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

D5.1 Software Structure and Preliminary Design

Deliverable No.	D5.1	**Due Date**	31-JUL-2021
Type	Report	**Dissemination Level**	Public
Version	1.0	**WP**	WP5
Description	Initial specification of 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
Ignacio Lacalle	P01 UPV	iglaub@upv.es
Alejandro Fornés	P01 UPV	alforlea@upv.es
Katarzyna Wasielewska-Michniewska	P03 IBSPAN	katarzyna.wasielewska@ibspan.waw.pl
Piotr Lewandowski	P03 IBSPAN	piotr.lewandowski@ibspan.waw.pl
Maria Ganzha	P03 IBSPAN	Maria.ganzha@ibspan.waw.pl
Marcin Paprzycki	P03 IBSPAN	marcin.paprzycki@ibspan.waw.pl
Eduardo Garro	P03 PRO	egarro@prodevelop.es
Georgios Stavropoulos	P04 CERTH	stavrop@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 S21SEC	olopez@s21sec.com
Jordi Blasi	P13 S21 SEC	jblasi@s21sec.com

History

Date	Version	Change
3-Jun-2021	0.1	Table of content
8-Jun-2021	0.2	First round of contributions (partially filled enablers templates)
12-Jun-2021	0.4	Second round of contributions – version ready for internal review
30-Jul-2021	1.0	Final version submitted to EC

Key Data

Keywords	Enablers, verticals, self-*, interoperability, manageability, scalability, federated learning, DLT
Lead Editor	Katarzyna Wasielewska-Michniewska (P03 IBSPAN)
Internal Reviewer(s)	Lambis Tassakos (P14 TwoTronic), Zbigniew Kopertowski (P15 OPL)

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 gathers the work and outcomes of the four tasks of the work package, which are devoted to the design and implementation of enablers required to implement the different verticals of the ASSIST-IoT architecture. Task 5.5 started in M9 so its results are not included in this deliverable.

The realisation of ASSIST-IoT architecture (outlined in deliverable D3.5 ASSIST-IoT Architecture Definition – Initial) requires the design and development of elements that support core functionalities. The ASSIST-IoT approach uses the abstraction term "enablers". Enabler will consist of a group of micro-services, each of them served over a container, acting towards a single goal (i.e., to provide a specific functionality) in the architecture. Enablers can be assigned to architectural layers (planes: Device and Edge, Smart Network and Control, Data Management, Application and Services) or verticals (Self-*, Interoperability, Security, Privacy and Trust, Scalability, Manageability) that intersect them. Plane (horizontal) enablers are addressed in D4.1 Initial Core Enablers Specification, whereas this deliverable addresses transversal (vertical) enablers. Note that, verticals represent functions targeting NGIoT properties that exist either independently on different planes or require the cooperation of elements from multiple planes.

By M9 of the project (July 2021), a total of 18 enablers have been identified and formalised:

- From Self-*: Self-healing device enabler, Resource provisioning enabler, Monitoring and notifying enabler, geo(Localisation) enabler, Automated configuration enabler.
- From Federated Machine Learning: FL Orchestrator, FL Training Collector, FL Repository, FL Local Operations, FL Privacy.
- From Cybersecurity: Cybersecurity monitoring enabler, Cybersecurity monitoring agent enabler, Identity manager enabler, Authorisation enabler.
- From DLT: Logging and auditing enabler, Data integrity verification enabler, Distributed broker enabler, DLT-based FL enabler.

These enablers have been identified responding to requirements presented in D3.2 Use Cases Manual & Requirements and Business Analysis Initial, as well from architecture specifications. This deliverable is the first out of two iterations (D5.4 by M24), so the content will be updated and expanded as the project evolves, and it will serve as the basis for the technical provision of the whole WP. Additionally, three iterations of Transversal Enabler Development deliverable (Preliminary D5.2 by M9, Intermediate D5.3 by M18, Final D5.5 by M30) will be prepared that will base on the content of this deliverable and its updates in time. Here, core information extracted from enabler templates are included. This information will be further detailed to provided full technical specifications of enablers that are to be provided.

Table of contents

List of tables

List of figures

List of acronyms

Acronym	Explanation
AI	Artificial Intelligence
API	Application Programming Interface
AR	Augmented Reality
CHE	Container Handling Equipment
CPU	Central Processing Unit
CSV	Comma Separated Value
DLT	Distributed Ledger Technology
DoS	Denial of Service
FAIR	Findable, Accessible, Interoperable, Reusable
FML	Federated Machine Learning
FL	Federated Learning
FLS	Federated Learning System
FLTC	Federated Learning Training Collector
GPS	Global Positioning System
HW	Hardware
I/O	Input/Output
JSON	JavaScript Object Notation
JVM	Java Virtual Machine
K8s	Kubernetes
LTS	Long-Term Storage
LTSE	Long-Term Storage Enabler
MANO	Management and Orchestration
NGIoT	Next Generation Internet of Things
NN	Neural Networks
noSQL	Not Only Structured Query Language
MITM	Man-In-The-Middle
ML	Machine Learning
MQTT	MQ Telemetry Transport
OEM	Original Equipment Manufacturer
PAP	Policy Administration Point
PCM	Powertrain Control Module
PDP	Policy Decision Point

PEP	Policy Enforcement Point
PIP	Policy Information Point
REST	Representational State Transfer
RSSI	Received Signal Strength Indicator
RTG	Rubber-Tyred Gantry (crane)
SDN	Software Defined Network
SoTA	State-of-the-Art
SQL	Structured Query Language
SMC	Secure Multi-Party Computation
SR	Semantic Repository
TBD	To Be Done/Defined
TRL	Technology Readiness Level
TTL/SSL	Time To Live/Secure Sockets Layer
UC	Use Case
WP	Work Package
XACML	eXtensible Access Control Markup Language
XML	Extensible Markup Language

1. About this document

The main goal of this deliverable is **to provide the specifications of the vertical enablers** that are going to be developed under the scope of WP5. 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.

It should be highlighted that this deliverable corresponds to the first out of two documents, and therefore its content will be expanded and adapted as the project evolves. This is motivated by different reasons, including the fact that both the requirements and the architecture produced by the work of WP3 are still evolving (and therefore new enablers or modifications in the current ones may be needed), and as a result the interactions between enablers from WP4 and WP5 may require adapting them (in the form of new interfaces, methods, components, etc.).

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	D5.1 takes input from: • T3.1 (state-of-the-art): Novel components and technologies research for further design choices • T3.2 & T3.3 (use cases and requirements): To be evaluated and fulfilled with the proposed enablers • T3.5 (architecture): Design principles and high-level functionalities to cover D5.1 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 D5.1 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 does not mark any specific milestone; still, it contributes to the realisation of *MS3 – Enablers defined*, that will be achieved in M12. Although far in time (M24), it is also central part of *MS6 – Software structure finished.*
Deliverables	This deliverable receives inputs from D3.1 (State-of-the-art and Market Analysis Report), D3.2 (Use Cases Manual & Requirements and Business Analysis Initial) and D3.5 (ASSIST-IoT Architecture Definition - Initial). Once enablers are being delivered, they will feed the deliverables of WP6 related to testing, integration, distribution and documentation, they will be the cornerstone of pilots' implementations of WP7, and they will be a key part in the technical evaluation to be performed under the scope of WP8.

1.2. The rationale behind the structure

This deliverable consists of 4 sections and two annexes. It starts with an introduction that outlines vertical enablers and their relation to task-specific division. Next section includes specification of enablers divided into tasks that they belong to. To facilitate the readers' comprehension, the corresponding templates with their initial

specification (composed of a set of tables and diagrams) have been moved to an annex at the end of the document, leaving in this section just the description and main functionalities provided by the identified enablers. Finally, the last section of this document concludes with a summary of the future work carried out in the work package that will be included in the second version of the deliverable. The first annex contains extended information on Federated Learning taxonomy that was used during the analysis of FL-related requirements and design of corresponding enablers. The second annex contains template tables and diagrams for identified enablers.

1.3. Outcomes of the deliverable

A set of enablers have been formalised in this deliverable (structured information is mostly provided in the annex for readers' convenience). Formalisation includes: functionality provided, relations to plane enablers and other vertical enablers, requirements and use cases mappings, and components that conform each enabler. Enablers are described with respect to two categorisations. Firstly, verticals from the architecture and enablers positioned inside are outlined (for more details on verticals description see D3.5). Secondly, enablers are summarised with respect to tasks in WP5 that correspond to "application areas" that do not map directly onto verticals, e.g., federated learning enablers belong to different verticals.

With respect to **Self-*** task, 5 enablers have been specified: (1) Self-healing device enabler (will be responsible for ensuring that devices will automatically heal-up when environment causes some disturbance), (2) Resource provisioning enabler (will horizontally scale (up or down) the resources devoted to a specific enabler (inside a node) in a dynamic fashion), (3) Monitoring and notifying enabler, (4) Geo(Localisation) enabler (will be directly used in pilots varying from coordinating heavy machinery in port to locating workers on construction site), and (5) Automated configuration enabler (will keep heterogeneous devices and services synchronised with their configurations).

Federated Machine Learning is an approach to train ML models that do not require sharing datasets with a central entity. Enablers identified here include: (1) FL Orchestrator (will be responsible for coordinating the overall Federated Learning process), (2) FL Training Collection (will be responsible for delivering back the updated model), (3) FL Repository (a set of different databases, including initial ML algorithms, already trained ML models, and auxiliary repositories for other additional functionalities), (4) FL Local Operations (will be embedded within each FL involved party/device of the FL systems to assure local data processing), and (5) FL Privacy (will guarantee that different parties are not able to derive insights about each other's training data).

Cybersecurity will provide protection against threats associated to ASSIST-IoT infrastructure. Cybersecurity related enablers are: (1) Cybersecurity monitoring enabler (will provide security awareness, visibility and infrastructure monitoring), (2) Cybersecurity monitoring agent enabler (will perform functions of an endpoint detection and response system), (3) Identity manager enabler (will be responsible for identifying and authenticating to have access to the resources by associating user rights with established identities), and (4) Authorisation enabler (will be responsible for the authorisation phase in the access control process).

In ASSIST-IoT, privacy and trust per design will be addressed by the introduction of **DLT-related** enablers. DLT-based enablers include: (1) Logging and auditing enabler (will allow the documentation of data usage), (2) Data integrity verification enabler (will provide DLT-based data integrity verification mechanisms that allow data consumers to verify the integrity of the exchanged data), (3) Distributed broker enabler (will support immutability and non-repudiation of selected aspects of connections between enablers), and (4) DLT-based FL enabler (will facilitates exchanges of parameters of on-device local data models in an immutable and decentralised way).

Work on **manageability and control** enablers has started in M9 so they will be included in the second iteration of this deliverable (D5.4 Software Structure and Final Design).

1.4. Lessons learnt

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

- MAPE-K[1] (Monitor, Analyse, Plan, Execute, Knowledge) proposed by IBM as reference model for autonomic control loops is a powerful tool to express most of the functionalities related to self-*. It inspired the design of enablers assigned to Self-* vertical.
- Federated Machine Learning is still in a research phase in AI/ML domain (low TRL) with no industrial developments. As a result, ASSIST-IoT aims at developing a stand-alone FL system, as several use cases of the project are candidates for its use.
- ASSIST-IoT Federated Learning related use cases require centralised structure and horizontal data partition (with respect to taxonomy discussed in the following sections).
- Principles of Federated Learning and DLT do not got together too well even though these areas initially seemed to be aligned. Specifically, all parties that belong to a DLT infrastructure can access data that is stored in DLT. On the other hand, FL imposes some restrictions on what should be accessed, e.g. from information about consecutive model updates one may probably detect the nature of the data. As a result, one should be very careful to decide what FL related information to store in DLT to protect privacy of information. Eventually, some additional data protection mechanisms may be needed.
- Most of the enablers designed are generic and loosely-coupled to the use cases. As a result, they can be easily leveraged in different environments but greater integration effort is needed to provide this flexibility.
- Identification of vertical enablers was a daunting task because of their transversal nature.
- Identifying and specifying all the interactions needed between enablers is a demanding task. It requires analysis in the context of specific endpoints, possible code modifications, addition or modification of a component, etc. It may happen that some interactions may not be discovered until actual developments/implementation starts. However, this analysis has to be pushed forward to avoid work duplicities and prepare a clean and consistent architecture.

1.5. Deviation and corrective actions

The Consortium has made a great effort to envision and formalise the enablers that will be needed for the realisation of the ASSIST-IoT architecture to address specific use cases. However, there are some deviations with respect to the initial plan that have to be tackled during the next phase:

- The following enablers, even though initially proposed, have been removed from Security, Privacy and Trust vertical: Collector enabler, Log management enabler, Storage and search analytics enabler, Visualisation enabler. They were proposed for collecting data and performing analysis in the context of cybersecurity, however their functionalities were not transversal, and could be covered by enablers provided on horizontal planes. Hence, these enablers have been removed from this vertical, and proper configurations will be applied when deploying them to offer the functions expected initially. Note, that this deviation has already been corrected.
- In both WP4 and WP5 enablers data is collected from different sources. A set of unified mechanisms should be proposed to avoid duplication of functionality and development effort.

Finally, all enablers were expected to reach the same level of maturity (in its definition and design) by M9. In order to have the same level of readiness before starting technical deployment, it was planned to have a clear description of each enabler functionalities, structure with listed components, including a selection of candidate technologies to be verified during their implementation, etc. However, differences in advancement of work for different enablers can be noticed. Importantly, this should not be considered a deviation on the execution but the usual progress of the WP. There is the commitment to reach a steady state on the descriptions and the development status of the enablers by M12, where the first Open Call of ASSIST-IoT will be launched.

[1] An Architectural Blueprint for Autonomic Computing. IBM, eds. , IBM (2005). Online: https://www-03.ibm.com/autonomic/pdfs/AC%20Blueprint%20White%20Paper%20V7.pdf

2. Introduction

As it was stated in D3.5, the ASSIST-IoT architecture is structured following a multidimensional approach composed of horizontal *Planes* and *Verticals.* The planes represent a classification of logical functions that fall under the scope of a particular domain, whereas verticals target NGIoT properties that exist on different planes, either independently or requiring cooperation of elements from different planes.

In D3.2 a study of requirements coming from stakeholders representing three industries (pilot-specific) was presented. Architectural approach proposed in ASSIST-IoT should fulfil not only these requirements but also be adoptable to other industry sectors providing solutions that will fill the gap between high-level design and actual implementation of NGIoT solutions.

The main building block in ASSIST-IoT architecture is an enabler - an abstraction term that represents a collection of components, running on nodes, that work together for delivering a particular functionality to the system. They can be fully independent or may require of the cooperation of other enablers to deliver its intended functionality. As explained in D3.5, software components will be containerised, following an encapsulation paradigm in which enablers can communicate among them only through dedicated exposed interfaces, avoiding the possibility of having direct communication between components of different enablers.

The enablers have been identified and designed based on requirements (D3.2) and SoTA analysis (D3.1). The document focuses on providing initial specification of transversal (vertical) enablers. Each enabler from this group is assigned to one or more vertical as outlined in the following subsections and described in detail in D3.5. This section outlines how proposed enablers are positioned with respect to verticals. Note that, features offered by verticals are also provided by designed choices guiding the ASSIST-IoT architecture (for details see D3.5).

