D3.16 – i4Q DATA REPOSITORY v2

WP3 – BUILD: Manufacturing Data Quality

i4Q has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 958205.

Document Information

GRANT AGREEMENT NUMBER	958205	ACRONYM	i4Q
FULL TITLE	Industrial Data Services for Quality Control in Smart Manufacturing		
START DATE	01-01-2021	DURATION	36 months
PROJECT URL	https://www.i4q-project.eu/		
DELIVERABLE	D3.16 i4Q Data Repository v2		
WORK PACKAGE	WP3 – BUILD: Manufacturing Data Quality		
DATE OF DELIVERY	CONTRACTUAL 31-Dec-2022	ACTUAL	30-Dec-2022
NATURE	Other	DISSEMINATION LEVEL	Public
LEAD BENEFICIARY	ITI		
RESPONSIBLE AUTHOR	ITI		
CONTRIBUTIONS FROM	2- ENG, 5-KBZ, 11-UNI, 12-TIAG, 22-RIAS		
TARGET AUDIENCE	1) i4Q Project partners; 2) industrial community; 3) other H2020 funded projects; 4) scientific community		
DELIVERABLE CONTEXT/ DEPENDENCIES	This deliverable is a public document developed as part of *"Task 3.5 - Manufacturing Data Storage and Use"*, that presents a technical overview of the i4Q Data Repository solution (i4Q^{DR}). This deliverable is the second version of D3.8 "i4Q Data Repository", that was delivered at M18. D3.16 receives input from deliverables D1.9, D2.6, D2.7, D3.7, and D3.15. Moreover, it provides feedback to D3.15.		
EXTERNAL ANNEXES/ SUPPORTING DOCUMENTS	Appendix I of this document contains the i4Q^{DR} web documentation which, indeed, consists of a link to the website including the web documentation of all i4Q solutions.		
READING NOTES	None		
ABSTRACT	This deliverable presents a technical overview of the i4Q Data Repository solution (i4Q^{DR}), including an explanation of its mapping against the i4Q Reference Architecture. Furthermore, it provides specific information regarding its implementation status up to M24, describing developments performed so far. This document is an updated version of D3.8 (submitted in M18) and presents the follow-up of part of the work performed in T3.5.		

Document History

VERSION	ISSUE DATE	STAGE	DESCRIPTION	CONTRIBUTOR
0.1	7-Nov-2022	Draft	First Version of Table of Contents, based on final version of D3.8.	ITI
0.2	28-Nov-2022	Draft	First complete version: content updated with the progress made from M18 to M24. Ready for internal review.	CERTH, ENG, ITI, KBZ, UNI
0.3	30-Nov-2022	Draft	Added new Section 3.5 providing overview on next steps after M24. Ready for internal review.	ITI
0.4	8-Dec-2022	Internal review	Internal review.	RIAS, TIAG
0.5	13-Dec-2022	Draft	Second complete version, including reviewers' comments.	ITI
1.0	30-Dec-2022	Final Document	Final quality check and issue of final document	CERTH

Disclaimer

Copyright message

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS/ACRONYMS

API	Application Programming Interface
DBMS	Data Base Management System
DIT	Data Integration and Transformation
DR	Data Repository
DSS	Decision Support System
GUI	Graphical User Interface
HA	High Availability
HA+Sec	High Availability with Security
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
MB	Message Broker
PDF	Portable Document Format
REST	Representational State Transfer
REST API	RESTful Application Programming Interface
SQL	Structured Query Language
SS	Single Server
SS+Sec	Single Server with Security
TLS	Transport Layer Security

Executive summary

D3.16 delivers the second release of the i4Q **Data Repository** (i4Q^{DR}) solution and presents an executive explanation of this solution. More specifically, it provides a general description of the solution, its technical specifications, and a report on its implementation status. The source code of the i4Q^{DR} solution is available in a private repository of Gitlab at: https://gitlab.com/i4q/dr.

This document is complemented by the technical documentation associated to the i4Q^{DR} solution, which is deployed on the website http://i4q.upv.es. This website contains the information of all the i4Q solutions developed in the project "Industrial Data Services for Quality Control in Smart Manufacturing" (i4Q). The i4Q^{DR} solution's technical documentation is publicly available at http://i4q.upv.es/8_i4Q_DR/index.html and provides information regarding the topics listed below:

- General description
- Features
- Images
- Authors
- Licensing
- Pricing
- Installation requirements
- Installation Instructions
- Technical specifications of the solution
- User manual

This deliverable extends *D3.8 – Data Repository* [1] by explaining the progress made from M19 to M24 in the development of the i4Q^{DR} solution. More specifically, the main advances in this period are:

- The development of new scenarios
- The implementation of a graphical user interface.
- The definition of a more complete user manual, which has been included in the technical documentation.
- And a preliminary integration with the i4Q Message Broker has been implemented, too.

This document, *"D3.16 – i4Q Data Repository v2"*, is an update of v1 (D3.8). For this reason, it contains information of the 1st version together with the updates developed in this 2nd version.

Document structure

Section 1: Contains a general description of the **i4Q Data Repository**, providing an overview and the list of features. It is addressed to final users of the i4Q solution.

Section 2: Contains the technical specifications of the **i4Q Data Repository**, providing an overview and its architecture diagram. It is mainly addressed to software developers.

Section 3: Details the implementation of the **i4Q Data Repository**. This section is the one that has been most modified with respect to D3.8 since the progress performed from M19 to M24 is presented here; more specifically in Section 3.2. However, the rest of the section, whose content was already included in D3.8, has been reorganised and slightly modified for readability purposes. Similarly, as Section 2, the content of this section is mainly addressed to software developers.

Section 4: Provides the conclusions.

APPENDIX I: Provides the **i4Q Data Repository** technical web documentation, which has been updated since M18 with a more complete user manual of the solution. The web documentation can be accessed online at: **http://i4q.upv.es/8_i4Q_DR/index.html**

1. General Description

1.1 Overview

The **i4Q Data Repository** (i4Q^DR^) is a distributed storage system that will oversee receiving, storing, and serving the data in an appropriate way to other solutions. Note that these operations will be performed according to standard data storage system's mechanisms so that no specific data transformations will be applied. Indeed, there is another solution for this purpose, namely the i4Q Data Integration and Transformation Services solution (i4Q^DIT^) which is presented in deliverable D4.9 [2]. The i4Q^DR^ solution is suitable to support and enhance a high degree of digitization in companies with most manufacturing devices acting as sensors or actuators and generating vast amounts of data.

Firstly, the i4Q^DR^ is expected to absorb large volumes of data reaching the system at high volumes and low latencies[1]. However, the demand of computing resources will vary over the time. Thus, the i4Q^DR^ is expected to adapt its computing resources to the actual demand at any given moment, so that it can use additional resources, when necessary. These resources could either be local in the factory manufacturing shopfloor or remote, such as public or private clouds depending on required operational and functional latencies.

Furthermore, the i4Q^DR^ solution provides proper tools for administrators to characterize and transform the data contained inside it. In this regard, the i4Q^DR^ allows exporting data in a different format to the one in which it was stored. Moreover, since the i4Q^DR^ will enable access to different data sources, data from a given data source can be enriched by correlating it with data from another data source. This solution is also useful for data scientists, who can use data stored in the i4Q^DR^ in their experimentations.

Finally, this solution includes some features to enhance secure data management. First, it will oversee data protection, serving as a secure data vault system for the information. This can be achieved by encrypting data, both in flight and at rest. Furthermore, the i4Q^DR^ will be in charge of ensuring administration of regulated access to the data and the related tools, so that only allowed entities are able to do so.

[1] No specific numbers were specified in the DoA of this project or by the industrial partners regarding these two performance features. However, the technologies used by the i4Q^DR^ are state-of-the art technologies already applied in real production scenarios. Thus, they fulfil the performance needs of this project's pilots.

