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

D4.1 Initial Core Enablers Specification

Deliverable No.	D4.1	**Due Date**	31-JUL-2021
Type	Report	**Dissemination Level**	Public
Version	1.0	**WP**	WP4
Description	Initial specification of Smart IoT Devices, Edge Nodes and enablers of the horizontal planes of ASSIST-IoT.		

Copyright

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

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

Disclaimer

Authors

Name	Partner	e-mail
Alejandro Fornés	P01 UPV	alforlea@upv.es
Ignacio Lacalle	P01 UPV	iglaub@upv.es
Eduardo Garro	P02 PRO	egarro@prodevelop.es
Paweł Szmeja	P03 IBSPAN	pawel.szmeja@ibspan.waw.pl
Piotr Sowiński	P03 IBSPAN	piotr.sowinski@ibspan.waw.pl
Nikolaos Vrionis	P06 INFOLYSIS	nvrionis@infolysis.gr
Antonios Varkas	P06 INFOLYSIS	avarkas@infolysis.gr
Alex van den Heuvel	P09 NEWAYS	alex.van.den.heuvel@newayselectronics.com
Ron Schram	P09 NEWAYS	Ron.Schram@newayselectronics.com
Fotios Konstantinidis	P10 ICCS	fotios.konstantinidis@iccs.gr
Konstantinos Naskou	P10 ICCS	konstantinos.naskou@iccs.gr
Zbigniew Kopertowski	P15 OPL	Zbigniew.Kopertowski@orange.com

History

Date	Version	Change
1-Jun-2021	0.1	ToC presented
2-Jul-2021	0.2	First round of contributions completed. Subsections 4.2 and 4.3 with their respective annex content partially provided (enabler templates missing)
9-Jul-2021	0.3	Second round of contributions completed. Ready for internal review
19-Jul-2021	0.4	Integration of changes from IR (INFOLYSIS). Minor changes in Section 1. Acronyms completed. Tables and figures completed and homogenised.
26-Jul-2021	0.5	Integration of changes from IR (MOW, ICCS). Minor changes in Section 3.
31-Jul-2021	1.0	Final version submitted to EC

Key Data

Keywords	Enablers, Hardware specifications, Edge nodes, Smart IoT devices
Lead Editor	Alejandro Fornés (P01 - UPV)
Internal Reviewer(s)	Theoni Dounia (P06 - INFOLYSIS), Piotr Dymarski (P08 - MOW), Konstantinos Naskou (P10 - ICCS)

Executive Summary

This deliverable is written in the framework of WP4 – *Core 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 and hardware elements required to implement the different planes of the ASSIST-IoT architecture.

The realisation of the ASSIST-IoT architecture requires the design and development of specific elements, both software and hardware, that support the exposed functionalities and in the way they have been conceived: these are the Smart IoT Devices, the Edge Nodes and the enablers of the architecture. This document presents the specifications of both the hardware elements to be implemented under the scope of the Device and Edge Plane, and the enablers of the upper three planes of the ASSIST-IoT architecture (no software developments will be presented in this deliverable, just design decisions).

Regarding the hardware specifications of the equipment, general information regarding operation, storage and mechanical conditions are provided. Two Smart IoT Devices have been formalised: the *ASSIST-IoT localisation tag* and the *ASSIST-IoT fall arrest device*, which are devoted to localise assess and people as per requirement from the use cases. These devices consider Ultra-Wide Band (UWB) as the key technology for providing the localisation accuracy expected for the project. Additionally, the specifications of the Edge node include all the electronics and firmware required for serving the use cases of the project, including wired and wireless interfaces, processor and memory, power, etc.

As a reminder, an enabler is an abstraction term that represents a collection of components, running on hardware nodes, that communicate among them for delivering a particular functionality to the system. Enablers can only be interacted with via their exposed interfaces. By M9 of the project (July 2021), a total of 19 enablers have been identified and formalised:

- From the Smart Network and Control plane: Smart Orchestrator, SDN Controller, Auto-configurable network enabler, Traffic classification enabler, Multi-link enabler, SD-WAN enabler, WAN Optimisation enabler, and VPN enabler.
- From the Data Management plane: Long-term Storage enabler, Edge Data Broker, Semantic Repository enabler, Semantic Translator enabler, and Semantic Annotator enabler.
- From the Applications and Services plane: Tactile Dashboard, Business KPI reporting enabler, Performance and Usage Diagnosis enabler, OpenAPI Management enabler, Video Augmentation enabler, and MR enabler.

These enablers have been identified responding to requirements presented in D3.2, as well from architecture specifications. Being the first document of a series of three iterations, the information provided in this deliverable is susceptible to change, both in the number of enablers provided and regarding design decisions during enablers' implementation. The report summarises the different technologies that can be used/enhanced to realise those enablers, as well as includes enough schemas and diagrams to start the technical developments in WP4. This deliverable will be updated later on in the project (D4.2 by M18, D4.3 by M30), serving as the basis for the technical provision of the whole WP.

Table of contents

Table of contents 6
List of tables 7
List of figures 7
List of acronyms 9
1. About this document 12
1.1. Deliverable context 12
1.2. The rationale behind the structure 12
1.3. Outcomes of the deliverable 13
1.4. Lessons learnt 14
1.5. Deviation and corrective actions 14
2. Introduction 16
3. Devices specifications 18
3.1. General specifications 18
3.2. Smart IoT devices 18
3.2.1. ASSIST-IoT localisation tag 19
3.2.2. ASSIST-IoT fall arrest device 19
3.3. Edge node 20
3.3.1. Edge node electronics 21
3.3.2. Edge node firmware 24
4. Initial horizontal enablers specification 25
4.1. Smart Network and Control enablers 26
4.1.1. Enablers' descriptions summary 27
4.2. Data Management enablers 29
4.2.1. Enablers' descriptions summary 30
4.3. Application and Services enablers 31
4.3.1. Enablers' description summary 32
5. Future Work 34
Annex A - Localisation 35
Annex B - Enablers templates 37

List of tables

Table 1. General information of the enabler 25
Table 2. Endpoints information of an enabler 26
Table 3. Specific information of an enabler component 26
Table 4. General information of the Smart Orchestrator enabler 37
Table 5. General information of the SDN Controller 39
Table 6. General information of the Auto-configurable Network enabler 42
Table 7. General information of the Traffic Classification enabler 44
Table 8. General information of the Multi-link enabler 46
Table 9. General information of the SD-WAN enabler 49
Table 10. General information of the WAN Acceleration enabler 52
Table 11. General information of the VPN enabler 53
Table 12. General information of the Semantic Repository enabler 55
Table 13. General information of the Semantic Translation enabler 58
Table 14. General information of the Semantic Annotation enabler 61
Table 15. General information of the Edge data broker 63
Table 16. General information of the Long-term Data Storage enabler 65
Table 17. General information of the Tactile Dashboard enabler 69
Table 18. General information of the Business KPI reporting enabler 70
Table 19. General information of the Performance and usage diagnosis enabler (PUD) 72
Table 20. General information of the OpenAPI management enabler 74
Table 21. General information of the Video augmentation enabler 76
Table 22. General information of the MR enabler 77

List of figures

Figure 1. Planes and Verticals of the ASSIST-IoT architecture 16
Figure 2. ASSIST-IoT enablers and hardware elements formalised 17
Figure 3. Block schematic diagram of the ASSIST-IoT localisation tag 19
Figure 4. Block schematic diagram of the ASSIST-IoT fall arrest device 20
Figure 5. General node functions 21
Figure 6. Block schematic diagram of the Edge node electronics. 22
Figure 7. High-level diagram of an enabler 25
Figure 8. Smart Network and Control plane functional blocks and enablers 27
Figure 9. Data Management Plane functional blocks and enablers 29
Figure 10. Application and Services plane functional blocks and enablers. 31
Figure 11. Localisation options in the Smart safety of workers' pilot 36
Figure 12. Possible position of tags 36
Figure 13. High-level diagram of the Smart Orchestrator enabler 38
Figure 14. High-level diagram of the SDN Controller 40
Figure 15. High-level diagram of the Auto-configurable Network enabler 43
Figure 16. High-level diagram of the Traffic Classification enabler 44
Figure 17. High-level diagram of the Multi-link enabler 47
Figure 18. High-level diagram of the SD-WAN enabler 50
Figure 19. High-level diagram of the WAN Acceleration enabler 52
Figure 20. High-level diagram of the VPN enabler 54
Figure 21. High-level diagram of the Semantic Repository enabler 56
Figure 22. High-level diagram of the Semantic Translation enabler 59
Figure 23. High-level diagram of the Semantic Annotation enabler 62
Figure 24. High-level diagram of the Edge data broker 63
Figure 25. High-level diagram of the Long-term Data Storage enabler 66

Figure 26. High-level diagram of the Tactile Dashboard enabler 69
Figure 27. High-level diagram of the Business KPI reporting enabler 71
Figure 28. High-level diagram of the Performance and usage diagnosis enabler (PUD) 73
Figure 29. High-level diagram of the OpenAPI management enabler 75
Figure 30. High-level diagram of the Video augmentation enabler 76
Figure 31. High-level diagram of the MR enabler 78

List of acronyms

Acronym	Explanation
AI	Artificial Intelligence
AIOTI	The Alliance for the Internet of Things Innovation
AoA	Angle of Arrival
AoD	Angle of Departure
API	Application Programming Interface
AR	Augmented Reality
ARM	Advanced RISC Machines (related to architecture of processors)
BLE	Bluetooth Low Energy
CAN	Controller Area Network
CHE	Container Handling Equipment
CLI	Command Line Interface
CNF	Cloud-native Network Function
CNI	Container Network Interface
CSV	Comma Separated Value
DC	Direct Current
DDR4 SDRAM	Double Data Rate 4 Synchronous Dynamic Random-Access Memory
DGPS	Differential Global Positioning System
DLT	Distributed Ledger Technology
E2E	End to End
EIA	Electronic Industries Association or Electronics Industries Alliance
eMMC	Embedded MultiMediaCard
ENI	Experiential Networked Intelligence
ETSI	European Telecommunications Standards Institute
exFAT	Extensible File Allocation Table
gNMI	gRPC Network Management Interface
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GUI	Graphical User Interface
HA	High Availability
HAL	Hardware Abstraction Layer
HMD	Head-Mounted Device
HRP	High Repetition Pulse
HTML	HyperText Markup Language

HTTP	HyperText Transfer Protocol
I/O	Input/Output
I2C	Inter-Integrated Circuit
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IMU	Inertial Measurement Unit
IoT	Internet of Things
JSON	JavaScript Object Notation
JTAG	Joint Test Action Group (related to test interface)
KPI	Key Performance Indicator
LED	Light Emitting Diode
LIDAR	Light Detection and Ranging
LTS	Long-Term Storage
LTSE	Long-Term Storage Enabler
MANO	Management and Orchestration
ML	Machine Learning
MPLS	Multiprotocol Label Switching
MPTCP	MultiPath TCP
MQTT	MQ Telemetry Transport
MR	Mixed Reality
NFV	Network Function Virtualisation
NFVI	Network Function Virtualisation Infrastructure
NFVO	Network Function Virtualisation Orchestrator
NGIoT	Next-Generation Internet of Things
noSQL	Not Only Structured Query Language
NS	Network Service
NSD	Network Service Descriptor
NXP	Next eXPerience (related to a family of processors)
OAM	Operations, Administration and Management (related to network traffic)
OEM	Original Equipment Manufacturer
ONOS	Open Network Operating System
OS	Operating System
OSM	Open Source MANO
PPP	Precise Point Positioning
PUD	Performance and Usage Diagnosis
QoS	Quality of Service

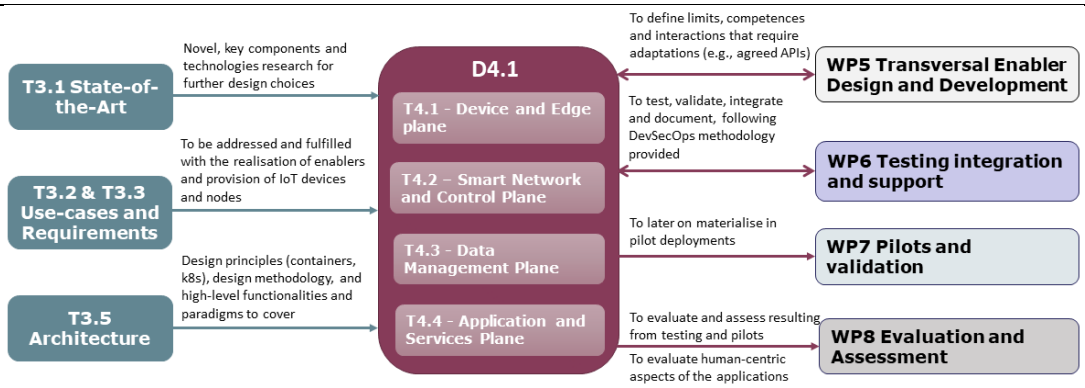
RADAR	Radio Detection and Ranging
RAT	Radio Access Technology
RDF	Resource Description Framework
REST	REpresentational State Transfer
RS	Recommended Standard
RTG	Rubber-Tyred Gantry (crane)
SD	Secure Digital
SDN	Software-Defined Networking
SD-WAN	Software-Defined Wide Area Network
SMARC	Smart Mobility ARChitecture
SotA	State-of-the-Art
SPDIF	Sony/Philips Digital Interface Format
SPI	Serial Peripheral Interface
SQL	Structured Query Language
SSH	Secure Shell
TIA	Telecommunications Industry Association
UI	User Interface
URL	Uniform Resource Locator
USB	Universal Serial Bus
UWB	Ultra-Wide Band
VIM	Virtualised Infrastructure Manager
VNF	Virtualised Network Function
VNFD	Virtualised Network Function Descriptor
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
WAN	Wide Area Network
WebRTC	Web Real-Time Communication
WiFi	Wireless Fidelity
WP	Work Package
WS02	Web Services Oxygenated
XML	Extensible Markup Language

1. About this document

The main goal of this deliverable is **to provide the specifications of the horizontal enablers** that are going to be developed under the scope of WP4. These enablers are the cornerstone of the project, since they will enable the deployment of an ASSIST-IoT architecture in a particular environment, allowing its further evaluation within the pilots involved in the project. Being the only document that gathers the outcomes of WP4, it will also include the specifications of the Smart IoT Devices and Edge Nodes that are to be provided for the scope of the project, despite not being enablers as such.