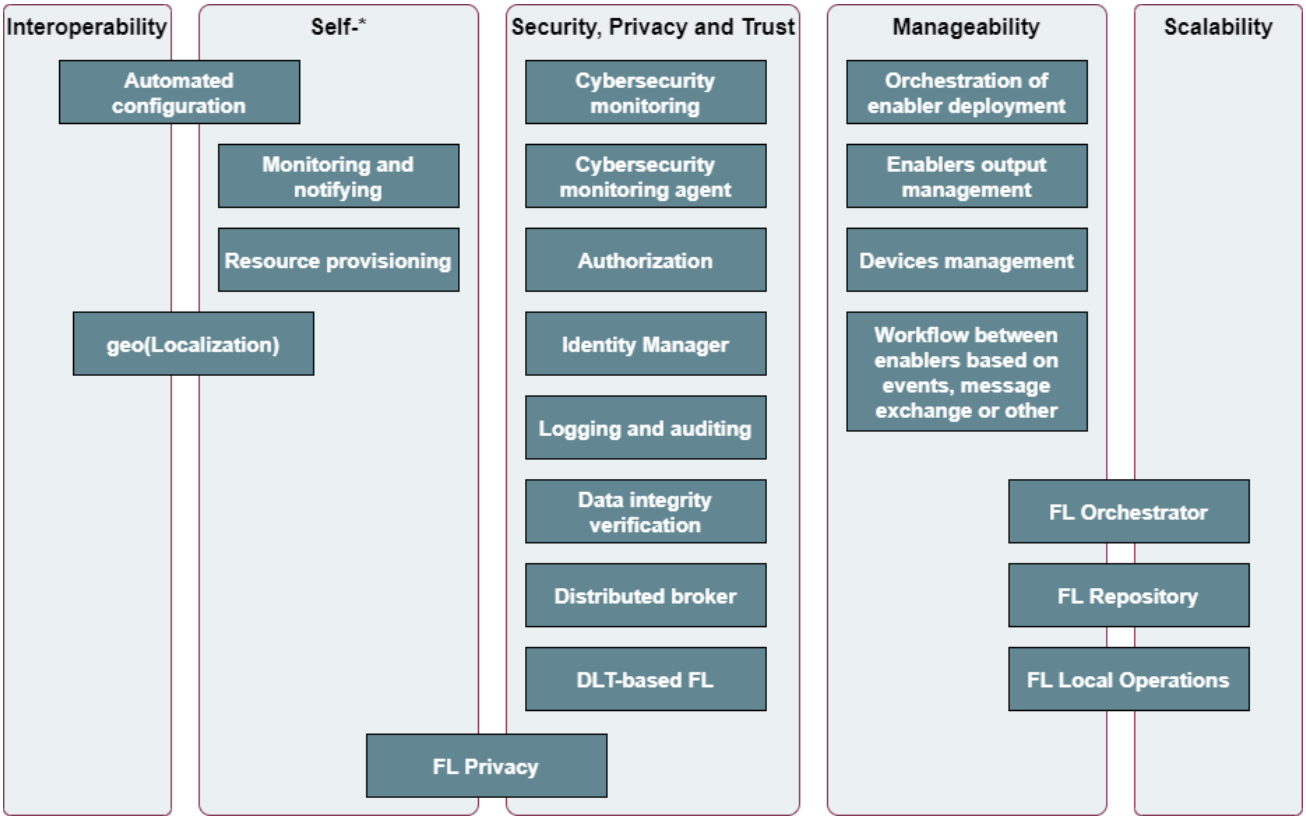

Figure 1. Enablers distribution among verticals

What is specific to WP5 is that enablers besides being distributed between verticals, are also designed and implemented within tasks that do not correspond directly to the verticals. Tasks indicate problem/application areas in which we may propose enablers that additionally can be classified within verticals. For example, Federated Machine Learning has a dedicated task that should propose solutions in this field that are not vertical-

specific but rather application-specific. On the other hand, scalability has no dedicated task but is addressed by various enablers and their components as a feature that the proposed architecture should provide by design.

Following subsections outline verticals and enablers (from different tasks) that were assigned to them. Next section contains short descriptions of enablers following task division.

2.1. Self-*

Self-* enablers will realise autonomous nature of an ASSIST-IoT deployment. Enablers will realise one of the Self-* capabilities (self-diagnosis, self-healing, self-awareness, self-organisation, self-configuration) fully or be a fundamental building block allowing developers to implement features that rely on Self-*.

Enablers included in this vertical:

- Self-healing device (see 3.1.1)
- Resource provisioning (see 3.1.2)
- Monitoring and notifying (see 3.1.3)
- Geo (Localisation) (see 3.1.4)
- Automated configuration (see 3.1.5)

Additionally, one enabler related to Federated Machine Learning has also been assigned to this vertical:

- FL Privacy (see 3.2.6)

2.2. Interoperability

Interoperability can be considered on several levels, among them: technical (the ability of two or more information and communication technology applications, to accept data from each other and perform a given task), syntactic (allows two or more systems to communicate and exchange data in case that the interface and programming languages are different) and semantic (the ability of different applications/artefacts/systems/… to understand exchanged data in a similar way, implying a precise and unambiguous meaning of the exchanged information).

Interoperability is provided by ASSIST-IoT architecture by design where approaches selected support this feature even without providing dedicated enablers. In ASSIST-IoT, as it was stated in D3.5, interoperability will be addressed in terms of scalability, security, privacy and heterogeneity of data sources.

Self-* enabler included also in this vertical (supporting interoperability):

- Automated configuration (see 3.1.5)

2.3. Scalability

The scalability principle of ASSIST-IoT, as it was stated in D3.5, aims at enabling elastic scaling deployments ranging from modest barely local operations up to large heterogeneous deployments based on demand features and functionalities. This scalability is essential in order to adapt to the different projects' workloads, performance, costs, and other business needs. Hence, the ASSIST-IoT will become a 3D-scalability approach: software, hardware and communication capabilities. For more details, please refer to D3.5.

Federated Machine Learning related enablers that have been assigned to this vertical:

- FL Orchestrator (see 3.2.3)
- FL Repository (see 3.2.4)
- FL Local Operations (see 3.2.5)

2.4. Security, Privacy and Trust

Cybersecurity objective is to provide protection against threats associated to ASSIST-IoT infrastructure that may conclude on an alteration of its characteristics to carry out activities not intended by owners, designers, or users.

Broadly accepted Cybersecurity objectives associated to Information Assets are: Confidentiality, Integrity, Availability, Authentication, Authorisation and Auditability. To provide effective threat mitigation initiatives to accomplish with the aforementioned cybersecurity objectives, the most extended method includes frameworks such as STRIDE or MITRE ATT&CK among others.

Enablers included in this vertical will provide response to cybersecurity objectives to mitigate cyber threats:

- Cybersecurity monitoring enabler (see 3.3.1)
- Cybersecurity monitoring agent enabler (see 3.3.2)
- Identity manager enabler (see 3.3.3)
- Authorisation enabler (see 3.3.4)

In ASSIST-IoT, privacy and trust per design will be addressed by the introduction of DLT-related enablers. DLT is known for the opportunity to decentralise procedures, resilience to changes, anonymity, and immutability to data. Enablers included in this vertical offering DLT-based functionalities include:

- Logging and auditing (see 3.4.1)
- Data integrity verification (see 3.4.2)
- Distributed broker (see 3.4.3)
- DLT-based FL (see 3.4.4)

Additional enabler in this vertical constitutes part of the Federated Machine Learning:

- FL Privacy (see 3.2.6)

2.5. Manageability

Manageability is one of the traits that ASSIST-IoT, as a next-generation IoT deployment, will need to have. This property is considered one of the building pillars (vertical) of the architecture, applying transversally to all four horizontal planes. The actual meaning of this vertical is the capacity of the system to be managed (i.e., changed, configured, interacted with) from a central point for the different type of users.

In practical terms, this vertical is aimed at allowing the system owner to handle the ASSIST-IoT deployment. Not only it means to be able to see what is implemented (enablers, devices, etc.) but to install new enablers, manipulate (e.g., create) new services and enablers and create overall workflows. The final objective is to achieve an all-encompassing management of the system by the user to provide a grasp of control over the deployment.

Regarding specific enablers, the manageability vertical will be composed of:

- Orchestration of enablers deployment and Workflow between enablers based on events, messaging exchange: This enabler will be the most important provision of this vertical. It will consist of a workflow-like (e.g., Node-Red) tool that will allow the user to deploy enablers at different locations of the network (provided that it is physically possible). This enabler will be in charge of receiving "instructions" from the user (administrator) on when, where and under which characteristics must an enabler/set of enablers be deployed and will trigger enough backend queries to enablers' interfaces (e.g., to the Smart Orchestrator) to make it happen.
- Enablers output management: This enabler will consist of a list of the enablers that are installed in the deployment, the results that they generate and some actions associated to those outputs. This enabler will as well include some backend interaction.
- Workflow between enablers based on events, messaging exchange, or other.

- Devices management: Enabler for monitoring devices and nodes in a deployment, allowing to monitor status and current work (in terms of enabler components).

Manageability tools will be incorporated to the solution once the rest of enablers of planes and verticals are advanced. The task that manages the creation of manageability enablers has started some months later than the others due to its cross-dependency nature, and therefore their interaction with the rest of the enablers of ASSIST-IoT in the form that will be related in detail later in the project.

Specifically, more information on manageability enablers will be provided in deliverable D5.2.

Finally, it is worth mentioning that the FL Orchestrator (see 3.2.2), FL Repository (see 3.2.4) and FL Local Operations (see 3.2.5) enablers would fit under this vertical as their mission is to provide the framework to facilitate federated learning. Although it is detailed in other parts of the document, FL is not considered as a "standalone" vertical in ASSIST-IoT, therefore it is reported and framed under the "manageability" distinction. However, it must be noticed that it does not fall under the work of task T5.5 (manageability enablers) neither it will be delivered at the same time or pace.

3. Initial vertical enablers specification

The specification of all the identified enablers is formalised in the ASSIST-IoT enabler template. The original template as provided in D3.5 is composed of four sections: (1) a table with general information about enabler, including its description; (2) a basic diagram depicting the high-level enabler's structure and communication among its components; (3) a table summarising all the functionalities provided by its endpoints (APIs, or other type of interfaces); and (4) a set of tables, one for each component, with information related to their rationale and implementation aspects.

Providing all the information required in the template is a work in progress and some design choices have not been taken yet. As a result, template used in this deliverable is a simplified version, e.g., the list of endpoints (third section) and some implementation aspects of components in the fourth section of the templates (e.g., hardware and software requirements) are not included. Finally, for facilitating the reading of the section, templates will be provided in Annex B, leaving just the description of the provided functionalities in the core document. All provided templates will include the next elements:

Table 1. General information of the enabler

Enabler	***Name of the enabler (follow glossary guidelines to name it)***
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 functionalities)*
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*
Relation with other enablers	*List of enablers (core or vertical) that interact with this one*
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*
Required components	*List of the IDs of components required by this enabler*

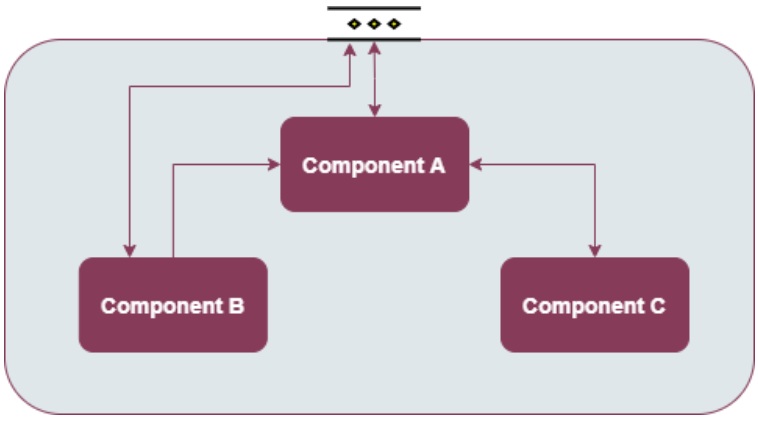

Figure 2. High-level diagram of an enabler

As aforementioned, any information related to endpoint will be provided in this iteration of the deliverable, since just a few of enablers have part of this information at this moment.

Table 2. Specific information of an enabler component

Enabler component	***Name of the enabler component***
id	*Short unique identifier*
Description and main functionality	*Functional description of the component (description paragraph and bullet points for describing its functionalities that are required by enabler/s)*
Target node/s	*Physical device in which it can be installed (edge node, smart IoT device, cloud…). If not decided just "to be decided"*
Candidate technologies	*Candidate technologies to implement it. In some cases, it might be more "final choices" than in others, but at least we must present implementation options (can be something like "custom component in Python using Flask")*

In the following sections each vertical enabler identified has been described in terms of the main functionalities it provides. For more detailed descriptions including structure see Annex B - Enabler templates.

It is necessary to callout that enabler definition might change during iterative development process.

3.1. Self-* enablers

3.1.1. Self-healing device enabler

The Self-healing device enabler is responsible for ensuring that devices will automatically heal-up when enabler will detect that device is unhealthy. The monitoring component is responsible for assessing the device's state of health. It collects and analyses data from multiple sources of information, such as memory usage, memory access, network connection metrics (RSSI levels), or CPU usage, providing a health score. The health score metrics are fed to a predefined set of rules (or to an anomaly-detection model) that determines whether the device is in a healthy state or not. The output of this component is used to determine if the remediation has been successful. When the device presents with symptoms of malfunctioning or intrusion, this enabler will determine from a user-defined set of remediation processes, which should be used for a proper treatment. If after the remediation, the device is not back to its normal state, the self-remediator is triggered to select another remediation process from the list.

3.1.2. Resource provisioning enabler

This enabler will be able to horizontally scale (up or down) the resources devoted to a specific enabler (inside a node) in a dynamic fashion. ASSIST-IoT aims at working on changing environments where resource availability must be used for the sake of the system as a whole. The objective of the enabler is to allow such scaling (per enabler and per node = K8s service in a topologically co-located cluster) so that software instances will be assigned more or less resources according to the pressure put on the system. The "pressure put on the system" will be represented through a series of indicators (metrics) that might be a combination of internal performance (of the enabler) and the status of other enablers within the ASSIST-IoT deployment. Finally, this enabler will be able to dynamically adapt (SELF-*) the provided resources to the enablers based on past data, historic trends, ML models.

3.1.3. Monitoring and notifying enabler

This enabler could be viewed as a general purpose by representing it as a combination of high-level monitoring module (which would allow to monitor devices, logs, etc.) and notifying module that could send custom messages to predefined system components. By being endowed with Self-* capabilities this enabler will be able to provide metrics predictions and smart notifications where applicable.

3.1.4. Geo(Localisation) enabler

To solve challenges of pilots we need to localise physical objects (containers in ports, workers on construction sites), some devices should be aware of their position in relation to each other (aligning cranes and tractors). This enabler will provide that functionality and will be directly used in pilot usages varying from coordinating

heavy machinery in port to locating workers on construction site. This enabler will be a basic building block providing Self-awareness (about location of devices) to the ASSIST-IoT deployment.

3.1.5. Automated configuration enabler

Automated configuration enabler keeps heterogeneous devices and services synchronised with their configurations. User can update configuration and define fall back configurations in case of errors. Self-* component will be responsible for reacting to changing environment and updating configuration as necessary. This enabler will realise self-configuration and will be a part of every non-trivial deployment.

3.2. Federated machine learning enablers

3.2.1. Introduction

The success of using Machine Learning (ML) to solve NGIoT problem will depend on the quality and quantity of available training data. Therefore, ML approaches typically rely on the central management of training data. However, centralising data for training is often not feasible or practical, for reasons of data privacy, regulatory compliance, and the huge amount of data involved.

Federated learning (FL) is an approach to train ML models that do not require sharing datasets with a central entity. In FL, a model is trained collaboratively among multiple parties, which keep with themselves their training dataset, but they collaboratively participate in a shared FL process. The notion of parties might refer to entities as different as data centres of an enterprise in different countries, mobile phones, cars, or different companies and organisations.

ASSIST-IoT aims at developing a stand-alone FL system, as several use cases of the project are candidates for its use. Due to still the infancy of FL approach into the AI/ML research ecosystem, it should be noticed that the FL system to be developed in the project is considered more like a research-oriented platform than the rest of the enablers of ASSIST-IoT platform. Hence, except only one FL enabler, it is expected that ASSIST-IoT platform will not rely on the contemplated enablers of FL for working appropriately in the different industrial environments of the project. In order to successfully design the appropriate ASSIST-IoT FL System, we have followed the FL taxonomy (proposed by Li et. al.[2]). A detailed explanation of them is included in Annex A. Following that taxonomy, the project partners ended with the below block diagram and flow chart for the ASSIST-IoT Federated Learning process, including its associated FL enablers.

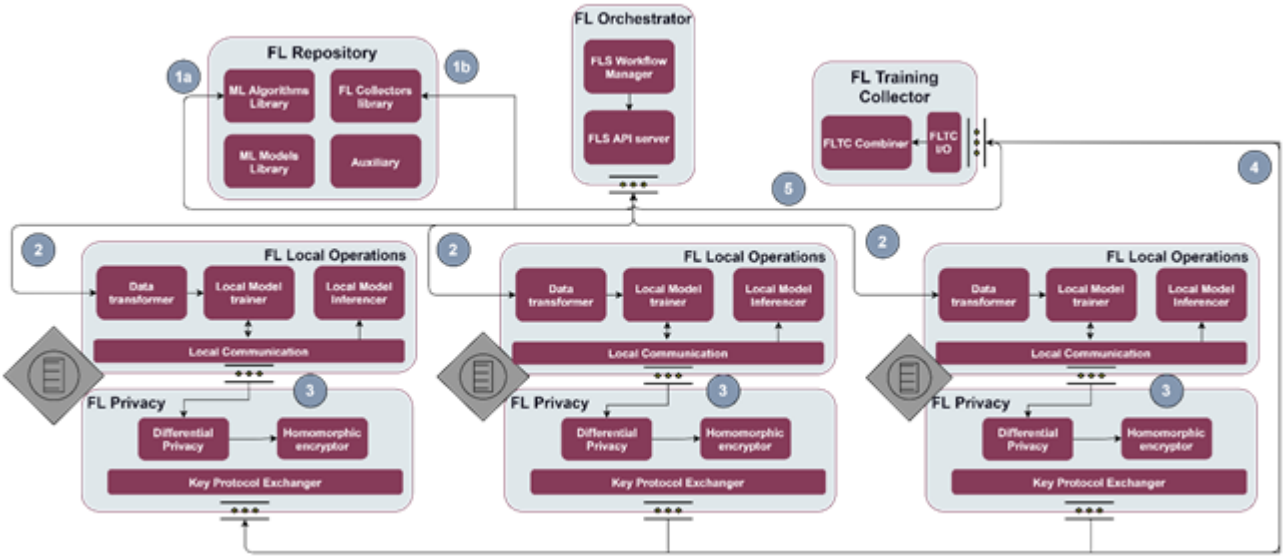

Figure 3. ASSIST-IoT FL system formed by five enablers

[2] Q. Li et. al., *"A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection"* arXiv:1907.09693v5

A brief description of the five different FL enablers is provided next, although a more detailed definition is as usual, included in the Annex B.