1.2 Features

The i4Q^{DR} will include the features explained below:

- An *access control mechanism*, to ensure that only authorised entities have access to the data and the related tools. i4Q^{DR} administrator users with the appropriate permissions will be able to configure this access control, in order to grant access to the allowed entities.
- Tools and technologies to *manage structured data*. This feature will be offered by means of DBMSs that may be relational SQL-based, document (e.g., JSON-based), general NoSQL tools, etc.
- Tools and technologies to *manage unstructured data i.e., blobs.* This feature will be offered through tools that offer support for blobs like some general-purpose DBMSs or even specific ones (e.g., MinIO[2]).
- Mechanisms to *query* the stored data and retrieve the results of such queries, for both structured data and blobs, in an efficient way.
- Mechanisms to *import/export data* to/from the i4Q^{DR} to ease the interoperability of this solution with others.
- Mechanisms to manage and configure the data repository itself.

[2] MinIO website: https://min.io/

2. Technical Specifications

2.1 Overview

The **i4Q Data Repository** (i4Q^{DR}) is aimed at transversally providing, in a centralized fashion, the functionality related to the storage of data in the whole i4Q system. Indeed, the i4Q^{DR} is involved in all pilots and is expected to interoperate with a large subset of the i4Q solutions.

The central nature of this solution allows taking and applying in a centralized manner some decisions related to the organization, management, and access to the data. For instance, access control criteria and data management policies criteria related to high availability and fault tolerance of the data storage tools, etc. These decisions are required for the proper design and the implementation of access and management mechanisms related to data storage. Therefore, the centralised nature of this solution reduces the complexity of such decision-making processes during implementation.

Moreover, when it comes to putting mechanisms and tools into practice, the centralised nature of the solution avoids effort duplication. More importantly, some problems derived from such duplicity, starting from having divergent criteria and ending in devoting duplicated efforts to the same tasks.

2.2 Architecture Diagram

The i4Q^{DR} solution is mapped against different sub-components of the i4Q Architectural Framework presented in D2.7 [3], which are summarized in an illustrative way in Figure 1. First of all, since the i4Q^{DR} solution covers the data storage requirements of the i4Q system, it is mapped to the *"Data Brokering and Storage"* sub-component of the **Platform Tier**. This mapping is highlighted in green colour in Figure 1.

Secondly, considering the functionalities which the i4Q^{DR} relies on, we can define mappings to some sub-components of the **Edge Tier**, namely, the *"Distributed Computing"*, the *"Data Collection"* and the *"Data Management".* More specifically:

- The mapping to the *"Distributed Computing"*, sub-component provides the i4Q^{DR} the necessary execution environments (containers, virtual machines, etc.) to support data replication.
- The mapping to the *"Data Collection"* sub-component, supplies the flows of data to store and the queries to execute to retrieve data.
- The mapping to the *"Data Management"* sub-component, enables the definition of the structures of the data to store.

These three mappings are highlighted in orange colour in Figure 1.

Finally, taking into account that the i4Q^{DR} provides data storage services to any other solution requiring them, we can define mappings to any sub-component of the **Enterprise** and **Edge Tiers**. In Figure 1, we have highlighted these mappings in red colour.

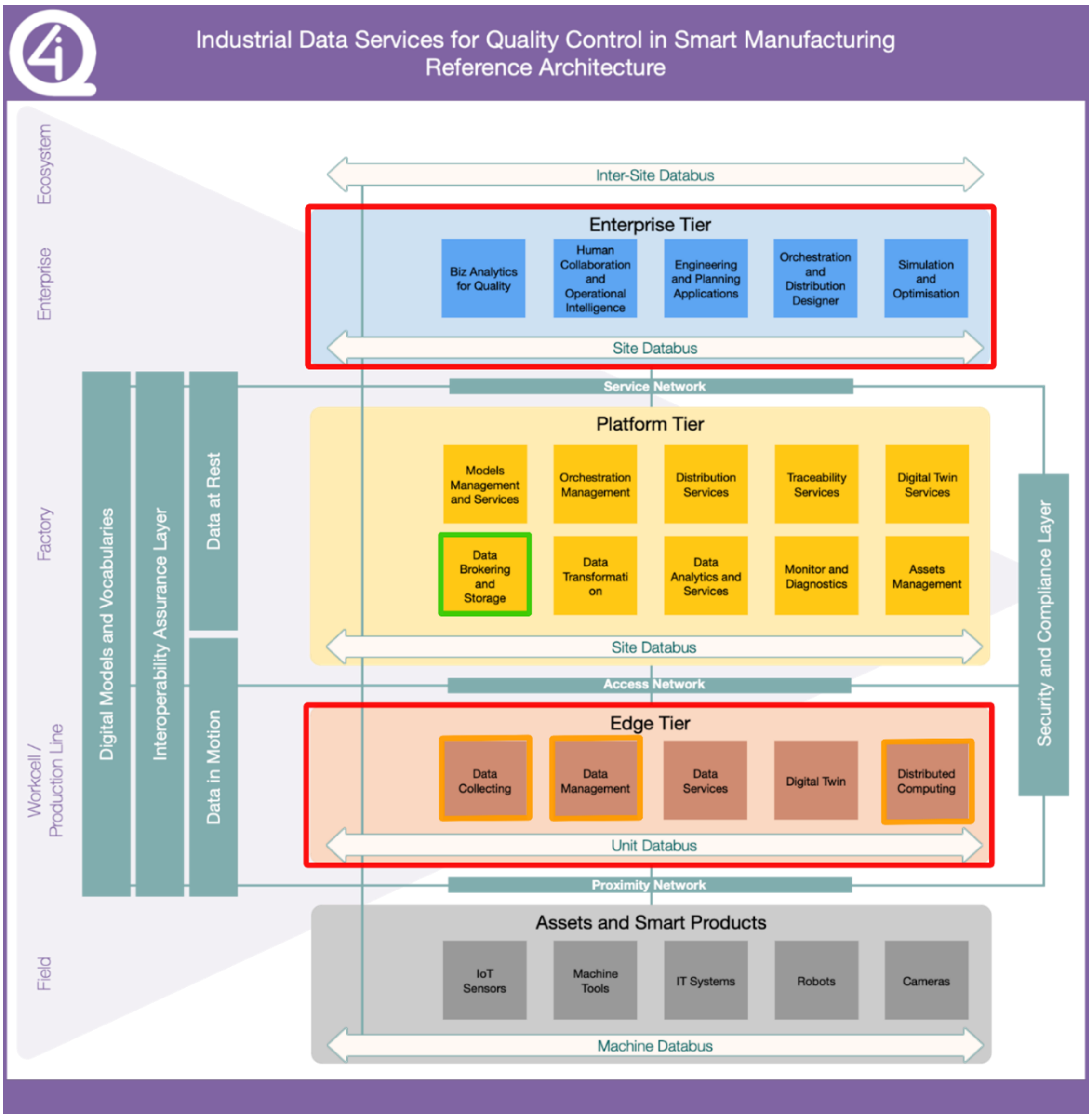


Figure 1. Mappings of i4Q^{DR} against the i4Q Reference Architecture

3. Implementation Status

The implementation of a solution such as the i4Q^{DR} is challenging since, as explained above in Section 2.1., the i4Q^{DR} solution must be interoperable with most of the other solutions of the i4Q system and is involved in all demonstrator pilots. This means that the implementation of the i4Q^{DR} must be flexible enough to adapt to different scenarios and technical requirements but, at the same time, approach them in the most uniform way possible. In *"D3.15* i4Q^{DR} *Guidelines for Building Data Repositories for Industry 4.0 v2"* [4], due on M24, a set of recommendations and design specifications that can be followed to address these challenges and requirements will be presented. The implementation of the i4Q^{DR} is driven by the guidelines that is described in D3.15 [4].

