This project has received funding from the European's Union Horizon 2020 research innovation programme under Grant Agreement No. 957258

Architecture for Scalable, Self-human-centric, Intelligent, Secure, and Tactile next generation IoT

assist-iot

D5.4 – Software Structure and Final Design

Deliverable No.	D5.4	**Due Date**	31-OCT-2022
Type	Report	**Dissemination Level**	Public
Version	1.0	**WP**	WP5
Description	*Final specification of the vertical enablers identified and developed in ASSIST-IoT.*		

Copyright

The ASSIST-IoT consortium consists of the following 15 partners:

UNIVERSITAT POLITÈCNICA DE VALÈNCIA	Spain
PRODEVELOP S.L.	Spain
SYSTEMS RESEARCH INSTITUTE POLISH ACADEMY OF SCIENCES IBS PAN	Poland
ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS	Greece
TERMINAL LINK SAS	France
INFOLYSIS P.C.	Greece
CENTRALNY INSTYUT OCHRONY PRACY	Poland
MOSTOSTAL WARSZAWA S.A.	Poland
NEWAYS TECHNOLOGIES BV	Netherlands
INSTITUTE OF COMMUNICATION AND COMPUTER SYSTEMS	Greece
KONECRANES FINLAND OY	Finland
FORD-WERKE GMBH	Germany
GRUPO S 21SEC GESTION SA	Spain
TWOTRONIC GMBH	Germany
ORANGE POLSKA SPOLKA AKCYJNA	Poland

Disclaimer

Authors

Name	Partner	e-mail
Alejandro Fornés	P01 UPV	alforlea@upv.es
Raúl Reinosa	P01 UPV	rreisim@upv.es
Rafael Vaño	P01 UPV	ravagar2@upv.es
Eduardo Garro	P02 PRO	egarro@prodevelop.es
Adrian Ramos	P02 PRO	aramos@prodevelop.es
Katarzyna Wasielewska-Michniewska	P03 IBSPAN	katarzyna.wasielewska@ibspan.waw.pl
Karolina Bogacka	P03 IBSPAN	k.bogacka@ibspan.waw.pl
Piotr Lewandowski	P03 IBSPAN	piotr.lewandowski@ibspan.waw.pl
Przemysław Hołda	P03 IBSPAN	pholda@ibspan.waw.pl
Anastasia Blitsi	P04 CERTH	akblitsi@iti.gr
Evripidis Tzonas	P04 CERTH	tzionasev@iti.gr
Ron Schram	P09 NEWAYS	Ron.Schram@newayselectronics.com
Alex van den Heuvel	P09 NEWAYS	alex.van.den.heuvel@newayselectronics.com
Oscar López Pérez	P13 S21 SEC	olopez@s21sec.com
Saioa Ros Jiménez	P13 S21 SEC	sros@s21sec.com

History

Date	Version	Change
08-Sep-2022	0.1	ToC and task assignments
30-Sep-2022	0.2	First round of contributions completed
13-Oct-2022	0.3	Second round of contributions integrated
18-Oct-2022	0.4	Third round of contributions. To internal review
31-Oct-2022	1.0	Final version submitted to EC

Key Data

Keywords	Enablers, verticals, self-*, security, manageability, scalability, federated learning, DLT
Lead Editor	P01 UPV – Alejandro Fornés
Internal Reviewer(s)	P14 TWOT – Lambis Tassakos P15 OPL – Zbigniew Kopertowski

Executive Summary

This deliverable is written in the framework of WP5 – Transversal enablers design and development of ASSIST-IoT project under Grant Agreement No. 957258. The document collects the work and outcomes of the five tasks of the work package, which focus on the design and implementation of the enablers required to implement the different verticals of the ASSIST-IoT architecture. These enablers (which may entail not just software but also hardware-related design and development) belong to at least one of the following verticals: (i) Self-*, (ii) Scalability, (iii) Interoperability, (iv) Manageability, and (v) Security, Privacy and Trust.

This document is the last iteration of a series of two deliverables devoted to the formalisation and design of the vertical enablers identified within the scope of the project, hence presenting their final software structure. It should be mentioned that the other series of deliverables belonging to this Work Package, devoted mainly to the release of the actual artifacts, also included an update of specifications. Thus, the present document is built upon both WP5 deliverable series (particularly, after D5.1 & D5.3), and since the latter was released just six months ago, this deliverable **focuses on incremental content** with respect to them, as, in general, enablers have not experienced major design changes during this time. The final design of the following enablers is included:

- From Self-*: Self-healing device enabler, Resource provisioning enabler, Monitoring and notifying enabler, Location tracking enabler, Location processing enabler and Automated configuration enabler.
- From Federated Machine Learning: FL Orchestrator, FL Training Collector, FL Repository and FL Local Operations.
- From Cybersecurity: Cybersecurity monitoring enabler, Cybersecurity monitoring agent enabler, Identity manager enabler and Authorisation enabler.
- From DLT: Logging and auditing enabler, Data integrity verification enabler, Distributed broker enabler and DLT-based FL enabler.
- From Manageability: Clusters and topology manager, Enablers manager and Composite services manager (the names of these enablers were modified in the last architecture document, D3.7).

For each of them, their respective (i) tables of general information (presented in D5.1), (ii) high-level structure (originally introduced in D5.1 and updated in D5.2 & D5.3), (iii) implementation technologies (candidate ones presented in D5.1 and formalised in D5.2 & D5.3), (iv) communication interfaces (introduced in D5.2 and updated in D5.3), and (v) enabler stories (presented in D5.2 and extended in D5.3) are revisited and updated, when needed.

.

Table of contents

List of figures

List of tables

List of acronyms

Acronym	Explanation
API	Application Programming Interface
BIM	Building Information Modelling
CA	Certificate Authority
CHE	Container Handling Equipment
CNI	Container Network Interface
CPU	Central Processing Unit
DLT	Distributed Ledger Technology
ES	Enabler Stories
FL	Federated Learning
gRPC	gRPC Remote Procedure Calls
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IdM	Identity Manager
IoT	Internet of Things
JSON	JavaScript Object Notation
K8s	Kubernetes
ML	Machine Learning
MQTT	MQ Telemetry Transport
NoSQL	Not Only SQL
OS	Operating System
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
REST	Representational State Transfer
SD-WAN	Software-Defined Wide Area Network
SQL	Structured Query Language
RAM	Random Access Memory
RTG	Rubber-Tyred Gantry (crane)
UV	Ultraviolet
VPN	Virtual Private Network

1. About this document

The main objective of this document is to present the final design and specifications of the vertical enablers developed within the scope of ASSIST-IoT project. These enablers, along with horizontal enablers proposed in WP4, are the technological backbone of the project, since they will enable the deployment of an ASSIST-IoT architecture.

Although this is the final iteration of a series of two deliverables devoted to the design of enablers, the complementing WP5 deliverable series (which focuses on actual development and outcomes) has also being providing specification updates, and hence this report has been arranged as a delta document built upon the content of its previous version (D5.1) and the complementary series (D5.3) for the sake of avoiding repeating many contents. Besides, the final deliverable of WP5 (D5.5), due in M28, will include all the information produced in an all-encompassing document, along with the developed artifacts.

1.1. Deliverable context

Keywords	**Lead Editor**
Objectives	**O3:** Definition and implementation of decentralised security and privacy exploiting DLT: Specification of DLT-based enablers in Security, Privacy and Trust vertical. **O4:** Definition and implementation of smart distributed AI Enablers: Specification of Federated Machine Learning related enablers.
Work plan	D5.4 takes input from: • T3.2 & T3.3 (use cases and requirements): To be revisited and linked to the proposed enablers. • T3.5 (architecture): Depicts the design principles and high-level functionalities to consider and follow. • D5.1 (initial transversal enablers specification): Initial design of vertical enablers, and previous iteration of this deliverable. • D5.2 & D5.3 (transversal enablers development preliminary and intermediate versions): Included updated specifications from D5.1 and new design-related content. D5.4 influences: • WP7 (pilots and validation): To later on materialise in pilot deployments. • WP8 (evaluation and assessment): To evaluate and assess results from testing within pilots. D5.4 must be in line with: • WP4 (core enablers): To define functional boundaries and interactions. • WP6 (testing, integration and support): To develop, test and deploy according to DevSecOps methodology.
Milestones	This deliverable is a key part of *MS6 – Software structure finished*, along with the previous content presented in D5.3. It also contributes to *MS7 – Integrated solution*, once enablers are deployed and working in pilot premises.
Deliverables	Being an incremental or delta document built upon previous deliverables of this WP, it receives inputs from its previous iteration (D5.1 – Initial Transversal Enablers Specification) and the deliverables related to the outcomes and artifacts, which included an update of the initial specifications (D5.2 & D5.3 – Transversal Enablers Development Preliminary/ Intermediate Version). It is also influenced by the main architecture document (D3.7) and linked to the project requirements and use cases (D3.3).

1.2. The rationale behind the structure

This deliverable consists of three sections beyond the current one. The first one (Section 2) is an introduction to the document, presenting the Verticals and allocating the different enablers within the context of one (or some) of them. The second section details the final design and specifications of each one of the vertical enablers, grouped by Verticals, including the data related to their context, software structure, technologies, endpoints and user stories. Finally, conclusions are drawn in the last section.

1.3. Outcomes of the deliverable

The main outcome of the present deliverable is the final design and software structure of the enablers envisioned within the scope of WP5. As updates (and enlargement) over the initial design (i.e., D5.1) were also documented in the complementary series of deliverables of the WP (the one related to actual developments and outcomes, i.e., D5.2 & D5.3), most of the content remains valid and designs have not suffered many major changes. Some relevant outcomes of the document and design modifications include:

- All vertical enablers have updated their respective tables with general information, pointing to the updated project's requirement and use cases.
- The features offered by the Location processing enabler have been extended, to achieve greater flexibility and be applicable to more use cases.
- Another enabler that has been subject of significant design modifications is the Automated configuration enabler, with a new internal structure and endpoints.
- New enabler stories have been added to some enablers, including: Self-healing device enabler, Location processing enabler, FL Repository, Enablers manager, Composite services manager, and Cluster and topology manager.

1.4. Lessons learnt

During the past months, the partners of the Consortium have focused their effort in developing and possibly adjusting the design of the enablers that will facilitate the realisation of the ASSIST-IoT architecture. From this work the following insights have been extracted:

- Manageability enablers (Enablers manager, Composite services manager, Clusters and topology manager) can be packaged and deployed together as their functionality is not foreseen to be used separately but rather will provide a package of functionalities to manage the ASSIST-IoT deployment.
- The decision was made not to containerise Cybersecurity monitoring agent enabler as it would negatively impact functionality offered by this enabler, e.g., as a result of a limited accessibility to the host system.
- DLT-based FL enabler functionality has been focused on storing model updates produced during the FL process. The decision was based on the considerations about the efficiency of the whole solution and possibilities of seamless integration with other FL enablers.

1.5. Deviation and corrective actions

The Consortium formalised in D5.1 (and now D5.4), and materialised in D5.2 and D5.3 the envisioned enablers. However, some deviations have slightly altered the initial plan:

- Manageability enablers have been renamed to better express their functionality.
- Monitoring and notifying enabler functionalities have been reviewed and refocused on the monitoring aspects to differentiate it more from Edge data broker enabler from WP4.

1.6. Version-specific notes

The Consortium formalised the initial design of the vertical enablers in M9. Being so early in the project, many relevant information required for their development was missing, and therefore these needs were covered in the

subsequent deliverables of WP5, namely D5.2 & D5.3, despite they should have focused just in releasing artifacts (in the form of MVPs, preliminary versions or actual releases). As D5.3 was in an already advanced date, most of the information depicted there is valid. Therefore:

- This version updates the general content of the enablers' template table from D5.1, and presents the updated high-level schema, implementation technologies, endpoints and user stories from D5.3 only when needed (incremental data).
- This entails that D5.3 remains as the main document for enablers design and specifications, presenting the current one just an update of the data presented six months ago.
- The final deliverable of D5.5 will include all the information in an all-encompassing document, along with the final outcomes. Only some minor design modifications might be needed to be documented in the last one, in which case they will be conveniently outlined.

1.7. Follow-up recommendations and comments from previous reviews

According to the formal feedback received out of RP1, documentation, deliverable D5.1 was accepted but the following comments were included, that have been tackled in the following way:

Q – (From D5.1) Is there a chance to test the decentralised learning approach within the project?

A – Yes. The outcomes from the Federated Learning task will be evaluated within the activities of the pilots, with special emphasis in pilot 3B.

Q – (From D5.1) Most of the enablers designed are generic and loosely-coupled to the use cases: how they are going to be tested? Isn't this a risk for the project?

A – The project detected this issue in early stages and it was decided that all enablers should be tested in at least one use case of the pilots, whenever possible. WP5 enablers are, in general, deployable independently of the scope of the use case or pilot, as they provide vertical, cross-cutting features. Therefore, no major risk is expected for testing them within the scope of pilots' activities.