3.2.2. FL Orchestrator

This enabler will be the responsible for coordinating the overall Federated Learning process. Hence, it will be in charge of specifying details of FL workflow(s)/pipeline(s), such as job scheduling, FL life-cycle management, or defining stopping criteria for the training. Whereas all these functionalities will be included within the FLS Workflow manager component, the associated directives for the FL orchestration will be sent and received via a FLS REST API server.

3.2.3. FL Training Collector

As explained before, as well as in the Annex A, the FL training process involves that several independent parties 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 will be tackled by the FL Training Collector, which will also be in charge of delivering back the updated model. The enabler will, therefore, form by two main components: the FLTC Combiner, and the FLTC I/O. On the one hand, the FLTC combiner will be mainly in charge of carrying out the aggregation/combination of local updates. It will include both homogeneous and heterogeneous FedAvg solutions for e.g., Logistic Regression Models, Decision-tree Models, or even Neural Network Models. FLTC I/O will provide a REST API to allow the input and output communication to and from the FL training collector enabler, being responsible of receiving and FL local updates that are sent to the FLC Combiner component, and communicating updates of the new FL model obtained in the FLC Component to involved training parties or to the FL repository.

3.2.4. FL Repository

The FL repository will be a set of different databases, including initial ML algorithms, already trained ML models suitable for specific data sets and formats, averaging approaches, and auxiliary repositories for other additional functionalities that may be needed, and are not specifically identified yet. ML libraries will provide a collection of ML models that can be either original ML models that can be adapted for FL (and, possibly, other modules necessary to set up FML such as optimisers, pre-processors, etc.), or already FL-off-the-shelf models. Among the FL Averaging modules, at this moment, FedAvg, and SecFed are expected to be supported. All these libraries are planned to provide multiple implementations of the same functionality, optimised for different computer hardware, or implemented in different programming framework languages. Each library/database will be demarcated with metadata, specifying all necessary aspects of its functionality.

3.2.5. FL Local Operations

One of key goals of FL is to assure protection of privacy of data, owned by individual stakeholders. Therefore, data is expected to be trained and used only locally. Therefore, an embedded enabler within each FL involved party/device of the FL systems is needed. It has been defined as FL Local Operations enabler. The different components and their rationale is explained next.

In IoT ecosystems each partner is likely to store data in its own format. Additionally, to apply a specific FL approach (e.g., Neural Networks) data must have format that will allow use of specific ML models and will allow joining locally trained models into the shared common model. Hence, one of the biggest problems with application of FL into IoT ecosystems comes from statistical properties of data stored in different nodes, as the use of FL should not depend on individual data formats. This can be achieved by transforming local data into format used by the ML approach within the FL Local Operations enabler, which by the Data transformer component, will perform transformations from local data formats to data formats required by the FL system. Once the data has been properly formatted, it will be use as data ground for FL training locally via the embedded ML trainer. The local results (i.e., the ML algorithm updates recommendations) will be sent to the FL training collector in order to carry out the appropriate aggregation methodology over the common shared model. To do so, both inputs and outputs updates will be sent through the Local Communications component, which it is foreseen to be developed as a REST API server. Finally, when the FL training process has concluded, the final

shared ML model will be also used to deliver specific functionality by means of the Local Inference Engine of the FL Local Operations enabler.

3.2.6. FL Privacy

The privacy of each party data is paramount in the FL process. Although the data will not be directly exposed and will not go out of the involved parties' premises, threatening models should be foresee over the communication of model updates via encryption mechanisms. These mechanisms will be carried out within ASSIST-IoT by means of the FL Privacy enabler, who will guarantee that different parties are not able to derive insights about each other's training data. To do so, the enabler will be formed by three components. On the one hand, a homomorphic encryptor will not permit outsiders to see the output model of each device/party (MITM attacks). On the other hand, methods of creating differentially private noise will guarantee that Malicious Aggregator cannot be allowed to infer which records are actual models and which not. Finally, the key protocol exchanger will ensure that the communication during the training process is not compromised via TTL/SSL certificates.

3.3. Cybersecurity enablers

3.3.1. Cybersecurity monitoring enabler

Cybersecurity monitoring enabler will consolidate the necessary information for cyber threat detection over the deployed architecture and pilots. Cybersecurity monitoring enabler provides cyber security awareness and visibility on cybersecurity objectives and will provide infrastructure cybersecurity monitoring.
The cybersecurity monitoring server will be responsible of collecting, processing, and analysing the incoming information from the infrastructure under study and will consolidate an output that will provide cybersecurity monitoring information related to different events. Figure 4 illustrates cybersecurity monitoring components and how cybersecurity monitoring output produce the alerts based on processing rules of security events.

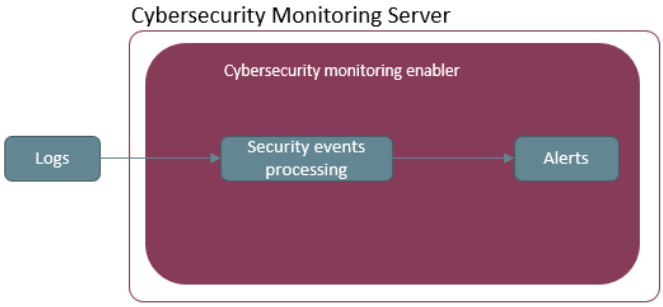

Figure 4. Security monitoring components

Monitoring server can rely on agents running on monitored endpoints that will forward collected data to the monitoring server, as detailed in Figure 5. Monitoring server can also collect data coming from agentless devices to consolidate log data via syslog among others possible methods.

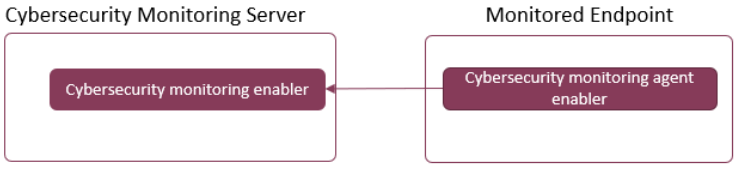

Figure 5. Cybersecurity monitoring server agent

Cybersecurity monitoring server will present information on T43E8 Long Term Storage Enabler.

3.3.2. Cybersecurity monitoring agent enabler

Cybersecurity monitoring agent enabler will report to security monitoring server. Security monitoring agent enables execution of processes on the system target under study to provide relevant information if a cybersecurity breach is produced. Security monitoring agent enabler will perform functions of an endpoint detection and response system, monitoring and collecting activity from end points that could indicate a cybersecurity threat.

3.3.3. Identity manager enabler

Identity manager enabler will be responsible for managing identities on the access control process. Authentication is a process by which the credentials provided by an identified entity (computer, application, or person) are compared with those memorised/created in the system to ensure that said entity is effectively who or what it claims to be.

The main goal of the 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).

Identity manager enabler will perform authentication phase of access control process. Identity manager will process and validate the identity for later control the access to the resources by the authorisation enabler. Identity Manager enabler will rely on OAuth2 protocol. OAuth2 model allows the delegation of the authentication process to a remote server, granting a communication that keeps entity (user or system) authentication data secure. Identity manager enabler using OAuth2 will communicate with Authorisation enabler implementing XACML policy.

3.3.4. Authorisation enabler

Authorisation enabler will be 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.

Authorisation enabler will be based on XACML standard security policies, results on obligations actions to be deployed after the evaluation process. It will be composed with different components as described below:

- PAP. Policy Administration Point. Point which manages access authorisation policies.
- PDP. Policy Decision Point. Point which evaluates access requests against authorisation policies before issuing access decisions.
- PEP. Policy Enforcement Point. The PEP component responds to where enforcement is going to take place.
- PIP. Policy Information Point. The role played by PIP is to provide attribute values upon request from the PDP context.

In ASSIST-IoT a federated authorisation enabler will distribute a security policy from cloud to the edge to be locally evaluated by the PDP and enforced locally by the PEP. Federated PAP policy will be controlled by an admin team and replicated locally in a local Access Control Policy.

Authorisation enabler on the edge will rely on T54E2 Data integrity verification enabler. The DLT Data integrity verification enabler will be used to provide resistance to unauthorised changes to policies.

Figure 6 describes the decoupled cloud-edge approach for the Authorisation enablers components

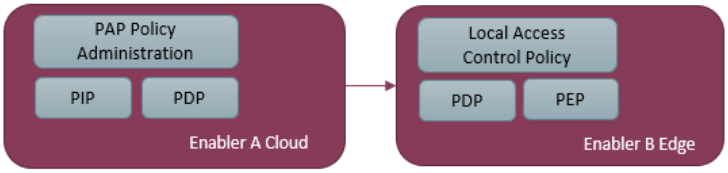

Figure 6. Authorisation enabler – cloud and edge

3.4. DLT-based enablers

3.4.1. Logging and auditing enabler

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 to help providing digital evidence and resolve conflicts between stakeholders, when applicable.

Billing mechanisms based on the logged actions may be developed if this is deemed necessary depending on the requirements of each use case, i.e. if the consumption of specific data can be regarded as billable.

3.4.2. Data integrity verification enabler

This enabler will provide DLT-based data integrity verification mechanisms that allow data consumers to verify the integrity of the exchanged data. The data integrity is possible due to the characteristics of the DLT itself. Ledgers in DLT allow only the data addition as they are append-only. Moreover, in permissioned Blockchain networks, only authorised users can issue transactions and only authorised nodes can validate the transactions that are added to blocks. Finally, the consensus algorithms used by Blockchain networks are implementing guarantees that attackers have to compromise the majority of the network to be in place to make malicious changes on the chain.

3.4.3. Distributed broker enabler

This enabler will provide secured data sharing mechanisms as regards the data exchange between different heterogeneous IoT devices belonging to various edge domains and/or between different enablers of the architecture.

The mechanism will focus on providing a distributed broker service that will serve as a registry of all the domains and/or ASSIST-IoT enablers that act as data producers and/or data consumers. Indexing and querying services will facilitate the efficient retrievability of the stored (meta)data.

This enabler may also act as a facilitator to the enablers that will provide semantic interoperability by providing searchable metadata of the interoperable domains complying with the FAIR principles.

3.4.4. DLT-based FL enabler

This enabler will foster the use of DLT-related components to exchange the local, on-device models (or model gradients) in a decentralised way allowing for decentralised verification of local model updates, avoiding single point of failures and acting as a component to manage AI contextual information in an immutable form, also avoiding alteration to the model data.

4. Future Work

This document is a first deliverable for WP5 and presents work done so far. Since this deliverable will have a second iteration, the specifications included here may be extended or updated as the project evolves. Enablers identified were classified according to verticals and tasks described using a template. It includes a table with main functionality and general information, a high-level diagram of its components, and a dedicated table for each of its components, in which rationale and candidate technologies for their implementation are highlighted. Next version of the deliverable will include enabler template extended with additional information such as endpoints specification.

The finalisation of this deliverable will initiate development of related software, therefore next tasks will be to:

- Fill in missing information such as endpoint/API specification for each enabler and component,
- Establish details of interactions between WP5 and WP4/5 enablers (so far relations have been identified but nature and protocols for the interactions need to be formalised),
- Make any adjustments necessary (e.g., slight modification in provided functionalities, change in enabler structure, change in selected technologies),
- Prepare the backlog of tasks, distribution of work and kick-off of implementation activities,
- Deliver first results - ready and available by M18 (however, the degree of development will not be homogeneous for all the enablers).

Annex A - Federated Learning taxonomy

A.1 - Introduction

In order to successfully design the appropriate ASSIST-IoT FL System, we have followed the FL taxonomy identified in Li et. al.[3] that relies on six main aspects: (i) communication architecture, (ii) scale of federation, (iii) data partitioning, (iv) ML model, (v) privacy mechanism, and (vi) motivation of federation. A detailed explanation of them is presented next.

A.1.1 - Communication architecture

There are two major ways of communications in FL Systems: ***centralised*** and ***decentralised design***. Both options are illustrated in Figure 7, and described next.

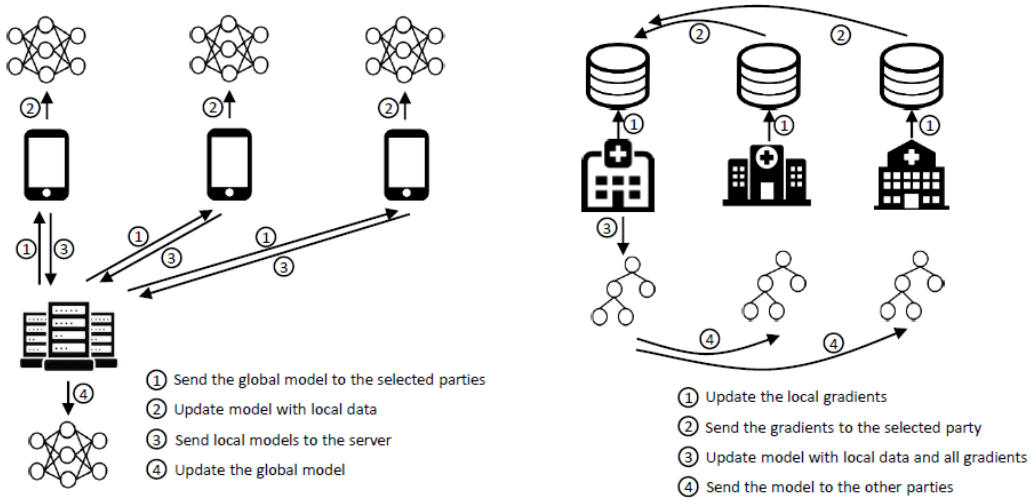

Figure 7. Communication Architecture options for FL (left: centralised, right: decentralised)

- In the ***centralised design***, the manager aggregates the local ML models obtained from the involved parties and send back the updated model for new training iterations. Therefore, the parameter updates on the global model are always done in this manager, sometimes also called Aggregator or Collector. The most commonly known example of Centralised FL architecture is the Google Keyboard – Gboard. In this system, the server collects local model updates from users' devices and train a global model. The global model is next sent back to the users for either new training rounds or for inferencing.

- In a ***decentralised design***, there is not a single point of truth, i.e., there is not a manager. In this design, the communications are performed among the parties, who are able to update the global parameters iteratively. While the centralised design has been regularly used in existing research studies, the decentralised design may be preferred on those scenarios in which concentrating the averaging process on a single server may bring potential risks or bias. However, the design of a decentralised FL system is far more challenging as all involved parties in the training have to be computational capable to support the Federated averaging during the learning process. Additionally, another major challenge is that it is hard to design a protocol that can treat every party almost fairly with a reasonable communication overhead. An example of a FL decentralised architecture can be a decentralised cancer diagnosis system

[3] Q. Li et. al., *"A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection"* arXiv:1907.09693v5

among hospitals, who do not want to rely on a single manager for the aggregation, and they agree on carrying out with the averaging in a distributed fashion.

A.1.2 - Scale of Federation

The FL Systems can be categorised into two typical types by the scale of federation: ***cross-silo*** and ***cross-device***. The differences between them lie on the number of parties and the amount of data stored in each party.

- In ***cross-silo***, the parties can be either independent organisations or independent data centres of a single organisation. The scale is usually assign for a relatively small number of parties, and each of them has a relatively large amount of data to be used for training as well as high computational power. For example, Amazon wants to recommend items for users by training the shopping data collected from hundreds of data centres around the world. Each data centre possesses a huge amount of data as well as sufficient computational resources.

- In ***cross-device***, on the contrary, the number of parties is relatively large and each party has a relatively small amount of data as well as computational power. The parties are usually IoT devices. The previous example of Gboard is a cross-device FL System. Under this scenario, the FL system should be therefore powerful enough to manage a large number of parties involved in the training, and deal with possible issues such as the unstable connection between the devices and the server. Another big challenge comes from the energy consumption concern, so that IoT devices cannot be asked to conduct complex training tasks, and restrict the ML model to be trained and inferenced.