The specifications provided in deliverables D1.9 [5] and D2.7 [3] have driven the implementation of the i4Q^{DR} solution. More specifically, the development work was organised in two phases.

In the first one, the goal was to provide the solution's core features. In this regard, the plan was to develop a *collection of toolkits*, that is, a set of software artifacts, to configure, bootstrap and manage a number of tools and technologies related to the collection, storage, and management of data, supporting different usage scenarios. The purpose of these toolkits was to provide *automatable* mechanisms to configure, deploy, manage, diagnose and undeploy the i4Q^{DR} tools and technologies. Such toolkits provide a basic version of the features described in Section 1.2.

The second phase of the implementation was aimed at providing a more advanced version of the solution. Not only by providing all the features described in Section 1.2, but also by providing them in a more refined way. In this regard, the work performed in the second phase has focused on:

- Offering a more unified interface to the toolkits implemented in the first phase.
- Simplifying and automating the deployment and use of the i4Q^{DR}.
- Improving the communication with the i4Q^{DR}.

A first version of the i4Q^{DR} solution (version v0.1) was released at the end of M18 and reported in D3.8 [1]. This version included most of the work planned for the first phase of the implementation, and some preliminary works of the second phase. The second, and updated version of the solution (version v0.2) will be released at the end of M24, together with this deliverable. Note, however, that the solution's version v0.2 might not be the final one, since it might be necessary to perform some modifications in the context of Task *"T6.8 - Continuous integration and validation"* (for further details see deliverable *"D6.9 - Continuous integration and validation"* [6], due at M24).

The rest of this section provides detailed information regarding the implementation of the two versions of the i4Q^{DR} solution released so far. First, Section 3.1, describes the i4Q^{DR} v0.1, that was released in M18. Then, Section 3.2 explains the advances included in version v0.2, that is, the progress made from M18 till M24. Section 3.3 analyses the features covered by the i4Q^{DR} and maps them to the set of user requirements defined for this solution in [5]. Then, Section 3.4 provides the history of the implementation. Finally, Section 3.5 gives an overview of the next steps planned for after M24.

3.1 Implementation of i4Q^{DR} v0.1 (M18)

For the implementation of i4Q^{DR} v0.1, the first task was to gather the requirements from the pilots and the rest of i4Q solutions interacting with the i4Q^{DR}, in order to select the storage technologies and to identify the different usage scenarios that had to be considered. This document briefly explains the storage technologies and usage scenarios that have been considered for the i4Q^{DR}. The decision-making process and the rationale followed to make such selections will be described in detail in deliverable D3.15 [4] (Section 4), as an illustrative example of the guidelines on how to build and design data repositories for Industry 4.0. With respect to the storage technologies that had to be supported by the i4Q^{DR}solution, the selection was performed based on two criteria. The first one was the need to satisfy the requirements of the project's pilots and the rest of the i4Q solutions. The second one was the wish of covering different types of data storage. More specifically, the selected storage technologies for the i4Q^{DR} are:

- Cassandra[3], a wide-column NoSQL distributed database server, appropriate for managing massive amounts of data.
- *MariaDB*[4], a SQL relational database server, very similar to MySQL.
- *MinIO*[5], which offers a high-performance, S3 compatible object storage. It is used to store files and is compatible with any public cloud.
- *MongoDB*[6], a JSON document-oriented database server.
- *MySQL*[7], a SQL relational database server.
- *Neo4J*[8], a graph database server that allows storing data relationships.
- *PostgreSQL*[9], a SQL relational database server.
- *Redis*[10], an in-memory data structure store, used as a distributed, in-memory key–value database, cache, and message broker, with optional durability.

Concerning the usage scenarios, four of them have been defined, addressing different needs in terms of computing resources and security features. Namely*:*

a) "Single Server" (SS)
b) High Availability" (HA)
c) Single Server with Security" (SS+Sec)
d) "High Availability with Security" (HA+Sec)

The "Single Server" scenarios (a,c) correspond to setups in which only one instance of the storage technology needs to be deployed. The "High Availability" scenarios (b, d) correspond to setups requiring the deployment of a cluster or a set of replicas of the storage technology, which are more suitable for cases with higher requirements in terms of computing resources and fault-

[3] Cassandra. https://cassandra.apache.org
[4] MariaDB. https://mariadb.org
[5] MinIO. https://www.min.io
[6] MongoDB. https://www.mongodb.com
[7] MySQL. https://www.mysql.com
[8] Neo4J. https://www.neo4j.com
[9] PostgreSQL. https://www.postgresql.org
[10] Redis. https://redis.io

tolerance. Finally, the "with Security" scenarios (c,d) refer to configurations involving the TLS protocols, mostly for using x509 certificates, whereas the regular ones do not include such a feature and, thus, are only recommended for development and experimental purposes.

After performing the selection explained above, the goal was to implement several toolkits, each one deploying one of selected storage technologies for each one of the defined scenarios. That is, a toolkit deploying Cassandra for the Single Server scenario, another one deploying Cassandra for the High Availability scenario, and so on. Some of these toolkits have already been implemented and their source code is available at the GitLab's private repository containing the source code of i4Q^{DR} at: https://gitlab.com/i4q/dr. Although each toolkit is closely related to the tools or technologies it handles, all the ones developed so far share a number of principles and characteristics:

- *Human-friendly and automatable configuration.* The toolkits can be configured by means of automatable artifacts like configuration files and environment variables. These artifacts are designed to be easily managed by human users as well as by external tools like shell scripts, automation tools, etc.
- *Human-friendly and automatable operation.* Each toolkit provides one *main shell script* per scenario, so each scenario can be easily bootstrapped by a human user. Moreover, the toolkits are provided in the form of Bash script files that offer a number of functions, which are specific to a given scenario, a given tool or technology or are generic. Thus, they can easily be used in a fully automatable fashion. For instance, they can be integrated in other Bash scripts, used by continuous integration tools, etc.
- *Independence among toolkits*. Each toolkit can be used independently from other toolkits.

More specifically, each one of these toolkits *builds* and *launches* one or more *Docker containers* containing an instance of the corresponding storage technology. In the case of the "Single Server" scenario, only one instance is deployed, whereas more than one are deployed in the case of the "High Availability" scenario. Note, however, that toolkits for any type of scenario may build and run containers for other tools and purposes. For instance, the MongoDB toolkit also deploys a container to run an instance of Mongo Express, a web-based interface to administrate MongoDB instances. The "with Security" scenarios involve the use of the TLS protocol, so that each node uses TLS certificates for authentication purposes when interacting among themselves.

Moreover, the toolkits implemented so far share a similar structure, and consist, mainly, of three sub-components:

- The *storage configuration file*, specifying the value of some properties of the storage technology that can be manually set by the user.
- The *orchestration configuration files*, in which the configuration and properties of the corresponding Docker container(s) are declared.
- A set of *executable files*, which automatically execute all the necessary functions to build and run the corresponding Docker container(s), in a transparent way to the user.

These sub-components allow a potential user to:

- Configure the tool or technology it handles, by means of configuration files and environment variables.

- Prepare the local filesystem (for instance, the directories shared between the local host and the Docker containers).
- Start the Docker containers, provision and configure them and start in them whatever servers that are needed.
- Check the status of the tools and technologies deployed.
- Undeploy and dispose the Docker containers (and the data managed by them, if required).

The efforts to develop the different toolkits have been distributed among the different task partners. Table 1 gathers, for each storage technology ("Storage Tech" column) and scenario (second column), which partner is responsible for the implementation of the corresponding toolkit (see columns "Responsible") and its status by M18, which is specified by the "Status" column. More specifically, three possible values are considered for the "Status" column, namely:

- *"Pending":* denoting that the implementation of this toolkit had not started yet (by M18).
- *"In progress":* indicating that the development of this toolkit had started but was not ready yet.
- *"Done":* showing that the development of the toolkit had finished, and the source code was available at the GitLab repository.