It should be highlighted that this deliverable corresponds to the first document of a series of three iterations, and therefore its content will be expanded and adapted as the project evolves. The rationale behind the iterative nature of its development is based on 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	**O2:** D4.1 presents the specifications of the enablers of the Network's plane (some of them, NFVs), including the auto-configurable enabler which will apply the models to improve the network performance. **O3:** Specifications of enablers focused on data (semantics, broker, storage) are provided. **O4:** Enablers from different planes, supported by AI (Artificial Intelligence) models, are described. **O5:** Human-centric interfaces for the use cases are presented.
Work plan	
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, it is also central part of *MS6 – Software structure finished.*
Deliverables	This deliverable receives inputs from D3.1 (State-of-the-art), D3.2 (requirements and use-cases) and D3.5 (architecture definition). 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 four sections and two annexes. It starts with an introduction followed by a section (Section 3) dedicated to the specifications of Smart IoT devices and Edge Nodes (which is complemented by

Annex A - , devoted to Smart IoT Devices for localisation). Afterwards, Section 4 is devoted to present the enablers' specifications. However, to facilitate the readers' comprehension, the corresponding templates with their initial specification (composed of a set of tables and diagrams) have been moved to Annex B - at the end of the document, leaving in this section the high-level scope of the horizontal Planes and just the description and main functionalities provided by the identified enablers. Last but not least, since this is the initial document of the deliverable series, the last section of this document concludes with a summary of the future work that will be carried out in the work package and is to be included in the next iterations of the deliverable.

1.3. Outcomes of the deliverable

The specifications of the hardware elements to be implemented under the scope of the Device and Edge Plane (Edge nodes and Smart IoT Devices), and the enablers of the upper three planes of the ASSIST-IoT architecture have been formalised. Regarding the hardware specifications, general ones for the Edge nodes and the Devices, such as operating, storage and mechanical conditions, are provided. On the one hand two **Smart IoT Devices** have been formalised: **the ASSIST-IoT localisation tag and the ASSIST-IoT fall arrest device**, which are devoted to assets localisation as per requirement from the use cases. The former aims at localising people or objects by means of UWB technology. From the use cases described in D3.2, it will be used in different use cases such as UC-P1-1 (Asset location management), UC-P1-2 (CHE location tracking), UC-P2-2 (Geofencing boundaries enforcement), and UC-P2-3 (Danger zone restrictions enforcement). Regarding the fall arrest device, besides localising the person, it incorporates a fall arrest detector, which is used by construction workers while working at height. It will be used specially for UC-P2-5 (Near-miss fall from height detection). On the other hand, the specifications of the **Edge node** include all the electronics and firmware required for serving the use cases of the project, including wired and wireless interfaces, processor and memory, power, etc.

A set of **enablers** has been **formalised** in this deliverable (information is mostly provided in Annex B - for readers' convenience). Formalisation includes the provided functionality, requirements and use cases mapping, components that comprise each enabler and, for some of them, the endpoints (at different degree of detail). With regards to the **Smart Network and Control plane**, eight enablers have been formalised: (1) the Smart Orchestrator enabler, for orchestrating and managing the virtualised network functions that will be deployed over the Kubernetes infrastructure; (2) the SDN Controller, based on ONOS, that will manage the control plane and will prioritise traffic in the network; (3) the Auto-configurable network enabler, that will adapt the policies of the SDN Controllers for ensuring QoS of applications based on specialised AI methods; (4) Traffic classification enabler, in charge of classifying traffic according to different classes (video streaming, traffic data, VoIP, etc.); (5) Multi-link enabler, which will allow reliable communication among different radio access technologies; (6) SD-WAN enabler, in charge of facilitating the private connection between different sites, forming a secure WAN with application QoS support; (7) WAN Optimisation enabler, which will consist of a set of components to improve the performance of WAN connections; and (8) VPN enabler, that will facilitate the connection of a Device from another network to the one of the site.

The **Data Management plane** considers all common data-related functions and services. The enablers are classified in two functional blocks: Data Governance, including (1) Long-term Storage enabler, that serves a secure and resilient storage, offering different storage sizes, individual storage space for other enablers; and (2) Edge Data Broker, in charge of enabling the efficient management of data demand supply from/to the Edge nodes, optimally distributing data where it is needed for application, services and further analysis; and then the Semantic enablers, which consists of (3) the Semantic repository enabler, which offers a "Nexus" for data models and ontologies; (4) the Semantic Translator enabler, that provides a configurable service to change the contents of semantically annotated data in accordance with translation rules; and lastly the (5) Semantic Annotator enabler, which offers a syntactic transformation service that annotates data in various formats and lifts it into RDF.

Lastly, six enablers have been defined under the scope of the **Applications and Services plane**. These enablers are: (1) the Tactile Dashboard, in charge of representing data from the use cases, through meaningful combined visualisations in real time. It will also contain the user interfaces for administrative instantiation and configuration of enablers and services; (2) the Business KPI reporting enabler, which facilitates the visualisation and combination of charts, tables, maps and other visualisation graphs for the desired KPIs and metrics; (3) the Performance and Usage Diagnosis enabler, that will collect performance metrics from monitored targets for

acting over them (manually or via another dedicated enablers); (4) the OpenAPI Management enabler, which will facilitate publishing all enablers APIs under a unified access, also for external users; (5) the Video Augmentation enabler, which will carry object recognition capabilities offered via computer vision; and (6) the MR enabler, that will facilitate the visualisation of data through head-mounted MR devices.

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. Additionally, the specifications of the hardware equipment (Edge nodes characteristics, Smart IoT Devices) that will be needed for the realisation of the use cases have been formalised as well. From all this work, the following insights have been extracted:

- UWB is the technology for implementing Smart Devices for localisation that better suits the requirements of the use cases of the project, mostly because of its accuracy and due to its robustness against interferences.
- Most of the enablers designed are generic and loosely-coupled to the use cases. This means that they can be easily leveraged in different verticals or environments. However, flexibility entails integration effort. For instance, the enablers of the Smart Network and Control plane require adaptation to the site's topology and needs (VPNs, WANs, network policies, etc.), whereas those from the Data Management plane need ontologies, data models and data collectors to transform, share and store data.
- Besides, specific enablers such as multi-link, video augmentation and MR, address direct requirements from the use cases. The will be developed in a way that could be used in other environments that could benefit from the provided functionalities
- Discovering all the interactions that will be needed between enablers is not a trivial task: specific endpoints, code modifications, addition of a component, etc. must be analysed, and in some cases, they may not be discovered until actual developments/implementation. This analysis has to be pushed forward to avoid work duplicities.

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:

- No enablers have been defined for the Device and Edge plane. Although some enablers from this plane have been discussed, work has been focused mostly towards the hardware equipment needed for the use cases. According to the ASSIST-IoT architecture, these enablers should facilitate or increase analytics, AI and communication capabilities. Now that the hardware specifications have been completed, these enablers will be formalised in the upcoming months.
- Some additional enablers might be needed, specifically for the Smart Network and Control Plane. It is still to be defined how particular networking aspects, such as interaction with Container Network Interface (CNI) Kubernetes plugins, will be tackled by the Smart Orchestrator. This is, whether this enabler would handle these features via specific configuration options over the deployed plugins, or by means of dedicated, complementary enablers. A dedicated analysis will be carried out under the scope of task 4.2.
- The DLT Communication enabler from the Data Management was expected to be included in this deliverable. It was intended to deliver blockchain distributed across many nodes, to provide data integrity verification, as well as communication auditing. However, its functionality was too similar to some of the enablers provided in WP5, and seemed to span across different planes, therefore it was removed from this iteration of the deliverable, at least until its scope is clearly defined.
- Last, all enablers were expected to reach the same level of maturity (in its definition) by M9. Meaning, having a clear description of their functionalities, listing the components that compose them, including an analysis of candidate technologies to leverage during their implementation, etc. in order to achieve

an equal level of readiness for technical development. However different degrees of completion have been experienced. This, in any case, is not considered a deviation on the execution, as the team considers an uneven distribution as the usual progress of the WP. There is the commitment to reach a steady state on the descriptions of the enablers by M12, where the first Open Call of ASSIST-IoT will be launched.

2. Introduction

The ASSIST-IoT architecture is structured following a multidimensional approach composed of horizontal *Planes* and *Verticals*. As explained in D3.5, the Planes represent a classification of logical functions that fall under the scope of a particular domain, whereas Verticals target Next Generation IoT (NGIoT) properties that exist on different Planes, either independently or requiring cooperation of elements from different Planes. The ASSIST-IoT Planes include a set of functional blocks aiming at addressing current concerns of different sectors, such as logistics and industry, leveraging NGIoT ecosystems.

The ASSIST-IoT architecture is a conceptual, high-level architecture, and therefore, there is a gap between its design and an actual implementation in an ecosystem that can benefit from utilising NGIoT technologies. Aiming at understanding the real needs of stakeholders of different industrial sectors, a first study of requirements was realised and presented in D3.2. These requirements need to be addressed, always considering that ASSIST-IoT does not solely target the three industrial sectors studied in the project, and thus both the architecture and the technical solutions should be applicable in other sectors as well.

In ASSIST-IoT, functionalities are provided by means of *enablers*. An **enabler** is an abstraction term that represents a **collection of components** (generally software, but without precluding the possibility of having hardware ones), running on nodes, that work together for **delivering a particular functionality** to the system. They can be fully independent or may require the cooperation with other enablers to deliver their intended functionality. As explained in D3.5, software components will be containerised, following an ***encapsulation* paradigm** in which enablers can communicate between them only through dedicated exposed interfaces, avoiding the possibility of having direct communication between components of different enablers.

These enablers have been designed considering not only the identified requirements but also the State-of-the-Art (SotA) of the technologies involved in their realisation, that was presented in D3.1 and therefore it is not part of the present deliverable. In any case, it should be mentioned that some of the main ASSIST-IoT enablers and technical decisions come almost naturally from architecture specifications and are later reassured by the requirements.

This document focuses mainly on providing the **initial specification of enablers of the horizontal Planes**. In a nutshell, the ASSIST-IoT planes are the following ones:

- The Device and Edge plane, which contains the physical and tangible components of the interconnected ecosystem and describes the collection of functions that can be logically appointed to them.
- The Smart Network and Control Plane, which manages virtual and wireless aspects of network connectivity, following the SDN/NFV paradigm.
- The Data Management plane, that handles all functions related to a virtual shared data ecosystem, including but not limited to semantics, governance and storage.
- The Application and Services plane, which supports and provides human-centric enablers for end-user applications and services.

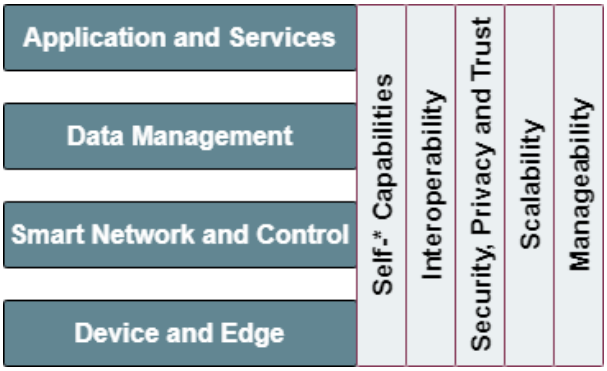


Figure 1. Planes and Verticals of the ASSIST-IoT architecture

Being the first document of the deliverable series, specifications can (and will) be expanded in the next two iterations that will be delivered later on in the project. It should be highlighted that the enablers, despite being the cornerstone of the ASSIST-IoT ecosystem and hence the main focus of WP4 work, are not the only outcome provided by this work package. The Smart IoT devices and Edge nodes, as well as other non-software artifacts such as ontologies and data processing rules, are also part of the same work package, and therefore their related outcomes will be presented either in this deliverable or in its future iterations. In particular, this deliverable will include the **initial specifications of Smart IoT devices and Edge nodes.** The hardware equipment and the enablers that have been formalised so far are depicted in Figure 2.