A.1.3 - Data partitioning

FL Systems are also categorised in horizontal, or vertical data partitioning based on how data are distributed over the sample and feature spaces.

- In ***horizontal data partitioning***, the datasets of different parties have the ***same feature space but little intersection on the sample space***. This is the natural data partitioning on cross-device setting (which will be the main structure of ASSIST-IoT FL use cases), where different parties try to improve their model. Since the local data are with the same feature space, the parties can train the local models using their local data with the same model architecture. The global model can simply be updated by averaging all the local models. It is most commonly data partitioning on the majority of FL studies up to date, and the most popular framework of horizontal FL is FedAvg. Apple's 'Hey Siri' or Google's 'OK Google' wake-word recognition are typical application of horizontal data partitioning (each user says same sentence but different voice).
- In ***vertical FL***, the datasets of different parties have the ***same sample space but differ in the feature space***. An example can be These two departments share the same sample space (i.e., all the residents in the country) but each of them only has one part of features (e.g. housing or tax related personal data). Entity alignment techniques to collect the overlapped samples of the parties are usually adopted in novel FL systems. Then, the overlapped data are used to train the shared model using encryption methods. Cooperation among government agencies can be treated as a situation of vertical partition. Suppose the department of taxation requires the housing data of residents, which are stored in the department of housing, to formulate tax policies. Meanwhile, the department of housing also needs the tax information of residents, which is kept by the department of taxation, to adapt their housing policies.

A.1.4 - ML models

Since FL is used to solve ML problems, the parties usually want to train state-of-the-art ML models, as described next:

- The most popular ML models are ***neural networks (NN)***, which achieve state-of-the-art results in many AI tasks, like image classification and word prediction. From FL perspective, there are many studies based on stochastic gradient descent (SGD), which can be used to train NNs, and many FL frameworks are proposed based on this approach. However, not all IoT devices are hardware-capable to exploit this type of ML models.

- Other widely used ML models are ***decision trees***, which re highly efficient to train and easy to interpret compared with NNs. A tree-based FL System is designed for the training for a single or multiple decision trees (e.g., gradient boosting decision trees (GBDTs) and random forests). Due to its lower complexity, GBDTs are especially popular recently in FL, and they have a very good performance in many classification and regression tasks.
- Besides NNs and trees, ***linear models*** (e.g., linear regression, logistic regression, support vector machines) are classic and easy-to-use models. These linear models are basically easy to learn compared with other complex models (e.g., NNs), but they have lower accuracy performance.

A.1.5 - Privacy mechanisms

Although, ideally, the local data is expected not to be exposed in FL, the exchanged model parameters may still leak sensitive information about the data. Since there may be many potential attacks against ML models such as model inversion attack and membership inference attack, privacy mechanisms should be adopted in the FL training process:

- ***Cryptographic methods*** such as homomorphic encryption, and secure multi-party computation (SMC) are widely used in privacy-preserving ML algorithms. Basically, the parties have to encrypt their messages before sending, operate on the encrypted messages, and decrypt the encrypted output to get the final result. By applying these methods, the user's privacy can be well protected. However, due to the additional encryption and decryption operations, cryptographic methods increases high computational overheads.
- ***Differential privacy*** guarantees that one single record does not influence much on the output of a function. Many studies adopt differential privacy for data privacy protection in order to guarantee that the parties cannot know whether an individual record contributes to the learning or not. By injecting random noises to the data or the ML model parameters, differential privacy provides statistical privacy as well as protection against the inference attacks. Nevertheless, the noise additions in the training process produce less accurate models.

A.1.6 - Motivation of Federation

In real-world applications of FL, individual parties need the motivation to get involved. The motivation can be ***regulations*** or ***incentives***. FL inside a company or an organisation is usually motivated by regulations (e.g., FL across different departments of a company). But in many cases, parties cannot be forced to provide their data by regulations. Taking Google's Gboard as an example, the company cannot prevent users who do not provide data from using it. However, those who agree to upload input data may enjoy a higher accuracy of word prediction. This kind of incentives can encourage every user providing their data to improve the performance of the overall model. However, how to design such a reasonable protocol remains challenging. Incentive mechanism design can be very important for the success of an FL System. There have been some successful cases for incentive designs in Blockchain. The parties inside the system can be collaborators as well as competitors. Other incentive designs are proposed to attract participants with high-quality data for FL.

A.2 - ASSIST-IoT business scenarios and FL needs matrix

The following table aims at gathering in a single and comprehensive form the requirements and needs expressed by pilot owners with respect to the most suitable development of FL in the project:

Table 3. Federated Learning aspects of ASSIST-IoT Use Cases

Pilot	Business Scenario	Potential FL adoption	Communication architecture	Scale of Federation	Data Partitioning	ML Model implementation	Privacy mechanism	Motivation of Federation
Port Automation	**RTG remote control with AR support**	UC-P1-7 Target visualisation during RTG operation	***Centralised***	***Cross-silo***	***Horizontal***	***TBD***	***TBD***	***Incentives***
Smart safety of workers	**Occupation safety and health monitoring**	UC-P2-1 Worker's health and safety assurance	***Centralised***	***Cross-device***	***Horizontal***	***TBD***	***TBD***	***Regulation***
Cohesive vehicle monitoring and diagnostics	**Vehicle diagnostics**	UC-P3A-2 Vehicles non-conformance causes identification	***Centralised***	***Cross-device***	***Horizontal***	***TBD***	***TBD***	***Regulation***
	Vehicle exterior condition inspection and documentation	UC-P3B-1 Vehicle's exterior condition documentation UC-P3B-2 Exterior defects detection support	***Centralised***	***Cross-silo***	***Horizontal***	***TBD***	***TBD***	***Regulation***

Annex B - Enabler templates

B.1 - Self-*: Self-healing device enabler

Table 4 General information for Self-healing device enabler

Enabler	***Self-healing device enabler***
Id	*SELF11*
Owner and support	*PRODEVELOP, SRIPAS, UPV*
Description and main functionalities	*This enabler aims at providing to IoT devices with the capabilities of actively attempting 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-*, Federated Learning, Security, Privacy and Trust, Manageability - as self-healing is an universal capability, it actions might affect all other enablers and/or functionalities across all verticals.*
Plane/s involved	• *Device and Edge Plane* • *Smart Network and Control Plane*
Relation with other enablers	• *T42E2: SDN controller* • *T42E3: Auto-configurable Network enabler* • *SELF15: Automated device connection and configuration* • *SELF14: Monitoring and Notifying* • *T44E3: Performance and usage diagnoses enabler*
Requirements mapping	• *R-C-5: Local processing capabilities* • *R-P1-2: CHE location availability* • *R-P1-21: Remote reliability capabilities* • *R-P2-3: Smart wristband for construction workers* • *R-P2-18: Temporary storage* • *R-P2-4: Continuous authentication for wristband* • *R-P2-16: Device reliability and durability* • *R-P3A-9: Edge Intelligence* • *R-P3A-11: Connectivity between OEM and fleet*
Use cases mapping	• *UC-P1-2: CHE location tracking* • *UC-P1-5: RTG-Truck alignment* • *UC-P1-6: Wireless remote RTG operation* • *UC-P2-1: Worker's health and safety assurance* • *UC-P2-5: Near-miss fall from height detection* • *UC-P3A-2: Vehicle non-conformance causes identification*

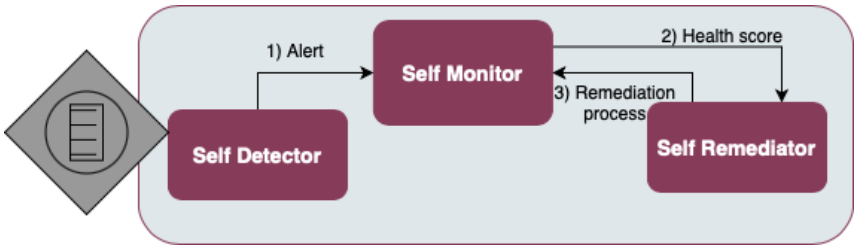

Figure 8. High-level structure for Self-healing device enabler

Components

Enabler component	***Self Detector***
Id	*SELF11_DETECTOR*
Description and main functionality	*The goal of this component is to collect information from the IoT device, and to determine whether the IoT device is being compromised. In principle is foreseen to be*

	implemented as a lightweight Host-based Intrusion Detection System. If the device is found to be compromised, the component issues an alert that is used by the Self Monitor component
Target node/s	*Edge Node, IoT Gateway*
Candidate technologies	*A Host-Based Instruction Detection System such as InfluxData Telegraf, CIoTA*

Enabler component	***Self Monitor***
Id	*SELF11_MONITOR*
Description and main functionality	*The Self monitor component is responsible for assessing the device's state of health. It collects and analyses data from multiple sources of information, such as memory usage, memory access, network connection metrics (RSSI levels), or CPU usage, providing a health score. The health score metrics are fed to a predefined set of rules (or to an anomaly-detection model) that determines whether the device is in a healthy state or not. The output of this component is used to determine if the remediation has been successful.*
Target node/s	*Edge Node, IoT Gateway*
Candidate technologies	*Node-Red, a custom based JavaScript framework; Thingsboard Community Edition*

Enabler component	***Self Remediator***
Id	*SELF11_REMEDIATOR*
Description and main functionality	*When the device presents with symptoms of malfunctioning or intrusion, this component's job is to determine from a set of remediation processes, which should be used for a proper treatment. The self-monitor component will assign a health score to the remediation process in order to reflect the effectiveness in threating the device. If after the remediation, the device is not back to its normal state, the self-remediator is triggered to select another remediation process from the list*
Target node/s	*Edge Node, IoT Gateway*
Candidate technologies	*Custom-based Node-Red, Thingsboard Community Edition*

B.2 - Self-*: Resource provisioning enabler

Table 5. General information for Resource provisioning enabler

Enabler	***Resource provisioning***
Id	*SELF12*
Owner and support	*SRIPAS, UPV*
Description and main functionalities	*This enabler will be able to horizontally scale (up or down) the resources devoted to a specific enabler (inside a node) in a dynamic fashion. ASSIST-IoT aims at working on changing environments where resource availability must be used for the sake of the system as a whole.* *The objective of the enabler is to allow such scaling (per enabler and per node = K8s service in a topologically co-located cluster) so that software instances will be assigned more or less resources according to the pressure put on the system.* *The "pressure put on the system" will be represented through a series of indicators (metrics) that might be a combination of internal performance (of the enabler) and the global status of the deployment (involving other enablers' information).* *Finally, this enabler will be able to dynamically adapt (SELF-*) the provided resources to the enablers based on past data, historic trends, ML models.*
Vertical, related capabilities and features	• *Self-**
Plane/s involved	• *Data Management Plane* • *Device and Edge Plane*
Relation with other enablers	• *T55E1: Devices Management - to check the status of the rest of devices, this enabler must know which other devices do exist and how to reach them* • *T55E3: Workflow between enablers - the relation with this enabler might be complementary/substitutive to the previous; the objective is to realise the*

	"world" around the particular node where this enabler will run in order to get "performance parameters/indicators" • *T44E2: Performance and usage diagnosis - to gather information - if needed- of the global performance of the system* • *T43E4: LTSE - to store metrics for further training of the ML model incorporated*
Requirements mapping	• *RC-7: Edge-oriented deployment* • *R-P3A-9: Edge intelligence* • *R-P3B- 20: Information Pre-fetching*
Use cases mapping	*First, let's note that this enabler will be present -as aforementioned- in any non-trivial deployment of ASSIST-IoT. Therefore, the following bullet points do not showcase an exclusive list of presence of the enabler but rather highlights those cases in which the use of the functionality is explicitly mentioned to meet the objectives of the pilot.* *UC-P3A-1: Fleet in-service emissions verification* *UC-P3B-1: Vehicle's exterior condition documentation*

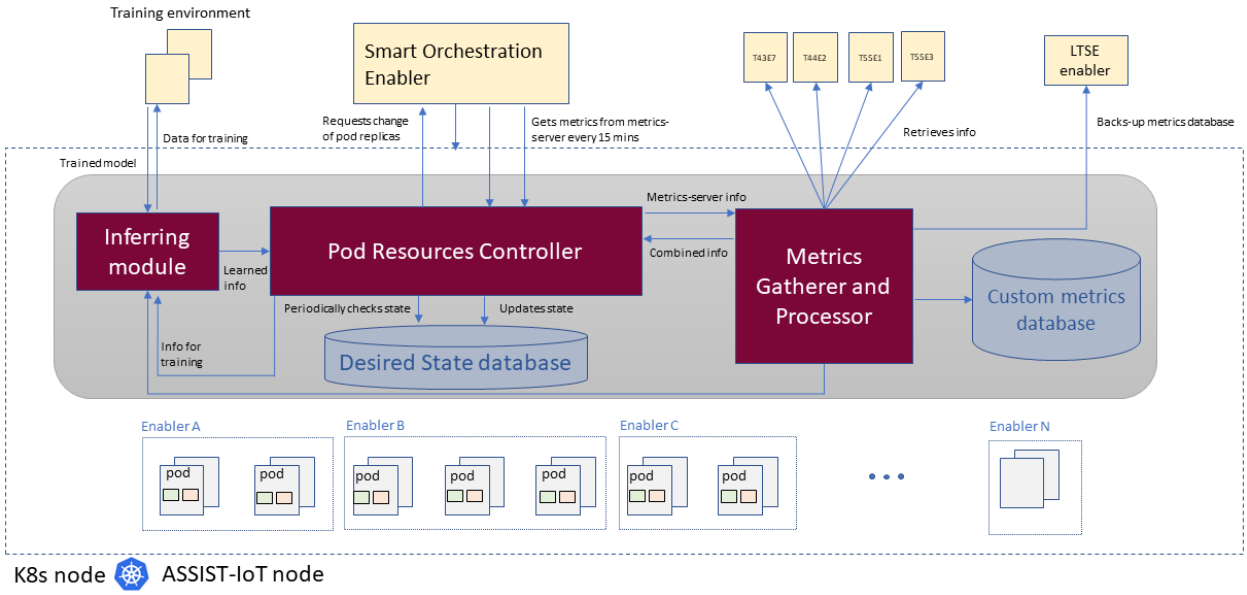

Figure 9. High-level structure for Resource provisioning enabler

Components

Enabler component	***Pod Resources controller***
Id	*T51E2_controller*
Description and main functionality	*When the device presents with symptoms of malfunctioning or intrusion, this component's job is to determine from a set of remediation processes, which should be used for a proper treatment. The self-monitor component will assign a health score to the remediation process in order to reflect the effectiveness in threating the device. If after the remediation, the device is not back to its normal state, the self remediator is triggered to select another remediation process from the list.*
Target node/s	*Edge node, IoT Gateway*
Candidate technologies	*Custom component in Python, K8s HPA*

Enabler component	***Inferring module***
Id	*T51E2_inferring*
Description and main functionality	*A module that will use a trained ML model (e.g., a time series forecasting method trained with historic of that very enabler and/or others) to infer how many resources will be needed to run it at a specific moment.*
Target node/s	*Any ASSIST-IoT node*
Candidate technologies	*Custom component in Python*

Enabler component	***Desired state database***
Id	*T51E2_state*
Description and main functionality	*A database (lightweight, structured) just to manage and the store the current desired state (in terms of pods replicas of the components) of each enabler. NOTE: This tiny database could be substituted by one Collection inside the MongoDB of the enabler. It has been left here for the sake of clarity and conceptual separation.*
Target node/s	*Any ASSIST-IoT node*
Candidate technologies	*SQLite*

Enabler component	***Metrics Gatherer and Processor***
Id	*T51E2_metrics*
Description and main functionality	*A module that retrieves information from other enablers, other edge nodes and the global system of the ASSIST-IoT deployment in order to build a custom metrics database which the Pod Resources Controller will act upon.*
Target node/s	*Any ASSIST-IoT node*
Candidate technologies	*Custom component in Python*

Enabler component	***Custom Metrics Database***
Id	*T51E2_database*
Description and main functionality	*A database for storing combined metrics per enabler, so ML models can be trained with these data.*
Target node/s	*Any ASSIST-IoT node*
Candidate technologies	*MongoDB*