Storage Tech	Scenario	Responsible	Status
Cassandra	SS	ITI	Done
	HA	ITI	Pending
	SS+Sec	ITI	Pending
	HA+Sec	ITI	Pending
MariaDB	SS	ENG	Done
	HA	ENG	Pending
	SS+Sec	ENG	In Progress
	HA+Sec	ENG	Pending
MinIO	SS	ITI	Done
	HA	ITI	Done
	SS+Sec	KBZ	In progress[11]
	HA+Sec	KBZ	In Progress
MongoDB	SS	ITI	Done
	HA	ITI	Done
	SS+Sec	ITI	Done
	HA+Sec	ITI	Done
MySQL	SS	ITI	Done

[11] In D3.8 the status for this scenario was mistakenly reported as "Done".

	HA	ENG	Pending[12]
	SS+Sec	ITI	Done
	HA+Sec	ENG	Pending[12]
Neo4j	SS	ITI	Done
	HA	ITI	Pending
	SS+Sec	ITI	Done
	HA+Sec	ITI	Pending
PostgreSQL	SS	UNI	Done
	HA	UNI	Pending
	SS+Sec	UNI	In Progress
	HA+Sec	UNI	Pending
Redis	SS	ITI	Done
	HA	UNI	Pending
	SS+Sec	ITI	Done
	HA+Sec	UNI	Pending

Table 1. Summary of toolkits' implementation status by M18.

i4Q^{DR} v0.1 included a preliminary experiment of one of the developments planned for the second implementation phase, namely the implementation of a more unified interface to the toolkits. For this purpose, a tool called Trino was integrated into the toolkit for MongoDB for the "Single Server with Security" scenario.

Since most of the work related to the second phase of the implementation was performed after M18, the explanation of that integration is provided in Section 3.2.

3.2 Implementation of i4Q^{DR} v0.2 (M24)

The main work related to the second implementation phase started in M19. As mentioned before, the main focus of this phase was to provide the features implemented in i4Q^{DR} v0.1, but in a more sophisticated way. The progress included in v0.2 with respect to version v0.1 can be summarised as follows:

1. Additional demonstration scenarios were included.
2. A new interoperability layer was implemented on top of the different toolkits using Trino [7]. The purpose of this layer is two-fold. On the one hand, to improve the interoperability of the i4Q^{DR} with other i4Q solutions and, on the other hand, to facilitate the support to other storage technologies in the future, if necessary.

[12] In D3.8 the status for these scenarios was mistakenly reported as "Done".

3. A graphical user interface was developed and implemented to allow for a better simplified user experience. This was a suggestion reported during the mid-term review of the project, in September 2022.
4. A preliminary functional integration with the i4Q Message Broker was implemented to enhance the interoperability of the i4Q^{DR}.

These advances are explained in detail in the rest of this section.

3.2.1 New scenarios

First, the new scenarios included i4Q^{DR} v0.2 were the SS+Sec scenario in MariaDB, MinIO and PostgreSQL toolkits. Additionally, a new storage technology has been included during M24 to cover a possible requirement for the generic pilot defined in T6.7, namely, TimeScaleDB[13], a time-series SQL database which is an extension of PostgreSQL. For this technology the SS and SS+Sec scenarios have been implemented.

The information regarding the new scenarios implemented in i4Q^{DR} v0.2 is shown in Table 2, which is quite similar to Table 1 but reflects the updates performed from M19 to M24. More specifically, the status of the new scenarios is now "Done" and the corresponding cell at that column is highlighted in light purple colour. The status of the other scenarios, which have status "Pending" in Table 1 is now "Discarded". These scenarios have not been implemented because, finally, they were not required by any pilot or i4Q solution and task partners decided to focus their work on other issues.

Storage Tech	Scenario	Responsible	Status	Trino Integration status
Cassandra	SS	ITI	Done	Done (v0.2)
	HA	ITI	Discarded	N/A
	SS+Sec	ITI	Discarded	N/A
	HA+Sec	ITI	Discarded	N/A
MariaDB	SS	ENG	Done	Done (v0.1)
	HA	ENG	Discarded	N/A
	SS+Sec	ENG	Done	Done (v0.2)
	HA+Sec	ENG	Discarded	N/A
MinIO	SS	ITI	Done	In progress (v0.2)
	HA	ITI	Done	In progress (v0.2)
	SS+Sec	KBZ	Done	In progress (v0.2)
	HA+Sec	KBZ	Discarded	N/A
MongoDB	SS	ITI	Done	Done (v0.1)

[13] TimeScaleDB. https://www.timescale.com/

	HA	ITI	Done	Done (v0.1)
	SS+Sec	ITI	Done	Done (v0.1)
	HA+Sec	ITI	Done	Done (v0.1)
MySQL	SS	ITI	Done	Done (v0.1)
	HA	ENG	Discarded	N/A
	SS+Sec	ITI	Done	Done (v0.2)
	HA+Sec	ENG	Discarded	N/A
Neo4j	SS	ITI	Done	N/A
	HA	ITI	Discarded	N/A
	SS+Sec	ITI	Done	N/A
	HA+Sec	ITI	Discarded	N/A
PostgreSQL	SS	UNI	Done	Done (v0.2)
	HA	UNI	Discarded	N/A
	SS+Sec	UNI	Done	Done (v0.2)
	HA+Sec	UNI	Discarded	N/A
Redis	SS	ITI	Done	Discarded
	HA	UNI	Discarded	N/A
	SS+Sec	ITI	Done	Discarded
	HA+Sec	UNI	Discarded	N/A
TimeScaleDB	SS	UNI	Done	To be decided
	SS+Sec	UNI	Done	To be decided

Table 2. Summary of toolkits' implementation status by M24.

3.2.2 Top layer using Trino

Regarding the implementation of a layer on top of the different toolkits, the goal is two-fold. Firstly, it will improve the interoperability of the i4Q^{DR} with the rest of the solutions, by offering a common interface for any of the toolkits. Secondly, it will ease the support to more storage technologies by the i4Q^{DR} if necessary, in the future, which is a need that has been identified along the first months of the i4Q solutions development. In the following, we will refer to this layer as *"top-layer".*

The tool used to implement the top-layer is Trino [7], which is a highly parallel and distributed ANSI SQL-compliant open-source query engine that offers a relational-like view of different data storage tools. Moreover, Trino allows the execution of *federated queries,* which means that several databases of different types (relational, object storage, streaming or NoSQL, etc.) can be accessed within the same query.

Trino fulfils the goals of the top-layer as follows. On the one hand, Trino offers a REST API which will allow for the interaction of the i4Q^{DR} with other solutions through HTTP protocol. Furthermore, it is possible to use a Python client package like the one provided in: https://github.com/trinodb/trino-python-client that enables the implementation of custom client applications connecting to Trino's servers. This package can be used to develop a subcomponent facilitating the interoperability of the top-layer with other solutions. On the other hand, Trino facilitates the integration of new storage technologies via the so-called "connectors".

Basically, a connector is a piece of software that adapts Trino to a data source, as if it was a driver for a database[14]. Trino contains several built-in connectors and many third parties have contributed with connectors for other technologies. The list of currently available connectors is provided at: https://trino.io/docs/current/connector.html. This list includes connectors for all the storage technologies mentioned in Section 3.1 except MinIO and Neo4j. However, in the case of MinIO, it seems that the connector for Hive can be used[15]. However, we believe that the use of Trino brings enough benefits to consider its use in the implementation of i4Q^{DR} whenever possible, even though it does not support all the selected storage technologies. Figure 2 provides an overview of the i4Q^{DR} architecture, showing the use of Trino on top of all the implemented toolkits (please, see D3.15 [4] for further details on the i4Q^{DR} architecture).

