS OpenAPI - Set QHD params

Parameters

Name	Description
owner string *(query)*	Owner name InterQ
asset string *(query)*	Asset name CNC
model string *(query)*	Model name test data
subject string *(query)*	Subject name test subject

Fig. 19. QHDKey rule.



Responses

Response content type: application/json

Curl

```
curl -X GET "http://localhost:8000/QHDKey?owner=InterQ&asset=CNC&model=test%20data&subject=test%20subject" -H "accept: application/json"
```

Request URL

```
http://localhost:8000/QHDKey?owner=InterQ&asset=CNC&model=test%20data&subject=test%20subject
```

Server response

Code	Details
200	Response body

```
"{'qhd_key':e8ff3c4dbd46429298e9745b131572fc}"
```

Response headers

```
connection: close
content-length: 46
content-type: application/json
date: Wed, 08 Jun 2022 08:46:27 GMT
server: gunicorn
```

Responses

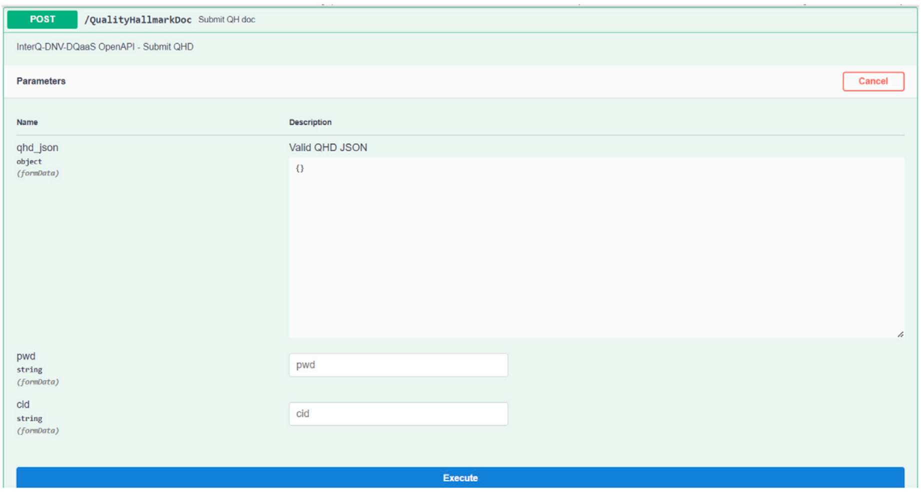
Code	Description
200	*OK*

Fig. 20. Output of QHDkey rule.

hallmarks were reported twice (similar to double spending in crypto blockchains), checking for consensus among stakeholders to ensure that all of them have the same PPD hallmarks, identifying sensor faults in the production line, and tracking device locations with faults using the PPD hallmarks

Security: The permissioned DLT ensures security in transactions between companies and users due its principles of cryptography, decentralization, and consensus. The transactions in the TF are typically quality hallmarks which can be seen as a form of guarantee/assurance reflecting the performance of a company/users involved in manufacturing, data acquisition, and metrology. First, an user or a company installs a Trusted Node which is deployed in with the scope of the TF network. The identity management service on the node is a Blockchain smart contract used to authenticate TF users. Owners of trusted nodes can write JSON files containing QHD. It is very important that these QHD are not tampered with after they have been recorded on the TF as it may be used to endanger to reputation of an user or cause roadblocks to certification. TF implements immutability, a concept from append-only computing where one may only add new hallmarks to the DLT but cannot erase previous entries. It is possible to add a hallmark with a correction but it is not possible to modify older entries. Moreover, entries into the DLT can only be recorded when there is consensus as verified by smart contracts. The replication of the ledger across multiple parties also ensures that hallmark entries are consistent across all stakeholders.