B.3 - Self-*: geo (Localisation) enabler

Table 6 General information for geo (Localisation) enabler

Enabler	***(geo) Localisation***
Id	*SELF13*
Owner and support	*SRIPAS, NEWAYS*
Description and main functionalities	*To solve challenges of pilots we need to localise physical objects (containers in ports, workers on construction sites), some devices should be aware of their position in relation to each other (aligning cranes and tractors). We might need to realize localisation using absolute coordinates (GPS) or relative (coordinates in a port).*
Vertical, related capabilities and features	• *Self-* (autonomy)* • *Self-* - system will react to the incoming events (detecting devices, devices joining, configuration changing, etc.) and taking informed decision on what action should be taken* • *Interoperability - this enabler will work with heterogeneous devices, hence interoperability must be taken into consideration*
Plane/s involved	• *Smart IoT Device - this enabler will connect and configured devices* • *Edge Plane - self-explanatory, this enabler will connect and configured devices* • *Smart Network and Control Plane - devices will be using various network mediums (wireless, 5G, physical connection)* • *Data Management Plane - this enabler will need to ensure interoperability between heterogeneous devices* • *Application and Services Plane - this enabler must ensure that there is an interface with which an end user can communicate between the different heterogeneous devices. The device location is track and traceable, be monitored, be analysed, made visible, and if necessary, notifications can be received and sent.*
Relation with other enablers	*Vertical:* • *T53E3: Identity Manager - devices that will be connecting must be properly identified*

	• *T53E1: Authorisation - devices that will be connecting must be properly authorised* • *SELF15: Automated Configuration enabler - devices that will be connecting must be properly configurated*
Requirements mapping	• *R-C-7: Edge-oriented deployment* • *R-P1-1: CHE location services* • *R-P1-2: CHE location availability* • *R-P1-3: CHE positioning accuracy* • *R-P1-5: Container ID tracking system* • *R-P2-1: Personal location tracking* • *R-P2-2: Construction plant location tracking* • *R-P2-11: Geofencing* • *R-P2-14: Evacuation Instructions*
Use cases mapping	• *UC-P1-1: Asset location management - for location management assets must know where they are* • *UC-P1-2: CHE location tracking - for location tracking ASSIST-IoT needs to know where it is* • *UC-P1-5: RTG-Truck alignment - to align both device types, we need to understand where they are* • *UC-P1-6: Wireless remote RTG operation - system needs to understand where it is* • *UC-P2-1: Workers' health and safety assurance* • *UC-P2-2: Geofencing boundaries enforcement - need for localisation is self-explanatory, of course within the buildings GPS cannot be used* • *UC-P2-3: Danger zone restrictions enforcement - need for localisation is self-explanatory, of course within the buildings GPS cannot be used* • *UC-P2-4: Construction site access control - might be a stretch, but if site access control could be location based* • *UC-P2-5: Near-miss fall from height detection; Height fall accident detection - if the fall occurs, we need to know the exact place of the accident including the location data* • *UC-P2-6: Safe navigation instructions - navigation requires localisation*

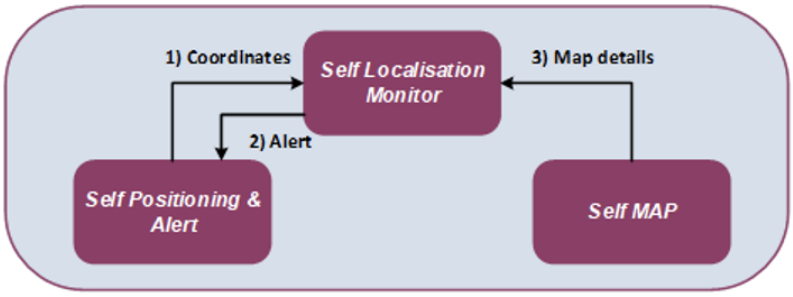

Figure 10. High-level structure of geo(Localisation) enabler

Components

Enabler component	***Self-localisation Positioning & Alert***
Id	*SELF13_Positioning & Alert*
Description and main functionality	*The purpose of the localisation positioning and alert is to; 1) collect information about (geo)location coordinates of the IoT device; 2) to give an alert to both the smart IoT device and the OSH (Occupational Safety & Health) manager in case the smart IoT device is located in an unauthorised zone or danger zone, or if an unsafe situation is created due to an incident.*
Target node/s	*Gateway/Edge node*
Candidate technologies	*A custom based framework; REST, MQTT*

Enabler component	***Self-localisation MAP***
Id	*SELF13_MAP*
Description and main functionality	*The purpose of MAP is to provide the latest information on indoor and outdoor buildings, floors, and its surroundings, including information on geofencing, border enforcement, danger zones, and escape and security routes. The incident log within the map will be updated for the OSH management system..*
Target node/s	*Edge Node, IoT Gateway*
Candidate technologies	*A custom based framework; REST, MQTT*

Enabler component	***Self-localisation Monitor***
Id	*SELF13_LOCALISATION MONITOR*
Description and main functionality	*The purpose of the localisation monitor is to collect the (geo)location coordinates and alarms of all connected smart IoT devices, to determine if the IoT device is within an authorised area and if there is a possible unsafe situation. The MAP component serves as 3) input to gain insight into the latest map details and (geo)location status.* *If an IoT device is in an unauthorised zone or danger zone, or if an unsafe situation is created due to an incident, an alert is issued to both the smart IoT device and the OSH manager, and an incident log is generated saved and the map updated.*
Target node/s	*Edge Node, IoT Gateway*
Candidate technologies	*A custom based framework; REST, MQTT*

B.4 - Self-*: Monitoring and notifying enabler

Table 7. General information for Monitoring and notifying enabler

Enabler	***Monitoring and Notifying***
Id	*SELF14*
Owner and support	*SRIPAS, CERTH*
Description and main functionalities	*This enabler could be viewed as a general purpose by representing it as a combination of high-level monitoring module (which would allow to monitor devices, logs, etc.) and notifying module that could send custom messages to predefined system components.* *For example, on construction sites we will monitor health signals of workers. Those signals should be monitored and in case of breaching some threshold notification should be sent and action might be taken.*
Vertical, related capabilities and features	• *Self-**
Plane/s involved	• *Device and Edge Plane – devices (smart, IoT) are part of the use cases for various actions (e.g., monitor data)* • *Application and Services Plane – monitoring will have to include an interface for an end user to interact. Notification is similar to the monitoring* • *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.* • *Smart Network and Control Plane (Network with SDN)*
Relation with other enablers	*Vertical:* • *A relation with T54E1 Logging and auditing enabler may exist. The enabler is included in the Security, Privacy and Trust vertical.*
Requirements mapping	• *R-P1-1: CHE location services* • *R-P1-2: CHE location availability* • *R-P1-10: CHE identification* • *R-P2-1: Personal location tracking* • *R-P2-3: Smart wristband for construction workers* • *R-P2-2: Construction plant location tracking* • *R-P2-4: Continuous authentication for wristband* • *R-P2-7: Monitoring the weather conditions at the construction site*

	• *R-P2-10: Motion Pattern Monitoring and Analysis* • *R-P2-12: Alerts and notifications minimisation* • *R-P2-14: Evacuation Instructions* • *R-P3A-10: Vehicle Dashboard Notifications* • *R-P3A-7: Active monitoring mode initiation by the Aftersales service technician capability* • *R-P3B-19: Critical Damage Identification Time*
Use cases mapping	• *UC-P1-1: Asset location management* • *UC-P1-2: CHE location tracking* • *UC-P1-4: RTG-truck identification and authentication* • *UC-P1-3: Container handling operations reporting - this might be a little bit of stretch, but CHE needs to notify its operations to other components* • *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 - tt 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 takes action when unauthorised access was detected)* • *UC-P2-2: Geofencing boundaries enforcement - when you breach fence the notification will be sent* • *UC-P2-3: Danger zone restrictions enforcement - similar to the above* • *UC-P2-4: Construction site access control* • *UC-P2-5: Near-miss fall from height detection* • *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's non-conformance causes identification* • *UC-P3B-1: Vehicle's exterior condition documentation*

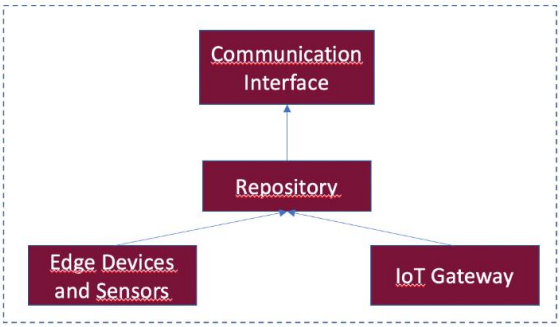

Figure 11. High-level structure of Monitoring and notifying enabler

Components

Enabler component	***Edge Devices and Sensors***
Id	*SELF14-01*
Description and main functionality	*The edge devices have to capture data from the real world in order to kick off the process. The edge devices and the measurements to provide are dependent on the use case. The port automation has the need to capture the location and operational status for CHE and the containers' location. Work safety requires a wider range of data like weather, personal health data, and location.*
Target node/s	*TBD*
Candidate technologies	*TBD*

Enabler component	***IoT Gateway***
Id	*SELF14-02*

Description and main functionality	*The IoT Gateways, their capabilities, and their usage can have an impact on the enabler. For instance, the gateway can perform an action and share that there is a critical situation.*
Target node/s	*TBD*
Candidate technologies	*TBD*

Enabler component	***Repository (Database)***
Id	*SELF14-03*
Description and main functionality	*The enabler has to handle data for monitoring and notification. An open issue to research/ determine the type of the database (SQL or NoSQL).*
Target node/s	*TBD*
Candidate technologies	*TBD*

Enabler component	***Communication Interface***
Id	*SELF14-04*
Description and main functionality	*A communication interface that will handle the message flow in the enablers' components.*
Target node/s	*TBD*
Candidate technologies	*Kafka*

B.5 - Self-*: Automated configuration enabler

Table 8. General information for Automated configuration enabler

Enabler	***Automated Configuration Enabler***
Id	*SELF15*
Owner and support	*SRIPAS*
Description and main functionalities	*Automated Configuration Enabler keeps heterogenous devices and services synchronised with their configurations. User can update configuration and define fallback configurations in case of errors. Self-* component will be responsible for reacting to changing environment and updating configuration as necessary.*
Vertical, related capabilities and features	• *Self-* - system will react to the incoming events (detecting devices, devices joining, configuration changing, etc.) and taking informed decision on what action should be taken* • *Interoperability - this enabler will work with heterogenous devices, hence interoperability must be taken into consideration*
Plane/s involved	• *Device and Edge Plane – self-explanatory, this Enabler will connect and configured devices.* • *Smart Network and Control Plane – devices will be using various network mediums (wireless, 5G, physical connection)* • *Data Management Plane – this enabler will need to ensure interoperability between heterogeneous devices*
Relation with other enablers	*Horizontal:* • *T43E1: Semantic repository - Semantic Repository will allow to retrieve data model from the SR enabler and easily communicate with already connected device and send the required configuration* • *T43E3: Semantic annotation - it would be great if we could have configuration in some independent form and then use SA to annotate that configuration and send it to the device* *Vertical:* • *T53E3: Identity Manager - devices that will be connecting must be properly identified* • *T53E1: Authorisation - devices that will be connecting must be properly authorised* • *T53E4: Monitoring Agent - required for threat detection* • *T53E4: Cybersecurity monitoring - required for threat detection*

Enabler	*Automated Configuration Enabler*
	• *T54E1: Logging and auditing - configure device events should be securely stored, so a user can easily retrace everything that happened*
Requirements mapping	• *R-C-1: Data sovereignty, R-C-2: Data governance, R-C-3: Compliance with legal requirements on data protection, R-C-5: Local processing capabilites, R-C-6: Data persistence and trust - we will be storing configuration data, which might include sensitive data, so all data related requirements apply* • *R-C-7: Edge-oriented deployment - this enabler should be easily deployed on the edge* • *R-P1-6, R-P1-16 or R-P1-24 - as a means of sending the configuration data* • *R-P2-11: Geofencing - updating geofencing area (TBD)* • *R-P2-14: Evacuation instructions - updating evacuation instruction (TBD)* • *R-P3A-5: Data Storage* • *R-C-5: Local processing capabilities* • *R-P1-21: Remote reliability capabilities* • *R-P3A-7: Active monitoring mode initiation by the Aftersales service technician capability* • *R-P3B-19: Critical Damage Identification Time*
Use case mapping	• *UC-P1-1: Asset location management - when new assets are joining - they need to automatically download config, environmental data, etc.* • *UC-P2-1: Workers' health and safety assurance - all workers' wearables need to be automatically be able to connect and be configured. Manual configuration would involve too much human operation* • *UC-P2-6: Safe navigation instructions - configuration for navigation (maps, updated danger zones, etc.) need to be keep updated automatically.* • *UC-P3A-3: Sending new configuration to PCM*

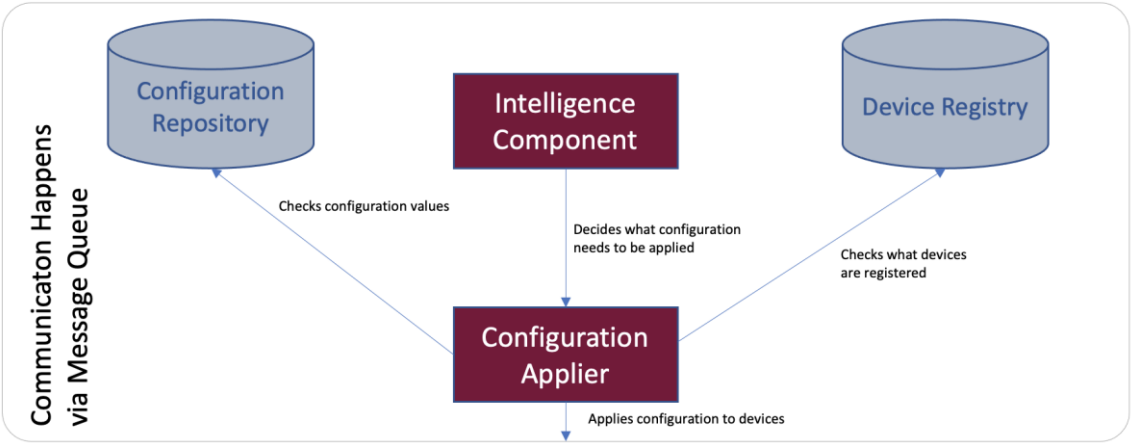

Figure 12. High-level structure for Automated configuration enabler

Components

Enabler component	*Configuration Repository*
Id	*SELF15-Db*
Description and main functionality	*A place to store details about configuration, configuration rules and information about whether all connected devices received configuration.*
Target node/s	*Node powerful enough to host database*
Candidate technologies	*Postgres*

Enabler component	*Configuration Applier*
Id	*SELF15-Cfgr*
Description and main functionality	*Component responsible for checking and applying configuration updates via Device/Service Connector as well as reacting to failure events.*

Enabler component	*Configuration Applier*
Target node/s	*Edge node powerful enough to host JVM*
Candidate technologies	*Akka Typed, Scala*

Enabler component	*Message Queue*
Id	*SELF15-mq*
Description and main functionality	*Transporting messages internally between components and connectors*
Target node/s	*At least 3 nodes, to ensure reliability*
Candidate technologies	*Kafka*

Enabler component	*Device Registry*
Id	*SELF15-rs*
Description and main functionality	*Component responsible for registering devices/connector types and providing list of devices for defined connector types.*
Target node/s	*Edge node powerful enough to host JVM*
Candidate technologies	*Akka Typed, Scala*

Enabler component	*Intelligence Component*
Id	*SELF15-smrt*
Description and main functionality	*Module will be responsible for deciding which configurations and configuration values should be applied. This could be realised in one of two ways:* • *User will define the desired state of the system and intelligence component will apply the values* • *User will define rules and in case of some events configuration will be updated.*
Target node/s	*Edge node powerful enough to host JVM*
Candidate technologies	*Akka Typed, Scala*

B.6 - Federated machine learning: FL Orchestrator

Table 9. General information for FL Orchestrator

Enabler	*FL Orchestrator*
Id	*T52E1*
Owner and support	*PRODEVELOP, SRIPAS, UPV*
Description and main functionalities	*In general, FL takes place on local nodes training local version of the shared model and producing parameters to be combined to deliver a new (improved) version of the global/shared model. Moreover, the basic version of FL consists of multiple workers and a central manager and follows the vanilla master-slave pattern. However, FL can involve also additional nodes (virtual or real) that combine selected groups of parameters in a scenario-specific order (e.g., mediator nodes). This may happen when, among others, the system is to deal with complex data ownership structures, or when data is unbalanced. The FL orchestrator is responsible of specifying details of FL workflow(s)/pipeline(s). This includes FL job scheduling, managing the FL life cycle, 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 - it shall manage/orchestrate the training process among the involved parties* • *Scalability - it should be able to add/remove desired/undesired parties from the training as well as inferencing processes*
Plane/s involved	• *Smart Network and Control Plane - a network interface should be managed in the communication between parties – mediators (if any) - masters*