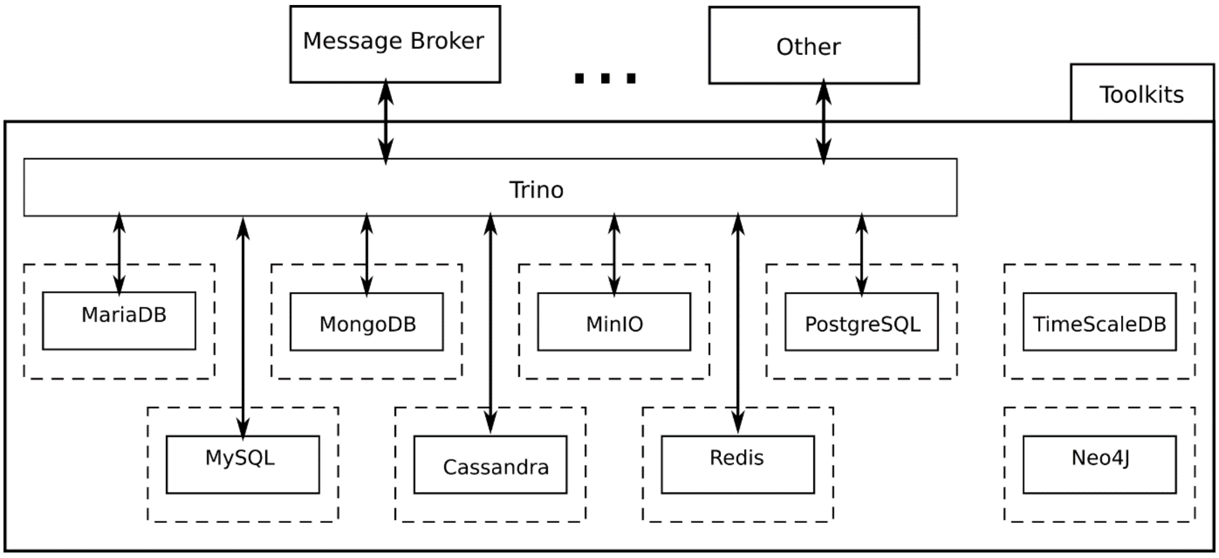


Figure 2. Architectural overview of i4Q^{DR} v0.2

Table 2 gathers information on which scenarios have been integrated with Trino, more specifically column *"Trino integration status"*. The values shown in that column are as follows:

- *Done*: denoting that the integration was implemented. In this case, it is also noted whether this was first included in v0.1, or in v0.2.

[14] https://trino.io/docs/current/overview/concepts.html, see "Connector" subsection.
[15] See post in Trino blog at: https://trino.io/blog/2020/10/20/intro-to-hive-connector.html.

- *In progress*: denoting that this case is being implemented at the time of writing this document. The implementation is expected to finish in the next weeks[16].
- *N/A (Not Applicable):* meaning that the integration was not implemented because the implementation of the scenario itself was discarded.
- *Discarded:* the integration has been discarded in this case because the scenario is not expected to be used by any pilot or any other i4Q solution.
- *To be decided:* it is necessary to check whether such an integration is feasible.

As shown in Table 2, v0.1 included the integration of some toolkits with Trino. More specifically, in that version, partner ENG started testing the “Single Server” scenario of both MariaDB and MySQL on Trino, whereas partner ITI did so for the “Single Server” scenario with MongoDB. In v0.2 such integrations have been refined and improved. For instance, instead of using a static Trino’s orchestration file, it is generated when bootstrapping the i4Q^{DR}, considering only the storage technologies selected by the user. Version v0.2 includes the integration of Trino into Cassandra and PostgreSQL toolkits, and into new scenarios of MySQL and MariaDB. The integration of Trino into MinIO is being implemented at the moment of writing this document and, thus, its status is reported as “in progress”. Finally, the integration of Trino into TimeScaleDB has status “To be decided” because the list of Trino connectors does not include this technology. However, the plan is to check whether the connector for PostgreSQL could be used. Since TimeScaleDB was included in the i4Q^{DR} at M24 it was not possible to do this before submitting this deliverable. In Table 2, the progress included in v0.2 is highlighted by using light purple as the background colour of the corresponding cell in the “Trino Integration status” column.

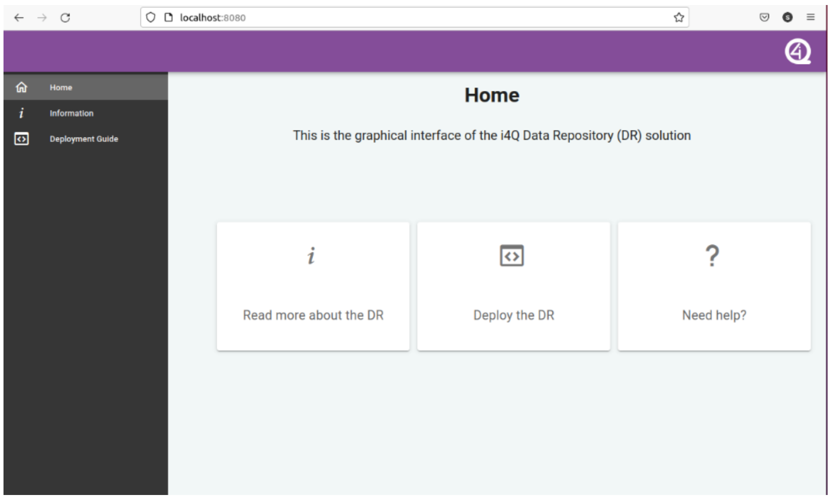
3.2.3 Graphical User Interface (GUI)

During the mid-term review of the project (September 2022), the reviewer suggested the implantation of a graphical user interface (GUI) for those i4Q solutions that did not have one, such as the case of the i4Q^{DR}. Thus, in the last months, a simple Web typified standard boxed interface has been implemented for the i4Q^{DR} using Vue.js[17], a JavaScript Framework for building web user interfaces.

More specifically, the GUI developed for the i4Q^{DR} consists of three screens:

- A “landing” screen “Home” (see Figure 3), providing general information on the GUI and links to the other screens.

[16] This work is planned to be finished before the end of M25, at the latest. In case the finishes after M24 (end of T3.5), this work will be performed in the context of T6.8, which ends at M36,
[17] Vue.js website: https://vuejs.org/

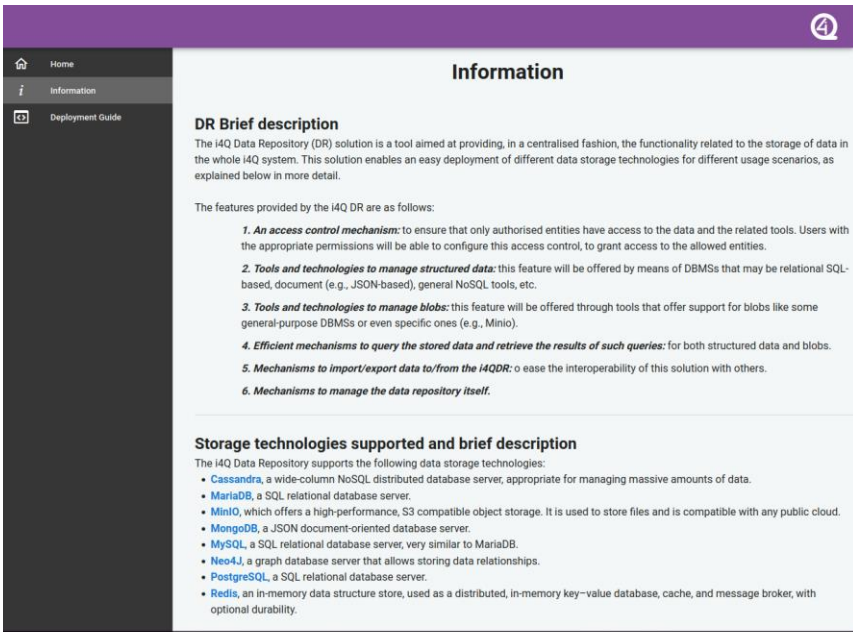


Home

This is the graphical interface of the i4Q Data Repository (DR) solution

Figure 3. i4Q^{DR} GUI, "Home" screen

- Screen "Information", which explains the purpose of the i4Q^{DR}, and describes the storage technologies and scenarios supported, as shown in Figure 4. This screen contains links to the web technical documentation of both the i4Q^{DR} and the i4Q^{DRG}.