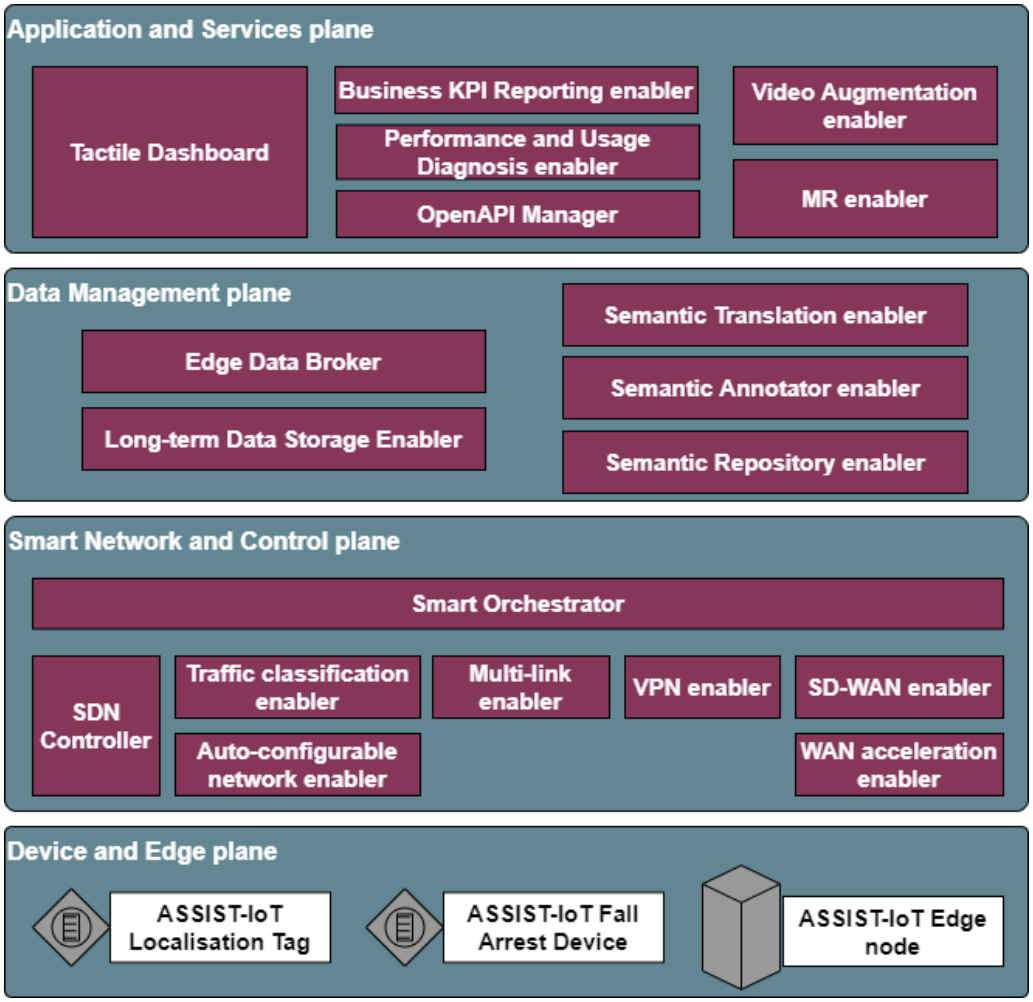


Figure 2. ASSIST-IoT enablers and hardware elements formalised

3. Devices specifications

The use cases formalised in D3.2 have been studied and used as input to determine the characteristics of the Edge node and the Smart IoT devices that will be developed as part of the ASSIST-IoT project. The conclusion is that for both the port automation pilot and the smart safety of workers' pilot, localisation is a key aspect that needs to be addressed by dedicated Smart IoT devices. In the port pilot, localisation is required for determine the position of Container Handling Equipment (CHEs) with an accuracy better than 1 meter, whereas in the smart safety's pilot it is needed for tracking people, geofencing, evacuation purposes and also to identify near-miss fall events (i.e., in case a worker is suspended by a fall arrest system, a rescue operation has to be initiated) in a construction site.

In order to fulfil these requirements, two Smart IoT devices will be developed to support localisation: the ***ASSIST-IoT localisation tag*** and the ***ASSIST-IoT fall arrest device***. They are described in Section 3.2 along with their particular specifications. UWB has been selected as key implementation technology for implementing localisation. More information about localisation and the rationale behind selecting UWB is presented in Annex A - . Apart from these devices, the Edge node that will be developed under the scope of ASSIST-IoT is presented in Section 3.3, with its hardware and software specifications. It will support/provide analytic, AI and communication capabilities, as well as localisation functionality, required by the use cases. The following section describes the general specifications that have to be met by the Edge node and the Smart IoT devices.

3.1. General specifications

The general specifications for the Smart IoT devices and Edge node are given in this section. These are common specifications that have to be fulfilled by the Edge node and the Smart IoT devices that will be developed within the ASSIST-IoT project.

Environmental operating conditions:

- Ambient temperature range for normal operation: -10 to 50 [ºC].
- Relative humidity range for normal operation: 20 to 80 [%] non-condensing.
- Test methods: IEC60068-2-2(Bd), IEC60068-2-1(Ad).

Environmental storage conditions:

- Storage temperature range: -20 to 70 [ºC].
- Storage relative humidity: 5 to 95 [%] non-condensing.

Mechanical conditions:

- Shock: 25 [g-force].
  - Shock test method according: IEC60068-2-29
- Random vibration:
  - Sine sweep: 0.5 [g-force].
  - Random vibration (Truck level): 0.37 [g_{rms}].
  - Sine dwell: 2-[g-force].
  - Test method according: IEC60068-2-6.

The enclosure of the Smart IoT devices and the Edge node must meet Ingress Protection code: IP53 or better (IP5x = Dust protected, IP x3 = protection against spraying water). The material and dimensions of the enclosure will be determined during implementation.

3.2. Smart IoT devices

Two Smart IoT Devices devoted to people localisation have been designed: The *ASSIST-IoT localisation tag* and the *ASSIST-IoT fall arrest device*. The former is used to localise the person and to warn that person in case it enters an unrestricted or dangerous area, while the latter acts as an emergency device. It incorporates as fall

arrest detector and an Inertial Measurement Unit (IMU) to automatically communicate a dangerous situation to the Edge node enabling the latter to act.

Additionally, for localisation, anchors and tags are used (see also Annex A -). The Edge node will be designed in a way that it can also be used as anchor. The localisation enabler receives the distance of a tag to the anchors on request of an enabler. The localisation enabler knows where the anchors are placed, meaning that it can determine the absolute position of the tag. The position of a tag can be displayed on a map from a construction side or on the floorplan of a building.

3.2.1. ASSIST-IoT localisation tag

The ASSIST-IoT localisation tag is a Smart IoT device used for people localisation. This device has tag functionality and it contains a buzzer and red LED to indicate to the person that he/she is in a restricted area. This device has been designed considering mainly the requirements for the smart safety of workers' pilot, however, many other use cases can be envisioned with it. Its block schematic diagram is given in the next figure:

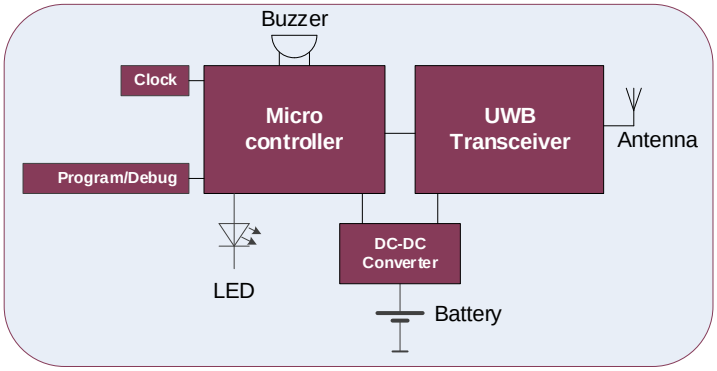


Figure 3. Block schematic diagram of the ASSIST-IoT localisation tag

A short description with the key specifications of the different parts of the ASSIST-IoT localisation tag is given:

- **UWB Transceiver:** Ultra-Wide Band (UWB) is used for localisation purposes. UWB is also used e.g., for communication with the anchors. Key specifications of UWB are:
  - Fully interoperable with IEEE 802.15.4 HRP UWB.
  - Ranging technique based on IEEE802.15.14z.
  - Position accuracy < 50 cm.
- **Battery:** The Smart IoT device is battery powered and can work stand-alone for more than 12 hours. The DC-DC converter creates a constant on-board voltage for the different devices.
- **Buzzer:** The Buzzer is used to give audio feedback to the user. The moment at which the buzzer is operated is determined by the localisation enabler.
- **LED:** The LED flashes to give a user visual feedback. The moment at which the LED flashes is determined by the localisation enabler.
- **Micro controller:** The micro controller controls the communication over UWB and controls the LED and buzzer when needed.
- **Clock:** The clock function generates the clock signal required by the micro controller. The exact clock circuitry and clock frequency will be determined during implementation.
- **Program/Debug:** This interface is used for development purposes.

3.2.2. ASSIST-IoT fall arrest device

The ASSIST-IoT fall arrest device is an emergency device. It is a localisation tag with a fall arrest sensor interface, an Inertial Measurement Unit (IMU) and a push button. The push button is used by the person wearing the tag, to indicate that this person is in an emergency situation and needs immediate help. The fall arrest device has a fall arrest detector interface which is used to connect the fall arrest sensor. This sensor detects a deployed fall-arrest system, indicating a person being suspended, and jointly with the IMU, the device determines if the

person has fallen or not. If an emergency situation is detected, a message is transmitted to the anchors immediately. An enabler can pick up this message and act accordingly. A block schematic diagram is given in the following figure:

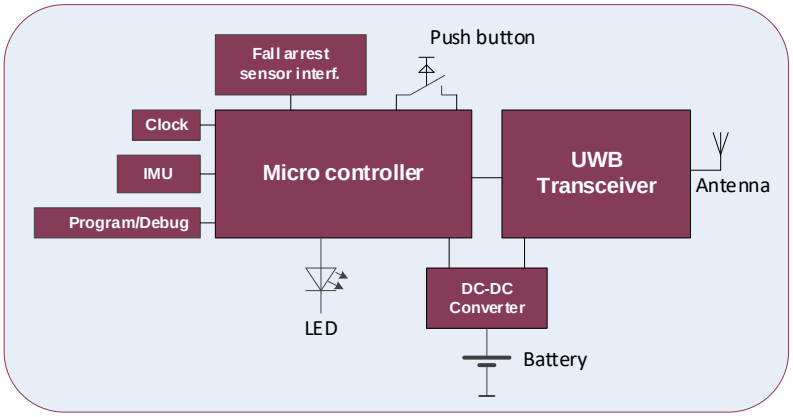


Figure 4. Block schematic diagram of the ASSIST-IoT fall arrest device

A short description of the different parts is given:

- **Fall arrest sensor interface:** It must be possible to connect a fall arrest detector to this device. The exact specifications for this interface are not known yet. The fall arrest sensor itself needs to be selected. This can be a switch or strain gauge device.
- **Inertia Measurement Unit (IMU):** The IMU will be a six-axis sensor, 3-axis to measure acceleration and 3-axis to measure angular rate. The microcontroller reads the measured values and determines if a dangerous situation occurs. If this is the case the microcontroller will send an emergency message with the UWB transceiver.
- **UWB Transceiver:** UWB is used for localisation purposes. UWB is also used e.g., for communication with the anchors. Key specifications of UWB are:
  - Fully interoperable with IEEE 802.15.4 HRP UWB.
  - Ranging technique based on IEEE802.15.14z.
  - Position accuracy < 50 cm.
- **Battery:** The Smart IoT device is battery powered and can work stand-alone for more than 12 hours. The DC-DC converter creates a constant on-board voltage for the different devices.
- **LED:** The LED is used to indicate the user that the device is operational. The LED will be green blinking when operational.
- **Micro controller:** The micro controller controls the communication over UWB and controls the LED and buzzer when needed.
- **Clock:** The clock function generates the clock signal required by the micro controller. The exact clock circuitry and clock frequency will be determined during implementation.
- **Program/Debug:** This interface is used for development purposes.

3.3. Edge node

In D3.5, the general Edge node functions were described (see Figure 5). To be able to implement the functional blocks, both hardware and firmware are needed. The different pilots have been studied and have been used as input to determine which hardware and firmware needs to be implemented. The compute power, memory, Physical Network Interface, Smart IoT device interfaces, etc., are part of the Edge node electronics and are described in Section 3.3.1. The hardware abstraction layer, operating system and the container runtime are described in Section o.

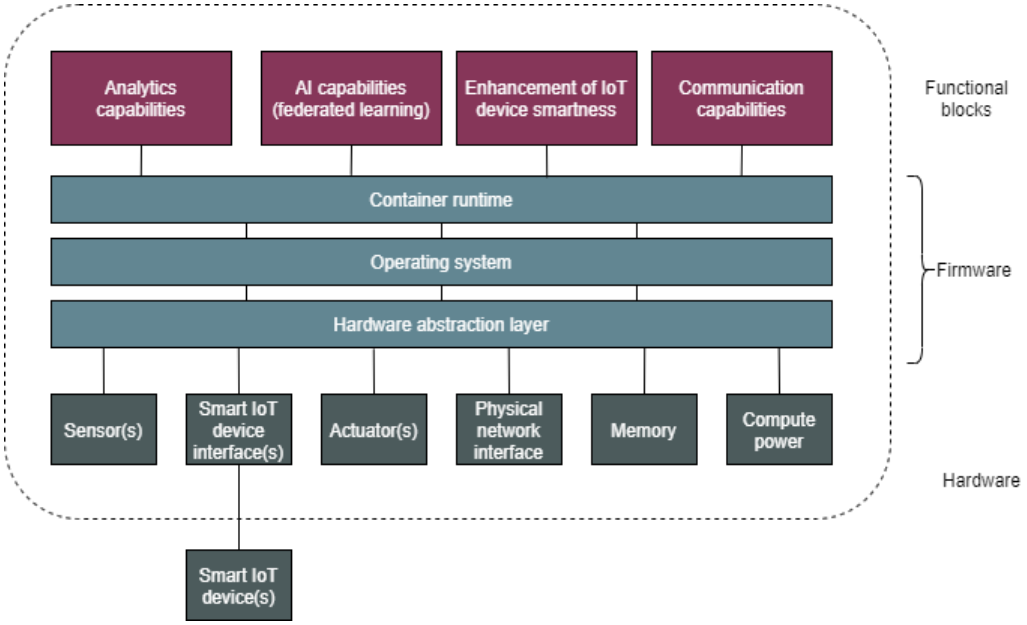


Figure 5. General node functions

3.3.1. Edge node electronics

As shown in Figure 5, the hardware of the Edge node consists of Compute power, Memory, Physical Network Interface, Smart IoT device interfaces, and dedicated sensors and actuators (if needed). A block diagram with the components that will be present to implement the hardware of the Edge node is detailed in Figure 6. Besides, the specifications of the different parts of the electronic components are also provided in this section.

Wired interfaces

The following wired interfaces are part of the Edge node electronics:

- **USB:** USB interfaces are used for many purposes; this can be to connect sensors but also to connect memory expansion. Key specifications are:
  - USB 2.0 Host interface with USB-A connector.
  - USB 3.0 host interface with USB-C connector.
- **2 x Ethernet:** Ethernet is used to connect to a local Ethernet network. Key specifications are:
  - 10/100/1000Mbps compatible.
  - Unshielded RJ45 connector.
- **CAN:** Within cars the CAN bus is used for communication between sensors/actuators and the ECU. By connecting to this CAN bus, the Edge node can gather the data, the software can decide which data will be processed and what will be communicated with the cloud. Key specifications are:
  - CAN 2.0 with female DB9 connector.
  - CAN FD with female DB9 connector.
- **RS485:** The RS485 physical layer can be used for many purposes e.g., to connect external sensors, I/O expansion or actuator control. RS-485 only specifies the electrical characteristics of the generator and the receiver: the physical layer. It does not specify or recommend any communication protocol. The communication protocol depends on the usage and will be defined when it is known what exactly is going to be connected. Key specifications:
  - Signal definition according to EIA RS-485.
  - 10 Mbps as maximum communication speed.
  - Can be used point-to-point and for bus structures.
  - Female DB9 connector.