2. Introduction

Considering the ASSIST-IoT reference architecture, Verticals represent properties or features that are present on different horizontal Planes (i.e., Device and edge, Smart network and control, Data management and Application and services), either independently or requiring cooperation with them. In this architecture, vertical capabilities can be grouped in five types, namely: (i) Self-*, (ii) Interoperability, (iii) Security, Privacy and Trust, (iv) Scalability, and (v) Manageability. Some of these capabilities become features of the system by design choices (e.g., using an underlying container orchestration framework such as Kubernetes, provides features that can be allocated under scalability and self-*), while others require of dedicated artifacts (i.e., enablers) to be incorporated.

A total of 21 enablers have been formalised in the project as deemed necessaries (or at least, encouraged) to be part of Next Generation IoT system realisations. Before implementing them, a set of initial specifications with expected features and candidate technologies were outlined in D5.1[1] and later on extended in D5.2[2] and D5.3[3]. As an agile approach has been applied, many specification changes have been already addressed by the two previous deliverables (despite the fact they were not initially expected to include such information). Hence, in the present report, a moderate number of changes are needed.

It should be mentioned that enablers, despite been originally envisioned belonging to a specific Vertical, might provide features that can be allocated under the umbrella of other ones, as it can be seen in Figure 1. One clear example are the enablers related to Federated Learning (FL). These enablers work together to provide a framework that can be deployed to train models in different computing nodes locally and combine their results in a central node, hence contributing to scalability as processing effort is shared. Also, data do not travel through the network to the central node, just the trained models and the inferred weights, preserving data privacy (hence enablers can be assigned to Security, Privacy and Trust vertical as well).

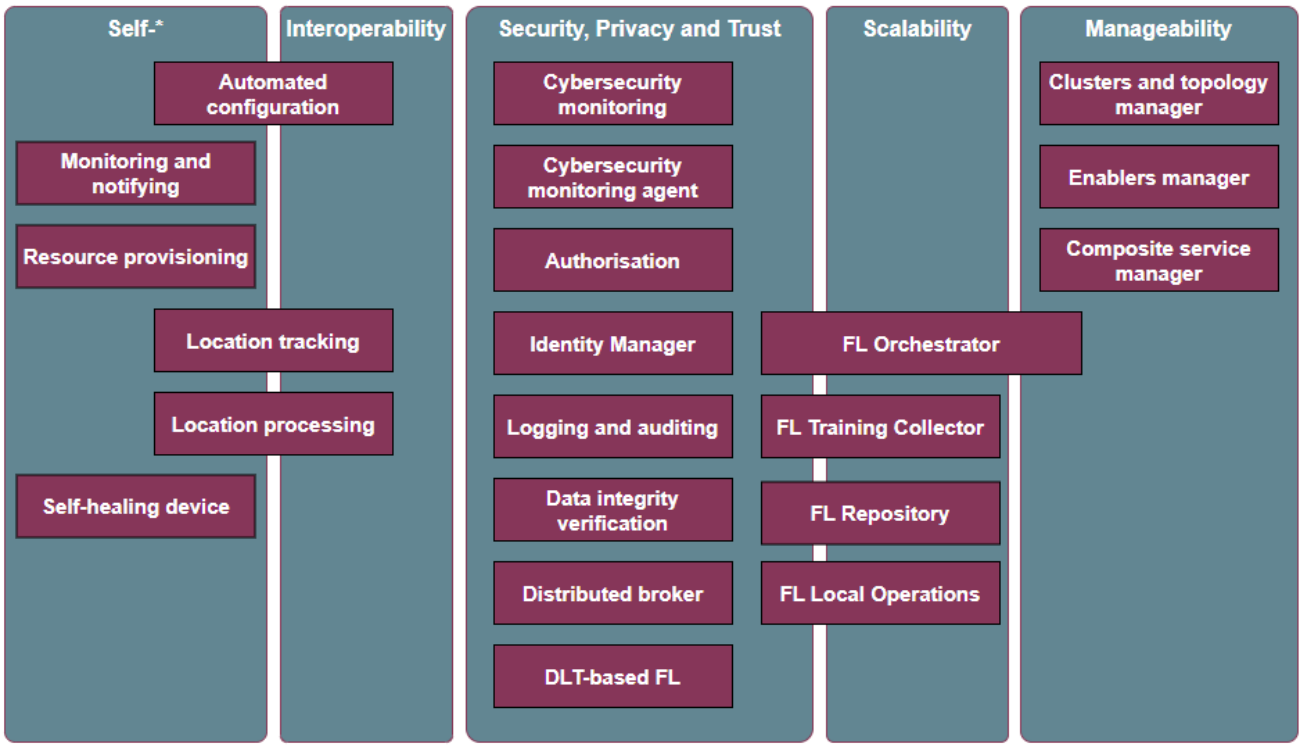

Figure 1. Vertical enablers distribution among verticals

[1] https://assist-iot.eu/wp-content/uploads/2021/12/ASSIST-IoT_D5.1_Software_Structure_and_Preliminary_Design_v1.pdf
[2] https://assist-iot.eu/wp-content/uploads/2021/12/ASSIST-IoT_D5.2-Transversal-Enablers-Development-Preliminary-Version-v1.0.pdf
[3] https://assist-iot.eu/wp-content/uploads/2022/05/D5.3_Transversal-Enablers-Development-Intermediate-Version.pdf

3. Final vertical enablers specification

The specification of the enablers of the project are presented following a common structure. First, a table with **general information** about a given enabler (see Table 1), including its description, is attached. Then, a basic diagram presenting the **high-level structure** of its internal components is depicted, followed by a table listing the **implementation technologies**, frameworks and/or programming languages leveraged to realise each of its components. Afterwards, the **endpoints** (generally APIs) that will be exposed to consume the enabler (either by end users, administrators or other enablers) are documented. Finally, a collection of **Enabler Stories** (ES), will be presented, including cases were an enabler performs an action as a response to a given event or condition (initiated by a user, or not).

Since most of the specifications indicated remain updated, in those enablers in which any change needs to be reported, the content of D5.3 will be pointed to (with the exception of the table with general information). Regarding the tables with implementation technologies and endpoints, the additions, modifications and removals are highlighted.

Table 1. General information of an enabler

Enabler	***Name of the enabler***
Id	*Short unique identifier/acronym*
Owner and support	*Lead and supporting beneficiaries*
Description and main functionalities	*Functional description of the enabler (description paragraph and bullet points for describing its features)*
Vertical, related capabilities and features	*Vertical to which this enabler belongs. Vertical groups together logically connected features and functionalities of a system, regardless of the plane on which they may be implemented. ASSIST-IoT defines 5 verticals:* • *Manageability* • *Scalability* • *Security, privacy and trust* • *Interoperability* • *Self-* (autonomy)* *Every vertical involves capabilities, a concretisation of a capability is called a feature.*
Plane/s involved	*Horizontal plane or planes on which the enabler's features are delivered*
Requirements mapping	*List of the IDs of the requirements addressed or considered*
Use case mapping	*List of the IDs of the use cases related to this enabler*
Internal components	*List of the internal components of this enabler*

3.1. Self-* enablers

3.1.1. Self-healing device enabler

Table 2. General information of the Self-healing device enabler

Enabler	**Self-healing device enabler**
Id	SELF11
Owner and support	PRO
Description and main functionalities	This enabler aims at providing to IoT devices with the capabilities of actively at-tempting to recover themselves from abnormal states, mainly divided in three categories: security (jamming, DoS), dependability (data corruption, network protocol violation), and long-term (HW's end-of-life, HW unsupported capabilities), based on a pre-established routines schedule.
Vertical, related capabilities and features	Self-* (autonomy)
Plane/s involved	Device and edge plane – as it provides features to heal the devices belonging to this plane

Enabler	Self-healing device enabler
	Smart network and control plane – as it brings features to evaluate some networking aspects
Requirements mapping	• R-C-5: Local Processing Capabilities • R-C-7: Edge-oriented deployment • R-C-18: Support for autonomous processing • R-C-19: Support for self-aware systems • R-C-20: Support for system self-healing • R-C-21: Reduction of computing demands for AI training • R-P2-3: Smart wristband for construction workers • R-P2-16: Device reliability and durability • R-P3A-9: Edge Intelligence
Use case mapping	• UC-P1-2: CHE location tracking • UC-P2-1: Worker's health and safety assurance • UC-P3A-2: Vehicle non-conformance causes identification
Internal components	Self-detector, Self-monitor, Self-remediator

High-level structure

No changes to D5.3.

Implementation technologies

After several attempts (in which the functionalities of the enabler could not reach the host OS environment if it was encapsulated), it was decided that the self-healing device enabler became an encapsulation exception of the ASSIST-IoT enablers list. In addition, instead of basic thresholds definition, more advanced monitoring options (i.e., smarter ML-based solutions) are under development.

Apart from this, the implementation technologies (Node-RED, JavaScript and Unix commands) remain as in D5.3.

Communication interfaces

The updated list of endpoints is presented here. From the three endpoints introduced in D5.3, five new endpoints (highlighted in green) have been added:

Table 3. Communication interfaces (API) of the Self-healing device enabler

Method	Endpoint	Description
POST	**/self-monitor-type**	To select between "basic" or "ML" the type of self-monitor tool to be used by the enabler
POST	**/cpuusage?threshold=XX**	If "basic" self-monitor-type is marked, this endpoint changes the maximum threshold of CPU usage to the XX value defined by the user
POST	**/ramusage?threshold=XX**	If "basic" self-monitor-type is marked, this endpoint changes the maximum threshold of RAM usage to the XX value defined by the user
POST	**/network?IP=XX**	If "basic" self-monitor-type is marked, this endpoint changes the IP address over which the service should ping to check network availability
POST	**/battery?threshold=XX**	If "basic" self-monitor-type is marked, this endpoint changes the battery life alerting action to the XX value defined by the user
POST	**/ml-cpuusage**	If "ML" self-monitor-type is marked, this endpoint defines the maximum threshold of CPU usage according to the trained results of the ML model
POST	**/ml-ramusage**	If "ML" self-monitor-type is marked, this endpoint changes the maximum threshold of RAM usage according to the trained results of the ML model
POST	**/ml-battery**	If "ML" self-monitor-type is marked, this endpoint changes the battery life alerting action according to the trained results of the ML model

Enabler stories

There two **enabler stories** described in D5.3 remain valid (monitoring of the CPU and RAM usage – ES1, and evaluation of the network interface operation – ES2). A **third** one, devoted to **battery usage mechanism**, is described below:

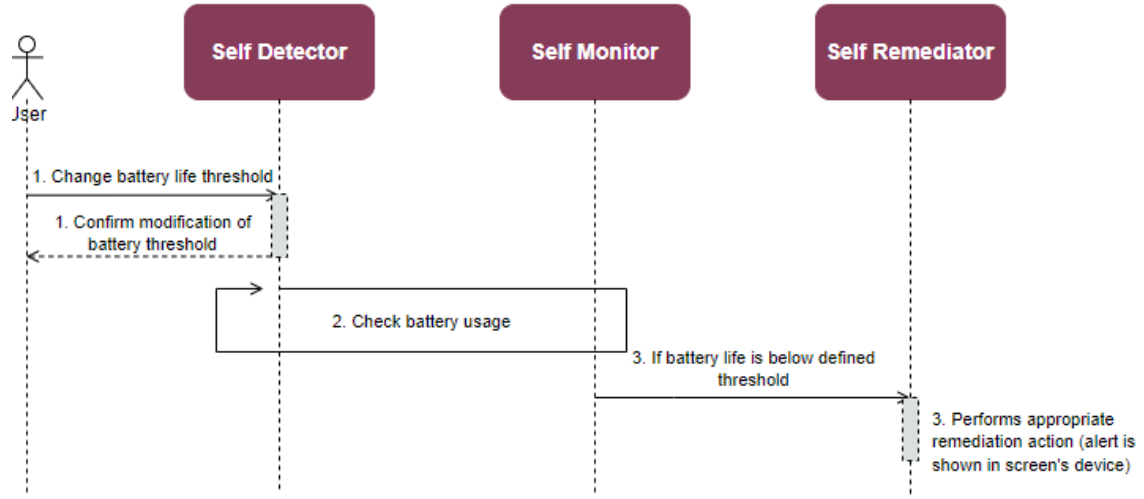

Figure 2. Self-healing Device enabler ES3 (battery usage monitoring and threshold update)

STEP 1: The user starts a device, installs the self-device enabler, and configures the battery usage threshold by interacting with the self-detector via API commands.

STEP 2: From this moment, the self-detector and self-monitor are in charge of discovering and monitoring the battery life in a happy path scenario.

STEP 3: If the monitored battery life metric goes down below certain threshold value defined by the user in STEP 1, the self-monitor informs the self-remediator to carry out the proper remediation action (prompts out an alert to the device's screen).