Enabler	*FL Orchestrator*
Relation with other enablers	• *T44E4: OpenAPI management enabler* • *T52E2: FL Training collector* • *T52E3: FL repository* • *T52E4: FL Local Operations enabler* • *T52E5: FL Privacy enabler*
Requirements mapping	• *R-C-2: Data governance* • *R-C-3: Compliance with legal requirements on data protection* • *R-C-5: Local processing capabilities* • *R-C-7: Edge-oriented deployment* • *R-P3A-9: Edge intelligence* • *R-P3A-12: Edge connectivity* • *R-P3B-15: Automatic defect detection* • *R-P3B-21: Automatic recognition* • *R-P3B-14: Defect classification categories*
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*

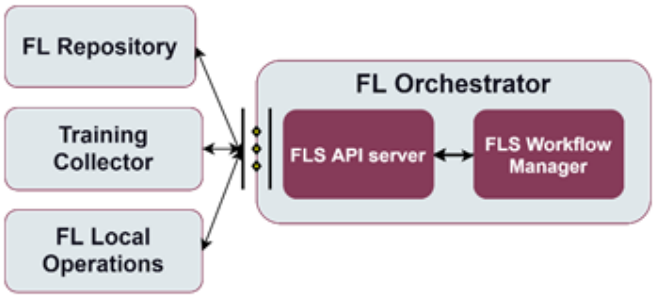

Figure 13. High-level structure of FL Orchestrator

Components

Enabler component	*FLS API server*
Id	*T52E1_API*
Description and main functionality	*Offers a REST API to allow the communication and interaction with FL Structure components. Hence, it allows to retrieve information or perform FL management actions, to FL parties or external actors.*
Target node/s	*Cloud server, High-tier edge node*
Candidate technologies	*Swagger*

Enabler component	*FLS Workflow Manager*
Id	*T52E1_WorkflowManager*
Description and main functionality	*This component is in charge of defining workflow for a specific incarnation of FL lifecycle. The workflow specifics are configured on the basis of a file containing information in JSON format (scheduling candidates are global sate and optimistic lock scheduling, single-party DAG scheduling, multi-party coordinated scheduling). Workflow description also specifies, among others, source of initial shared algorithm, ML technique to be used (possibly multi-component one), topology for information exchange (parameter flow), use of additional/auxiliary nodes, method used for parameter aggregation, handling of error conditions (e.g., one of workers going offline/not delivering results), management module(s). It is also finally in charge of defining when the training process should stop based on e.g., the federated model convergence, or reaching a predefined max-iteration threshold .*
Target node/s	*Cloud server, High-tier edge node*
Candidate technologies	*FATE FLOW*

B.7 - Federated machine learning: FL Training Collector

Table 10. General information for FL Training Collector

Enabler	***FL Training Collector***
Id	*T52E2*
Owner and support	*SRIPAS, NEWAYS, UPV, PRODEVELOP*
Description and main functionalities	*In the context of this enabler, two aspects of the FL process need to be supported.* 1. *There exist multiple ways of combining local results to deliver new version of the shared model (e.g., FedSGD, FedAvg); moreover, averaging can be completed in a single step, or updates can be applied sequentially in a specific order.* 2. *Different topologies of FL-running systems may be supported. For instance, beyond the master-slave topology, in order to deal with unbalanced data, additional layer of "Mediator Training collectors" have been recently proposed. In this context one should acknowledge that as research on FL continues, new methods of combining local results, and novel topologies (matching needs of complex use cases) can be proposed. Hence, this enabler should be able to accommodate them.* *Hence, the FL training collect can be univocally instantiated, or in order to realise more advanced topologies like the one using Mediators, multiple instances can be configured within a given workflow (T52E1) to realise a hierarchical FL. 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.*
Plane/s involved	*Smart Network and Control plane, a network interface should be managed in the communication between parties – mediators (if any) - masters*
Vertical	*N/A*
Relation with other enablers	• *T43E8: Long-term data storage enabler* • *T44E4: OpenAPI management enabler* • *T52E2: FL Orchestrator* • *T52E3: FL repository* • *T52E4: FL Local Operations enabler* • *T52E5: FL Privacy enabler*
Requirements mapping	• *R-P2-7: Monitoring the weather conditions at the construction site* • *R-P2-8: Personal cooling system* • *R-P3A-9: Edge intelligence* • *R-P3B-15: Automatic defect detection* • *R-P3B-21: Automatic recognition* • *R-P3B-14: Defect classification categories*
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.*

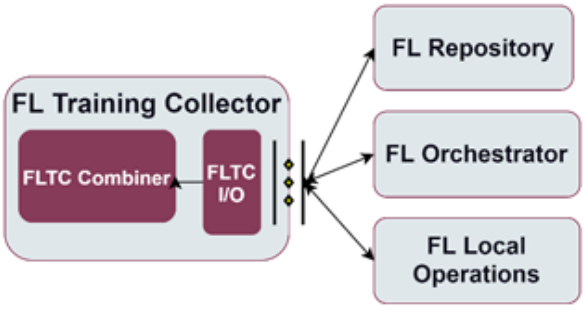

Figure 14. High-level structure for FL Training Collector

Components

Enabler component	*FLTC I/O*
Id	*T52E2_IO*
Description and main functionality	*Provides a REST API to allow the input and output communication to and from the FL training collector enabler. On the one hand it is responsible of receiving and FL local updates that are sent to the FLC Combiner component. On the other hand, it is responsible of communicating updates of the new FL model obtained in the FLC Component to involved training parties or to the FL repository. The communication capabilities of this component will be designed so that it can conceptually deal with situations in which more complex topologies are used.*
Target node/s	*Cloud server, High-tier edge node*
Candidate technologies	*Swagger*

Enabler component	*FLTC Combiner*
Id	*T52E2_Combiner*
Description and main functionality	*This component will receive "suggestions" from a certain number (possibly all) local nodes and combine them to generate an updated FL model. It can include both homogeneous and heterogeneous FedAvg solutions for e.g., Logistic Regression Models, Decision-tree Models, or even Neural Network Models. Finally, it will broadcast the aggregated model back to involved parties.*
Target node/s	*Cloud server*
Candidate technologies	*FedML*

B.8 - Federated machine learning: FL Repository

Table 11. General information for FL Repository

Enabler	*FL Repository*
Id	*T52E3*
Owner and support	*SRIPAS, NEWAYS, PRODEVELOP, UPV*
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 is proposed. This repository will store (and deliver upon request/need) the ML algorithms or ML models. The FL repository will consist 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). Note that each component will be demarcated with metadata, specifying all necessary aspects of its functionality. Moreover, multiple implementations of the same functionality, optimised for different computer hardware and/or implemented in different languages, may also be stored. While it is assumed that majority of components will be stored as source code, use of binaries are not precluded.*
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*
Vertical	• *Manageability - it shall manage/orchestrate the training process among the involved parties* • *Scalability - it should be able to add/remove desired/undesired parties from the training as well as inferencing processes*
Relation with other enablers	• *T43E8: Long-term data storage enabler* • *T44E4: OpenAPI management enabler* • *T52E1: FL Orchestrator* • *T52E2: FL Training collector* • *T52E3: FL repository*

Enabler	*FL Repository*
Requirements mapping	• *R-C-2: Data governance* • *R-C-3: Compliance with legal requirements on data protection* • *R-C-5: Local processing capabilities* • *R-C-7: Edge-oriented deployment* • *R-P2-7: Monitoring the weather conditions at the construction site* • *R-P2-8: Personal cooling system* • *R-P3A-9: Edge intelligence* • *R-P3A-12: Edge connectivity* • *R-P3B-15: Automatic defect detection* • *R-P3B-21: Automatic recognition* • *R-P3B-14: Defect classification categories*
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*

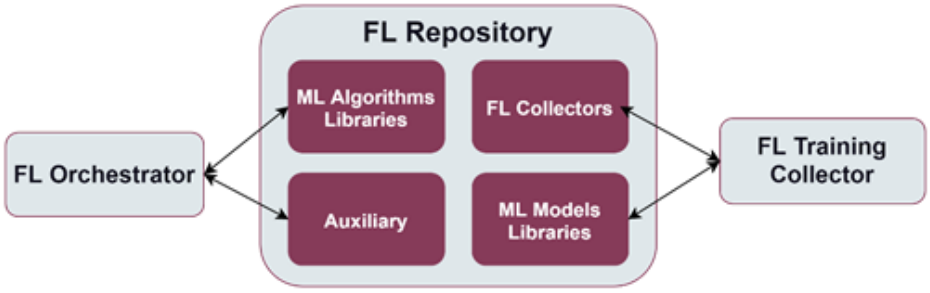

Figure 15. High-level structure for FL Repository

Components

Enabler component	*ML Algorithms library*
Id	*T52E3_ML_Algorithms*
Description and main functionality	*These libraries will be used by local nodes to instantiate local processes (T52E1). The way that libraries (modules) will be stored will be similar to the way that standard ML libraries It will made available ML algorithms that can be used for either regular ML modelling, or for FL modelling. Moreover, as in the well-known cases of use of external ML modules, appropriate ML library modules are to be downloaded to the local node, installed and used to complete model training.*
Target node/s	*Cloud server, High-tier edge node*
Candidate technologies	*Decision concerning way in which the metadata will be implemented and used needs to be further investigated. The two main approaches are: (i) use of semantic technologies, or (b) use of XML-based demarcation, such as Scikit-learn*

Enabler component	*ML Models library*
Id	*T52E3_ML_Algorithms*
Description and main functionality	*The repository will also persist ML trained models. These models can be conceptualised in two "scenarios".* a. *If the enabler is installed on a local node, it will store models that are currently in training (note that local node can be involved in multiple FL processes realised independently) and/or are "in use" by this node.* b. *If the repository is instantiated in some "more central location" it will store current versions of shared models (including initial models). Here, depending on the topology, shared models may represent a group of nodes (e.g., in the case of use of Mediators), or be common to all nodes.*
Target node/s	*Cloud server, High-tier edge node*
Candidate technologies	*TensorFlow YOLO*

Enabler component	*FL Collectors library*
Id	*T52E3_FL_Collectors*
Description and main functionality	*As described in the FL Training Collector (T52E2), different Federated averaging algorithms can be applied to combine local results. This component of the FL repository will store them.*
Target node/s	*Cloud server, High-tier edge node*
Candidate technologies	*FedML*

Enabler component	*Auxiliary component*
Id	*T52E3_Auxiliary*
Description and main functionality	*Any other modules that may be needed to instantiate FL can be also stored in the FL repository. Among them possible modules related to process verification, error handling, stopping criteria, authorisation, belong to this category.*
Target node/s	*Cloud server, High-tier edge node*
Candidate technologies	*FATE Flow, FedML*

B.9 - Federated machine learning: FL Local Operations

Table 12. General information for FL Local Operations

Enabler	*FL Local Operations Enabler*
Id	*T52E4*
Owner and support	*SRIPAS, PRODEVELOP, UPV*
Description and main functionalities	*One of key goals of FL is to assure protection of privacy of data, 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 has to be in the same format, which is imposed by the ML model that is being employed. In order to carry out with all these local operations, the FL Local Operation enabler is proposed. It will consist of four components:* ***Local data transformer component*** *(that will be in charge of guaranteeing that data is appropriately formatted for the FL model in use),* ***Local Model training component, Local Model inference component****, and* ***Communication component*** *(to enable in and out communications between involved local parties and FL orchestrator and FL collector).*
Vertical, related capabilities and features	• *Manageability - it shall manage/orchestrate the training process among the involved parties* • *Scalability - it should be able to add/remove desired/undesired parties from the training as well as inferencing processes.*
Plane/s involved	• *Smart Network and Control plane, a network interface should be managed in the communication between parties – mediators (if any) - masters*
Relation with other enablers	• *T43E8: Long-term data storage enabler* • *T44E3: Performance and Usage diagnosis enabler* • *T44E4: OpenAPI management enabler* • *T52E1: FL orchestrator* • *T52E2: FL Training collector* • *T52E3: FL repository* • *T52E4: FL Local Operations enabler* • *T52E5: FL Privacy enabler*

Requirements mapping	• *R-C-2: Data governance* • *R-C-3: Compliance with legal requirements on data protection* • *R-C-5: Local processing capabilities* • *R-C-7: Edge-oriented deployment* • *R-P2-7: Monitoring the weather conditions at the construction site* • *R-P2-8: Personal cooling system* • *R-P3A-9: Edge intelligence* • *R-P3A-12: Edge connectivity* • *R-P3B-15: Automatic defect detection* • *R-P3B-21: Automatic recognition* • *R-P3B-14: Defect classification categories*
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.*

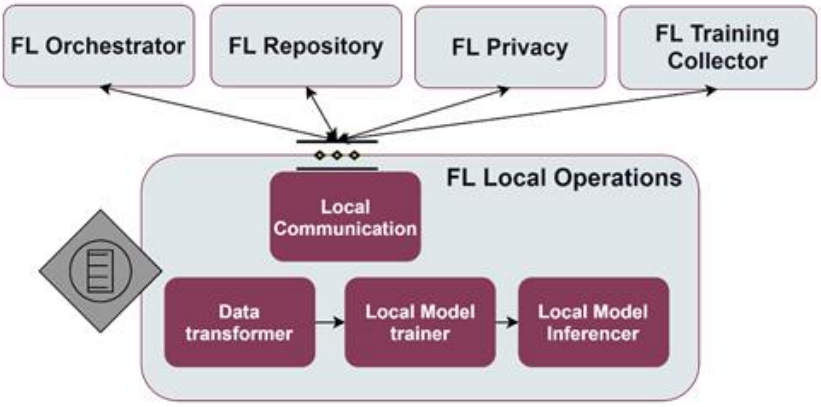

Figure 16. High-level structure for FL Local Operations

Components

Enabler component	***FL Local Data transformer***
Id	*T52E4_DataTransformer*
Description and main functionality	*In IoT ecosystems, each partner may (and is likely to) store data in its own (private/local) format. Use of FL requires transformation of appropriate parts of local data into the correct format. This format has to be described as part of the FL configuration, and all participating nodes have to oblige. This may be achieved by node owner providing appropriate transformation component. However, such component can be envisioned as being downloaded from the FL Repository (T52E3). Moreover, transformation may involve creation of a separate (sub-)repository consisting of only FL-bound data, or be based on performing translation "on the fly", during model training and/or inference. Note that it is easy to formulate rationale why one could create a (sub-)repository for FL training data, while to apply translation on the fly during model application. It should be stressed that transformations are applied only to data that participates in FL-related activities*
Target node/s	*Edge node*
Candidate technologies	*Scikit-learning, LabelEncoder*

Enabler component	***FL Local Model trainer***
Id	*T52E4_Local_Model_trainer*
Description and main functionality	*The Local Model training component is responsible for local model training. During configuration it instantiates appropriate ML training libraries and, if this is the beginning of the process, initial version of the shared model. This step can be completed locally by the node owner, but this is unlikely. The main problem would be assuring uniformity of training methods across nodes belonging to different owner. More likely,*

Enabler component	*FL Local Model trainer*
	the necessary modules (ML algorithm libraries and the initial version of the shared model) will be downloaded from the FL Repository (T52E3).
Target node/s	*Edge node*
Candidate technologies	*TensorflowFederated, or FATEClient*

Enabler component	*FL Local Data Inference*
Id	*T52E4_ Local_Model_Inference*
Description and main functionality	*The third component is responsible for use of the trained model. Here, the model may be used (1) after the FL process is completed, or (2) it may start to be used from a certain (predefined by the owner) level of quality of the shared model. In the latter case, each new version of the shared model would replace the previous one. Obviously, it is implicitly assumed that each new version of the shared global model will deliver better quality of results. Here, data to be fed into the trained model is transformed using the Data transformer component. Interpretation of the results of application of the model to specific input data (including actions to be, possibly, undertaken on the basis of the results) is likely to be provided by the data owner. However, it is also possible that appropriate module is going to be downloaded from the FL Repository (T52E3).*
Target node/s	*Edge node*
Candidate technologies	*OpenVINO, OpenCV (for video inference)*

Enabler component	*FL Local Communication*
Id	*T52E4_Local_Communication*
Description and main functionality	*Use of this component facilitates the only "correct" way to communicate in and out with the node that is participating in FL. It is to be configured to send parameters (parameter update proposals) to the FL Training Collector (T52E2) and receive back the updates to be applied to obtain the next version of the shared model.*
Target node/s	*Edge node*
Candidate technologies	*Swagger*