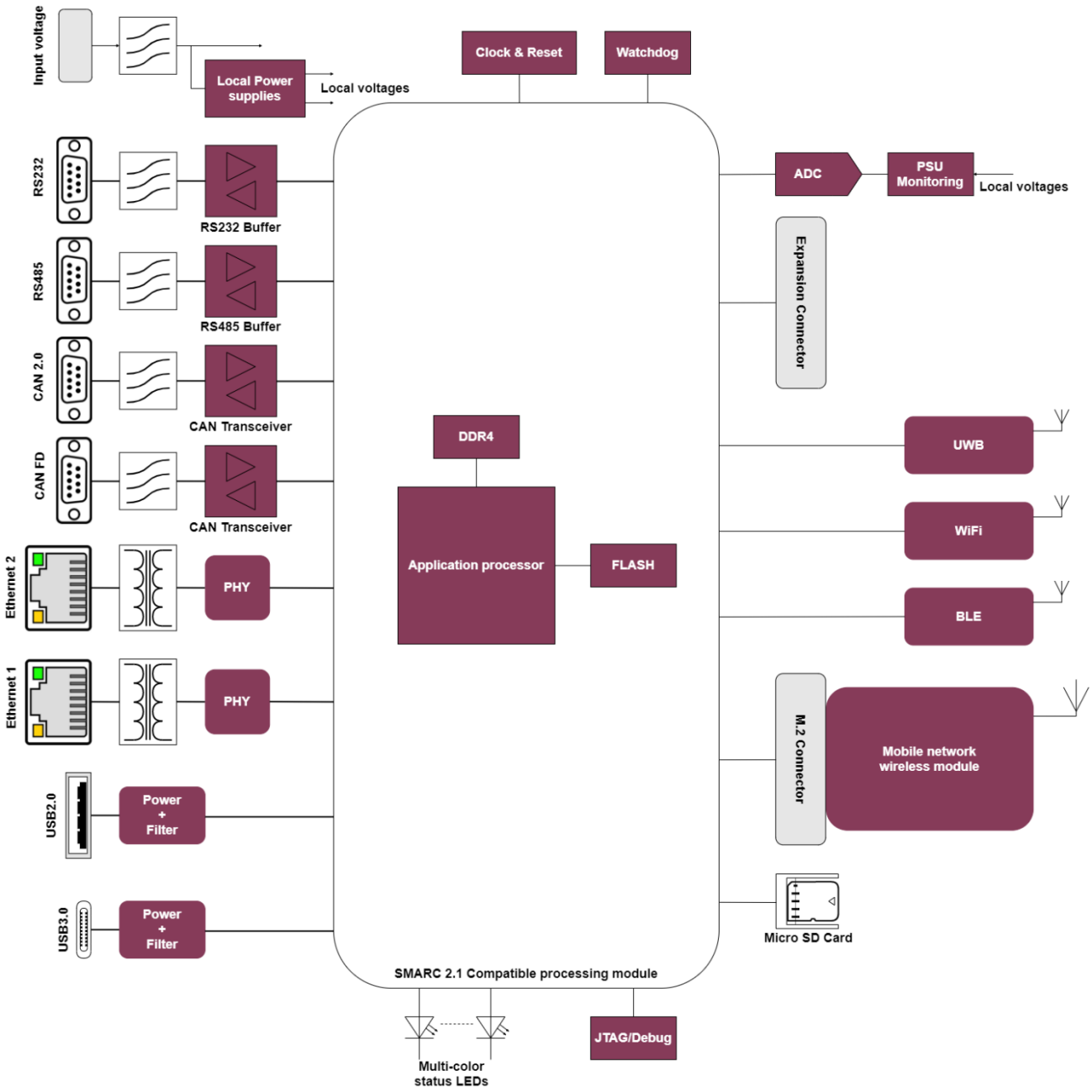


Figure 6. Block schematic diagram of the Edge node electronics.

- **RS232:** The RS232 physical layer can also be used for many purposes e.g., to connect external sensors, I/O expansion or actuator control. RS-232 only specifies the electrical characteristics of the generator and the receiver: the physical layer. It does not specify or recommend any communication protocol. The communication protocol depends on the usage and will be defined when it is known what exactly is going to be connected. Key specifications:
    - Signal definition according TIA/EIA RS-232-F.
    - 250kbps as maximum communication speed.
    - Can be used for point-to-point connections.
    - Female DB9 connector.

Wireless interfaces

The following wireless interfaces are part of the Edge node electronics:

- **UWB:** This interface is mainly used for localisation purposes but can also be used for communication. E.g., for communication with the tags, this enables the Edge node to act as an anchor. Key specifications of UWB are:
  - Fully interoperable with IEEE 802.15.4 HRP UWB.
  - Ranging technique based on IEEE802.15.14z.
  - Position accuracy < 50 cm.
- **WiFi:** WiFi can be used for connecting smart IoT devices or other wireless sensors. WiFi can also be used to connect to a network e.g., to create a communication channel with a server or a bac office in the cloud.
- **BLE:** Bluetooth Low Energy is used for low power short range communication with, smart IoT devices or other wireless sensors.
- **Mobile network wireless module:** A mobile network wireless module with M.2 interface will be used. M.2 is a standardised interface for internally mounted expansion cards. Depending on what is needed and what is available in the market a 4G module, a 5G module or a module that combines 4G/5G can be used.

Compute power

The compute power is implemented as follows:

- **Processing module:** A processing module with a SMARC 2.1 (Smart Mobility ARChitecture) interface will be used. SMARC 2 is a standardised small form factor computer module definition. Various modules are available which enables to scale compute power for a specific application. For the processing module the key specifications are listed below. These specifications are based on NXP's i.MX 8M Plus processor and are minimum specifications, hence the final chosen module will have equal or better specifications.
  - Dual ARM Cortex-A53 application processor up to 1.8GHz.
  - ARM Cortet-M7 real time processor @ 800MHz.
  - Machine learning accelerator.
  - 4GB DDR4 SDRAM memory.
  - 8GB eMMC FLASH.
- **Micro SD Card interface:** The Micro SD Card is used for non-volatile data storage. This interface will be according the SD3.0 specification with exFAT as filesystem.

Supportive functions

Besides the electronics to implement the functions itself, the following supportive functions are specified:

- **Clock & Reset:** The clock and reset function generates the clock signals required by the different parts of the Edge node. The exact clock circuitry and clock frequencies will be determined during implementation of the Edge node. The reset circuitry keeps the Edge node in reset until the input voltage and the generated local voltages are stable and within the required range. When a local supply voltage goes out of range, e.g. due to a fault, the Edge node will be reset. When the fault is removed the Edge node can come out of reset. As soon as the Edge node comes out of reset the boot process is started.
- **Watchdog:** The software of the Edge node stops kicking the watchdog when it has been crashed. After a timeout period the watchdog will reset the Edge node and the boot process will start. When the reason for crashing is removed the Edge node will start operating as expected.
- **Local power supplies:** The local power supplies generate the on-board required voltages from the input voltage. The specifications for the local power supplies are determined during implementation. The specifications for the input voltage are:
  - Input voltage: 12V ± 5%.
  - Ripple & noise 150mVpp.
  - Connector type: Panel mount power jack connector 2.1 x 5.5 mm.
- **PSU monitoring:** The local voltages are measured and can be read back by software. This is used for diagnostic purposes and is useful during production of the Edge node. When voltages are out of range, the

Edge node will stay in reset to prevent faulty or unexpected behaviour. The Edge node will also enter the reset state when one of the power supplies becomes out of range. The root cause could be that a hardware fault occurs during normal operation or that the Edge node is used outside the specified input voltage range.

- **Expansion connector:** The expansion connector is going to be implemented for future use. Interfaces like RS232, RS485, USB, I2C, SPI, SPDIF, I2S, etc., can be made available at this connector. During implementation it will be determined which interfaces will be made available at the expansion connector.
- **JTAG/Debug:** The JTAG/Debug interface is used for debug purposes during development of the Edge node and for test purposes during production. The exact specifications of this interface will be determined during implementation. This interface is not available for normal use.
- **Status LEDs:** These LEDs are used to indicate the status of the Edge node (e.g., power OK, Ethernet link status, etc). The number of LEDs and the accompanying function will be determined during implementation. Some LEDs will be visible at the outside of the enclosure to give the user status information, some LEDs will be inside the box and provide status feedback during implementation for debugging purposes.

3.3.2. Edge node firmware

As shown in Figure 5, the firmware of the Edge node consists of an Operating System (OS) which is built on top of a Hardware Abstraction Layer (HAL), a container runtime and in addition pre-installed software to support enablers will be used. This pre-installed software operates on top of the OS, next to the container runtime so custom containers can use this pre-installed software.

Key specifications of the Edge node firmware are given:

- **Operating System (OS):** Yocto[1], based on Linux, is used as OS. The Yocto Project is an open source collaboration project that helps developers creating custom Linux-based systems regardless of the hardware architecture. The project provides a flexible set of tools and a space where embedded developers worldwide can share technologies, software stacks, configurations, and best practices that can be used to create tailored Linux images for embedded and IOT devices, anywhere a customised Linux OS is needed.
- **Hardware Abstraction Layer (HAL):** The HAL consists of device driver as interface between the electronics and the OS. The Yocto project supports several kinds of peripherals and provides device drivers which implement hardware specific functionality for these peripherals. Besides, not supported peripherals of the Edge node will need own developed device drivers. These are also part of the HAL.
- **Configuration and initialisation:** The configuration and initialisation of the standard interfaces (Ethernet, Serial, etc.), SSH and a default user will be preconfigured on the Edge node, making the node fully functional and ready to run enablers on.
- **Container runtime:** For the container runtime, Docker is used. Docker is a set of platform-as-a-service products that use OS-level virtualisation to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. As the enablers will be implemented as containers, Docker will be preinstalled.
- **Pre-installed software:** It is expected that some software will be used by several enablers and at the same time, it must be possible to update the Edge node firmware. For this reason, the following supportive software will be pre-installed at the Edge node:
    - *Python:* Python is a general-purpose programming language that will be used by many enablers. It is used for web development, AI, machine learning, mobile application development, etc. As Python will be used on the Edge node, it will be preinstalled.
    - *Software update support:* Software update support is needed to be able to update the HAL, OS and specific device drivers e.g., to apply security patches, deploy new features or bug fixes. The update mechanism is to be determined e.g. “swupd” or “mender.io” can be used. The software update support is used to support the configuration enabler.

[1] https://www.yoctoproject.org/

4. Initial horizontal enablers specification

The specification of all the identified enablers is formalised following the ASSIST-IoT enabler template. The original template, and provided in D3.5, is composed of four sections: (1) a table with general information about the enabler, including its description, plane to which it belongs and the requirements that it addresses; (2) a basic diagram depicting the high-level 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 table, one for each of its components, with dedicated information about them.

A few considerations have to be considered about the information provided in this deliverable. First of all, the template has evolved, and therefore some slight differences are present (e.g., in the first table of the template, now the mapping with the use cases is included). Besides, because of being at early stages of WP4, some design choices have not been taken yet, and therefore (i) the list of endpoints (third section of the template) is preliminary (and in some cases, not provided), and (ii) some implementation aspects of their components in the last section of the templates will not be included (e.g., hardware and software requirements, and the row devoted to implementation technologies has been modified for this deliverable to "Candidate Technologies"). 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. More specifically, the delivered templates include:

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)*
Plane/s involved	*Horizontal plane or planes on which the enabler's features are delivered [Only one for horizontal enablers, transversal enabler intersect more than one]*
Relation with other enablers	*List of enablers (core or vertical) that interact with this one. Just list.*
Requirements mapping	*List of the IDs of the requirements addressed or considered. Attach a short rationale (1/2 sentences)*
Use case mapping	*List of the IDs of the use cases related to this enabler.*
Required components	*List of the names of the components that form this enabler*

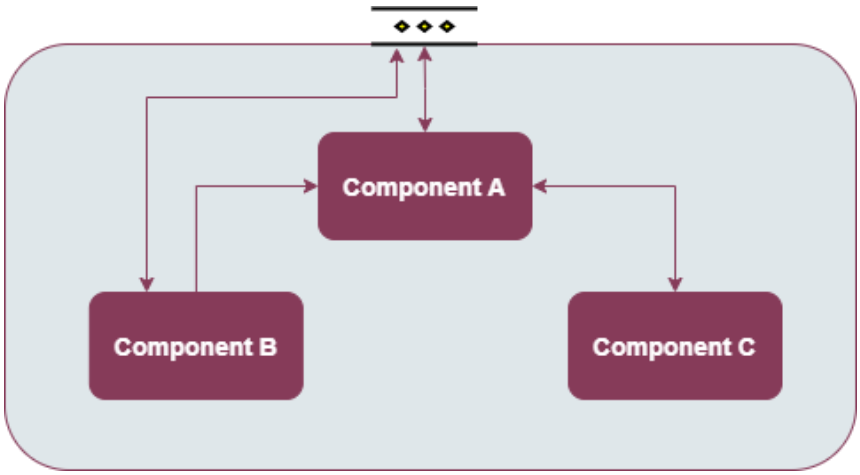


Figure 7. High-level diagram of an enabler

Table 2 describes the endpoints that will be offered by each enabler. As aforementioned, because of being in a very early stage, some enablers have a very basic table of endpoints described (hence, to be improved) or even it is not present. Besides, for each of the components that are part of a particular enabler, a table like the one presented in Table 3 is included.