3.1.2. Resource provisioning enabler

Table 4. General information of the Resource provisioning enabler

Enabler	Resource provisioning enabler
Id	T51E2
Owner and support	UPV
Description and main functionalities	Working on edge deployments, where resources are not as large as in the cloud, it is unfeasible to set a static resource projection to each node. This is due to the difference in the use of these resources depending on the workload at the time the task is performed, being dependent on several factors. This enabler aims to adapt the auto-scaling of nodes and clusters more dynamically, achieving optimal use in relation to resource utilisation and general operation.
Vertical, related capabilities and features	Self-* (autonomy)
Plane/s involved	All Planes – it monitors and adapts the resources of enablers from all planes
Requirements mapping	• R-C-7: Edge-oriented deployment • R-C-9: Workload placement • R-C-16: Resource monitoring • R-C-18: Support for autonomous processing • R-C-19: Support for self-aware systems
Use case mapping	• UC-RPE-1: Information about enablers and active components • UC-RPE-2: Data training (deep learning)

Enabler	Resource provisioning enabler
	• UC-RPE-3: Obtain data training range • UC-RPE-4: Update data training range • UC-RPE-5: Data inference • UC-RPE-6: Select enablers to manage
Internal components	API, Pod Resources Controller (PRC), Training module (TM), Inference module (IM), History data – MySQL Database, Predicted data – MySQL Database

High-level structure

No changes to D5.3.

Implementation technologies

No changes to D5.3.

Communication interfaces

No changes to D5.3.

Enabler stories

No changes to D5.3.

3.1.3. Location tracking enabler

Table 5. General information of the Location tracking enabler

Enabler	Location tracking enabler
Id	T51E3-A
Owner and support	NEWAYS
Description and main functionalities	The main task of the location tracking enabler is to receive the position of tags. Each tag transmits its position with a fixed repetition rate of around 1 second. This position represents the triangulation of the tag relative to 3 reference anchors with a fourth anchor for failure reference. The localization tracking enabler translates these positions into a format which can be handled by a data broker.
Vertical, related capabilities and features	Self-* (autonomy)
Plane/s involved	Device and edge plane – coordinates physical tags and anchors to locate assets
Requirements mapping	• R-P1-1 (CHE location services) • R-P1-2 (CHE location availability) • R-P1-3 (CHE positioning accuracy) • R-P2-1 (Personal location tracking) • R-P2-2 (Construction plant location tracking) • R-P2-3 (Localisation tag for construction workers) • R-P2-11: Geofencing
Use case mapping	• UC-P2-1: Workers' health and safety assurance • UC-P2-2: Geofencing boundaries enforcement • UC-P2-4: Construction site access control
Internal components	Tag and anchor configuration, localisation engine

High-level structure

No changes to D5.3.

Implementation technologies

No changes to D5.3.

Communication interfaces

No changes to D5.3.

Enabler stories

No changes to D5.3.

3.1.4. Location processing enabler

Table 6. General information of the Location processing enabler

Enabler	Location processing enabler
Id	T51E3-B
Owner and support	SRIPAS
Description and main functionalities	The Location Processing enabler provides highly configurable and flexible geofencing capabilities based on location data. It runs user-defined queries against the data storage. The enabler handles data updates and queries using both the HTTP-based request-response and streaming approach.
Vertical, related capabilities and features	Self-* (autonomy)
Plane/s involved	Data management plane – it provides location awareness through location data
Requirements mapping	• R-P1-1: CHE location services • R-P2-1: Personal location tracking • R-P2-2: Construction plant location tracking • R-P2-11: Geofencing • R-C-5: Local Processing Capabilities • R-C-18: Support for autonomous processing • R-C-19: Support for self-aware systems
Use case mapping	• UC-P2-1: Workers' health and safety assurance • UC-P2-2: Geofencing boundaries enforcement • UC-P2-4: Construction site access control
Internal components	Application and Database

High-level structure

The design of the whole enabler has been rethought since D5.3. Initially, the scope covered defining regions and points and then querying them. It was decided that the enabler should incorporate more features provided by the geolocation storage to achieve greater flexibility. In effect, the enabler can be used in various use cases.

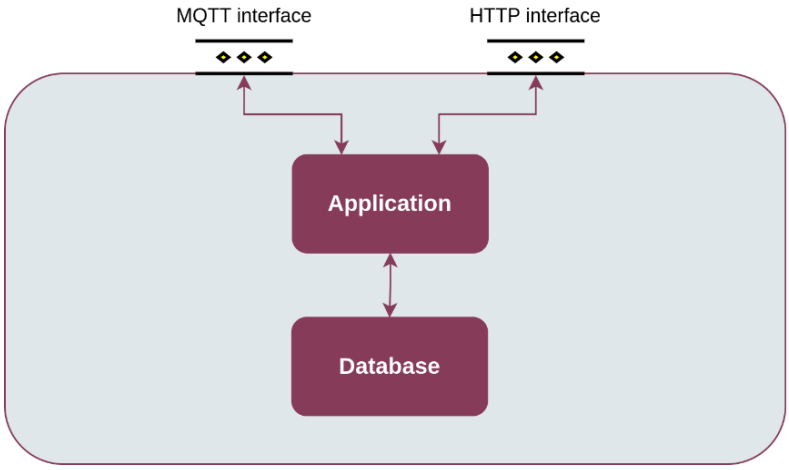

Figure 3. Location processing enabler structure

The application is written with the Akka framework to provide scalability and durability. It runs user-defined SQL queries against the database. The queries can be parametrised using the specialised syntax. In effect, the enabler is highly customisable for different scenarios. The incoming data is collected from input streams or HTTP requests; it allows for streaming the query results. The transferred data is in JSON format. The behaviour of the application is configurable through an HTTP interface. The application streaming capabilities are compatible with the MQTT protocol. The database is shipped with the PostGIS extension. It stores the geolocation data and the application configuration.

Implementation technologies

Although the technologies mentioned were already described in previous deliverables, they are stated again to map them with the final components, because of the redesign of the enabler.

Table 7. Implementation technologies for the Location processing enabler

Technology	Justification	Component(s)
Scala	Main language for developing custom functions for all components of the enabler. Selected for its high quality, good support and adequacy for the task	~~All components except database~~ Application
Akka	All the components of the enabler will be developed using Akka (and Akka Streams) libraries, due to the excellent support they provide for features and communication standards/protocols required for the enabler	~~All components except database~~ Application
PostGIS	PostGIS is a mature spatial extension for PostgreSQL. It natively supports all the GIS-oriented functions needed by the Location processing enabler. Additionally, PostGIS is well supported by existing Java-based libraries (hence it is also Scala-friendly)	~~Location data storage and processing~~ Database
~~**Apache Kafka**~~	~~Selected for its state-of-the-art support for asynchronous buffering and stream handling mechanisms~~	~~Streaming updates/queries~~

Communication interfaces

The design has been rethought since D5.3, and therefore **the previous API calls are invalid**. The following table presents the endpoints that the enabler will include:

Table 8. Communication interfaces of the Location processing enabler

Method	Endpoint	Description
GET	**/v1/queries**	Returns all queries.
GET	**/v1/queries/<name>**	Returns a specific query.
POST	**/v1/queries**	Creates a new query with specified configuration.
POST	**/v1/queries/<name>/input**	Sends input data to a specific query.
PUT	**/v1/queries/<name>**	Updates a query.
DELETE	**/v1/queries/<name>**	Deletes a query.

Enabler stories

The design has been rethought since D5.3, and hence the previous **enabler stories** are substituted by a **new set**.

The **first enabler story** is related to the **configuration of a query** via the HTTP interface. Its flow and steps are the following:

STEP 1: The client invokes the /v1/queries API endpoint, passing the configuration details.

STEP 2: The application verifies the request data.

STEP 3: The application communicates with the database that stores query configurations to perform the requested action.

STEP 4: The client receives the confirmation message.

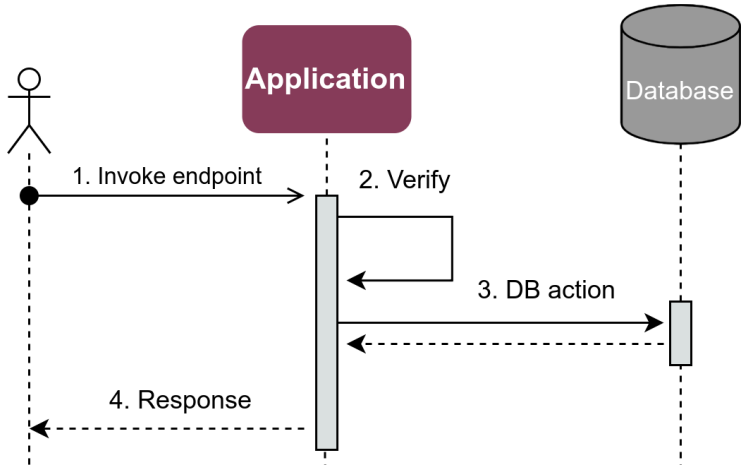

Figure 4. Location processing enabler ES1 (query configuration)

The **second enabler story** involves **passing the input data to a running query** via HTTP interface. Its flow and steps are the following:

STEP 1: The client invokes the */v1/queries/<name>/data* endpoint and passes the input data in the request body.

STEP 2: The application communicates with the called query and inputs the received data.

STEP 3: Query runs SQL statement.

STEP 4: The application responds with the query output data.

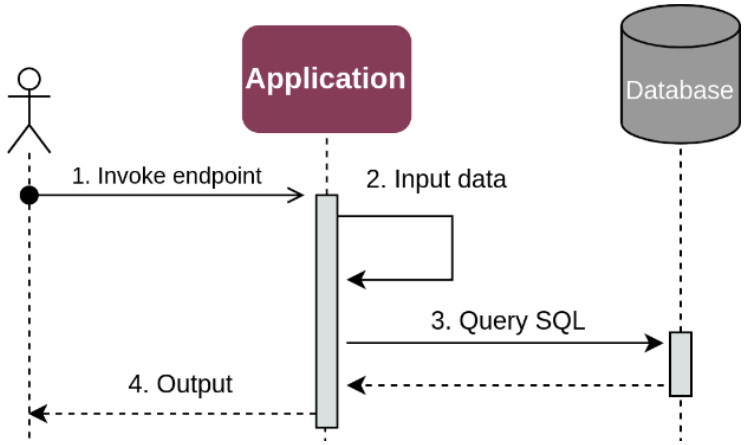

Figure 5. Location processing enabler ES2 (running query input)

3.1.5. Monitoring and notifying enabler

Table 9. General information of Monitoring and notifying enabler

Enabler	Monitoring and notifying enabler
Id	T51E4
Owner and support	CERTH
Description and main functionalities	This enabler IS responsible for monitoring the uninterrupted functionality of devices and notifying in case of malfunction incidents. Specifically, it must ensure the departure of data, the arrival, the validity and its own self-monitoring functionality. • **Device Monitoring**: Another functionality of the enabler is the device monitoring. The enabler ensures that the IoT device reads the required data in fixed time intervals, to control data flooding or data interruption. If not, a notification will be created.

Enabler	Monitoring and notifying enabler
	• **Edge Monitoring**: Furthermore, the enabler is to guarantee the edge monitoring. In more details, the enabler ensure communication with connected IoT devices. If communication between the linked components is lost, a notification will be created. Additionally, it will check for attacks (i.e., sybil attack).
Vertical, related capabilities and features	Self-* (autonomy)
Plane/s involved	Device and edge plane – Devices (smart, IoT) are part of the use cases for various actions Data management plane – Monitoring and notification is based on data (real-time with streaming, historical as reports). Depending on the case, the need for streaming or historical data may arise Application and services plane – Monitoring will have to include an interface for an end user to interact. Notification is similar to the monitoring
Requirements mapping	• R-C-16: Resource monitoring • R-C-18: Support for autonomous processing • R-C-19: Support for self-aware systems • R-P1-1: CHE location services, indirectly, to monitor its correct functionality • R-P2-1: Personal location tracking, indirectly, to monitor its correct functionality • R-P2-3: Smart wristband for construction workers, indirectly, to monitor its correct functionality • R-P2-2: Construction plant location tracking, indirectly, to monitor its correct functionality • R-P2-7: Monitoring the weather conditions at the construction site, indirectly, to monitor its correct functionality • R-P2-9: Assessment of Personal Exposure to UV Radiation, indirectly, to monitor its correct functionality • R-P2-12: Alerts and notifications minimisation
Use case mapping	• UC-P1-1: CHE location tracking • UC-P1-2: Container handling operations reporting • UC-P1-5: RTG-Truck alignment; RTGs and Trucks needs to notify each other about their positions • UC-P2-1: Workers' health and safety assurance; It is required in this UC that after breaching some threshold values we need to send notifications across various components of the system (for example notifying OSH manager or to some components that acts when unauthorised access was detected) • UC-P2-2: Geofencing boundaries enforcement; When you breach fence the notification will be sent. • UC-P2-4: Detection of falls and immobility • UC-P2-5: Safe navigation instructions • UC-P2-6: Health and safety inspection support • UC-P3A-1: Fleet in-service emissions verification • UC-P3A-2: Vehicle's non-conformance causes identification • UC-P3B-1: Vehicle's exterior condition documentation and visualisation
Internal components	Communication Interface, Database, Message queue, Registry, and Module for implementing the logic

High-level structure

No changes to D5.3.