Information

DR Brief description

The i4Q Data Repository (DR) solution is a tool aimed at providing, in a centralised fashion, the functionality related to the storage of data in the whole i4Q system. This solution enables an easy deployment of different data storage technologies for different usage scenarios, as explained below in more detail.

The features provided by the i4Q DR are as follows:

1. ***An access control mechanism:*** to ensure that only authorised entities have access to the data and the related tools. Users with the appropriate permissions will be able to configure this access control, to grant access to the allowed entities.
2. ***Tools and technologies to manage structured data:*** this feature will be offered by means of DBMSs that may be relational SQL-based, document (e.g., JSON-based), general NoSQL tools, etc.
3. ***Tools and technologies to manage blobs:*** this feature will be offered through tools that offer support for blobs like some general-purpose DBMSs or even specific ones (e.g., Minio).
4. ***Efficient mechanisms to query the stored data and retrieve the results of such queries:*** for both structured data and blobs.
5. ***Mechanisms to import/export data to/from the i4QDR:*** o ease the interoperability of this solution with others.
6. ***Mechanisms to manage the data repository itself.***

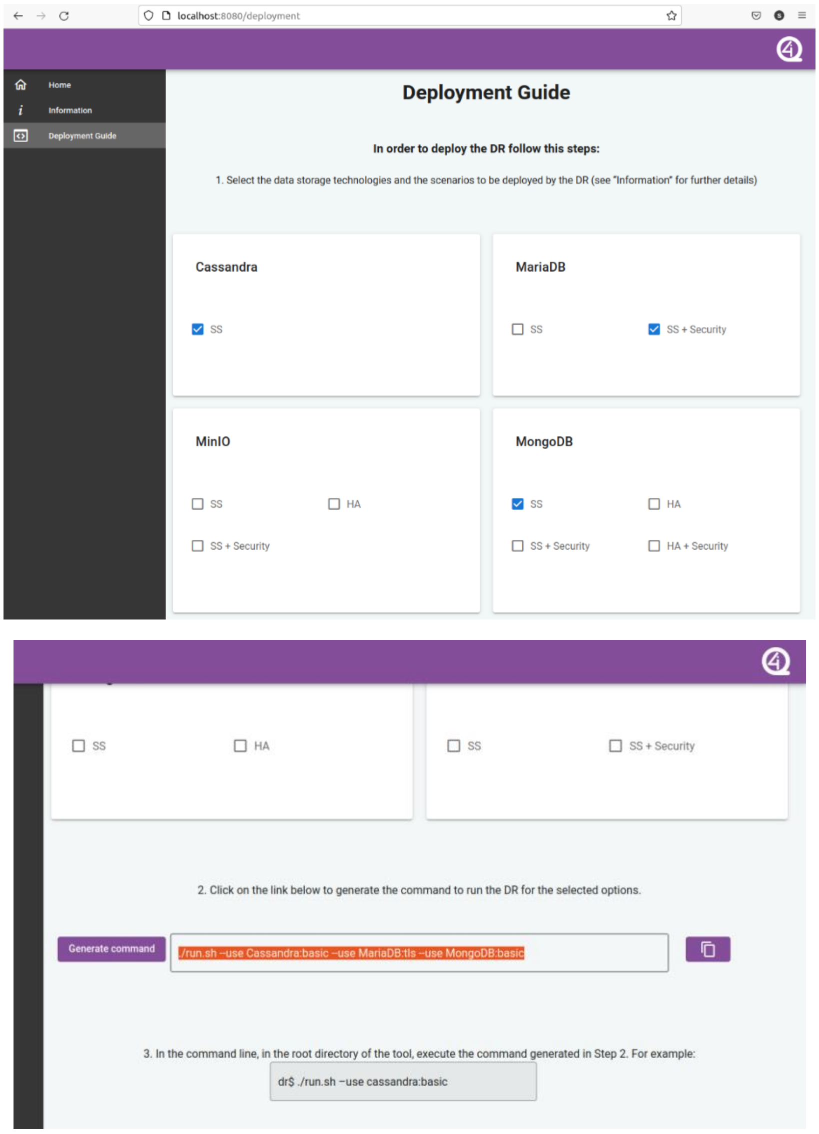
Storage technologies supported and brief description

The i4Q Data Repository supports the following data storage technologies:

- Cassandra, a wide-column NoSQL distributed database server, appropriate for managing massive amounts of data.
- MariaDB, a SQL relational database server.
- MinIO, which offers a high-performance, S3 compatible object storage. It is used to store files and is compatible with any public cloud.
- MongoDB, a JSON document-oriented database server.
- MySQL, a SQL relational database server, very similar to MariaDB.
- Neo4J, a graph database server that allows storing data relationships.
- PostgreSQL, a SQL relational database server.
- Redis, an in-memory data structure store, used as a distributed, in-memory key–value database, cache, and message broker, with optional durability.

Figure 4. i4Q^{DR} GUI, "Information" screen

- Screen "Deployment", which shows the available scenarios for each one of the supported storage technologies, as displayed in Figure 5. This screen allows the user to select the scenarios selected to be deployed by i4Q^{DR} and, according to such a selection, generated the exact command that must be executed in the shell in order to deploy the i4Q^{DR}.



Deployment Guide

In order to deploy the DR follow this steps:

1. Select the data storage technologies and the scenarios to be deployed by the DR (see "Information" for further details)

Cassandra: [✓ SS]

MariaDB: SS [✓ SS + Security]

MinIO: SS, HA, SS + Security

MongoDB: [✓ SS] HA, SS + Security, HA + Security

2. Click on the link below to generate the command to run the DR for the selected options.

```
./run.sh –use Cassandra:basic –use MariaDB:tls –use MongoDB:basic
```

3. In the command line, in the root directory of the tool, execute the command generated in Step 2. For example:

```
dr$ ./run.sh –use cassandra:basic
```

Figure 5. i4Q^{DR} GUI, "Deployment" screen

The source code of the GUI is included as part of the i4Q^{DR} solution and is available at the same repository. However, the GUI is deployed as a Docker container, independent of i4Q^{DR}.

3.2.4 Integration with the i4Q Message Broker

Finally, another feature included in v0.2 is a preliminary integration with the i4Q Message Broker (i4Q^{MB}), an additional i4Q component developed by partner CERTH responsible to provide a fast and secure way of inter-solution communication through data streaming. The i4Q^{MB} is based on Apache Kafka as distributed by the Confluent platform.

The communication in Kafka is conducted through the use of Kafka topics. A topic is an abstraction that acts as an intermediary between the communication of the solutions. Thus, a first approach to communicate solutions may consist of one or multiple solutions producing messages in a certain topic and other solutions consuming these messages via subscribing to that specific topic. Figure 6 summarises this approach of communication between solutions. The data format of the messages can be either JSON or Avro[18] which is a more efficient and compact way of exchanging messages, with a data model similar to JSON.