Table 2. Endpoints information of an enabler

Method	URL	Payload (if needed)	Description	Response format
GET/POST/ PUT/DELETE	/{something}/…	Data needed to be included	Main functionality provided	In the future, a table should be provided for each method to show data format, response codes…

Table 3. Specific information of an enabler component

Enabler component	*Name of the enabler component*
id	*Short unique identifier. E.g., T4XEY_{Key_name/concept}*
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, they might be already "final choices" whereas for others it is under decision, but at least implementation options must be presented (can be something like "custom component in Python using Flask")*

4.1. Smart Network and Control enablers

The main objective of the Smart Network and Control plane is the realisation of a networking infrastructure for NGIoT ecosystems, facilitating the interconnection of both hardware (i.e., nodes and devices) and software (i.e., enablers) elements either from inside or outside the site's network, and the configuration of different networking aspects that provide the required characteristics for different use cases in terms of latency, bandwidth, Quality of Service (QoS), etc.

The architecture considered in the ASSIST-IoT architecture follows the softwarisation paradigm based on SDN/NFV[2] that has been evolved during the last decade in the telecommunication industry, as also exposed in the deliverable devoted to the State-of-the-Art (D3.1). It provides three main advantages with respect to former architectures:

- Large reduction of costs related to network equipment (most network functions, now virtualised, can be instantiated in general purpose equipment).
- Faster reconfiguration and flexibility, since the instantiation and configuration of network functions (and routing aspects) can be automated, also for elastic scaling of resources.
- Openness, facilitating the utilisation of open-source software while reducing the dependency of proprietary elements, either software or hardware.

In this paradigm, the NFV Management and Orchestration (MANO) and the SDN Controller are the main elements, being the former in charge of instantiating and managing the lifecycle of Virtualised Network Functions (VNFs), and the latter in controlling the packet forwarding, among other networking aspects, based on specified policies. In the ASSIST-IoT ecosystem we aim at leveraging this paradigm, but adapted to an industrial ecosystem in which containers (and Kubernetes as container orchestrator) are expected to be prevalent, instead of leveraging (or at least minimising the necessity of) virtual machines.

The enablers of the Smart Network and Control plane can be classified under four functional blocks, namely "Smart Orchestrator", "SDN Controller", "VNFs" and "Self-contained network", as it can be seen in Figure 8. The logical classification in different functional blocks is purely conceptual, not having any further impact neither in design nor in deployment stages. Most of these enablers will be instantiated and controlled by the Smart Orchestrator enabler, regardless of being their belonging functional block.

[2] M. S. Bonfim, K. L. Dias and S. Fernandes, "Integrated NFV/SDN Architectures: A Systematic Literature Review", ACM Comput. Surv. vol 51, 6 (2019).

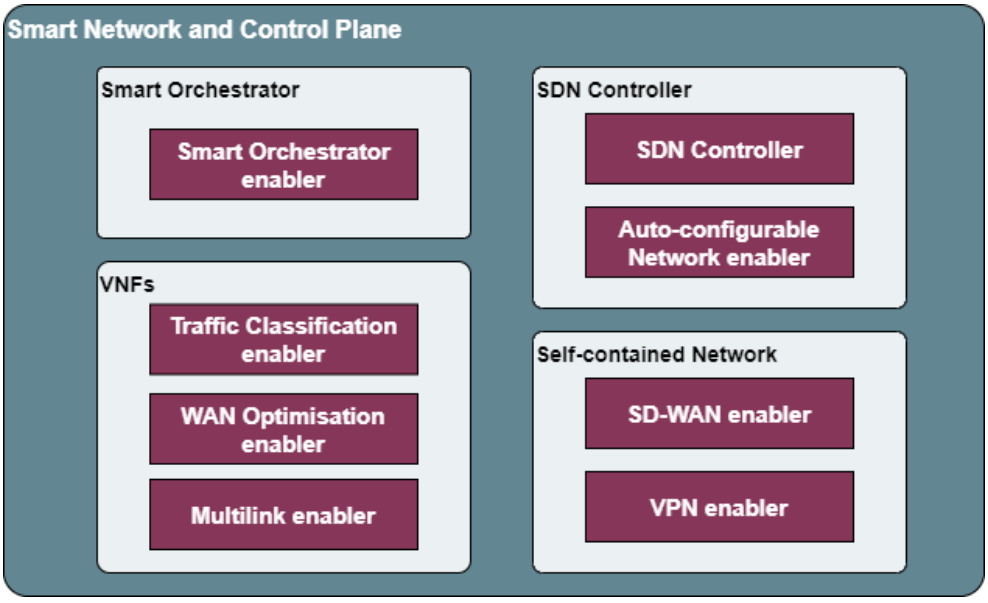


Figure 8. Smart Network and Control plane functional blocks and enablers

These enablers have been identified following the requirements defined in D3.2. It should be mentioned that the number and the scope of these enablers can be modified since requirements are still evolving, and therefore they should not be taken as close at this point. All the enablers will be related to requirements and use cases that they will address, but in summary they target demands from the use cases like:

- Ensuring specific minimum latencies, bandwidths and/or QoS.
- Network reliability to prevent absence or loss of connections.
- Provisioning of machine-to-machine communication or mesh networks, when required.
- Multi-link wireless network capabilities.
- Support for Kubernetes-based networking and its interaction with SDN (pending to be defined).

4.1.1. Enablers' descriptions summary

Smart Orchestrator

This enabler facilitates the interaction of User Interfaces (UIs) and other enablers with the main components of the MANO framework, namely the Network Function Virtualisation Orchestrator (NFVO) and the Kubernetes clusters, exposing only the required inherent functionalities. In particular, this enabler will control the whole lifecycle of Virtualised Network Functions (VNFs), from their instantiation to their termination, allowing their deployment in any k8s cluster available. If needed, adaptations for lightweight alternatives such as k3s will be studied.

This enabler is also in charge of performing additional functionalities, like the validation of submitted descriptors for Virtualised Network Functions (VNSD) and Network Services (NSD). Finally, it also acts as an abstraction layer, decoupling the underlying selected NFVO technology from the ASSIST-IoT ecosystem. In this way, in case the NFVO component is changed in the future, only this layer would need to be adapted without having to modify the interfaces of the rest of enablers that communicate with this one.

SDN Controller

The SDN Controller is the key element of an SDN-enabled network, being the software that takes over the responsibilities of the control plane from the hardware elements (switches mostly), including monitoring and management of packet flows. Although typically installed in a dedicated machine, its functionalities are intended to be provided following the ASSIST-IoT architecture based on enablers, either by adopting a distribution that matches it or by making the necessary adaptations to fulfil them. The main functionalities are related to network management, operation and maintenance, allowing topology management, network configuration, network control and network operations, among other features.

Auto-configurable network enabler

This enabler provides optimised network routing configuration capabilities to the SDN Controller of an ASSIST-IoT ecosystem. This enabler will consist of an application that consumes the northbound APIs of the SDN Controllers to generate a policy that improves the performance of one/many KPIs of the network (e.g., latency). The selection of the most suitable model depends on the use cases, and the objective of this enabler is to provide **three different strategies** (pending to define) to be compatible with the selected Controller.

Traffic classification enabler

The aim of this enabler is to classify network traffic into a number of application classes (video streaming, VoiP, Network control, best effort, OAM, etc.), making use of a AI/ML framework and dedicated algorithms. The traffic classification enabler can be seen as a service of the application layer of the general SDN architecture. To provide its functionalities, three stages are needed: (i) data processing and feature extraction, (ii) model selection training, and (iii) inference and results. For developing this particular enabler, we will rely on modules of the SDN Controller for providing the features of the data to perform the inference, but in case this information it is not enough the necessary components will be added.

The results provided by this enabler can be used for supporting network policy-making processes for ensuring QoS (application priority), optimising traffic, or enhancing security (therefore results must be consumed by other enablers).

Multi-link enabler

Multi-link wireless network capabilities provide the possibility of sending streams of data over different Radio Access Networks and different channels in each of them (for instance, regarding cellular, using more than 1 connection). Besides, it should provide reliability mechanisms: in case one channel is down, signal cannot be lost or at least it should be recovered almost in real time. The reliability mechanism will be integrated by means of redundancy rather than diversity (e.g., MPTCP), since it is better considering latency aspects, especially when switching.

Regarding redundancy, signal will be sent through two channels, of (at least) two different RATs, so in case one is down the other can take over. In case of video streaming, the redundant channel will transmit at lower resolution, and in case it has to take over this resolution will dynamically increase. Since this enabler comes from a requirement of the port automation pilot, the involved RATs will be cellular, WiFi and fluidmesh (starting with two of them). When the main link is lost and the second takes over, a third channel should be initiated for acting as the redundant one. This enabler will be expanded in the future to support not just multi-link connections for video but also for other classes of low-latency traffic.

SD-WAN enabler

The objective of this enabler is to provide access between nodes from different sites based on SD-WAN technology. This enabler will consist of three main virtualised elements, (i) an edge node in each one of the sites to be interconnected (SD-WAN edge), which will act as the endpoint between the site's network and the different access networks that connect them (MPLS, Internet, 4G, Satellite, etc.), creating/finalising the necessary tunnels; (ii) an SD-WAN controller, which includes, IP address management, and pushing down policies onto SD-WAN Edges, among other functionalities; and (iii) a configuration server, also known as WAN Service Configurator, that contains the application-based policies and functionalities (configuration of end-to-end WAN services, QoS/business optimisation, etc.).

Besides, the SD-WAN edge will consist of two main components, a virtual switch and monitoring agents. The monitoring agents will collect network metrics that will be processed in the WAN monitoring module, getting outcomes that in turn will feed the WAN Service Orchestrator for applying the optimal configurations (in tunnel selection and application prioritisation).

WAN Acceleration enabler

This enabler aims at increasing the efficiency of data transfer in Wide Area Network. This enabler will contain a set of independent, standalone VNFs with that purpose. These functions can be either chained (so data that requires of different techniques travels through the different functions) or selected for specific purposes. For now, three functions are expected to be provided: traffic shaping, data compression and data decompression.

VPN enabler

This enabler will facilitate the access to a node or device from a different network to the site's private network using a public network (e.g., the Internet) or a non-trusted private network. VPN enabler must work in High Availability (HA) mode, with VPN servers distributed in different nodes, and failover. As a first step, the site's network will be considered trusted, so VPNs will not be needed to connect nodes or devices that belong to it.

It should be highlighted that SD-WAN enabler will be the primary choice for connecting sites' networks while VPN will (primarily) connect particular external elements to the site's network since (i) VPN lacks both network and application-level performance optimisation, and (ii) it requires extensive manual effort to add different sites to the entire WAN.

4.2. Data Management enablers

The Data Management plane organises common data-related functions and services. To support highly dynamic and heterogeneous data flow pipelines the enablers on this plane are designed to be the building blocks of decentralised systems, where operations on data are performed exactly where and when they are needed. To this end, this plane delivers solutions to control the data traffic (edge data broker), process the data (semantic translation and annotation) and safely persist it (long-term storage), as required. The data transformation functions are designed to support static (i.e., bulk) data, as well as streaming data. In order to use consistent security mechanisms across the project, the data access control uses the functions delivered by the authentication and authorisation enablers from the Security, Privacy and Trust vertical (see deliverable D5.1, or its later revisions D5.2, D5.3).

The Data Management plane is conceptually divided into two functional blocks: Data Governance and Semantics (please, refer to individual enabler descriptions for detailed explanations).

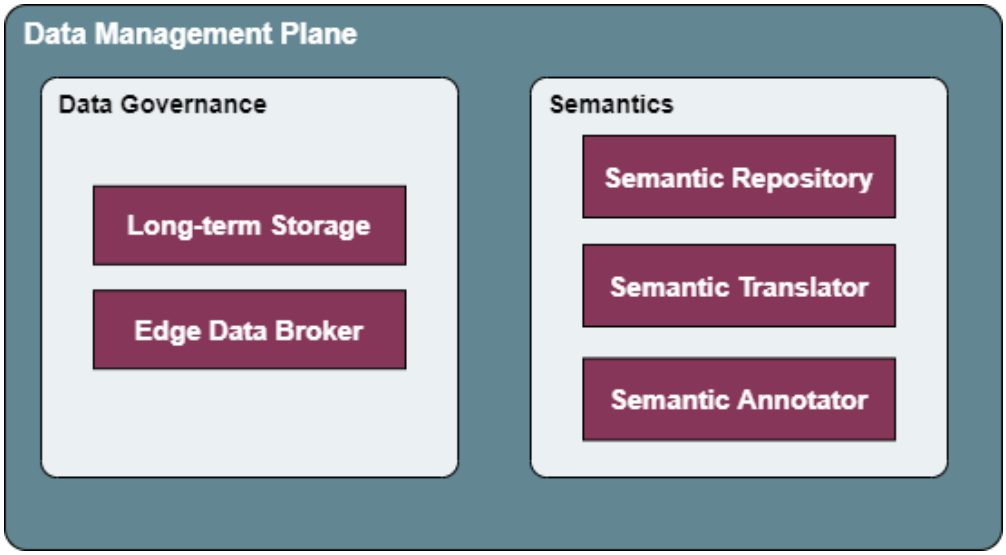


Figure 9. Data Management Plane functional blocks and enablers

On the one hand, the Data Governance block gathers enablers that, in principle, do not need to modify data, but rather offer a service that improves data safety and availability, security, or helps to optimise the flow of data, based on its contents. On the other hand, the Semantics functional block delivers semantically enabled transformations that process and, when needed, change the data to conform to the requirements of every node in a data pipeline. Semantic data processing is supported by the repository enabler, which organises models and configuration files used in semantic translation and annotation in order to ease the design of semantic data pipelines.

4.2.1. Enablers' descriptions summary

Semantic repository enabler

This enabler offers a "Nexus" for data models and ontologies, that can be uploaded in different file formats, and served to users with relevant documentation. This enabler is aimed to support files that describe data models or support data transformations, such as ontologies, schema files, semantic alignment files etc.

Additionally, human-readable documentation for the models will be served. Offered files, their metadata and documentation are, in principle, public, so that this enabler could be used as support for a shared semantic ecosystem. The secure access restrictions may be more tightly controlled with an API manager in future versions.