A new functionality is introduced, adding the ability to check the health of edge/IoT devices and gateways on demand and/or in fixed time intervals. If the result of the query discovers devices/gateways which are unreachable, the result is pushed to the DLT distributed broker enabler. In addition, the user will be able to register/delete devices and gateways in the internal storage.

Note: It is under discussion if an additional functionality can be introduced, in order to check the health of K8s nodes/pods on demand and/or on fixed time intervals.

Implementation technologies

No changes to D5.3.

Communication interfaces

No changes to D5.3.

Enabler stories

The **enabler stories** of D5.2 and D5.3 (IoT device which stops receiving data from its integrated sensor – ES1, device entering restricted zone – ES2, and querying vehicle conditions – ES3) remain valid. A new one (**#4**) has been added and formalised below. In this story, a user wants to **check the health of devices or gateways** in order to see if they are connected and communicating with each other.

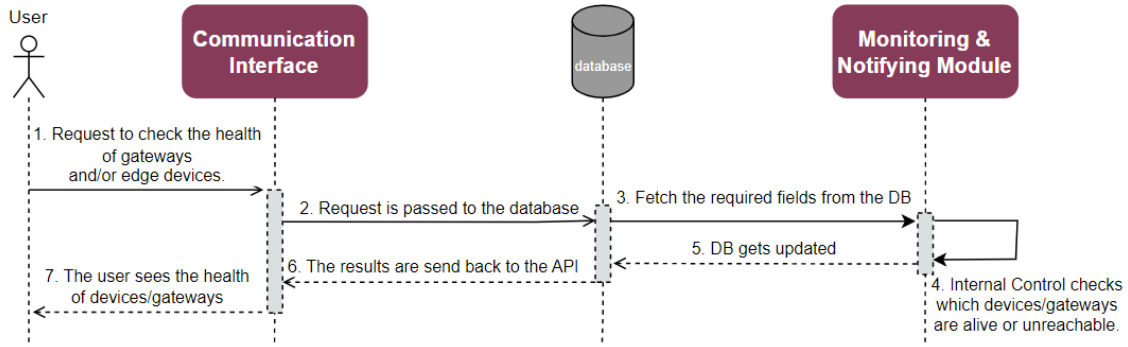

Figure 6. Monitoring and notifying enabler ES4 (check gateways/devices health)

STEP 1: The user sends the request to the enabler's API to check the health of devices and/or gateways.

STEP 2: The request is passed to the database.

STEP 3: The database returns the required fields to the monitoring and notifying module which implements the logic.

STEP 4: The module's internal control system checks which devices and/or gateways are alive or unreachable.

STEP 5: The result is sent to the DB and it gets updated.

STEP 6: The results are sent back to the API for user consumption.

STEP 7: The user checks the results and acts accordingly.

3.1.6. Automated configuration enabler

Table 10. General information of the Automated configuration enabler

Enabler	Automated configuration enabler
Id	T51E5
Owner and support	SRIPAS
Description and main functionalities	This enabler aims to allow users to abstractly define requirements for functionalities and then to check whether those requirements are met by registered resources. The enabler can also automatically react to external actions based on predefined rules. • *Administrator* can define *configuration* of a system. The *configuration* describes what *resources* are required by a specific *functionality*.

Enabler	Automated configuration enabler
	• Administrator can define *reactions* - rules on how the system should behave when a specific *action* has happened. *Reactions* can modify existing *configuration* and emit notifications. • *Self-Configurator* can autonomically decide which *functionalities* will be kept in the event of limited available *resources*. • *Resources* can notify the Self-Configurator about going live and going down.
Vertical, related capabilities and features	Interoperability, Self-* (autonomy)
Plane/s involved	Device and edge plane – the enabler will be used for configuring devices Data management plane – the automated configuration enabler will provide/support interoperability mechanisms for heterogeneous environments
Requirements mapping	• R-C-2: Data governance • R-C-5: Local processing capabilities • R-C-6: Data persistence and trust • R-C-7: Edge-oriented deployment • R-C-18: Support for autonomous processing • R-C-19: Support for self-aware systems • R-C-28: Distributed Configuration • R-P1-16: Open/accessible remote capabilities • R-P3A-5: Data Storage • R-P3A-12: Edge connectivity
Use case mapping	• UC-P1-1: Asset location management • UC-P2-1: Worker's health and safety assurance • UC-P2-6: Safe navigation instructions • UC-P3A-3: Sending new configuration to PCM
Internal components	Self-configurator, Eventstore

High-level structure

Although conceptually the enabler provides the high-level features described in past deliverables, the internal components have been refactored, with the final design as shown in the following figure:

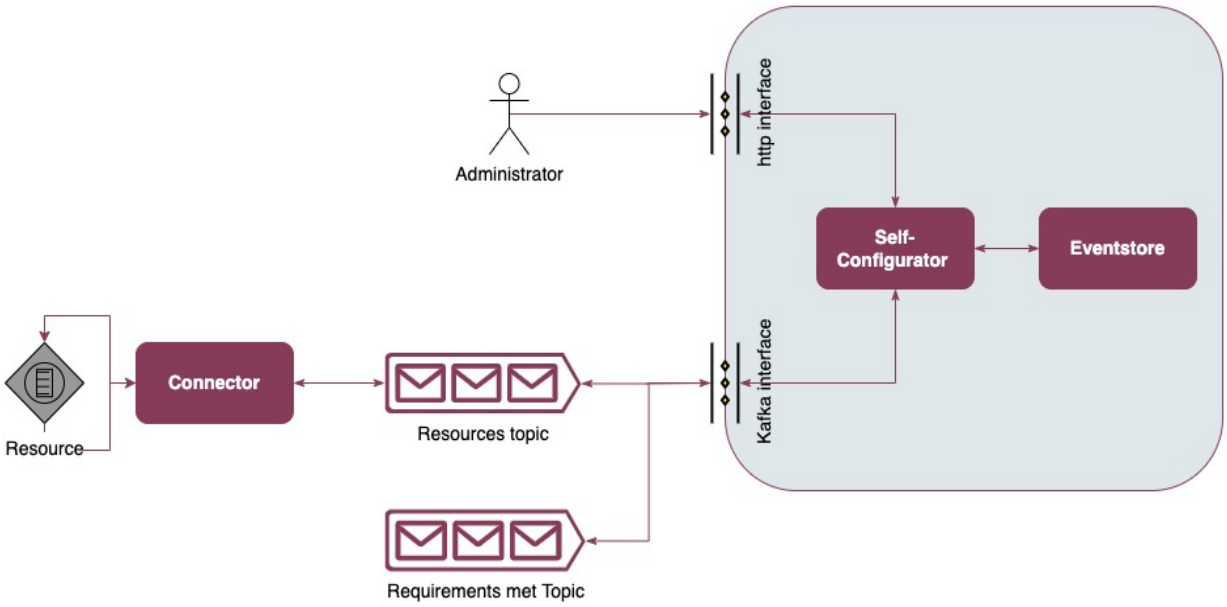

Figure 7. Automated configuration enabler structure

Implementation technologies

Although some of the technologies mentioned were already described in previous deliverables, they are stated again to map them with the final components, because of the conceptual redesign of the enabler. EventStore DB and Akka Persistence have been added to the used technologies for implementing the enabler.

Table 11. Implementation technologies for the Automated configuration enabler

Technology	Justification	Component(s)
Scala 3	Scala is a programming language that supports both object-oriented and functional programming. It can be run on a Java virtual machine or compiled to JavaScript and run in a browser. The latest iteration of Scala language (Scala 3) was released in May 2021 and it is just being adopted by the industry. The language has strong static typing, which can support exhaustive pattern matching, opaque data types, and type classes.	Self-Configurator ~~Configuration applier, Registry, Intelligence~~
Akka and Akka Persistence	Akka is a set of libraries for Scala and Java that implements actor-based concurrency model. It provides abstractions for both message-driven (via communicating actors) and stream-based applications. It is one of the most popular Scala frameworks. It is widely used in both academia and industry. Akka Persistence enables event sourcing for Akka's actors. It realises it by providing Finite State Machine inspired abstraction for actors. By default, it stores only events that change the internal state of the actors, not the state itself. Akka is being actively developed, and it already provides support for Scala 3. Akka Persistence FSM-based abstraction together with an exhaustive pattern matching (Scala feature) can ensure that all defined scenarios are handled.	Self-Configurator ~~Configuration applier, Registry, Intelligence~~
EventStore DB	EventStoreDB is a NoSQL database optimised for Event Sourcing. It stores data as an immutable series of events. As mentioned previously, JDBC-compliant databases were out of the picture, but EventStoreDB provides a plugin that integrates with Akka Persistence and Scala 3 automatically.	Eventstore
Kafka	Kafka is a distributed event streaming platform that provides publisher-subscriber mechanics \cite{kafka-homepage}. It is the most popular solution of its kind in manufacturing, banking, insurance, and telecom companies. Apart from the publisher-subscriber model, Kafka can store streams of events durably and process streams of events either online or offline.	Self-Configurator, Connector ~~All of them~~

Communication interfaces

During implementation, a redesign of the **endpoints** was performed so those **defined in past deliverables are no longer valid**. Currently, the following endpoints apply for the enabler:

Table 12. Communication interfaces of the Automated configuration enabler

Method	Endpoint	Description
POST	**/requirements-model**	Creates or updates requirements model
DELETE	**/requirements-model/{id}**	Deletes requirements model with specific id
POST	**/reaction-model**	Creates or updates reaction model
DELETE	**/reaction-model/{id}**	Deletes reaction model with specific id
RegisterResource	**Resources topic**	Message with which resources register to Self-Configurator
DeregisterResource	**Resources topic**	Message with which resources deregister from Self-Configurator
CustomMessage	**Resources topic**	A custom message that can trigger reactions
RequirementsMet	**Requirements met topic**	Message sent if all requirements are met with available resources
RequirementsNotMet	**Requirements met topic**	Message sent if not all requirements are met with available resources

Enabler stories

No changes to D5.3.

3.2. Federated learning enablers

3.2.1. FL Orchestrator

Table 13. General information of the FL Orchestrator

Enabler	FL Orchestrator
Id	T52E1
Owner and support	PRO, SRIPAS
Description and main functionalities	The FL orchestrator is responsible of specifying details of FL workflow(s)/pipeline(s). This includes FL job scheduling, managing the FL lifecycle, selecting and delivering initial version(s) of the shared algorithm, as well as modules used in various stages of the process, such as training stopping criteria. Finally, it can specify ways of handling different "error conditions" that may occur during the FL process.
Vertical, related capabilities and features	Manageability, Scalability, Privacy
Plane/s involved	Smart network and control plane – a network interface should be managed in the communication between parties – mediators (if any) – masters
Requirements mapping	• R-C-3: Compliance with legal requirements on data protection • R-C-7: Edge-oriented deployment • R-C-10: Transmission security • R-C-21: Reduction of computing demands for AI training • R-C-22: Support for data privacy during the training process • R-C-23: Multi-model FL support • R-C-24: Cooperative ML training support • R-C-25: Holistic security/privacy approach • R-C-28: Distributed configuration • R-P3A-9: Edge intelligence • R-P3B-13: Automatic defect detection
Use case mapping	• UC-P1-7: Target visualisation during RTG operation • UC-P2-1: Worker's health and safety assurance • UC-P3A-2: Vehicle non-conformance causes identification • UC-P3B-1: Vehicle's exterior condition documentation
Internal components	FLS API Server, FLS Workflow manager

High-level structure

No changes to D5.3.

Implementation technologies

No changes to D5.3.

Communication interfaces

Since there are many changes in the API with respect to D5.3, it is preferred to attach the new and final set of interfaces rather than point to all the changes that it has suffered from the previous version. Hence, the current endpoints are as follows:

Table 14. Communication interfaces (API) of the FL Orchestrator

Method	Endpoint	Description
GET	**/configurationsbyModel/<id>**	Recover configurations of model <id> collected in the FL Repository

GET	**/AddModelData/<id>**	Store into the internal FL Orchestrator database, the retrieved FL algorithm from the FL Repository
GET	**/ModelData/<id>**	Append FL training configuration with the FL algorithm retrieved from the FL repository of the /appModelData endpoint
POST	**/StoreConfigurationModel**	Store into the internal FL Orchestrator database, the complete FL algorithm + training configuration (i.e., store the resulting JSON from /ModelData endpoint)
GET	**/configurationsbyModel/<id>**	Recover configurations of model <id> collected in the FL Orchestrator database
GET	**/GetConfigurations**	Recover all FL algorithms plus training configurations stored in the FL Orchestrator database
POST	**/UpdateConfigurationModel**	Update training configuration of the FL algorithm to be used for training
POST	**/TrainingModel/<id>**	Start FL training with the configuration <id> defined in the WebUI
GET	**/RecoverStatusfromEnablers**	Receive the status of all the FL Local Operations involved in the training
POST	**/FLTrainingRound**	Recover last finalised training round from the FL Training Collector
GET	**/RecoverTrainingEpochs/<epochs>/<total_epochs>**	Get the current epoch being trained and the total number of epochs to be trained

Enabler stories

No changes to D5.3.