B.10 - Federated machine learning: FL Privacy

Table 13. General information for FL Privacy enabler

Enabler	*FL Privacy enabler*
Id	*T52E5*
Owner and support	*CERTH, SRIPAS, PRODEVELOP, UPV, NEWAYS*
Description and main functionalities	*Enabler that guarantees that different parties are not able to derive insights about each other's training data, based on messages exchanged during the training process (e.g., weights). Methods of creating differentially private noise, and homomorphic encrytpion will be available.*
Vertical, related capabilities and features	• *Self-* (Self-healing, Self-monitoring)* • *Security, Privacy and Trust*
Plane/s involved	• *Device and Edge Plane* • *Data Management Plane*
Relation with other enablers	• *T43E8: Long-term data storage enabler* • *T44E3: Performance and Usage diagnosis enabler* • *T44E4: OpenAPI management enabler* • *T52E1: FL orchestrator* • *T52E2: FL Training collector* • *T52E3: FL repository* • *T52E4: FL Local Operations enabler*
Requirements mapping	• *R-C-2: Data governance* • *R-C-3: Compliance with legal requirements on data protection* • *R-C-5: Local processing capabilities* • *R-C-7: Edge-oriented deployment*

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*

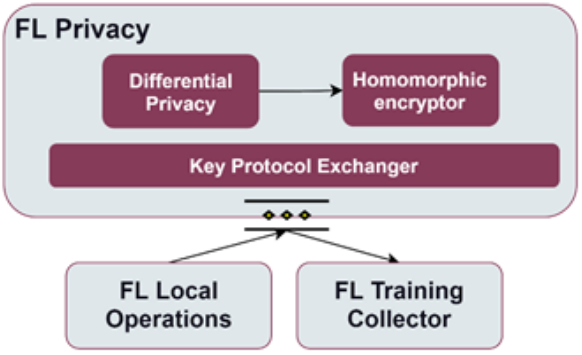

Figure 17. High-level structure for FL Privacy enabler

Components

Enabler component	***Homomorphic encryptor***
Id	*T52E5_Homomorphic_Encryptor*
Description and main functionality	*An encryption model is needed to ensure data confidentiality. Encryption models are transforming the data and are valuable to be executed prior to any data transferring. As data need to be stored or perform calculations, a homographic model may be the most appropriate model.*
Target node/s	*Nodes that have to send the FL output*
Candidate technologies	*Paillier Encryption, Affine Homomorphic Encryption*

Enabler component	***Differential privacy***
id	*T52E5_DiffPrivacy*
Description and main functionality	*Differential privacy mechanism is to run prior to any data transaction. The mechanism adds noise in the data making it harder for attackers to get the data. The process is a rigorous mathematical framework and calls for computational power.*
Target node/s	*Nodes that have to send the FL output*
Candidate technologies	*TBD*

Enabler component	***Keys protocol Exchanger***
id	*T52E5_ProtocolExchanger*
Description and main functionality	*Key exchange protocols can enhance the confidentiality and integrity in the network's communication. A paradigm that can be implemented is the SSL/TLS.*
Target node/s	*Nodes that have to send the FL output*
Candidate technologies	*TLS/SSL*

B.11 - Cybersecurity: Cybersecurity monitoring enabler

Table 14. General information for Cybersecurity monitoring enabler

Enabler	***Cybersecurity monitoring enabler***
Id	*T53E3*
Owner and support	*S21Sec*
Description and main functionalities	*Security monitoring enabler for threat detection and incident response.* *Provides security awareness and visibility and infrastructure monitoring.*

	Having raw data as input will set a series of processing steps will enable the discovery of cybersecurity threats. This process goes through a sequence of these steps: • *Collect, parse and normalise input events* • *Enrich normalised events* • *Correlate events to detect cybersecurity threats and produce alerts*
Vertical, related capabilities and features	• *Security, Privacy and Trust*
Plane/s involved	*all*
Relation with other enablers	• *T53E4: Cybersecurity monitoring agent enabler* • *T53E1: Authorisation enabler* • *T53E2: Identity Manager enabler* • *T43E8: Long Term Storage enabler*
Requirements mapping	• *R-P1-1: CHE location services and operational status* • *R-P2-4: Continuous authentication for wristband. Un-authorised use is detected and reported* • *R-P3A-11: Connectivity between OEM and fleet. Network connection assessment*
Use case mapping	• *UC-P1-1: Asset location management* • *UC-P1-2: CHE location tracking* • *UC-P1-4: RTG-truck identification and authentication* • *UC-P2-1: Worker's health and safety assurance* • *UC-P3A-1: Fleet in-service emissions verification* • *UC-P3A-2: Vehicle non-conformance causes identification*

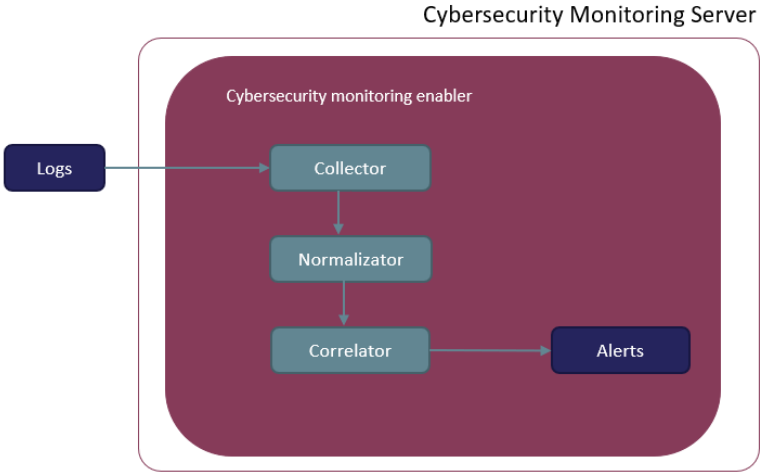

Figure 18. High-level structure of Cybersecurity monitoring enabler

Figure 18 describes the components of the solution, receiving as input logs from the infrastructure and produced cybersecurity alerts as output.

Components

Enabler component	***Collector***
Id	*T53E3-01 Cybersecurity monitoring enabler (collector, normalisation, event correlator)*
Description and main functionality	*The cybersecurity monitoring server will be responsible of collecting, processing, and analysing the incoming information from the infrastructure under study and will consolidate an output that will provide cybersecurity monitoring information related to different events.*
Target node/s	*Cloud, Edge*
Candidate technologies	*SYSLOG, RSYSLOG, NGLOG, ossec, wazuh*

Enabler component	*Normalisator*
Id	*T53E3-02*
Description and main functionality	*The normalisator shall receive structured data and will normalise and complete it.*
Target node/s	*Cloud, Edge*
Candidate technologies	*JSON, ossec, wazuh*

Enabler component	*Correlator*
Id	*T53E3-03*
Description and main functionality	*The correlator shall receive normalised data events and generate and alert when policy requires it.*
Target node/s	*Cloud, Edge*
Candidate technologies	*Simple Event Correlator SEC, ossec, wazuh*

B.12 - Cybersecurity: Cybersecurity monitoring agent enabler

Table 15. General information for Cybersecurity monitoring agent enabler

Enabler	*Cybersecurity monitoring agent enabler*
Id	*T53E4*
Owner and support	*S21Sec*
Description and main functionalities	*Perform functions of an endpoint detection and response system, monitoring and collecting activity from end points that could indicate a threat.* *Security agent runs at a host-level, combining anomaly and signature-based technologies to detect intrusions or software misuse. It can also be used to monitor user activities, assess system configuration, and detect vulnerabilities.* *Agent enabler communicates with Monitoring enabler. Agent module can run inside other enablers to report Monitoring enabler.*
Vertical, related capabilities and features	• *Security, Privacy and Trust*
Plane/s involved	*all*
Relation with other enablers	• *T53E3: Cybersecurity monitoring enabler* • *T53E1: Authorisation enabler* • *T53E2: Identity Manager enabler* • *T43E8: Long Term Storage enabler*
Requirements mapping	• *R-P1-1: CHE location services and operational status* • *R-P2-4: Continuous authentication for wristband - un-authorised use is detected and reported.* • *R-P3A-11: Connectivity between OEM and fleet - network connection assessment*
Use case mapping	• *UC-P1-1: Asset location management* • *UC-P1-2: CHE location tracking* • *UC-P1-4: RTG-truck identification and authentication* • *UC-P2-1: Worker's health and safety assurance* • *UC-P3A-1: Fleet in-service emissions verification* • *UC-P3A-2: Vehicle non-conformance causes identification*

Figure 19 describes the agent component of the of the Cybersecurity monitoring agent enabler.

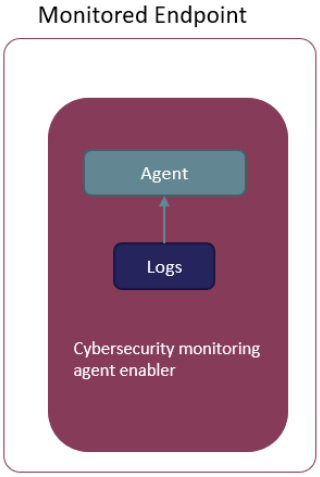

Figure 19. Cybersecurity monitoring with monitoring agent

Components

Enabler component	***Agent module***
Id	*T53E4-01*
Description and main functionality	*Agent is installed and running at hosts or node level and reporting to cybersecurity monitoring server.*
Target node/s	*Cloud, Edge*
Candidate technologies	*ossec-agent, wazuh-agent, other monitoring log based agent technologies*

B.13 - Cybersecurity: Identity manager enabler

Table 16. General information for Identity Manager enabler

Enabler	***Identity Manager enabler***
Id	*T53E2*
Owner and support	*S21Sec*
Description and main functionalities	*Identity Management enabler will store user credentials and data. Using OAuth2 protocol, it will offer a federated identification service where service requester and provider will be able to establish a trusted relation without previously knowing each other.* *When a requester (Access Entity) asks for a service (Access Controlled Entity), the provider will redirect the request to a third-party identity server, known by both parties, so the requester can identify itself and obtain a session token. The service provider will ask the identity server to validate the token and provide data about the requester.* *This way a secure identification process is completed without the service provider having received the requester credentials.* *This enabler works in collaboration with the Authorisation enabler. Interaction with Authorisation enabler components (PEP) are described in the corresponding section.*
Vertical, related capabilities and features	• *Security, Privacy and Trust*
Plane/s involved	• *Data Management Plane* • *Application and Services Plane*
Relation with other enablers	• *T53E1: Authorisation enabler* • *T43E8: Long Term Storage Enabler* • *T53E3: Cybersecurity monitoring enabler* • *T53E4: Cybersecurity monitoring agent enabler*
Requirements mapping	• *R-P1-10: CHE Identification* • *R-P1-6: Terminal data access* • *R-P1-13: CHEs authentication range*

	• *R-P2-11: Geofencing* • *R-P2-4: Continuous authentication for wristband* • *R-P3A-6: Active monitoring mode initiation by the OEM software engineer capability* • *R-P3A-7: Active monitoring mode initiation by the Aftersales service technician capability* • *R-P3A-8: Active monitoring mode initiation by the driver capability. (Auth and Authz REQUIRED)*
Use case mapping	• *UC-P1-1: Asset location management* • *UC-P2-2: Geofencing boundaries enforcement* • *UC-P2-1: Worker's health and safety assurance* • *UC-P3A-2: Vehicle non-conformance causes identification*

The diagram describes the interactions of the Identity Manager enabler with Authorisation Enabler and Policy Enforcement Point (PEP) component which will require interaction with Identity Manager.

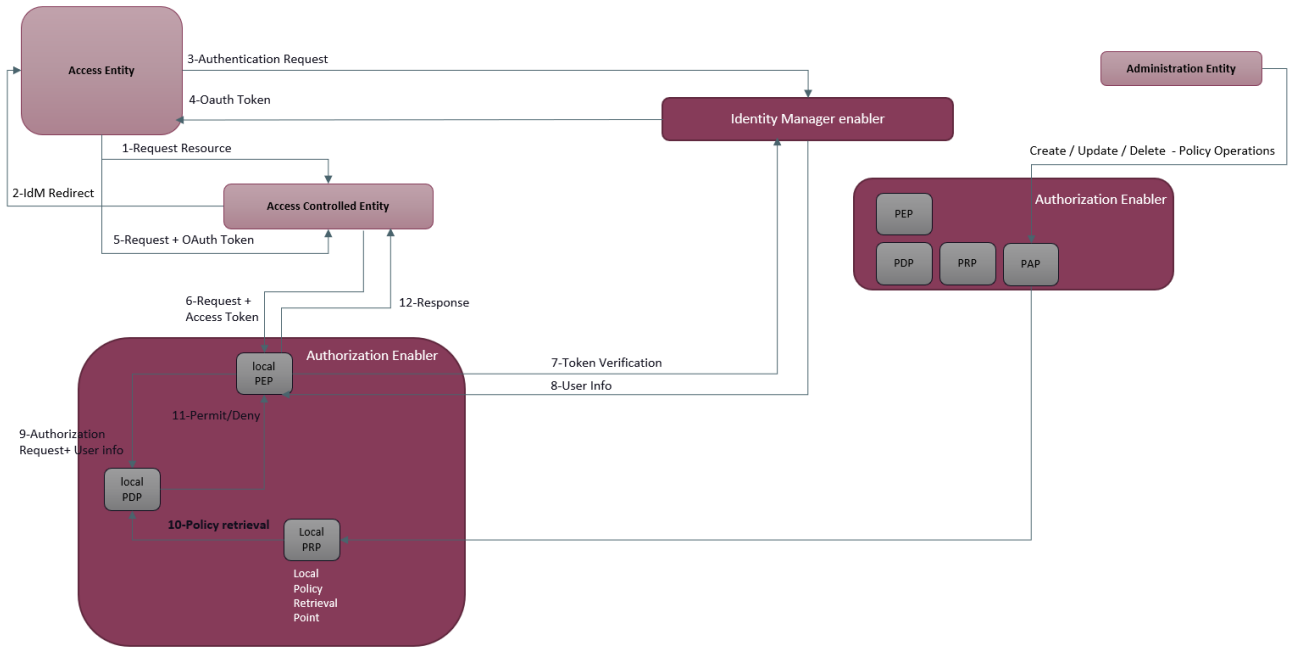

Figure 20. High-level structure of Identity Manager enabler

Components

Enabler component	***Identity Manager module***
Id	*T53E2-01*
Description and main functionality	*Contains the user or entities credentials store and data. It will offer a OpenId/OAuth2 interface.*
Target node/s	*Cloud*
Candidate technologies	*OAuth2 protocol.* *Identity manager Keyrock. https://fiware-idm.readthedocs.io/en/latest/* *Identity manager Keycloack. https://www.keycloak.org/*

B.14 - Cybersecurity: Authorisation enabler

Table 17. General information for Authorisation enabler

Enabler	***Authorisation enabler***
Id	*T53E3*
Owner and support	*S21Sec*
Description and main functionalities	*Authorisation server offers a decision-making service based on XACML policies. It has different modules that interact and can be deployed independently such as, PEP (Policy Decision Point), PAP (Policy Administration Point), PIP (Policy Information Point) and PDP (Policy Decision Point). Server will present a Rest interface and will respond to an external authorisation request. The decision may be accompanied by a set of action to be launched, in the form of external requests.* • *PAP. Edit and publish policy* • *PIP. Serve context information* • *PEP. Validate identity and request validation* • *PDP. Make decision and launch related actions* *Interactions with the Identity Manager enabler. Are described in the corresponding section.*
Vertical, related capabilities and features	• *Security, Privacy, and Trust*
Plane/s involved	• *Data Management Plane* • *Application and Services Plane*
Relation with other enablers	• *T53E1: Authorisation enabler* • *T43E8: Long Term Storage Enabler* • *T53E3: Cybersecurity monitoring enabler* • *T53E4: Cybersecurity monitoring agent enabler*
Requirements mapping	• *R-P1-10: CHE Identification* • *R-P1-6: Terminal data access* • *R-P1-13: CHEs authentication range* • *R-P2-11: Geofencing* • *R-P2-4: Continuous authentication for wristband* • *R-P3A-6: Active monitoring mode initiation by the OEM software engineer capability* • *R-P3A-7: Active monitoring mode initiation by the Aftersales service technician capability* • *R-P3A-8: Active monitoring mode initiation by the driver capability. (Auth and Authz REQUIRED)*
Use case mapping	• *UC-P1-1: Asset location management* • *UC-P2-1: Workers' health and safety assurance* • *UC-P2-2: Geofencing boundaries enforcement* • *UC-P3A-2: Vehicle non-conformance causes identification*

Figure 21 describes the interactions of the Authorisation enabler with the Identity Manager enabler and includes the access entity (user or system which perform the request and the access-controlled entity (as application on which access control is enforced).