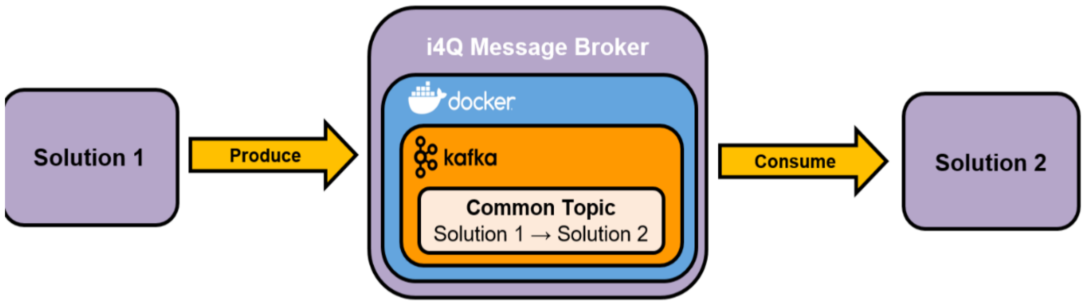


Figure 6. Communication among solutions via the i4Q^{MB}

For communication flows in which the i4Q^{DR} must compile information produced by other solutions, there are two possible approaches. The first one is developing a simple application that:

- Subscribes to the common topic agreed by both the solution producing the information and the i4Q^{DR}, and
- Stores the information into a schema and table in the i4Q^{DR} by means of the Trino instance deployed by the i4Q^{DR}. The specific schema and table involved could be defined, among other possible ways, by the solution providing the information, the i4Q^{DR} itself, or by agreement between the i4Q^{DR} and other solutions retrieving such an information later on. The implementation of such an application can make use of client packages for Trino, such as the one mentioned in Section 3.2.2.

This approach is summarised in Figure 7.

The second possibility is using the Kafka connector for Trino[19], which allows access to live topic data from Apache Kafka through Trino. With the appropriate configuration, the Kafka connector allows to transform the unstructured data produced in a Kafka topic into structured data that can be stored into a table of a schema defined in Trino's. Such a transformation is performed according to a mapping from the unstructured data of a Kafka message to the table's schema defined before deploying the Trino instance. In the following, an example is provided to illustrate how this integration works. The flow of information of that example is summarised in Figure 8.

[18] Apache Avro website: https://avro.apache.org/
[19] https://trino.io/docs/current/connector/kafka-tutorial.html

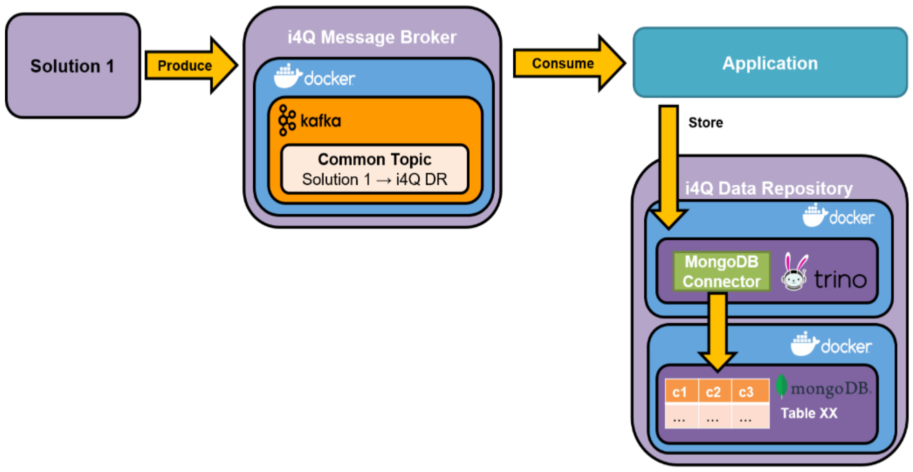


Figure 7. Storing information in the i4Q^{DR} via the i4Q^{MB} using an application.

As an example, let's assume a scenario involving the following components:

- An i4Q solution "Solution 1" producing some message
- An instance of the i4Q^{MB}, which internally deploys an instance of Kafka[20]. In this instance, at least, the Kafka topic "Users" has been defined.
- An instance of the i4Q^{DR}, which deploys a Docker container running an instance of Trino. The Trino instance includes the Kafka connector for Trino, which has been configured to have access to the Kafka instance deployed by the i4Q^{MB}. Furthermore, a mapping for Kafka messages with topic "Users" has been defined in the corresponding configuration file (see [8] for further details).

Then, when Solution 1 produces a message (in JSON format) of the form:

```
{
    "id": 1,
    "name": "Mary",
    "age": 30
}
```

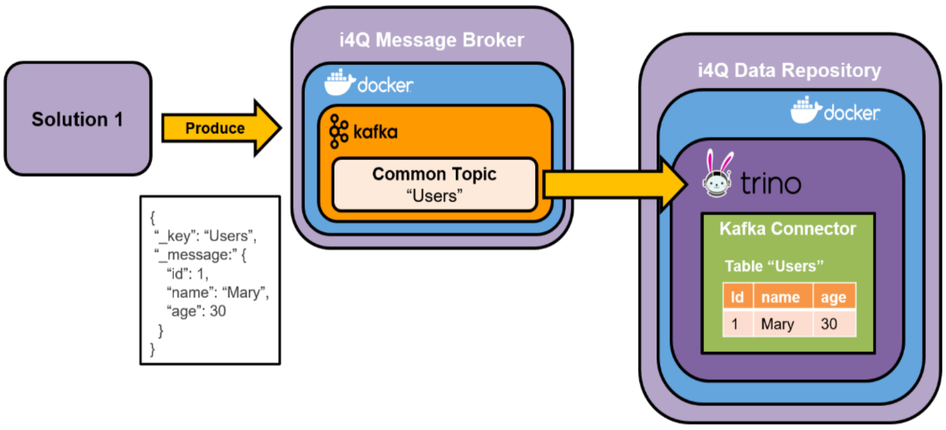
with the topic "Users" via the i4Q^{MB}, the corresponding Kafka message is something of the form:

```
{
 "_key": "Users",
 "_message:" {
    "id": 1,
    "name": "Mary",
    "age": 30
  }
}
```

[20] This can imply the deployment of more than one Docker container.

Such a message is processed by the i4Q^{MB} and received in the Kafka connector for Trino in the i4Q^{DR}. First, Trino creates a table called "Users" in case it does not exist. In this case, the mapping from the Kafka message has been defined in such a way that the message fields are mapped to columns of that table with that name (e.g., the field "id" corresponds to a column called "id" in the table "Users"). Therefore, when the Kafka connector receives the message, it stores its content as a new record into the table "Users". That is, in this case, the result will be a table "Users" of the form shown below:

id	name	age
1	Mary	30



```
{
"_key": "Users",
"_message:" {
  "id": 1,
  "name": "Mary",
  "age": 30
  }
}
```

Id	name	age
1	Mary	30

Figure 8. Storing information in the i4Q^{DR} via the i4Q^{MB} using the Kafka connector for Trino.

Preliminary implementations of both approaches of integration with the i4Q^{MB} have been included in i4Q^{DR} v0.2. The plan is to improve both implementations after M24, in the context of task *"T6.8 – Continuous integration and validation*".

3.3 Solution features analysed and mapping with user requirements

A set of features has already been developed for i4Q^{DR}, based on the set of user requirements referring to i4Q^{DR} [3] and in line with the functional viewpoints [5]. Similar requirements have been assigned into common categories of tasks based on an extensive technical study conducted on user requirements, available datasets, etc., introduced to ensure the generalization abilities of the i4Q^{DR} solution.

Below, we explain in more detail which solution's features address and cover each one of the requirements defined in D1.9 [3] for the i4Q^{DR}.

- *PC1r2.4 "Capability of storing data for future retrieval and analysis":* is covered by the feature of being able to save image blob and structured data into a database, so that it can be retrieved later for its analysis.
- *PC1r3.2 "Capability of extracting relevant (requested) features from ingested data/signals":* which is supported by the fact that the i4Q^{DR} can execute a given query to retrieve the desired data from the corresponding storage technology/tool.