3.2.2. FL Training collector

Table 15. General information of the FL Training Collector

Enabler	**FL Training Collector enabler**
Id	T52E2
Owner and support	SRIPAS, PRO
Description and main functionalities	The FL training process involves several independent parties that commonly collaborate in order to provide an enhanced ML model. In this process, the different local updates suggestions shall be aggregated accordingly. This duty within ASSIST-IoT is tackled by the FL Training Collector, which is also be in charge of delivering back the updated model. In the context of this enabler, two aspects of the FL process need to be supported: • The combination of local results to deliver new version of the shared model, as there exist multiple ways of combining them (e.g., FedSGD, FedAvg) . • Different topologies of FL-running systems. The FL Training Collector will consist of two components: (i) the combiner, responsible of providing updates with respect to the shared averaged model, and (ii) the I/O component, which will carry out the input and output communications of the enabler.
Vertical, related capabilities and features	Privacy, Scalability
Plane/s involved	Smart network and control plane – a network interface should be managed in the communication between parties – mediators (if any) – masters
Requirements mapping	• R-C-3: Compliance with legal requirements on data protection • R-C-7: Edge-oriented deployment • R-C-10: Transmission security • R-C-21: Reduction of computing demands for AI training • R-C-22: Support for data privacy during the training process • R-C-23: Multi-model FL support • R-C-24: Cooperative ML training support • R-C-25: Holistic security/privacy approach • R-C-28: Distributed configuration • R-P3A-9: Edge intelligence

Enabler	FL Training Collector enabler
	• R-P3B-13: Automatic defect detection
Use case mapping	• UC-P1-7: Target visualisation during RTG operation • UC-P2-1: Worker's health and safety assurance • UC-P3A-2: Vehicle non-conformance causes identification • UC-P3B-1: Vehicle's exterior condition documentation
Internal components	FLTC I/O, FLTC Combiner

High-level structure

No changes to D5.3.

Implementation technologies

No changes to D5.3.

Communication interfaces

No changes to D5.3.

Enabler stories

No changes to D5.3.

3.2.3. FL Repository

Table 16. General information of the FL Repository

Enabler	FL Repository enabler
Id	T52E3
Owner and support	SRIPAS, PRO
Description and main functionalities	One of key aspects of application of Federated Learning in IoT ecosystems is making it configurable. In this context, the FL Repository enabler was proposed. This repository stores (and delivers upon request/need) the ML algorithms or ML models. It consists of four main components: ML Algorithms libraries (that gathers ML algorithms in its first stage, i.e., without involving any modelling associated with a particular training data set), ML models libraries (intermediary or final versions of ML models, once they have been already trained with a particular data set), Collectors (averaging algorithms to be used on the FL training process – if used), and Auxiliary component (for any needed additional module). The FL Repository is a set of different databases.
Vertical, related capabilities and features	Privacy, Scalability
Plane/s involved	Device and edge plane – as the FL repository may be instantiated with edge devices Data management plane – in order to collect several ML data repositories
Requirements mapping	• R-C-3: Compliance with legal requirements on data protection • R-C-7: Edge-oriented deployment • R-C-10: Transmission security • R-C-21: Reduction of computing demands for AI training • R-C-22: Support for data privacy during the training process • R-C-23: Multi-model FL support • R-C-24: Cooperative ML training support • R-C-25: Holistic security/privacy approach • R-C-28: Distributed configuration • R-P3A-9: Edge intelligence • R-P3B-13: Automatic defect detection
Use case mapping	• UC-P1-7: Target visualisation during RTG operation

Enabler	FL Repository enabler
	• UC-P2-1: Worker's health and safety assurance • UC-P3A-2: Vehicle non-conformance causes identification • UC-P3B-1: Vehicle's exterior condition documentation
Internal components	ML Algorithms Libraries, FL Collectors, Auxiliary, ML Model Libraries

High-level structure

No changes to D5.3.

Implementation technologies

No changes to D5.3.

Communication interfaces

No changes to D5.3.

Enabler stories

The stories of the enabler described in deliverable D5.2 remain valid (FL Orchestrator retrieves ML algorithm – ES1, FL Training Collector retrieves ML collector algorithm – ES2, and FL Training Collector sends updated model – ES3). Apart from them, a new **use story (#4)** related with the **trained FL model offering of previous FL trainings**. Its diagram and flows are the following:

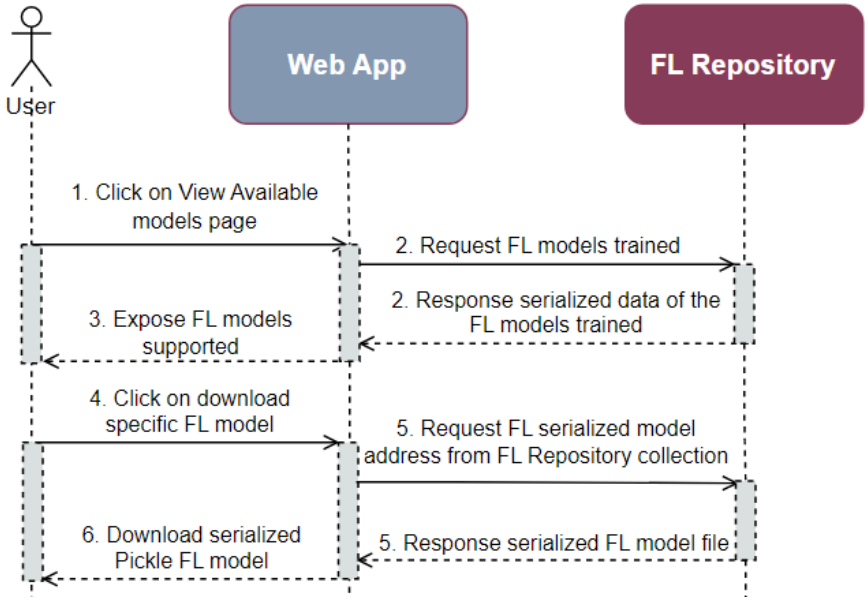

Figure 8. FL Repository ES4 (download FL serialised Pickle file)

STEP 1: User opens the second tab of the FL system webpage.

STEP 2: Automatically, the FL Orchestrator perform an API request to the FL repository to visualise the already stored and finished FL models.

STEP 3: These models are presented in a table format to the user, which can download them as serialised pickle files by clicking on the download button.

STEP 4: The user decides to download a FL model.

STEP 5: A request is made to the repository, which provides the serialised model.

STEP 6: The user obtains the model.

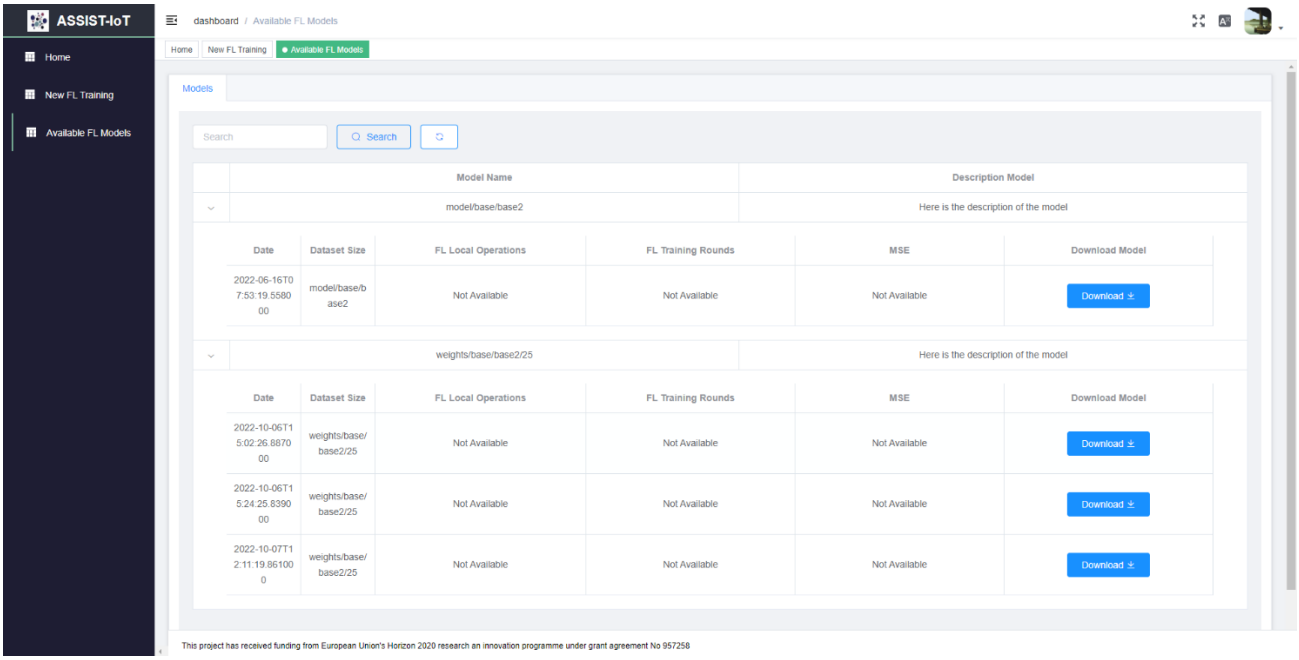

Figure 9. Screenshot of FL WebApp download page

3.2.4. FL Local operations

Table 17. General information of the FL Local Operations enabler

Enabler	**FL Local Operations**
Id	T52E4
Owner and support	SRIPAS, PRO
Description and main functionalities	One of key goals of FL is to assure protection of data privacy, owned by individual stakeholders. Therefore, data is expected to be used only locally, to train local version of the shared model, and only parameters update proposals of the ML algorithm are shared with other master or other participants. When the FL training process has concluded, the final shared ML model is used to deliver specific functionality, also called inference engine. Both operations (model training and model inference) involve access to private data. This means that it is crucial to "encapsulate" local processes within a single "node" (that is controlled by data owner). However, it should be noticed that the data that is being used in both FL training processes must be in the same format, which is imposed by the ML model that is being employed. To carry out with all these local operations, the FL Local Operations enabler was proposed. FL Local Operations enabler is an embedded enabler within each FL involved party/device of the FL systems.
Vertical, related capabilities and features	Privacy, Scalability
Plane/s involved	Smart Network and Control plane – a network interface should be managed in the communication between parties – mediators (if any) – masters
Requirements mapping	• R-C-3: Compliance with legal requirements on data protection • R-C-7: Edge-oriented deployment • R-C-10: Transmission security • R-C-21: Reduction of computing demands for AI training • R-C-22: Support for data privacy during the training process • R-C-23: Multi-model FL support • R-C-24: Cooperative ML training support • R-C-25: Holistic security/privacy approach • R-C-28: Distributed configuration

Enabler	FL Local Operations
	• R-P3A-9: Edge intelligence • R-P3B-13: Automatic defect detection
Use case mapping	• UC-P1-7: Target visualisation during RTG operation • UC-P2-1: Worker's health and safety assurance • UC-P3B-1: Vehicle's exterior condition documentation • UC-P3B-2: Exterior defects detection support
Internal components	Local Data Transformer (that is charge of guaranteeing that data is appropriately formatted for the FL model in use), Local Model Training, Local Model Inferencer, Privacy

High-level structure

No changes to D5.3.

Implementation technologies

The implementation technologies of the enabler have not changed since the deliverable D5.3 (scikit-learn, pyTorch, TensorFlow, Flower, OpenVINO, OpenCV, Python, homomorphic encryption algorithms and FastAPI) besides the extension indicated in the following table:

Table 18. Implementation technologies for the FL Local Operations enabler

Technology	Justification	Component(s)
Sonora	ADDED Sonora is a Python-first implementation of gRPC-Web built on top of standard Python APIs like WSGI and ASGI for easy integration. It will be used to provide gRPC interface for the inference	Local Model Inferencer

Communication interfaces

The API presented in D5.3 remains valid. Additionally, a new interface has been added to the enabler:

Table 19. Communication interfaces of the FL Local Operations enabler

Method	Endpoint	Description
Inference	**gRPC**	Added gRPC interface for inference with high throughput requirements

Enabler stories

No changes to D5.3.

3.3. Cybersecurity enablers

3.3.1. Identity manager enabler

Table 20. General information of the Identity Manager enabler

Enabler	Identity manager enabler
Id	T53E2
Owner and support	S21Sec
Description and main functionalities	Identity manager enabler (IdM) performs the authentication phase of access control process, that is, it is responsible for managing identities on the access control process. Identity manager processes and validates the identity, for later control of the access to the resources by the authorisation enabler. The main goal of identity management is to ensure that only authenticated entities are granted access to the specific resource (applications, systems, or IT environments) for which they are authorised. This includes control over entities (i.e., user provisioning, or entities provisioning) and the process of onboarding new entities (i.e., users, systems, etc.). IdM enabler relies on OAuth2 protocol.