In short, the core function of this enabler is to be a database of data models and ontologies, with public read access. In summary, it supports:

- Versioning: different versions of data models.
- Ownership: only the data model owner may update a data model.
- Provision & search: Data models are public and browseable.
- Documentation: Documentation provided by data model owner is served.

Semantic translation enabler

Semantic Translation enabler offers a configurable service to change the contents of semantically annotated data in accordance with translation rules – so called "alignments", or alignment files. The core use-case, around which this enabler is designed, is to move data between semantic ontologies (which can be thought of as data schemas or vocabularies) that can express the same information, without changing the meaning of the information.

Flexibility of design and expressivity of configuration files allow for other use-cases, such as semantic reduction (removing selected information, e.g., because of privacy reasons), further annotation (adding additional information based on data content and possibly external variables), or even encoding or encrypting selected data items into a serialised form.

The Semantic Translator supports RDF as the only modern standard for semantic data. By design it supports and promotes the "core ontology" design, in which data transformations are always unidirectional and done to, or from a central ontology, and paired into "translation channels" to achieve bidirectional transformations. In this manner, n-to-n translations can be easily implemented, and the cost of including a new data model in existing deployments does not grow exponentially.

Translation services are offered as a "static" API for batch data, or through a publish-subscribe broker for streaming data.

Semantic annotation enabler

This enabler offers a syntactic transformation service, that annotates data in various formats and lifts it into RDF. Full list of formats is yet to be decided and the first version will support JSON.

The enabler is lightweight and stateless, so that it may have many independent deployments. The core functionality is designed to be integrated into a pipeline before the Semantic Translation enabler, which requires the use of RDF. In essence, using Semantic Annotator enables usage of the Semantic Translator with formats other than RDF.

Edge data broker enabler

The edge data broker enables the efficient management of data demand and data supply from/to the Edge Nodes. It optimally distributes data where it is needed for application, services and further analysis. Data distribution is based on reported demand and available resources at the Edge Nodes. It provides: (i) subscriptions and messages between the broker and the Edge Nodes; (ii) management of message scheduling, routing and delivery; (iii) common interfaces for searching; and (iv) finding information.

Long-term data storage enabler

The role of this enabler is to serve as a secure and resilient storage, offering different storage sizes and individual storage space for other enablers (which could request back when they are being initialising in Kubernetes pods). It also guarantees that the data will be kept safe, in face of various kinds of unauthorised access requests, or hardware failures, by only allowing access to the data once the Identity Manager and the Authorisation enablers have confirmed their access rights.

4.3. Application and Services enablers

Both devices' specifications (physical plane) and the two upper lying ASSIST-IoT horizontal Planes (Smart Network and Control plane, and Data management plane) will facilitate the collection of real-time data from massively distributed sets of sensors and heterogeneous networks. The main goal of the enablers of the Applications and Services plane is, therefore, to expose, interact and provide added value (over such collected data) via human-centric dashboards and tactile interfaces (AR and MR). To do so, the Applications and Services enablers' definitions have followed a similar approach as the App Entities recommendation proposed in the AIOTI framework[3]. It defines features to provide application logic, including data visualisation and user interaction services, data analytics capabilities, various kinds of data protection support, and data management logic.

ASSIST-IoT Application and Services enablers will consequently help human decision-makers to coordinate operations from different responsibility angles/roles (i.e., accessing rights). In particular, the project foresees three types of user roles:

1. End-users with read-only access: These users will be able to access the platform and analyse business KPIs aided by the customisable charts and graphs that will be offered via enablers of this plane. These results will help them to take decisions based on the provided insights, and supported by means of the Intelligent Decision-Making services provided.
2. ASSIST-IoT system administrators with read-and-write access: These users will be able to manage the on-boarding of all the devices and enablers of the platform, as well as to monitor their performance status and proceed accordingly in case of unexpected behaviour or even platform failures.
3. ASSIST-IoT external developers or practitioners: The stakeholders responsible for the ASSIST-IoT infrastructure, may allow external parties to include additional capabilities. In order to grant them access to particular areas of enhancement, specific clauses can be enabled or disabled over the overall platform.

As it can be seen in the Figure 10, the enablers of the Application and Services plane can be classified under three main functional blocks, "Dashboards", "AR/MR features", and "ASSIST-IoT API Management".

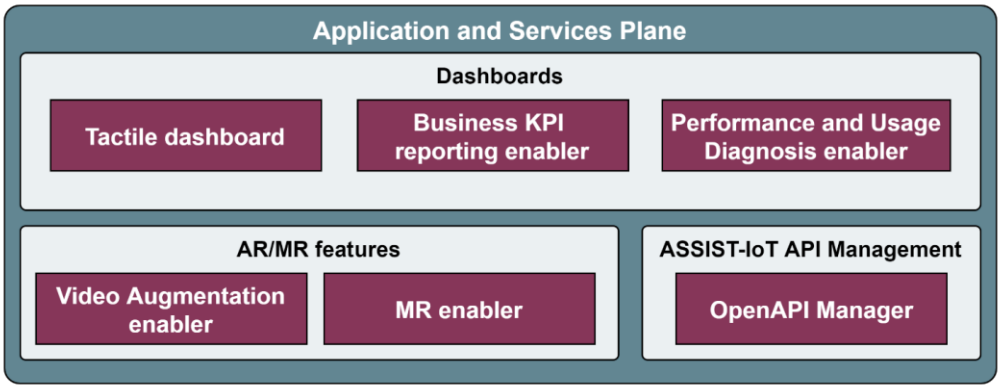


Figure 10. Application and Services plane functional blocks and enablers.

The Dashboard functional block is formed by three enablers: Tactile dashboard, Business KPI reporting enabler, and Performance and Usage Diagnosis enabler. They have both server-side and client-side modules that will enable users to see and interact with content in user-friendly interfaces. AR/MR features block gathers two enablers that will expose the Intelligent Decision-Making information in more advance human-to-machine

[3] https://aioti.eu/wp-content/uploads/2018/06/AIOTI-HLA-R4.0.7.1-Final.pdf

interfaces thanks to Computer Vision and Mixed Reality engines. Finally, the ASSIST-IoT API management block is in turn formed by the OpenAPI manager who will let expose the HTTP trigger function endpoints as REST APIs for those users with granted access rights. An individual enabler description is detailed next.

4.3.1. Enablers' description summary

Tactile dashboard enabler

The Tactile dashboard has the capability of representing data stored in all the ASSIST-IoT deployments through meaningful combined visualisations in real time. It is expected that it will also be able to integrate the Business KPI reporting, and the Performance and Usage Diagnosis enablers in order to provide the capability of exposing KPIs needed to evaluate the success of a particular organisation activities in which it is engaged. It will also be in charge of sending notifications based on the status of the data received from all the monitored ASSIST-IoT devices and enablers, respectively. To do so, the enabler will provide all the required web environment, split in two main components: (i) a frontend, that offers a web application based on a customised Vue.JS framework through which the ASSIST-IoT users and administrators can interact with; and (ii) a backend, that handles all the services needed for generating the frontend dashboard interfaces. Moreover, in order to manage the roles permissions, the enabler will connect with the Identity Manager and Authorisation enablers in order to ensure which users are authorised.

Business KPI reporting enabler

The best way to understand data of those companies that can make use of ASSIST-IoT platform is to visualise them. With the Business KPI reporting enabler it is expected that end-users can get the collected data from one or more storages and databases (such as the Edge broker enabler or the LTSE) into a collection of panels that will bring clarity, extract business insights, and allow to focus on only the KPI that are relevant to the organisation. To accomplish that goal, the Business KPI reporting enabler will provide a graphical and intuitive interface to make custom graphics, allowing to choose, in a very intuitive way among several types of graphs.

Performance and usage diagnosis enabler (PUD)

Performance and Usage Diagnosis (PUD) enabler aims at collecting performance metrics from monitored targets by scraping metrics to HTTP endpoints and highlighting potential problems in the ASSIST-IoT platform. This will allow to autonomously act in accordance (if enabled) or to notify the platform administrator for fine-tuning the associated machine resources. Four main components form the PUD enabler: (i) the PUD server, responsible for storing the time series data or metrics from long stable platform jobs; (ii) a Push Gateway to allow ephemeral and batch jobs to also expose their metrics; (iii) a WebUI used to visualise the aforementioned collected data; and (iv) an alert manager that will trigger specific actions depending on previously configured rules.

Hence, the PUD enabler will provide an end-to-end approach to infrastructure and application monitoring, covering all levels with easy instrumentation. The enabler will collect performance metrics trying to maintain operational simplicity while being able to adapt to varying scales/levels of the ASSIST-IoT infrastructure. By integrating it with a wide range of service discovery systems via REST APIs, PUD enabler will stay synchronised with the ASSIST-IoT infrastructure that it is actually monitoring.

OpenAPI management enabler

API management is a process that encompasses publishing, documenting, and overseeing APIs in a secure, and scalable environment. It allows organisations to monitor the interfaces' lifecycles of their published APIs, and guarantee that the demands for both internal and external developers, as well as applications that are communicating with the associated APIs, are appropriately met. Therefore, it provides the competencies for ensuring successful API usage in developer, business, and security perspectives.

API specification and documentation is a key factor for technology adoption, regardless of the software that is used internally or by external third parties. With the growth of publicly available APIs, some standards and tools have emerged. ASSIST-IoT, as an open platform, will be able to allow additional features out of the project development itself. To do so, a unified API access through the Open API manager will be provided. In particular, it will consist of an API design document for each ASSIST-IoT enabler based on Swagger definitions, followed by an API publisher interface. Next, the user will be able to subscribe to the APIs through the API subscription

web GUI, namely API Portal. In order to guarantee some level of management privileges, different set of access rights will also be able to be adopted by configuring them via the Cybersecurity enablers.

Video augmentation enabler

The Video Augmentation enabler will allow to perform computer vision functionalities over captured images or video streams either from ASSIST-IoT Edge nodes, or from ASSIST-IoT databases. The enabler will be formed (i) firstly by a Machine Leaning training block over which object detection of particular end-user assets can be detected (e.g., cargo containers or cars damages), followed by (ii) an Inference Engine, aiming to overlap new unseen data as input. Then, it will provide to the end user visual insights to help on enhancing operational efficiency or operational safety on the considered use cases. As it can be noticed, the Video Augmentation enabler will interact with the ML algorithms repository in order to select the most appropriate ML algorithm according to the objected data (although this is a user decision, not an autonomous decision).

MR enabler

The MR enabler is designed for providing immersive experience to practitioners of ASSIST-IoT in general, and for the Worker's safety pilot in particular as a first action line. The enabler will receive data from the Edge data broker or from the LTS enabler and will transform this data into a suitable format for visualisation capabilities over head-mounted MR displays (in principle, it is foreseen for MS Holo-Lens MR goggles). Information will then be displayed to the user, according to their authorisation/access rights, supporting user interaction with the virtual content and view customisation.

5. Future Work

This deliverable presents the work developed so far under the scope of WP4. It corresponds to the first document of a series of three iterations, and therefore the specifications depicted in the document are susceptible to change. In this document, the main results presented are the specifications of the Edge nodes and Smart IoT devices that will be developed in the project, and the specifications of the enablers that will be provided (for the upper three planes of the ASSIST-IoT architecture, namely the Smart Network and Control Plane, the Data Management plane and the Application and Services plane).

Regarding enablers specifications, results are presented following a template developed specifically for that purpose. It includes a table with main functionality and general information, a high-level diagram of its components, a table with endpoints, and a dedicated table for each of its components, in which rationale and candidate technologies for their implementation are highlighted. For the sake of facilitating the reading, most of this information has been placed into an annex, leaving in the core document the functionalities provided by each enabler.

The finalisation of this deliverable kicks off the software developments of the work package. Hence, the following actions are expected for each enabler:

- Distribution of tasks among the partners that contribute into its realisation.
- Finalisation of the endpoints/API calls required for the enabler (some of the enablers did not have it ready at this point).
- Initiation of the software developments of each of its components.
- Definition of possible interactions with WP5 or other WP4 enablers. Although they are intended to be as standalone as possible, so they can be reutilised by third parties and integrated in systems beyond ASSIST-IoT, some functionalities (such as authentication or long-term storage) are provided by other enablers, so they are not replicated within many enablers. Interactions will be formalised in the next deliverable devoted to architecture (D3.6).
- First results ready and available by the next deliverable iteration (the degree of development will not be homogeneous for all the enablers).

First software results are expected to be delivered with the second iteration of the deliverable. It is likely that some deviations occur during their development, such as:

- Slight modification of the provided functionalities.
- Change of the components that conform the enabler. For instance, some components may have been described at high level and thus may require to be split into more components.
- Modification of the technologies used for implementing the enabler with respect to the candidate ones highlighted in this document. Despite the fact that several discussions have taken place for identifying them, these decisions may not be valid during implementation.
- Identification of the necessity of more enablers to support the others.

Apart from the enablers identified so far, there are some actions that are specific for the Device and Edge plane. First of all, the formalisation of the enablers for this plane is still to be performed, work that it is to be carried out shortly. Besides, regarding the hardware elements developed for the project, the number of Smart IoT Devices will be (likely) increased once the requirements of the use cases and the necessities of the pilots are refined. All the outcomes that come out from the action points highlighted in this section will be presented in the next iteration of this deliverable, i.e., D4.2.

Annex A - Localisation

For both the port automation pilot and the smart safety of workers' pilot, indoor and outdoor localisation is needed. As per requirement of the use cases, people and equipment needs to be localised with an accuracy of less than 50 cm. With standard inexpensive GPS/GNSS, this accuracy cannot be reached. Still, with more sophisticated and expensive solutions, accuracy can be improved to decimetre level, e.g., with GNSS using geodetic grade and dual band antennas. Sub-decimetre accuracy can be reached with Differential GPS (DGPS) using a refence receiver with known location, being also possible to use a global Precise Point Positioning (PPP) service.

GPS/GNSS can be used for outdoor localisation, however, it cannot be used for indoor localisation since reception of the satellite signals cannot be guaranteed. Different principles of operation exist for indoor localisation:

- Signal strength measurement.
- Angle of Arrival (AoA) and Angle of Departure (AoD) measurement.
- Time of flight measurement.
- Phase-based measurement.
- Combination of Time of flight and Phased based measurement e.g. as used by Lidar
- Image-based triangulation or 3D reconstruction (with cameras).