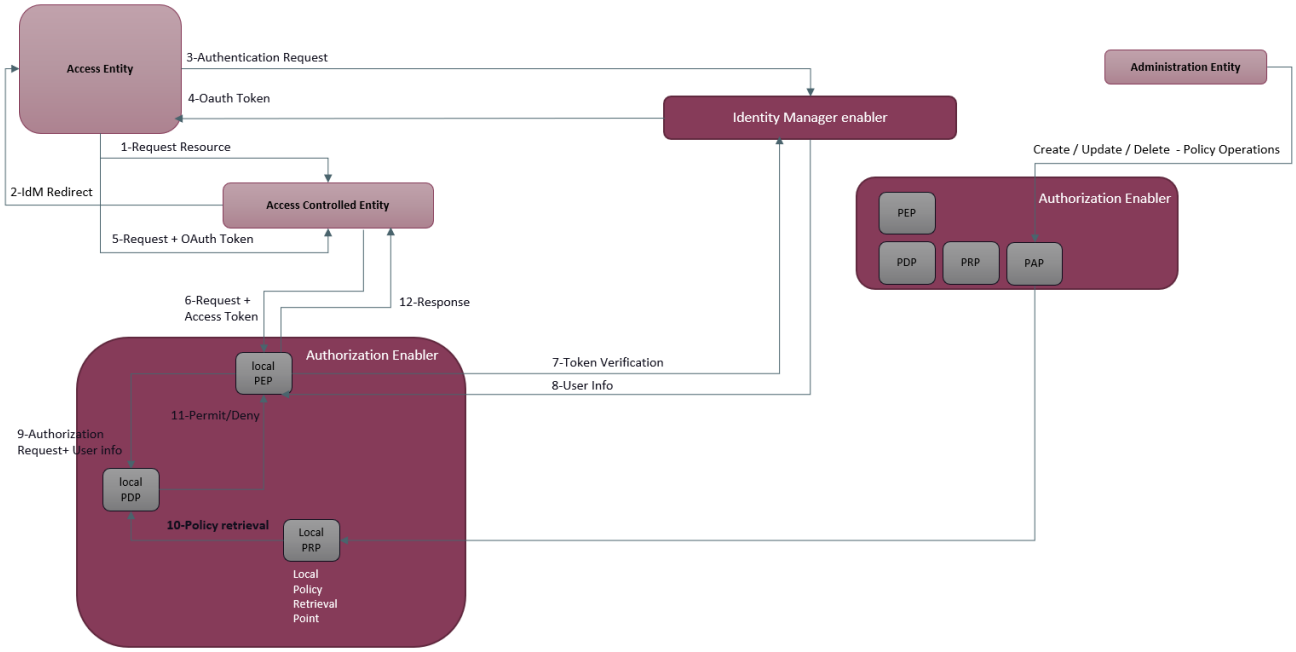

Figure 21. High-level structure of Authorisation enabler

Figure 22 describes the cloud-edge decoupling of Authorisation enable components.

Figure 22. Authorisation enabler – cloud and edge

Components

Enabler component	***PAP (federated) or local Policy Retrieval Point PRP***
Id	*T53E1-01*
Description and main functionality	*Policy Administration Point offers a web interface to edit the policy and publish it in XACML format to the location where the PDP will use it.* • *Present a web interface to build a policy* • *Transform to XACML and place it in the PDPs repository* • *Provide policy to PDP*
Target node/s	*Cloud*
Candidate technologies	*Docker, Rest, Java*

Enabler component	***PDP (local or federated)***
Id	*T53E1-02*
Description and main functionality	*Policy Decision Point is the module responsible of making the actual decision based on the context information compiled and the policy available.* • *Receive a who / what / where question in a Rest interface for validation* • *Obtain context information from external sources* • *Build a request compiling all the available data* • *Validate against the policy* • *Return Permit or Deny response* • *Launch related post decision actions*

Target node/s	*Edge Node, Gateway, Cloud*
Candidate technologies	*Docker, Rest, Java*

Enabler component	***PEP***
Id	*T53E1-03*
Description and main functionality	*Policy Enforcement Point is the responsible of requesting a decision to the PDP.* • *Validate identity against third party IS* • *Launch request to PDP*
Target node/s	*Edge Node, Gateway, Cloud*
Candidate technologies	*Docker, Rest, Java*

Enabler component	***PIP***
Id	*T53E1-04*
Description and main functionality	*Policy Information Point presents a Rest interface to publish context data to be used by the PDP when resolving the decision. It will also offer another interface (Rest and web) to add data.* • *Present an interface to add data to the storage* • *Serve context information in a Rest interface*
Target node/s	*Edge Node, Gateway, Cloud*
Candidate technologies	*Docker, Rest, Java*

B.15 - DLT-based: Logging and auditing enabler

Table 18. General information for Logging and auditing enabler

Enabler	***DLT Logging and Auditing***
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 providing digital evidence and resolve conflicts between stakeholders, when applicable. Billing mechanisms based on the logged actions may be developed depending on the requirements of the related use cases.*
Vertical, related capabilities and features	• *Security, Privacy, and Trust*
Plane/s involved	• *Data Management Plane* • *Edge and Device Plane* • *Applications and Services Plane*
Relation with other enablers	• *SELF14-03: Notifying and monitoring enabler* • *T53E3: Cybersecurity monitoring enabler* • *T53E4: Cybersecurity monitoring agent enabler* • *T53E2: Identity Manager enabler* • *T53E2: Authorisation enabler*
Requirements mapping	• *R-P1-1: CHE location services* • *R-P1-2: CHE location availability* • *R-P1-10: CHE identification* • *R-P2-1: Personal location tracking* • *R-P2-7: Monitoring the weather conditions at the construction site* • *R-P2-10: Motion Pattern Monitoring and Analysis* • *R-P2-11: Geofencing* • *R-P2-10: Motion Pattern Monitoring and Analysis* • *R-P2-13: Fall arrest detection* • *R-P2-3: Smart wristband for construction workers*

	• *R-P2-2: Construction plant location tracking* • *R-P2-9: Assessment of Personal Exposure to UV Radiation* • *R-P3A-6: Active monitoring mode initiation by the OEM software engineer capability* • *R-P3A-5: Data Storage* • *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*

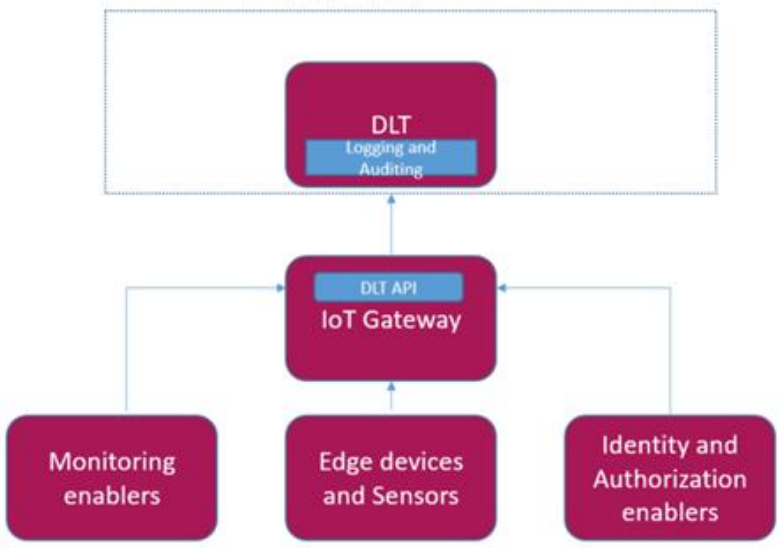

Figure 23. High-level structure of DLT Logging and Auditing enabler

Components

Enabler component	***DLT Logging and Auditing***
Id	*T54E1-01*
Description and main functionality	*DLT techniques will enhance the security in sharing data, enforce access control mechanisms, enhance data integrity verification, allow auditing, and support federated learning (to be conceptualised) with its decentralisation. In this specific enabler, they will use immutable properties of the DLT/Blockchain to store critical information in relation to the requirements of the involved use cases. This information can be used as a digital evidence and resolve conflicts.*
Target node/s	*Cloud*
Candidate technologies	*IDS (Blockchain-based) Clearing House, Hyperledger Fabric Chaincode (Smart Contracts), cryptographic techniques*

B.16 - DLT-based: Data integrity verification enabler

Table 19. General information for Data integrity verification enabler

Enabler	***Data integrity Verification enabler***
Id	*T54E2*
Owner and support	*CERTH, ICCS, Konecranes, S21SEC GES*
Description and main functionalities	*This enabler will provide DLT-based data integrity verification mechanisms that allow data consumers to verify the integrity of the exchanged data.*
Vertical, related capabilities and features	• *Security, Privacy, and Trust*
Plane/s involved	• *Data Management* • *Edge and Device* • *Applications and Services*
Relation with other enablers	• *T53E2: Authorisation enabler* • *Enablers used in Fleet in-service emissions verification (e.g.T43E7 Edge Data Broker)*
Requirements mapping	• *R-P3A-9: Edge Intelligence*
Use case mapping	• *UC-P3A-1: Fleet in-service emissions verification*

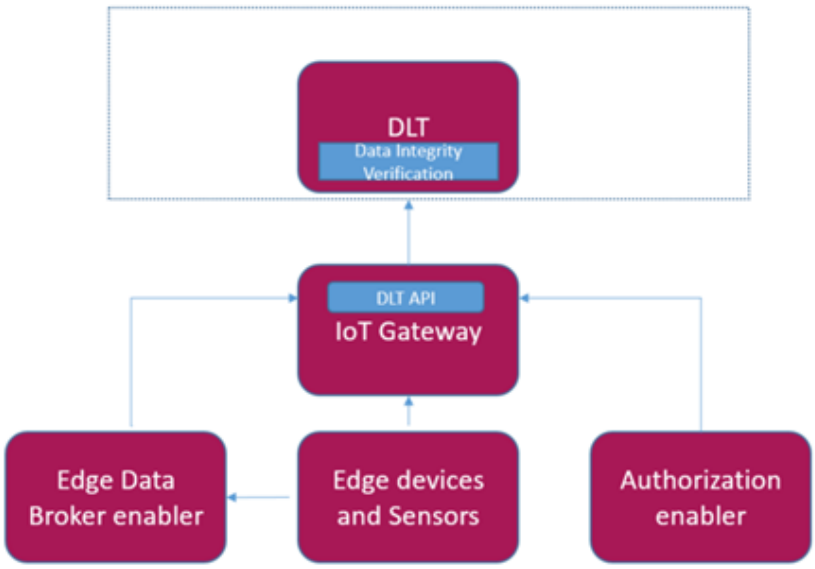

Figure 24. High-level structure of Data integrity verification enabler

Components

Enabler component	***DLT Data Integrity Verification***
Id	*T54E2-01*
Description and main functionality	*DLT techniques will enhance the security in sharing data, enforce access control mechanisms, enhance data integrity verification, allow auditing, and support federated learning (to be conceptualised) with its decentralisation. In this specific enabler, they will use immutable properties of the DLT/Blockchain to store data related to the requirements of the involved use cases. This data can be used for data integrity verification.*
Target node/s	*Cloud*
Candidate technologies	*Hyperledger Fabric Chaincode (Smart Contracts), cryptographic techniques (e.g., data hashing)*

B.17 - DLT-based: Distributed broker enabler

Table 20. General information for Distributed broker enabler

Enabler	***Distributed Broker service***
Id	*T54E3*
Owner and support	*CERTH, ICCS, Konecranes, S21SEC GES*
Description and main functionalities	*This enabler will provide secured data sharing mechanisms as regards the data exchange between different heterogeneous IoT devices belonging to various edge domains and/or between different enablers of the architecture.* *The mechanism will focus on providing a distributed broker service that will serve as a registry of all the domains and/or ASSIST-IoT enablers that act as data producers and/or data consumers. Indexing and querying services will facilitate the efficient retrievability of the stored (meta)data.* *This enabler may also act as a facilitator to the enablers that will provide semantic interoperability by providing searchable metadata of the interoperable domains complying with the FAIR principles.*
Vertical, related capabilities and features	• *Security, Privacy, and Trust*
Plane/s involved	*all*
Relation with other enablers	• *T43E3: Semantic Annotation Enabler* • *T43E2: Semantic Translation Enabler* • *T43E1: Semantic Repository Enabler* • *T43E7: Edge Data Broker* • *T52E2: FL Data Enabler* • *T53E2: Identity Manager enabler*
Requirements mapping	• *R-P3A-1: Monitored Data channels (TBD)* • *R-P2-15: BIM data models and interoperability compliance*
Use case mapping	• *UC-P3A-1: Fleet in-service emissions verification (TBD)* • *UC-P3A-2: Vehicle non-conformance causes identification (TBD)* • *UC-P3A-3: Updating the diagnostics methods pool (TBD)* • *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*

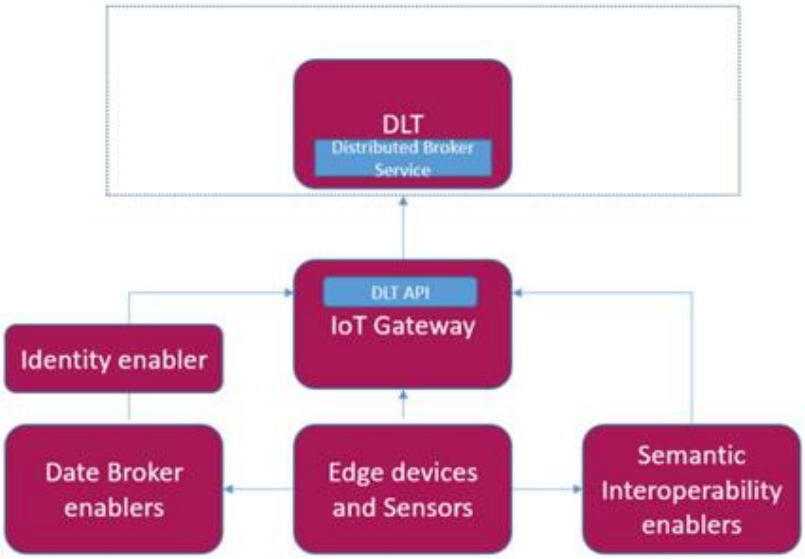

Figure 25. High-level structure of Distributed broker service enabler

Components

Enabler component	***DLT Broker Service***
Id	*T54E3-01*
Description and main functionality	*DLT techniques will enhance the security in sharing data, enforce access control mechanisms, enhance data integrity verification, allow auditing, and support federated learning (to be conceptualised) with its decentralisation. In this specific enabler, they will be used to ensure secure data sharing and facilitate (semantic) interoperability among the heterogeneous IoT devices belonging to various edge domains and/or of the ASSIST-IoT enablers.*
Target node/s	*Cloud*
Candidate technologies	*Hyperledger Fabric Chaincode (Smart Contracts), cryptographic techniques*

B.18 - DLT-based: DLT-based FL enabler

Table 21. General information for DLT-based FL enabler

Enabler	***DLT-based Federated Learning***
Id	*T54E4*
Owner and support	*CERTH,ICCS, Konecranes, S21SEC, GES*
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 avoiding single point of failures acting as a component to manage AI contextual information in an immutable form, and avoiding as well alteration to the data.*
Vertical, related capabilities and features	• *Security, Privacy, and Trust*
Plane/s involved	• *Device and Edge Plane* • *Data Management Plane*
Relation with other enablers	• *T43E7: Edge Data Broker* • *T52E2: FL Data enabler* • *T52E4: FL Privacy enabler*
Requirements mapping	• *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*

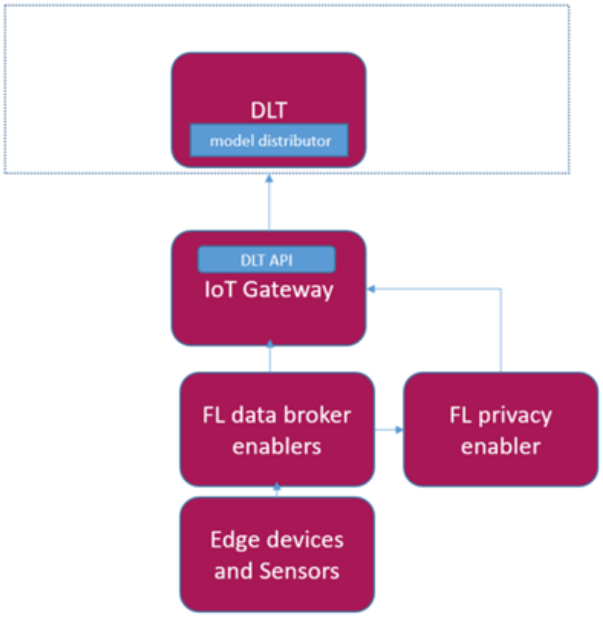

Figure 26. High-level structure of DLT-based FL enabler

Components

Enabler component	***DLT Model Distributor***
Id	*T54E4-01*
Description and main functionality	*DLT techniques will enhance the security in sharing data, enforce access control mechanisms, enhance data integrity verification, allow auditing, and support federated learning (to be conceptualised) with its decentralisation. In this specific enabler, they will use immutable properties of the DLT/Blockchain to store local model updates of Federated Learning Processes in a decentralised manner.*
Target node/s	*Cloud*
Candidate technologies	*Hyperledger Fabric clients - light nodes, Hyperledger Fabric Chaincode (Smart Contracts)*