Enabler	Identity manager enabler
	The IdM provides a central user database and management console. It is also able to work federated with remote user databases, unifying remote user stores. It provides Single-Sign-On capabilities through OAuth2 protocol. The IdM integrates with the Authorisation enabler in order to offer a common authorisation and authentication process.
Vertical, related capabilities and features	Security, privacy and trust
Plane/s involved	All planes – as it can provide identity management for accessing configuration and/or services offered by enablers of any plane
Requirements mapping	• R-C-7: Edge-oriented deployment • R-C-25: Holistic security/privacy approach
Use case mapping	• UC-P1-4: RTG-Truck identification and authentication • UC-P1-5: RTG-Truck alignment • UC-P2-5: Safe navigation instruction • UC-P2-6: Health and safety inspection support • UC-P3A-3: Updating the diagnostics methods pool • UC-P3B-2: Exterior defects detection support
Internal components	IdM administration module, IdM authentication module, Local user database, Remote user database (optional)

High-level structure

No changes to D5.3.

Implementation technologies

No changes to D5.3. Extended to work over machines with low resources, based on ARM64 architectures.

Communication interfaces

No changes to D5.3.

Enabler stories

No changes to D5.3.

3.3.2. Authorisation enabler

Table 21. General information of the Authorisation enabler

Enabler	Authorisation enabler
Id	T53E1
Owner and support	S21Sec
Description and main functionalities	Authorisation enabler is responsible for the authorisation phase in the access control process. Authorisation is a process of granting, or automatically verifying, permission to an entity (computer, application, or person) to access requested information after the entity has been authenticated. The enabler is based on XACML standard security policies, results on obligations actions to be deployed after the evaluation process. There are two different modes of deploying the same enabler, it can function as federated server, autonomous edge service or interact between both. In ASSIST-IoT a federated Authorisation enabler distributes a security policy from cloud to the edge to be locally evaluated and enforced. The authorisation enabler provides a Web administrator (PAP) to create and deploy the security policy to the different devices. In the service side that wants to use the authorisation service, there is an enforcement point (PEP) to make request to the authorisation server, this is, asking whether the access should be granted or not, through a REST interface available to receive the request and orchestrate the process. It is also possible to add Information Points (PIP) to generate the context for the request and obtain any data that

Enabler	Authorisation enabler
	external provider can offer to be incorporated to the request. Finally, the Authorisation enabler evaluates the requests against the policy, that is locally stored in the policy repository, and makes the decision (PDP) about the authorisation. It is also possible to invoke an obligation server to launch external requests to perform the derived actions (obligations) obtained as a result of the policy decision, through a REST requests.
Vertical, related capabilities and features	Security, privacy and trust
Plane/s involved	All planes – as it can implement authorisation schemas for deciding who (user, enabler, external service) can configure and/or consume enablers offered from any plane
Requirements mapping	• R-C-7: Edge-oriented deployment • R-C-25: Holistic security/privacy approach
Use case mapping	• UC-P1-4: RTG-Truck identification and authentication • UC-P1-5: RTG-Truck alignment • UC-P2-5: Safe navigation instruction • UC-P2-6: Health and safety inspection support • UC-P3A-3: Updating the diagnostics methods pool • UC-P3B-2: Exterior defects detection support
Internal components	PAP (Policy Administration Point), PDP (Policy Decision Point), PEP (Policy Enforcement Point), PIP (Policy Information Point)

High-level structure

No changes to D5.3.

Implementation technologies

No changes to D5.3. Extended to work over machines with low resources, based on ARM64 architectures.

Communication interfaces

The updated list of endpoints is presented here. From D5.3, four new endpoints (highlighted in green) have been added:

Table 22. Communication interfaces (API) of the Authorisation enabler

Method	Endpoint	Description
GET	**/evaluate?resource=<domain>@<resource>&action=<action>&code=<id>**	Evaluates a request performed and authorises it or not depending on the stored policies
GET	**/evaluate/trace?resource=<domain>@<resource>&action=<action>&code=<id>**	Provides detailed info of evaluation data, such as, Context, XACML REQUEST, XACML RESPONSE and JSON RESULT of the XACML Evaluation.
POST	**evaluate?resource=<domain>@<resource>&action=<action>&code=<id>**	Enables to create a local PIP (Policy Information Point), with the aim of consulting any information should be needed for the authorization decision.
GET	**/rest/fedprov/<pip_name>/<in_value>**	Enables the exchange of security policies between two Authorization servers.
POST	**/rest/poldbimport**	Evaluates a request performed and authorises it or not depending on the stored policies

Enabler stories

No changes to D5.3. But a previous (out of band) step should be added to the flow to reflect a possible exchange of a security policy between an authorisation enabler in the Cloud to an Authorisation enabler at the edge:

STEP 0: A policy defined in the Authorisation enabler in the Cloud is exported to and received by the Authorisation enabler in the Edge, to be evaluated locally in the Edge when corresponds.

3.3.3. Cybersecurity monitoring enabler

Table 23. General information of the Cybersecurity monitoring enabler

Enabler	Cybersecurity monitoring enabler
Id	T53E3
Owner and support	S21Sec
Description and main functionalities	Cybersecurity monitoring enabler provides cyber security awareness and visibility on cybersecurity objectives and provides infrastructure cybersecurity monitoring. The enabler is responsible of collecting, processing, and analysing the incoming logs and information from the infrastructure under study. It decodes the information and applies security active rules from an existing ruleset to this information. If there is a match, it generates an alert. It normalises and enriches the alert to provide a consolidated output that provides cybersecurity monitoring information related to different events and facilitates the assignment of the risk level of the incident and the response actions to be done. The cybersecurity enabler can do predefined actions for incident mitigation depending on the incident, such as to communicate with the agent so that it performs an action, send an email or send the incident to a ticketing system. This enabler will update information on a graphical interface, so that an admin user can see the status of the alert/incident information, as well as the status of the related monitoring agents.
Vertical, related capabilities and features	Security, privacy and trust
Plane/s involved	All planes – as it can monitor all layers, from device threats to network, services and application threats and vulnerabilities. Responses to certain events can be implemented and automated
Requirements mapping	• R-C-7: Edge-oriented deployment • R-C-18: Support for autonomous processing • R-C-19: Support for self-aware systems • R-C-25: Holistic security/privacy approach • R-C-26: Optimised Security notification • R-C-28: Distributed Configuration
Use case mapping	• UC-P1-4: RTG-Truck identification and authentication • UC-P1-5: RTG-Truck alignment • UC-P2-5: Safe navigation instruction • UC-P2-6: Health and safety inspection support • UC-P3A-3: Updating the diagnostics methods pool • UC-P3B-2: Exterior defects detection support
Internal components	Alert treatment module, Incident response module, GUI

High-level structure

No changes to D5.3.

Implementation technologies

No changes to D5.3. It has been extended to work over machines with low resources, based on ARM64 architectures.

Communication interfaces

No changes to D5.3.

Enabler stories

No changes to D5.3.

3.3.4. Cybersecurity monitoring agent enabler

Table 24. General information of the Cybersecurity monitoring agent enabler

Enabler	Cybersecurity monitoring agent enabler
Id	T53E4
Owner and support	S21Sec
Description and main functionalities	Cybersecurity monitoring agent enabler performs functions of an endpoint detection and response system, monitoring and collecting activity from end points that could indicate a cybersecurity threat. The Cybersecurity monitoring agent enables the execution of processes on the system target under study to collect relevant information if a cybersecurity breach is produced. It reports to the Cybersecurity monitoring enabler. The Cybersecurity monitoring agent collects and processes system events and system log messages. It monitors file integrity of critical files and audit data of the system. It also monitors the security of the Docker engine API and the container at runtime. It is also able to perform some actions such as blocking network connection or stopping running processes if the Cybersecurity monitoring enabler requests it.
Vertical, related capabilities and features	Security, privacy and trust
Plane/s involved	Device and edge plane – installed at the monitored hosts, this agent will collect all the necessary logs and data to be processed and analysed by the Cybersecurity monitoring enabler Rest of the Planes – logs and data belong to enablers/services/interfaces belonging to them
Requirements mapping	• R-C-5: Local Processing Capabilities • R-C-25: Holistic security/privacy approach • R-C-26: Optimised Security notification
Use case mapping	• UC-P1-4: RTG-Truck identification and authentication • UC-P1-5: RTG-Truck alignment • UC-P2-5: Safe navigation instruction • UC-P2-6: Health and safety inspection support • UC-P3A-3: Updating the diagnostics methods pool • UC-P3B-2: Exterior defects detection support
Internal components	The agent module

High-level structure

No changes to D5.3.

Implementation technologies

No changes to D5.3.

It should be noted here that, although initially the Cybersecurity monitoring agent was expected to run on containerised approach, it has not been finally implemented in this way, as described in D3.7.

The monitoring agent requires to be linked to the monitoring server as well as a registering process to this component. Due to the ephemeral behaviour of containerised solutions, the register of the agents to the server would be lost if the agent container disappears. Hence, it could be considered as an exception to run agents in an encapsulated environment, like a containerised solution. Also, security agent enabler might need to monitor host services and interfaces that might not reachable if deployed as a container. The proposed solution for overcoming this exception is to deploy the cybersecurity monitoring agent as a service in the underlying host system running the containerised solution, and then evaluating the way to redirect the information that the agent needs to collect from the monitored components running encapsulated.

Communication interfaces

No changes to D5.3.

Enabler stories

No changes to D5.3.

3.4. DLT-based enablers

3.4.1. Logging and auditing enabler

Table 25. General information of the Logging and auditing enabler

Enabler	Logging and auditing enabler
Id	T54E1
Owner and support	CERTH
Description and main functionalities	This enabler will log critical actions that happen during the data exchange between ASSIST-IoT stakeholders to allow for transparency, auditing, non-repudiation and accountability of actions during the data exchange. It will also log resource requests and identified security events to help to provide digital evidence and resolve conflicts between stakeholders, when applicable. If any requirement of filtering prior to logging, a filtering module will be considered to be deployed. The DLT API is the candidate component for performing any filtering.
Vertical, related capabilities and features	Security, privacy and trust
Plane/s involved	Device and edge plane – critical events are stored in the ledger Rest of the Planes – any critical action of the rest of the planes can be stored
Requirements mapping	• R-C-7: Edge-oriented deployment • R-C-27: Automated accountability • R-P1-1: CHE location services • R-P2-1: Personal location tracking • R-P2-2: Construction plant location tracking • R-P2-7: Monitoring the weather conditions at the construction site • R-P2-9: Assessment of Personal Exposure to UV Radiation capability • R-P3A-6: Active monitoring mode initiation by the OEM software engineer • R-P3B-19: Critical Damage Identification Time
Use case mapping	• UC-P1-1: Asset location management • UC-P1-2: CHE location tracking • UC-P2-1: Workers' health and safety assurance • UC-P2-2: Geofencing boundaries enforcement • UC-P2-3: Danger zone restrictions enforcement • UC-P2-4: Construction site access control • UC-P2-5: Near-miss fall from height detection • UC-P2-7: Health and safety inspection support • UC-P3A-1: Fleet in-service emissions verification • UC-P3B-1: Vehicle's exterior condition documentation
Internal components	Logging and auditing business logic (Smart Contracts), Hyperledger Fabric peers and orderers, Certificate Authorities (CAs), REST API

High-level structure

No changes to D5.2.

Implementation technologies

No changes to D5.3.

Communication interfaces

No changes to D5.3.

Enabler stories

No changes to D5.3.

3.4.2. Data integrity verification enabler

Table 26. General information of the Data integrity verification enabler

Enabler	Data integrity verification enabler
Id	T54E2
Owner and support	CERTH
Description and main functionalities	This is an enabler responsible for providing DLT-based data integrity verification mechanisms that allow data consumers to verify the integrity of any data at question. Network peers host smart contract (chaincode) which includes the data integrity business logic. It stores hashed data in a data structure and it compares it with the hashed data of the queries made by clients in order to verify their integrity.
Vertical, related capabilities and features	Security, privacy and trust
Plane/s involved	All planes – verification is performed specifically over data generated or processed at the different planes
Requirements mapping	• R-C-6: Data Persistence and Trust • R-C-7: Edge-oriented deployment • R-P3A-9: Edge Intelligence
Use case mapping	• UC-P3A-1: Fleet in-service emissions verification
Internal components	Data integrity verification business logic (Smart Contracts), Hyperledger Fabric peers and orderers, Certificate Authorities (CAs), REST API

High-level structure

No changes to D5.2.

Implementation technologies

No changes to D5.3.

Communication interfaces

No changes to D5.3.

Enabler stories

No changes to D5.3.