Privacy: The permissioned DLT greatly enhances privacy by using cryptographic techniques that can help protect sensitive information from unauthorized access, enhance pseudonymity of stakeholders' privacy by preventing their company information from being associated with their PPD hallmarks on the ledger if need be, data fragmentation across multiple nodes of the network can help protect the data from being accessed by a single malicious entity, support zero-knowledge proof to validate the correctness of a PPD hallmark without revealing any information about the statement, enable privacy-preserving protocols like zk-SNARKs allowing stakeholders to interact with the contract



POST /QualityHallmarkDoc Submit QH doc

InterQ-DNV-DQaaS OpenAPI - Submit QHD

Parameters

Cancel

Name	Description
qhd_json object *(formData)*	Valid QHD JSON {}
pwd string *(formData)*	pwd
cid string *(formData)*	cid

Execute

Fig. 21. Submitting a QHD.

without revealing their identities or the details of their interactions, and be implemented in a private network, where only authorized stakeholders have access to the PPD hallmarks, this makes it hard for outside parties to access the hallmarks.

Zero-waste: The DLT can provide a tamper-proof record of all transactions and interactions within the supply chain, allowing stakeholders to track the origin and movement of parts, identify any inefficiencies, and reduce waste. By using smart contracts to automate reporting PPD hallmarks, it could be possible to ensure that parts and materials meet certain quality standards, helping to prevent waste caused by defective or non-compliant products. Analysis of PPD hallmarks on the DLT can enable predictive maintenance and schedule repairs accordingly, reducing downtime and the associated waste. The DLT can facilitate collaboration among stakeholders in the supply chain, allowing them to share data and identify potential cross-organizational inefficiencies, reducing waste and optimize their operations.

9. Conclusions

In this paper, we have presented the Product – Process – Data (PPD) quality hallmark (QH) and the distributed ledger-based Trusted Framework (TF) that enable the exchange of trustworthy and traceable quality data between factories (of the same or other enterprises) in the product supply chain. As data is key, the quality of critical data must be checked. The meta-data about the quality of data used for the assessment must be also shared and trusted. We have shown how data quality services as a representative type services that share data quality metrics using the TF. The TF is a concrete service infrastructure that meets the generic requirements of traceability, trust and security for the specific goal of collaborative PPD QH management, for secure horizontal- and vertical quality data integration. Future work will be dedicated to further implementation of the PPD QH and the TF to detect the practical limitations and enrich the concept. So far, we have mainly done experiments of the TF using the data quality services. Along with the progress of our project, we will improve our solution by integrating new applications to build PPD QH having the harmonization of product quality and process quality to interchange PPD quality information through the supply chain.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: The authors Phu Nguyen, Arda Goknil, Simeon Tverdal, Erik Johannes Husom, Sagar Sen are colleagues at SINTEF with the following Guest editors:

Prof. Dr. Daryl Powell (daryl.powell@sintef.no)

Dr. Odd Myklebust (odd.myklebust@sintef.no)

The authors Phu Nguyen, Arda Goknil, Simeon Tverdal, Erik Johannes Husom, Sagar Sen currently work with the following Guest editors in the same research project Horizon 2020 DAT4.Zero.

Prof. Dr. Daryl Powell (daryl.powell@sintef.no)

Dr. Maria Chiara Magnanini (mariachiara.magnanini@polimi.it)

Dr. Odd Myklebust (odd.myklebust@sintef.no)

Data availability

The data that has been used is confidential.

Acknowledgements

The research leading to these results has received funding from the European Union’s Horizon 2020 Research and Innovation programme under Grant Agreement No. 958357 (InterQ).

References

Ahmed, M., Taconet, C., Ould, M., Chabridon, S., Bouzeghoub, A., 2021. IoT data qualification for a logistic chain traceability smart contract. Sensors 21 (6).

Akkaoui, R., Hei, X., Cheng, W., 2020. EdgeMediChain: a hybrid edge blockchain-based framework for health data exchange. IEEE Access 8, 113467–113486.