Signal strength measurement is not accurate enough for fulfilling the requirements, and therefore this method will not be used. A solution using Lidar is relatively expansive compared to other solutions, whereas image-based triangulation would need relatively high compute power to be working continuously. This would lead to an energy consuming solution which is not that suitable for portable or resource-constrained solutions.

Another solution is to place *anchors* at fixed and known locations and give the person or object, of which the position needs to be known, a *tag*. This tag communicates with the anchors and by means of time-of-flight measurement, the distance between tag and anchor is determined (the Edge node can serve as anchor, as it can be seen in Figure 11). Since several tags will be used to create a 3D, live view of the location of people (and assets, see Figure 12), these have to be cost effective. The are several possibilities for implementation:

- UWB: with time of flight, it is the most precise indoor localisation technology available for industrial applications. Accuracy of 10 cm can be reached, which fulfils the aforementioned requirement of 50 cm. This is also a standard: candidate technology.
- WiFi: Accuracy of 1-5 meters: not selected.
- Bluetooth: Accuracy of meters, with BLE 5.1 accuracy of 50 cm can be reached: candidate technology.
- Camera: Camera is on fixed position, making use of person recognition. A person cannot give feedback or emergency signalling to the specific person in case of danger, being complex: not selected.

Two technologies remain: BLE and UWB. For indoor use, the received signal may have travelled many paths. On the one hand, for BLE, this means that the signal is attenuated and thus the position accuracy is reduced compared to free line of sight. On the other hand, due to the wide band aspect of UWB, multipath received signals and distance measurement is not influenced that much. Therefore, because of being more accurate and performing better with disturbances from the environment compared to BLE, UWB is selected to implement localisation.

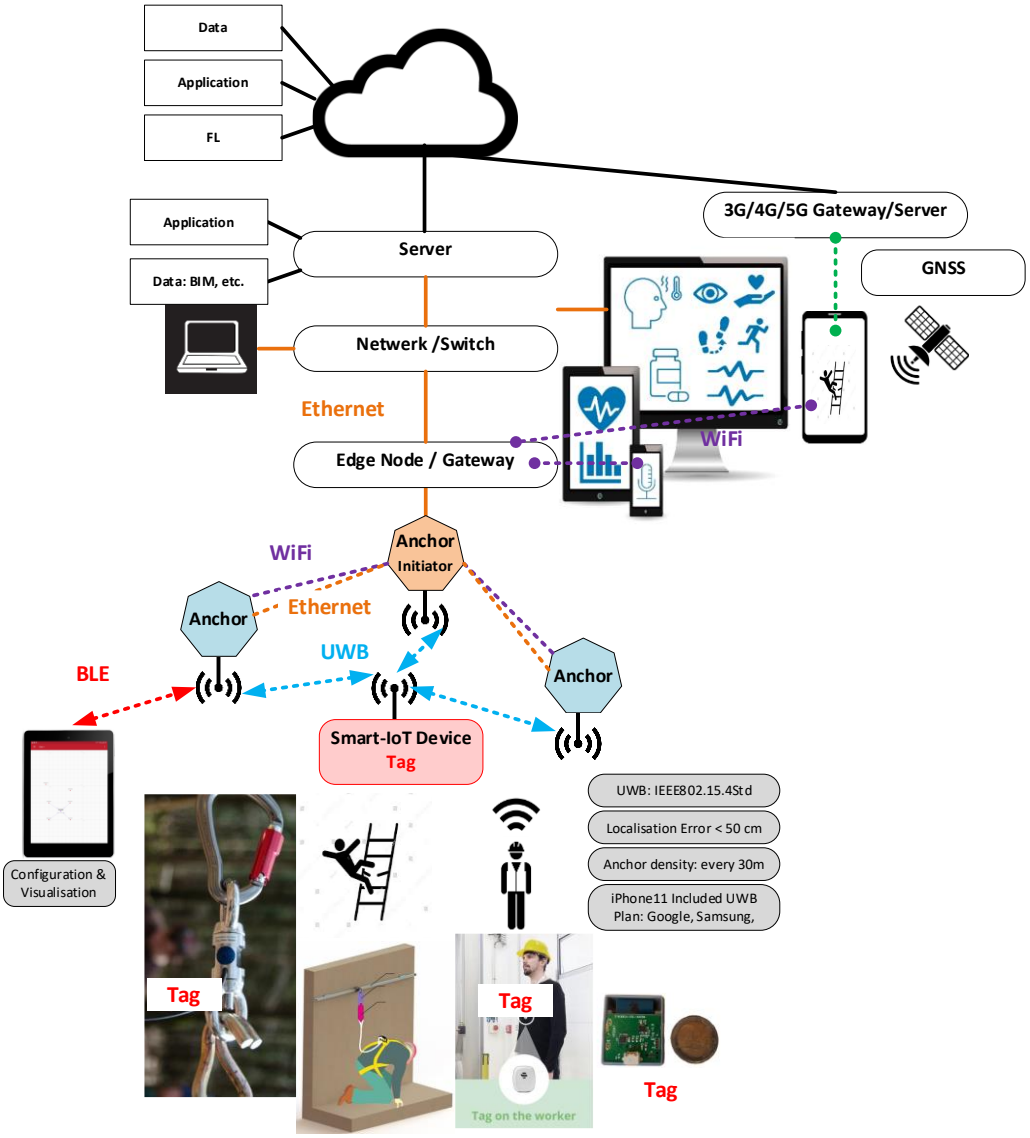


Figure 11. Localisation options in the Smart safety of workers' pilot

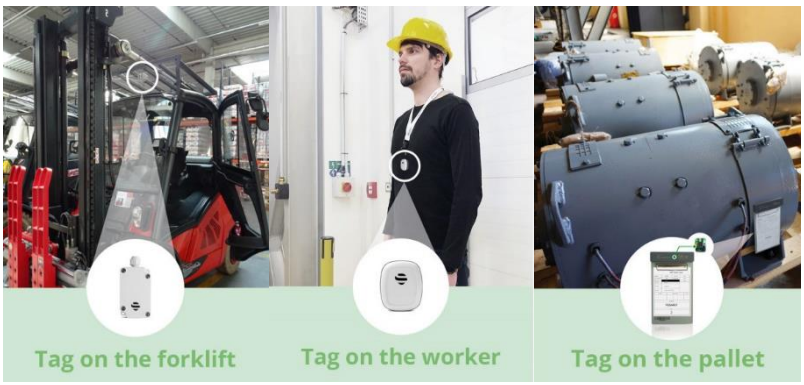


Figure 12. Possible position of tags

Annex B - Enablers templates

B.1 - Smart orchestrator enabler

Table 4. General information of the Smart Orchestrator enabler

Enabler	***Smart Orchestration enabler***
id	*T42E1*
Owner and support	*UPV, NEWAYS, OPL*
Description and main functionalities	*This enabler facilitates the interaction of user interfaces and other enablers with the main components of the MANO framework, namely the Network Function Virtualisation Orchestrator (NFVO) and the Kubernetes clusters, exposing only the required inherent functionalities. In particular, this enabler will control the whole lifecycle of Virtualised Network Functions (VNFs), from their instantiation to their termination, allowing their deployment in any k8s cluster available. If needed, adaptations for lightweight alternatives such as k3s will be studied.* *This enabler is also in charge of performing additional functionalities, like the validation of submitted descriptors for Virtualised Network Functions (VNSD) and Network Services (NSD). Finally, it also acts as an abstraction layer, decoupling the underlying selected NFVO technology from the ASSIST-IoT ecosystem. In this way, in case the NFVO component is changed in the future, only this layer would need to be adapted without having to modify the interfaces of the rest of enablers that communicate with this one.*
Plane/s involved	*Smart Network and Control Plane*
Relation with other enablers	• *T42E2: SDN Controller* • *T42E6: SD-WAN enabler* • *T55EX: (task has not started hence not formally documented yet) – Orchestration of enablers deployment*
Requirements mapping	• *R-P1-20: Remote latency capabilities (it will be in charge of deploying those CNFs in charge of ensuring it)* • *R-P1-22: Multilink wireless network capabilities (it will be in charge of deploying the related CNFs)* • *R-P3A-11: Connectivity between OEM and fleet (it may/can deploy a ping-based CNF to evaluate connection between fleet and OEM prior to an update, and instantiate those VNFs needed for stablishing the connection)* • *R-P3A-12: Edge Connectivity (it may/can deploy CNFs to support required latencies)*
Use case mapping	*This enabler is inherent to an ASSIST-IoT ecosystem and therefore it should be present at all pilots, otherwise it would not be possible to orchestrate VNFs and hence the Smart Network and Control plane would not be present. Among the use cases of the project, the ones with higher need of it are:* • *UC-P1-6: Wireless remote RTG operation* • *UC-P1-7: Target visualisation during RTG operation* • *UC-P2-6: Safe navigation instructions* • *UC-P3B-1: Vehicle's exterior condition documentation*
Required components	*MANO Abstraction Layer, Descriptor Validator, NFVO, Kubectl Proxy*

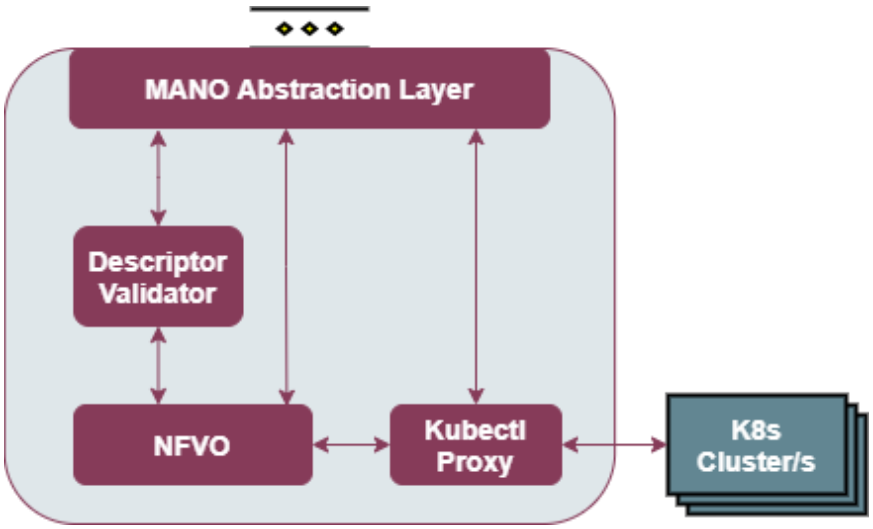


Figure 13. High-level diagram of the Smart Orchestrator enabler

Endpoints

This list of endpoints is pending to be increased. The revised list will be provided in the next iteration of the deliverable (D4.2).

Method	URL	Description
GET	**/vnfd**	Returns all the available VNFs in the catalogue.
GET	**/vnf/{id}**	Find VNFD by id
POST	**/vnfd**	Adds a VNFD to the catalogue
PUT	**/vnf/{id}**	Updates a VNFD
DELETE	**/vnf/{id}**	Deletes a VNFD
GET	**/nsd**	Return all the available NSs in the catalogue.
GET	**/nsd /{id}**	Find NSD by id
POST	**/nsd**	Adds a NSD to the catalogue
PUT	**/nsd/{id}**	Updates a NSD
DELETE	**/nsd/{id}**	Deletes a NSD
GET	**/nsi**	Returns all running NS instances
POST	**/nsi/{id}**	Creates an instance of a NS
DELETE	**/nsi/{id}**	Terminates a NS instance
GET	**/k8s**	Returns all the registered k8s clusters
POST	**/k8s**	Registers a new k8s in a VIM
DELETE	**/k8s/{id}**	Deletes an existing VIM
POST	**/vnf/name**	Uploads the container image of a VNF

Components

Finally, in the following tables are summarised the functionalities and rationale of the components that are needed, also shown in Figure 14 for implementing the enabler. There is no need of splitting the components over different nodes.

Enabler component	***MANO Abstraction Layer***
id	*T42E1_abstraction*
Description and main functionality	*This component will be the main entrypoint to the functionalities provided by the enabler, hence exposing the APIs, so other enablers (or user interfaces) can communicate with the NFVO and the k8s clusters of the MANO framework. It will act as an abstraction layer, meaning that an ASSIST-IoT deployment can be agnostic to the final choice of NFVO technology. If other technologies are to be used, the endpoints should be adapted accordingly.*
Target node/s	*High-tier Edge node*
Candidate technologies	*Custom python component, Flask, Open API.*

Enabler component	***Descriptor Validator***
id	*T42E1_validator*
Description and main functionality	*This component of the enabler will be responsible of checking that (i) VNFD and NSD are compliant with their respective JSON schemas, and (ii) there is a package for them already uploaded to the containers repository.*
Target node/s	*High-tier edge node*
Candidate technologies	*Custom python component, Flask, JSON Schema.*

Enabler component	***NFVO***
id	*T42E1 _NFVO*
Description and main functionality	*This component is responsible of orchestrating and managing the lifecycles of Network Services, from their instantiation to their termination. To this end, it communicates with the k8s clusters, which manages the virtualised infrastructure (NFVI). It should also provide connectivity among the VNFs involved. Although represented as a single component, it consists of many different services, each of them in its respective container.*
Target node/s	*High-tier edge node (it has to be able to communicate with all the K8s clusters of the site)*
Candidate technologies	*EMCO, OSM*

Enabler component	***Kubectl Proxy***
id	*T42E1_kubectl*
Description and main functionality	*This component is in charge of the communication between the NFVO (and the abstraction layer) and the k8s clusters. The native kubectl proxy of k8s will be used for this purpose.*
Target node/s	*High-tier edge node (it has to be able to communicate with all the K8s clusters of the site)*
Candidate technologies	*Kubernetes kubectl*

B.2 - SDN Controller

Table 5. General information of the SDN Controller

Enabler	***SDN Controller***
id	*T42E2*
Owner and support	*OPL, UPV*
Description and main functionalities	*The SDN Controller is the key element of a SDN-enabled network, being the software that takes over the responsibilities of the control plane from the hardware elements (switches mostly), including monitoring and management of packet flows. Although typically*

Enabler	***SDN Controller***
	installed in a dedicated machine, its functionalities are intended to be provided following the ASSIST-IoT architecture based on enablers, either by adopting a distribution that matches it or by making the necessary adaptations to fulfil them. The main functionalities are related to network management, operation and maintenance, allowing topology management, network configuration, network control and network operations, among other features.
Plane/s involved	*Smart Network and Control Plane*
Relation with other enablers	• *T42E1: Smart Orchestration enabler* • *T42E3: Auto-configurable Network enabler* • *T42E4: Traffic Classification enabler*
Requirements mapping	• *R-P3A-11: Connectivity between OEM and fleet (it provide connectivity setup and control)* • *R-P3A-12: Edge Connectivity (it provides network core connectivity for edge systems)*
Use case mapping	*This enabler is one of the main components of networking system in an ASSIST-IoT ecosystem, and it should be present at all pilots, where network connectivity is required among edge systems and core platform components. Exemplary use cases in the project are:* • *UC-P1-6: Wireless remote RTG operation* • *UC-P1-7: Target visualisation during RTG operation* • *UC-P2-6: Safe navigation instructions* • *UC-P3B-1: Vehicle's exterior condition documentation*

In the ONOS architecture (the one that will be leveraged for the project), one can distinguish core functional modules like Configuration, Control, Operation, Topology, and Northbound (NB) and Southbound (SB) API. Core subsystems are related to device, link, host, topology, etc. On the one hand, the usage of the SB API on the network level facilitates the integration of different vendors' devices. On the other hand, the NB API is available for application developers. Being ONOS the implementation that will be used for the project, it is possible to leverage REST API and also new generation of control and configuration interfaces like gNMI, gNOI, P4Runtime, NetDisco. The main functions envisioned in the project to be useful are the following: Device, Link, Host, Topology, Path, Flow , Flow Objectives, Group, Meter, Intent,, Application, Component Configuration.