3.4.3. Distributed broker enabler

Table 27. General information of the Distributed broker enabler

Enabler	Distributed broker enabler
Id	T54E3
Owner and support	CERTH
Description and main functionalities	This enabler will provide a mechanism that will facilitate data sharing between different heterogeneous IoT devices belonging to various edge domains and/or between different enablers of the architecture. In coordination with other enablers that will ensure trust between data sources (i.e. Identity and Authorisation providers), it will deal with data source metadata management and provide trustable, findable, and retrievable metadata for the data sources.
Vertical, related capabilities and features	Security, privacy and trust

Enabler	Distributed broker enabler
Plane/s involved	All planes – data source metadata are stored in the ledger
Requirements mapping	• R-C-7: Edge-oriented deployment • R-P3A-1: Monitored Data channels* • R-P2-15: BIM data models and interoperability compliance
Use case mapping	• UC-P2-2: Geofencing boundaries enforcement • UC-P2-3: Danger zone restrictions enforcement • UC-P2-6: Safe navigation instructions • UC-P2-7: Health and safety inspection support • UC-P3A-1: Fleet in-service emissions verification* • UC-P3A-2: Vehicle non-conformance causes identification* • UC-P3A-3: Updating the diagnostics methods pool*
Internal components	Distributed broker business logic (Smart Contracts), Hyperledger Fabric peers and orderers, Certificate Authorities (CAs), REST API

***** Pending to be confirmed

High-level structure

No changes to D5.2.

Implementation technologies

No changes to D5.3.

Communication interfaces

No changes to D5.3.

Enabler stories

No changes to D5.3

3.4.4. DLT-based FL enabler

Table 28. General information of the DLT-based FL enabler

Enabler	DLT-based FL enabler
Id	T54E4
Owner and support	CERTH
Description and main functionalities	This enabler will foster the use of DLT-related components to exchange the local, on-device models (or model gradients) in a decentralised way. The DLT can act as a component to manage AI contextual information and prevent any alteration to the data. The alteration of data is a threat to the Federated Learning approach and the DLT can help in mitigating the threat. Moreover, the enabler will allow mitigating single-point of failures. Finally, the enabler can be charged with validating the individually trained models to rule out malicious updates that can harm the global model.
Vertical, related capabilities and features	Security, privacy and trust
Plane/s involved	Device and edge plane – metadata are stored in the ledger Data management plane – metadata are stored in the ledger
Requirements mapping	• R-C-7: Edge-oriented deployment • R-C-22: Support for data privacy during the training process • R-P3A-9: Edge Intelligence • R-P3A-12: Edge Connectivity
Use case mapping	• UC-P3A-2: Vehicle's non-conformance causes identification • UC-P3B-1: Vehicle's exterior condition documentation

Enabler	DLT-based FL enabler
Internal components	DLT-base FL business logic (Smart Contracts), Hyperledger Fabric peers and orderers, Certificate Authorities (CAs), REST API

High-level structure

No changes to D5.2.

Implementation technologies

No changes to D5.3.

Communication interfaces

No changes to D5.3.

Enabler stories

No changes to D5.3.

3.5. Manageability enablers

3.5.1. Enablers manager

Table 29. General information of the Enablers manager

Enabler	Enablers manager
Id	T55E1
Owner and support	UPV
Description and main functionalities	This enabler will serve as a registry of enablers and, in case they are deployed, the retrieval of their status. In particular, it will: • Enable registering enabler repositories. • Allow the deployment of an enabler (this is, from an ASSIST-IoT repository), as well as its termination, re-instantiation and elimination. Deployments can be made in specific clusters or nodes, or in all clusters. • Retrieve a list of currently-running enablers, with access to dedicated graphical management interfaces (when available) as well as logs (if the enabler with log collection capabilities is in place).
Vertical, related capabilities and features	Manageability
Plane/s involved	All planes – as it can manage enablers belonging to any of them.
Requirements mapping	• R-C-7: Edge-oriented deployment • R-C-9: Workload placement • R-C-28: Distributed Configuration
Use case mapping	All use cases will require of this enabler to be fulfilled
Internal components	Dashboard and Backend

High-level structure

No changes to D5.3, although the naming of the enabler has been changed.

Implementation technologies

No changes to D5.3.

Communication interfaces

No changes to D5.3.

Enabler stories

The stories depicted in deliverable D5.3 remain valid (show deployed enablers – ES1, deploy an enabler – ES2, terminate an enabler – ES3, delete an enabler – ES4, and show enabler logs – ES5). The logic of the backend will be modified to support an additional **enabler story (#6)**, which will consist in **deploying a previously terminated enabler**. The flow will be as follows, essentially identical to the second story described in D5.3, with some modifications in the backend's logic:

STEP 1: The user interacts with the tactile dashboard and clicks on the "re-deploy enabler" button, starting the flow.

STEP 2: The dashboard sends an HTTP POST request to its backend to re-deploy a terminated enabler.

STEP 3: Before asking for it, the backend request information about the terminated enabler to the Smart Orchestrator API.

STEP 4: Once the backend has it, it interacts with the Smart Orchestrator API again to re-deploy it, which returns the result of the operation.

STEP 5: If the enabler has been deployed successfully, the dashboard shows to the user the updated list of enablers. To that end, the first enabler story takes over. It should include the recently-added enabler.

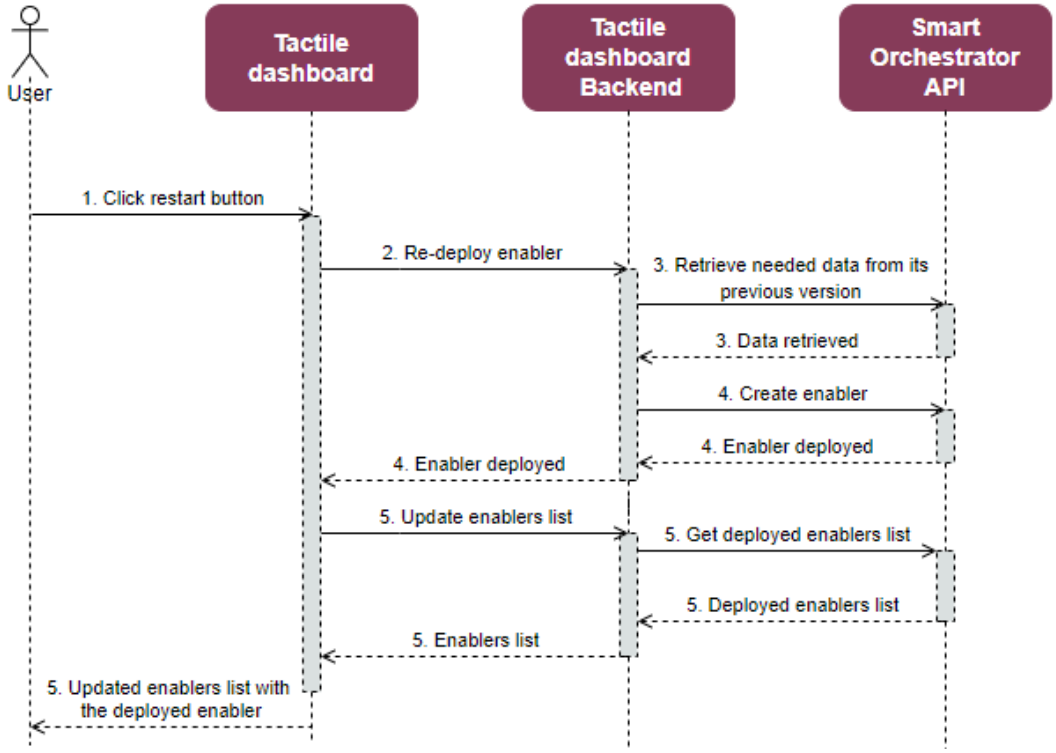

Figure 10. Enablers manager ES6 (re-deploy a terminated enabler)

3.5.2. Composite services manager

Table 30. General information of the Composite services manager

Enabler	Composite services manager
Id	T55E3
Owner and support	UPV
Description and main functionalities	This enabler provides a graphical environment where ASSIST-IoT administrators can connect different enablers to compose a chain, or composite service, easing the realisation of data pipelines. To do so, this enabler is able to provision basic agents to ease data movement among other enablers, by: • Bridging protocols (HTTP & MQTT, at least) • Offering graphical configuration possibilities • Deploying these agents in the right spot of the computing continuum

Enabler	Composite services manager
	• Abstracting IP addresses and port-related information of the involved services • Transmitting payloads as they come, or performing basic payload transformations.
Vertical, related capabilities and features	Manageability
Plane/s involved	Data management plane – it will primarily work with enablers that manage data. Some enablers belonging to the Device and edge plane might also be involved
Requirements mapping	• R-C-7: Edge-oriented deployment • R-C-9: Workload placement • R-C-28: Distributed Configuration • R-C-29: Configurable data flows
Use case mapping	All use cases will require of this enabler to be fulfilled
Internal components	Dashboard and Backend

High-level structure

No changes to D5.3, although the naming of the enabler has been changed. Regarding its backend logic, this enabler will not have the ability to deploy enablers in case these are not already present in a system, and therefore is features have been changed from the previous deliverable iteration.

Implementation technologies

No changes to D5.3 (PUI9, Vue, Java, Spring and PostrgreSQL) besides the selection of the visual technology that will be used in the frontend:

Table 31. Implementation technologies for the Composite service manager

Technology	Justification	Component(s)
Node-RED	This technology is used to graphically setup the agents to interconnect services, generating a JSON file with data related to the involved services, protocols and payload transformations, if any. The dashboard will send the generated file to the backend.	Dashboard

Communication interfaces

No changes to D5.3.

Enabler stories

Four enabler stories are foreseen for this enabler. The **first one** involves **listing the existing pipelines** present in the system. From this list, a user can add pipelines with new agents, modify or delete existing ones. Its flow and steps are the following:

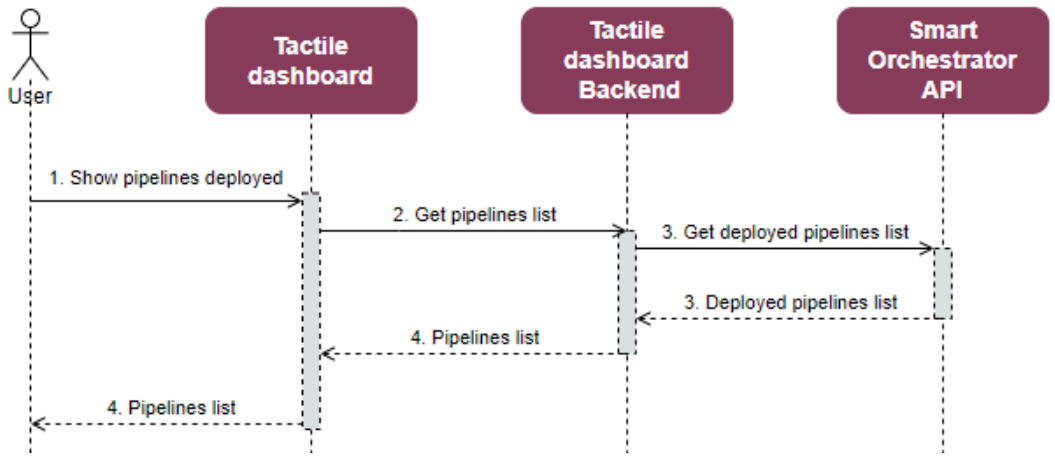

Figure 11. Composite services manager ES1 (list pipelines)

STEP 1: The user interacts with the tactile dashboard to retrieve the pipelines deployed, starting the flow.

STEP 2: The dashboard sends an HTTP POST request to its backend to get the current list of active pipelines.

STEP 3: Then, the backend request information about the active pipelines to the Smart Orchestrator API.

STEP 4: The answer from the Orchestrator is sent to the backend, which makes a first filtering of information which is later on formatted in the dashboard.

The **second enabler story** consists in the creation of **new pipelines**, and in this case the interaction is the following:

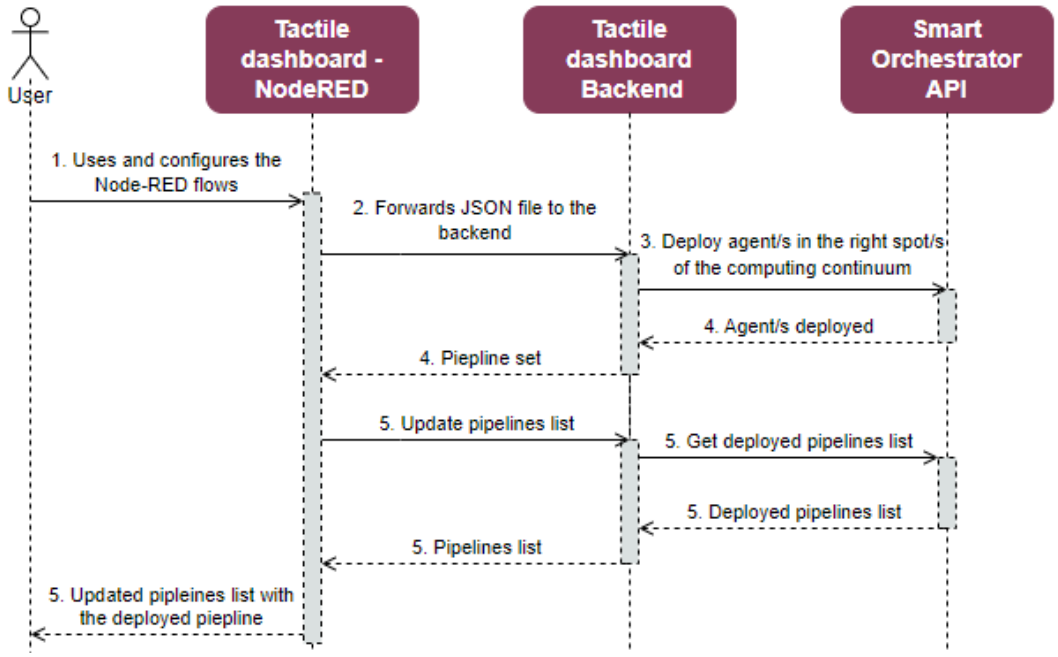

Figure 12. Composite services manager ES2 (create pipeline)

STEP 1: The user interacts with the tactile dashboard, specifically with the graphical Node-RED interface, and sets up the pipeline (selecting the involved input-output protocols, endpoints and specifying the data transformations, if any).