Atluri, V., Hong, Y., Chun, S.A., 2020. Security, privacy and trust for responsible innovations and governance. In: The 21st Annual International Conference on Digital Government Research. In: dg.o ’20, Association for Computing Machinery, New York, NY, USA, pp. 365–366. http://dx.doi.org/10.1145/3396956.3396978.

Bai, L., Hu, M., Liu, M., Wang, J., 2019. BPIIoT: A light-weighted blockchain-based platform for industrial IoT. IEEE Access 7, 58381–58393.

Bajoudah, S., Dong, C., Missier, P., 2019. Toward a decentralized, trust-less marketplace for brokered IoT data trading using blockchain. In: 2019 IEEE International Conference on Blockchain (Blockchain). pp. 339–346.

Bartol, J., Souvent, A., Suljanović, N., Zajc, M., 2020. Secure data exchange between IoT endpoints for energy balancing using distributed ledger. In: 2020 IEEE PES Innovative Smart Grid Technologies Europe. ISGT-Europe, pp. 56–60.

Casado-Vara, R., de la Prieta, F., Prieto, J., Corchado, J.M., 2018. Blockchain framework for IoT data quality via edge computing. In: BlockSys’18. pp. 19–24.

Cassoli, B.B., Jourdan, N., Nguyen, P.H., Sen, S., Garcia-Ceja, E., Metternich, J., 2022. Frameworks for data-driven quality management in cyber-physical systems for manufacturing: A systematic review. In: 15th CIRP Conference on Intelligent Computation in ManufacturingEngineering, 14–16 July 2021. Procedia CIRP 112, 567–572. http://dx.doi.org/10.1016/j.procir.2022.09.062, URL https://www.sciencedirect.com/science/article/pii/S2212827122012252.

Colledani, M., Coupek, D., Verl, A., Aichele, J., Yemane, A., 2014. Design and evaluation of in-line product repair strategies for defect reduction in the production of electric drives. Procedia CIRP 21, 159–164. http://dx.doi.org/10.1016/j.procir.2014.03.186, 24th CIRP Design Conference. URL https://www.sciencedirect.com/science/article/pii/S2212827114007525.

Dedeoglu, V., Jurdak, R., Putra, G.D., Dorri, A., Kanhere, S.S., 2019. A trust architecture for blockchain in IoT. In: MobiQuitous'19. pp. 190–199.

Eger, F., Coupek, D., Caputo, D., Colledani, M., Penalva, M., Ortiz, J.A., Freiberger, H., Kollegger, G., 2018. Zero defect manufacturing strategies for reduction of scrap and inspection effort in multi-stage production systems. In: 11th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 19–21 July 2017, Gulf of Naples, Italy. Procedia CIRP 67, 368–373. http://dx.doi.org/10.1016/j.procir.2017.12.228, URL https://www.sciencedirect.com/science/article/pii/S2212827117311721.

Husom, E.J., Tverdal, S., Goknil, A., Sen, S., 2022. UDAVA: An unsupervised learning pipeline for sensor data validation in manufacturing. In: Proceedings of the 1st International Conference on AI Engineering: Software Engineering for AI. pp. 159–169.

IBM, 2022. What is blockchain technology?. URL https://www.ibm.com/topics/what-is-blockchain. (Accessed 23 February 2022).

Jernigan, S., Kiron, D., Ransbotham, S., 2016. Data sharing and analytics are driving success with iot. MIT Sloan Manage. Rev. 58 (1).

Kang, J., Yu, R., Huang, X., Wu, M., Maharjan, S., Xie, S., Zhang, Y., 2018. Blockchain for secure and efficient data sharing in vehicular edge computing and networks. IEEE Internet Things J. 6 (3), 4660–4670.

Lopez, D., Farooq, B., 2020. A multi-layered blockchain framework for smart mobility data-markets. Transp. Res. C 111, 588–615.

Ma, Z., Wang, L., Zhao, W., 2020. Blockchain-driven trusted data sharing with privacy protection in IoT sensor network. IEEE Sens. J. 21 (22), 25472–25479.