- *PC2r3 "I4Q - DATA REPOSITORY: Define the right data repository":* is covered by the features of managing and configuring the data repository itself, and the inclusion of tools and technologies to manage both, structured, and unstructured (i.e., blobs) data.
- PC4r1.1.2 *"Gather information from human source and take it into account for analysis":* is covered by the solution's features allowing to import data from a database and being able to save both structured, and unstructured (i.e., blobs) data.
- *PC4r4.2 "Possibility of configure the data taking process":* is covered by the solution's features that enable the management of both structured, and unstructured (i.e., blobs) data.
- PC4r4.1 *"Extract valuable information from existing data and make it as an input for other exploitation or analysis"*: can be achieved by combining several features of the solution. More specifically, by: (i) mechanisms to import/export data from/to a database, (ii) mechanisms to query the stored data and retrieve the results of such queries, and (iii) tools and technologies to manage data, especially to save new data or update previously stored data. These features refer to the two main types of data considered by this solution: structured, and unstructured (i.e., blobs) data.
- PC4r5.1.1 *"Avoid losses and corruption of data due to communication failure":* is covered by the feature that allows creating several replicas to store data and saving both image blob, and structured data.
- PC4r5.4.1 *"Extract valuable info from existing data"*: is covered by the solution's mechanisms to import data from an existing database and to query the stored data and retrieve the results of such queries.
- PC6r8.2 *"Dataflow from machines to database shall be established":* is mainly supported by the features that allow one to (i) manage the data repository so that it can receive the information provided by other components (e.g., other tools or solutions) as input, and (ii) managing both structured and unstructured data, especially to save into a database unstructured data.
- PC6r8.3 *"Image Data Compression":* is covered by the feature of managing unstructured data, especially to save and update it.

Furthermore, there are several requirements that are supported by the capability of the solution to save structured and unstructured (i.e., blobs) data, namely:

- PC3r3.1 "Prediction Result Storage"
- PC3r3.2 "Importance and post analytical Storage"
- PC4r1.1.1 "Disponibility[21] of production data taking from the CNC program and production orders"
- PC4r1.1.3 "Current measuring machines data gathering" which refers to the necessity of collecting and saving all data provided by measurement equipment automatically in digital format.
- PC4r1.3 "Store the data to be used in future purposes"
- PC4r5.3 "Collect data from status sensors (time-stamped)"

[21] In the sense of "availability"

- PC5r3.3 "Data storage"
- PC6r1.4.1 *"Store the data",* which refers to the storage of injection machine parameters.
- PC6r1.4.2 *"Store the data (2)",* which stands for the storage of energy analyser parameters.
- PC6r1.4.3 *"Store the data (3)",* which refers to the storage of water pump parameters.
- PC6r8.1 "Historical image data should be managed in the data repository".

3.4 History

This section provides the version history of the i4Q^DR implementation up to M24, which is gathered in the table below. More specifically, it shows, for each version of the implementation (denoted by the first column), when it was released (see "Release date" column), and what functionality was added (described in column "New features").

Version	Release date	New features
V0.1	21/01/2022	Added toolkit for MongoDB (SS and HA scenarios)
V0.2	24/01/2022	Added toolkit for MySQL (SS, SS+Sec scenarios), and preliminary version of tookit for Redis (SS scenario)
V0.3	25/01/2022	Added SS+Sec scenario to Redis tookit. Included technical improvements.
V0.4	26/01/2022	Added toolkit for MinIO (SS scenario)
V0.5	27/01/2022	Applied minor changes and refactoring and added documentation about the tookits' commons.
V0.6	28/01/2022	Added tookit for Cassandra (SS), and applied minor technical improvements
V0.7	31/01/2022	Added HA scenario to MinIO toolkit
V0.8	01/02/2022	Added toolkit for Neo4j (SS scenario)
V0.9	02/02/2022	Added scenario SS+Sec to Neo4j toolkit
V0.10	15/02/2022	Updated MongoDB Docker image to v5.0.6, added SSL initialization, improved MongoDB HA scenario, and applied other technical improvements.
V0.11	30/04/2022	Added PostgreSQL toolkit (SS scenario)
V0.12	06/05/2022	Added preliminary integration of MongoDB toolkit for SS scenario with Trino
V0.13	04/05/2022	Added toolkit for MariaDB (SS scenario)
V0.14	11/05/2022	Added preliminary integration of MariaDB and MySQL toolkits for SS scenario with Trino
V1.0	30/05/2022	M18 solution release
V1.1	29/09/2022	Fixed bugs, and improved MongoDB SS+Sec scenario.
V1.2	11/10/2022	Integrated Trino into Cassandra SS+Sec scenario
V1.3	13/10/2022	Preliminary implementation of MinIO SS+Sec scenario

Version	Release date	New features
V1.4	13/10/2022	Preliminary implementation of MariaDB SS+Sec scenario
V1.5	24/10/2022	Preliminary implementation of PosgreSQL SS+Sec scenario
V1.6	15/11/2022	Final implementation of Minio SS+Sec scenario
V1.7	2/12/2022	Final implementation of MariaDB SS+Sec scenario
V1.8	12/12/2022	Integration of Trino into MariaDB SS+Sec scenario
V1.9	19/12/2022	Integration of Graphical User Interface
V1.9	20/12/2022	Preliminary integration with i4Q Message Broker
V2.0	22/12/2022	M24 solution release

Table 3. i4Q^{DR} Version history

3.5 Next steps

After M24 the development plan is to integrate i4Q^{DR} in all pilot use cases or to improve its integration in the pilots in which it has already been involved. This work will be performed in the context of the tasks corresponding to each pilot (tasks T6.1 to T6.7) and in *"T6.8 - Continuous integration and validation".*

During this integration work, i4Q^{DR} will be integrated with other i4Q solutions, in order to implement the pipelines defined for each pilot in a real production environment. Consequently, it might be necessary to modify the i4Q^{DR}, to fix problems found or to adapt the solution to the specific pilot scenario. Furthermore, it is expected to improve the graphical user interface, mainly to facilitate the configuration and deployment of the i4Q^{DR} and make it easier to be used.

Thus, new versions of the i4Q^{DR} may be produced in the M24-M36 period. In such a case, those versions will be mainly reported in the deliverables of T6.8 (D6.17 and D6.18).

4. Conclusions

Deliverable *"D3.16 - i4Q Data Repository v2"* is a technical specification document presenting a technical overview of the i4Q Data Repository solution (i4Q^{DR}). This deliverable describes in detail the role, the functionalities, and the conceptual architecture of the i4Q^{DR}.

Moreover, the main features of the solution have been explained, including a description of its architecture diagram with respect to i4Q Reference Architecture.

Furthermore, this document explained in detail the implementation work of the i4Q^{DR} solution. It provided an overview of the approach followed and the functionalities implemented for each one of the two versions of the i4Q^{DR} released so far, namely version v0.1 in M18, and version v0.2 in M24. In this regard, the analysis and engineering of the pilots' requirements for this solution has been included, too, to clarify the technical specifications.

Finally, a summary of the i4Q^{DR} version history is provided, too.

References

[1] i4Q, «D3.8 - Data Repository,» June 2022.

[2] i4Q, «D4.9 - i4Q Data Integration and Transformation Services v2,» December 2022.

[3] i4Q, «D2.7 – i4Q Reference Architecture and Viewpoints Analysis v2,» September 2021.

[4] i4Q, «D3.15 – i4Q Guidelines for Building Data Repositories for Industry 4.0 v2,» December 2022.

[5] i4Q, «D1.9 – Requirements Analysis and Functional Specification v2,» September 2021.

[6] i4Q, «D6.9 - Continuous integration and validation,» December 2022.

[7] «Trino official website,» [En línea]. Available: https://trino.io/.

[8] Trino, «Kafka connector for Trino tutorial,» [En línea]. Available: https://trino.io/docs/current/connector/kafka-tutorial.html.

[9] i4Q, «D2.6 – Technical Specifications,» September 2021.

Appendix I

The i4Q **Data Repository** (i4Q^{DR}) technical web documentation can be accessed online at: http://i4q.upv.es/8_i4Q_DR/index.html.