STEP 2: The dashboard sends an HTTP POST request to its backend to deploy the required agents. To that end, the backend interprets the JSON file received from the dashboard and translates it to commands that the Orchestrator can manage.

STEP 3: The backend asks to deploy the required agent/s to the Orchestrator.

STEP 4: Once these are deployed, a confirmation message is sent to the backend.

STEP 5: If the pipeline has been deployed successfully, the dashboard shows to the user the updated list of pipelines. To that end, the first enabler story takes over. It should include the recently-added pipeline.

The **third enabler story** comes to play when a **pipeline** is to be **deleted**. In this case, the flow and involved steps are the ones indicated below:

STEP 1: The user interacts with the tactile dashboard, specifically with the list of deployed pipelines, to delete one of them.

STEP 2: The dashboard sends an HTTP POST request to its backend to remove the involved agent/s. Then, backend recovers the data related to this pipeline.

STEP 3: The backend asks to remove the required agent/s to the Orchestrator.

STEP 4: Once these are removed, a confirmation message is sent to the backend.

STEP 5: If the pipeline has been deleted successfully, the dashboard shows to the user the updated list of pipelines. To that end, the first enabler story takes over. It should not include the recently-removed pipeline.

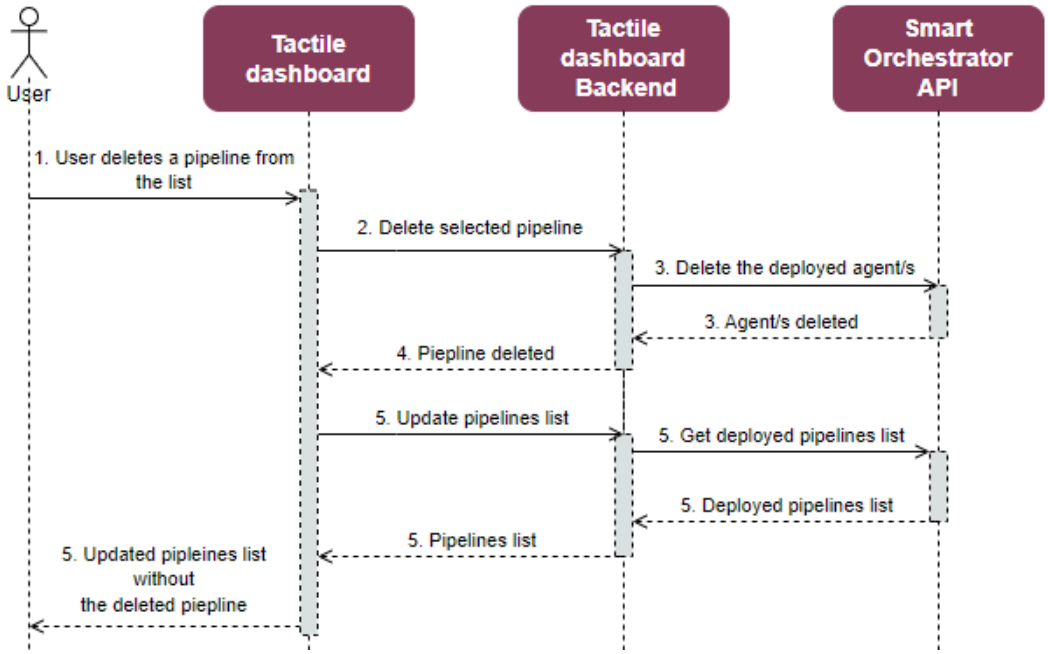

Figure 13. Composite services manager ES3 (delete pipeline)

The **fourth** and last **enabler story** corresponds to the **modification** of an **existing pipeline**. Right now, the flow is quite straightforward, consisting in a deletion of the previous version (ES#3) and followed by the creation of a new pipeline (ES#4). In future versions, the feasibility of updating on the fly those agents that do not need to be reallocated in other spots of the computing continuum will be considered, only deleting and creating those that actually need reallocation.

3.5.3. Clusters and topology manager

Table 32. General information of the Clusters and topology manager

Enabler	**Clusters and topology manager**
Id	T55E4
Owner and support	UPV
Description and main functionalities	Integrated in the tactile dashboard, the main functionality of this enabler will be to register new clusters of computing nodes (or a single computing node) in an ASISST-IoT deployment. Additionally: • It will perform all the necessary actions to provision the necessary actions related to K8s networking connectivity (e.g., cluster mesh) • It will manage also those clusters added via VPN or SD-WAN technologies • It will allow monitoring any registered node in the deployment, including its status (i.e., availability and used resources) and current instantiated enablers' components
Vertical, related capabilities and features	Manageability
Plane/s involved	Device and edge plane – addition of computing node to an ASSIST-IoT deployment Smart network and control plane – manageability actions related to K8s networking and topology management Application and services plane – as a service for system administrators, it will allow deploying enablers in specific computing nodes.
Requirements mapping	• RC7: Edge-oriented deployment
Use case mapping	This enabler will be present at all use cases, for administration purposes
Internal components	Dashboard and Backend

High-level structure

No changes to D5.3, although the naming of the enabler has been changed.

Implementation technologies

No changes to D5.3.

Communication interfaces

No changes to D5.3.

Enabler stories

The stories depicted in deliverable D5.3 are valid (show registered clusters – ES1, register a new K8s cluster – ES2, and delete cluster – ES3), but two of them have suffered some minor modifications. In the case of registering a cluster (ES2) and deleting a registered cluster (ES3), when interacting with the Smart Orchestration API, some operations related to the configuration of the selected CNI plugin of K8s (i.e., Cilium multi-cluster) need to be performed. In any case, the overall flow is barely affected (some additional API calls between the backend and the orchestrator).

Besides, the logic of the backend has been modified to support additional enabler stories. The **fourth story** consists in **depicting the topology** of the deployment, including the clusters and the nodes managed by the system as well as showing the enablers deployed in each computing node. Its flow and steps are the following ones:

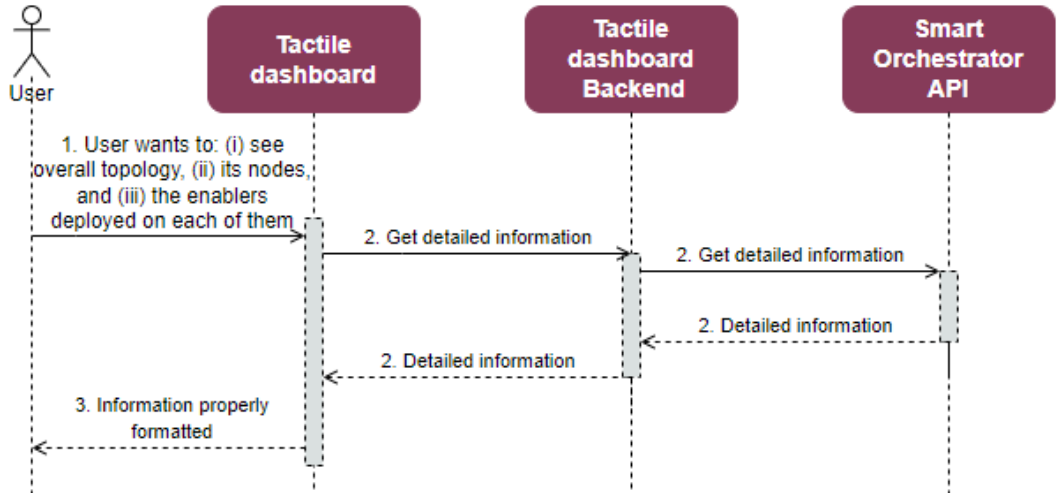

Figure 14. Clusters and topology manager ES4 (depict topology)

STEP 1: The user interacts with the tactile dashboard to get topology information (if clicking on a computing node, the interface will show additional information from it, as the enablers deployed on it).

STEP 2: The dashboard sends an HTTP POST request to its backend to gather the requested data from the Smart orchestrator.

STEP 3: Once the data is available, the dashboard formats it conveniently.

Finally, the **fifth enabler story** consists in the possibility of manually **deploying an enabler in a specific node of the topology**, with the interface of the previous story as stating point. Its flow is as follows:

STEP 1: The user interacts with the tactile dashboard, specifically with the topology view, to select a computing node to deploy an enabler on it.

STEP 2: The form to deploy enablers is shown to the user, but with some fields locked and filled with default information. Then, the enablers manager takes over.

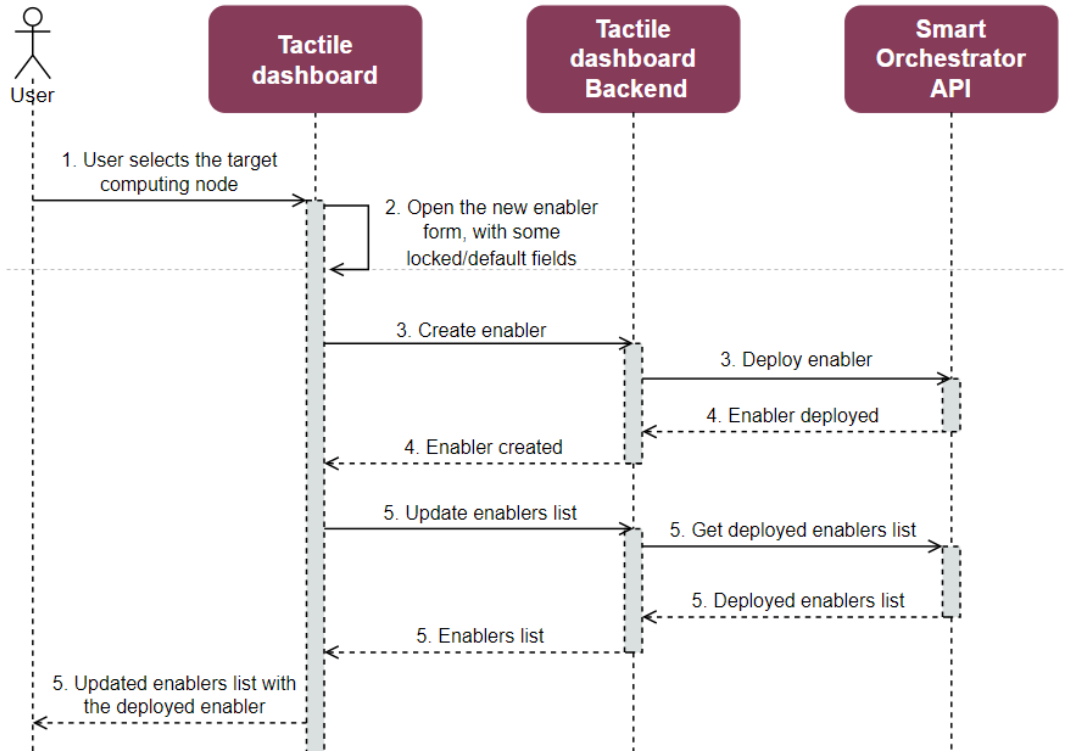

Figure 15. Clusters and topology manager ES5 (deploy enabler in target node)

4. Conclusions

This document is an update and extension of the specifications provided in the first iteration of this deliverable series constituting a final design of enablers proposed in WP5. It provides insight about the design and technical information for the transversal enablers, extending concepts introduced in deliverables D5.1, D5.2 and D5.3. The software outcomes of WP5 are at different levels of development, but mostly advanced enough to be integrated and validated in pilots. Specifically, some of them are containerised, others integrated with K8s (manifests ready) or prepared for packaging (in Helm charts), whereas the implementation of all of them is beyond minimum viable version. Although this document should mark the finalisation of the design of the enablers, as an agile approach is followed in the project, further improvements are expected from their deployment in integration and pilot environments.

The enablers developed so far allows for continuing efforts related to this and other work packages:

- To finish the development of the components of the enablers (WP5).
- To containerise, and/or generate the K8s manifests required to deploy them in those cases that have not virtualised the overall solution (WP5).
- To perform the testing and integration methodologies for each enabler (WP6).
- To package, publish and release the enablers as Helm charts (WP6).
- To start implementing them in pilots (WP7) for further validation and assessment (WP8), either fully or partially packaged.