Magnanini, M.C., Colledani, M., Caputo, D., 2020. Reference architecture for the industrial implementation of zero-defect manufacturing strategies. In: 53rd CIRP Conference on Manufacturing Systems 2020. Procedia CIRP 93, 646–651. http://dx.doi.org/10.1016/j.procir.2020.05.154, URL https://www.sciencedirect.com/science/article/pii/S2212827120305308.

Manogaran, G., Alazab, M., Shakeel, P.M., Hsu, C.-H., 2021. Blockchain assisted secure data sharing model for internet of things based smart industries. IEEE Trans. Reliab. 71 (1), 348–358.

Monrat, A.A., Schelén, O., Andersson, K., 2019. A survey of blockchain from the perspectives of applications, challenges, and opportunities. IEEE Access 7, 117134–117151. http://dx.doi.org/10.1109/ACCESS.2019.2936094.

Nguyen, D.C., Pathirana, P.N., Ding, M., Seneviratne, A., 2021. A cooperative architecture of data offloading and sharing for smart healthcare with blockchain. In: 2021 IEEE International Conference on Blockchain and Cryptocurrency. ICBC, pp. 1–8.

Nguyen, P.H., Sen, S., Jourdan, N., Cassoli, B., Myrseth, P., Armendia, M., Myklebust, O., 2022. Software engineering and AI for data quality in cyber-physical systems - SEA4dq'21 workshop report. SIGSOFT Softw. Eng. Notes 47 (1), 26–29. http://dx.doi.org/10.1145/3502771.3502781, https://doi.org/10.1145/3502771.3502781.

Patel, H., Shrimali, B., 2021. AgriOnBlock: Secured data harvesting for agriculture sector using blockchain technology. ICT Express.

Powell, D., Magnanini, M.C., Colledani, M., Myklebust, O., 2022b. Advancing zero defect manufacturing: A state-of-the-art perspective and future research directions. Comput. Ind. 136, 103596. http://dx.doi.org/10.1016/j.compind.2021.103596, URL https://www.sciencedirect.com/science/article/pii/S0166361521002037.

Powell, D.J., Romero, D., Gaiardelli, P., 2022a. New and renewed manufacturing paradigms for sustainable production. Sustainability 14 (3), http://dx.doi.org/10.3390/su14031279, URL https://www.mdpi.com/2071-1050/14/3/1279.

Rahman, M.U., Baiardi, F., Ricci, L., 2020. Blockchain smart contract for scalable data sharing in IoT: a case study of smart agriculture. In: 2020 IEEE Global Conference on Artificial Intelligence and Internet of Things. GCAIoT, IEEE, pp. 1–7.

Sen, S., Husom, E.J., Goknil, A., Tverdal, S., Nguyen, P., Mancisidor, I., 2022. Taming data quality in AI-enabled industrial internet of things. IEEE Softw. 39 (6), 35–42.

Shafagh, H., Burkhalter, L., Hithnawi, A., Duquennoy, S., 2017. Towards blockchain-based auditable storage and sharing of IoT data. In: Proceedings of the 2017 on Cloud Computing Security Workshop. pp. 45–50.

Shen, B., Guo, J., Yang, Y., 2019. MedChain: Efficient healthcare data sharing via blockchain. Appl. Sci. 9 (6), 1207.

Sunyaev, A., 2020. Distributed ledger technology. In: Internet Computing: Principles of Distributed Systems and Emerging Internet-Based Technologies. Springer International Publishing, Cham, pp. 265–299. http://dx.doi.org/10.1007/978-3-030-34957-8_9, https://doi.org/10.1007/978-3-030-34957-8_9.

Tang, B., Kang, H., Fan, J., Li, Q., Sandhu, R., 2019. Iot passport: A blockchain-based trust framework for collaborative internet-of-things. In: Proceedings of the 24th ACM Symposium on Access Control Models and Technologies. pp. 83–92.