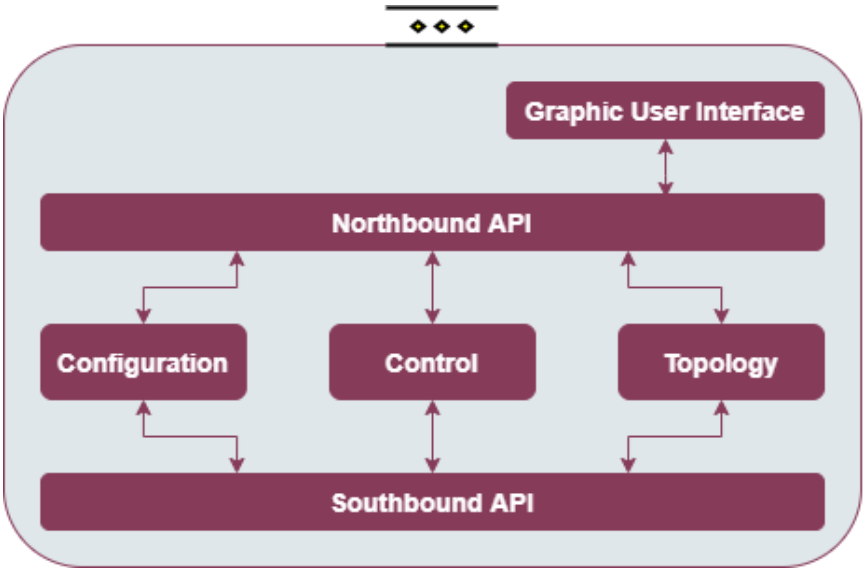


Figure 14. High-level diagram of the SDN Controller

Endpoints

This list of endpoints is pending to be increased. The revised list will be provided in the next iteration of the deliverable (D4.2).

Method	URL	Description
GET/POST/ PUT/DELETE	**/link/ ?{device=deviceId} {port=portNumber} {direction=[ALL,INGRESS,EGRESS]}**	Lists all infrastructure links, creates, update, deletes device
GET/POST/ PUT/DELETE	**/devices/{deviceid}/ports**	Lists all infrastructure devices, creates, update, deletes device
GET/POST/ PUT/DELETE	**/hosts/{hostId}**	Lists all end-stations hosts.
GET	**/topology/clusters/{clusterId}**	Gets list of topology cluster overviews.
GET/POST /DELETE	**/paths/{elementId}/{elementId}**	Gets set of pre-computed shortest paths between the specified source and destination network elements
GET/POST /DELETE	**/flows/{deviceId}/{flowId}**	Creates, lists, deletes a single flow rule applied to the specified infrastructure device
GET/POST /DELETE	**/meters/{deviceId}**	Creates, lists, deletes a single meter entry applied to the specified infrastructure device.
GET/POST /DELETE	**/intents/{app-id}/{intent-id}**	Gets the details for the given Intent object. Creates, deletes a new Intent object.
GET/POST/ PUT/DELETE	**/applications/{app-name}**	Gets a list of all installed applications. Activates, deactivates the named application.
GET/POST /DELETE	**/configuration/{component}**	Gets the configuration values for a single component. Adds, removes a set of configuration values to a component

Components

Enabler component	***Northbound API***
id	*T42E2_NB*
Description and main functionality	*Northbound API provide REST API and new generation interfaces using gNMI, gNOI, P4Runtime, NetDisco. It is needed for developing applications for network control and orchestrations and can be used by other external enablers. It consists of set of functions for mentioned above modules.*
Target node/s	*High-tier Edge node*
Candidate technologies	*Java and microservices will be used for this purpose*

Enabler component	***Southbound API***
id	*T42E2_SB*
Description and main functionality	*Southbound API provide protocols like NETCONF and new generation interfaces using gNMI, gNOI, P4Runtime, NetDisco. It is needed for network devices control provided by different vendors.*
Target node/s	*High-tier Edge node*
Candidate technologies	*Java and microservices will be used for this purpose*

Enabler component	***Control Module***
id	*T42E2_Control*
Description and main functionality	*This component is responsible for network flow control and meter API. It allows for network routing and traffic management.*
Target node/s	*High-tier Edge node.*

Enabler component	***Control Module***
Candidate technologies	*Usage of NB and SB API technologies*

Enabler component	***Configuration Module***
id	*T42E2_Configuration*
Description and main functionality	*This component is in charge of configuration of network devices, tracking the changes in the configuration of the network.*
Target node/s	*High-tier Edge node.*
Candidate technologies	*Usage of NB and SB API technologies*

Enabler component	***Topology Module***
id	*T42E2_Topology*
Description and main functionality	*This component is responsible for topology management of the network. It manages and keeps information about the network graph and network devices, links, and hosts.*
Target node/s	*High-tier Edge node.*
Candidate technologies	*Usage of NB and SB API technologies.*

Enabler component	***Operation Module***
id	*T42E2_Operation*
Description and main functionality	*This component is responsible for topology management of the network. It manages and keeps information about the network graph and network devices, links and hosts.*
Target node/s	*High-tier Edge node.*
Candidate technologies	*Usage of NB and SB API technologies.*

Enabler component	***Graphic User Interface***
id	*T42E2_Interface*
Description and main functionality	*This component will expose the functionalities of the internal modules of the SDN Controller for administrative purposes.*
Target node/s	*High-tier Edge node.*
Candidate technologies	*Usage of NB and SB API technologies.*

B.3 - Auto-configurable Network enabler

Table 6. General information of the Auto-configurable Network enabler

Enabler	***Auto-configurable Network enabler***
id	*T42E3*
Owner and support	*OPL, UPV*
Description and main functionalities	*This enabler provides optimised network routing configuration capabilities to the SDN Controller of an ASSIST-IoT ecosystem. This enabler will consist of an application that consumes the northbound APIs of the SDN Controllers to generate a policy that improves the performance of one/many KPIs of the network (e.g., latency). The selection of the most suitable model depends on the use cases, and the objective of this enabler is to provide* ***three different strategies*** to be compatible with the selected Controller.
Plane/s involved	*Smart Network and Control Plane*
Vertical, related capabilities and features	*Not Applicable*

Enabler	***Auto-configurable Network enabler***
Relation with other enablers	• *T42E2: SDN Controller*
Requirements mapping	• *R-P1-20: Remote latency capabilities (this enabler can help prioritising involved traffic)* • *R-P3A-12: Edge Connectivity (this enabler can prioritise traffic related to PCM calibration updates)*
Use case mapping	*This enabler fits all those use cases in which there is interest in that a particular traffic could be prioritised later on by the SDN Controller. The listed use cases refer to those in which it would be key (or just recommended) that traffic of either video streams, mission critical systems or image data have priority.* • *UC-P1-7: Target visualisation during RTG operation* • *UC-P2-6: Safe navigation instructions* • *UC-P3B-1: Vehicle's exterior condition documentation* • *UC-P3B-2: Exterior defects detection support*
Required components	*Policy module, monitoring module*

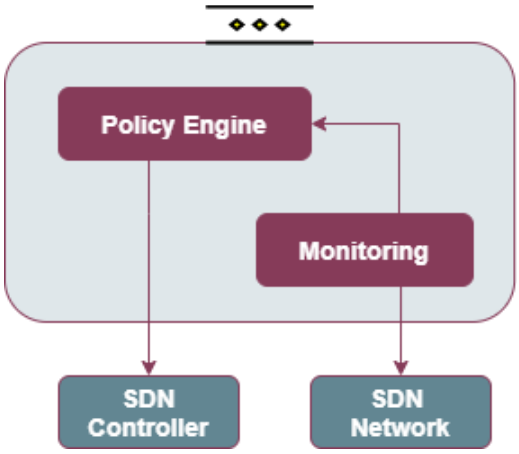


Figure 15. High-level diagram of the Auto-configurable Network enabler

Components

Enabler component	***Policy Engine***
id	*T42E3_Policy*
Description and main functionality	*This component is in charge of creation of polices and its execution in the SDN network for optimising the network traffic and creation of routing paths. It obtains the network information using SDN controller and data traffic using monitoring system. The optimising algorithms can be supported by AI techniques.*
Target node/s	*High-tier Edge node.*
Candidate technologies	*AI techniques, software module (python, java).*

Enabler component	***Monitoring Module***
id	*T42E3_Monitoring*
Description and main functionality	*This component is responsible for collecting network traffic statistics. The KPI specification will be based on use cases requirements. Potential open source tool is rt-sflow to be used.*
Target node/s	*High-tier Edge node.*
Candidate technologies	*Monitoring tools, rt-sflow, ifstat*

B.4 - Traffic Classification enabler

Table 7. General information of the Traffic Classification enabler

Enabler	***Traffic Classification enabler***
id	*T42E4*
Owner and support	*OPL, UPV*
Description and main functionalities	*The aim of this enabler is to classify network traffic into a number of application classes (video streaming, VoiP, Network control, best effort, OAM, etc.), making use of a AI/ML framework and dedicated algorithms. The traffic classification enabler can be seen as a service of the application layer of the general SDN architecture. To provide its functionalities, three stages are needed: (i) data processing and feature extraction, (ii) model selection training, and (iii) inference and results. For developing this particular enabler, we will rely on modules of the SDN Controller for providing the features of the data to perform the inference, but in case this information it is not enough the necessary components will be added.* *The results provided by this enabler can be used for supporting network policy-making processes for ensuring QoS (application priority), optimising traffic, or enhancing security (therefore results must be consumed by other enablers).*
Plane/s involved	*Smart Network and Control Plane*
Relation with other enablers	• *T42E2: SDN Controller* • *T42E3: Auto-configurable Network enabler*
Requirements mapping	• *R-P1-20: Remote latency capabilities (this enabler can help prioritising involved traffic)* • *R-P3A-12: Edge Connectivity (this enabler can prioritise traffic related to PCM calibration updates)*
Use case mapping	*This enabler fits all those use cases in which is interesting that a particular traffic could be prioritised later on by the SDN Controller. The listed use cases refer to those in which it would be key (or just recommended) that traffic of either video streams, mission critical systems or image data have priority.* • *UC-P1-7: Target visualisation during RTG operation* • *UC-P2-6: Safe navigation instructions* • *UC-P3B-1: Vehicle's exterior condition documentation* • *UC-P3B-2: Exterior defects detection support*
Required components	*Traffic Classification API, Data Stream Network Classifier, Knowledge Database, Training Module*

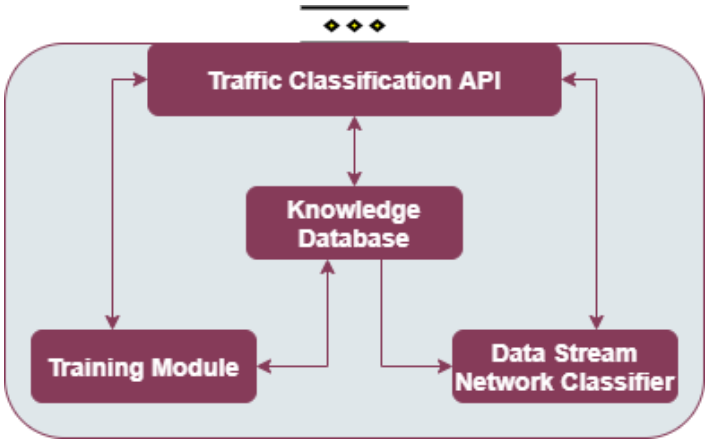


Figure 16. High-level diagram of the Traffic Classification enabler

Endpoints

This is a preliminary list of endpoints expected to be improved and increased. The revised list will be provided in the next iteration of the deliverable (D4.2).

Method	URL	Payload (if needed)	Description	Response format
GET	**/training/execute**		Starts a training session, with the model and data stored in the knowledge database (model in T52E3, ML algorithms repository)	
GET	**/database/model**		Returns de last trained model so it can be reused	
POST	**/inference/start**		Starts listening to a specific network interface to analyse the incoming traffic	
POST	**/inference/ configuration**		Method for configuring specific parameters of the inference method	

Components

Finally, in the following tables are summarised the functionalities and candidate technologies of the components that are needed, shown in Figure 16, for implementing the enabler. Given the large size that the dataset may have, it would be recommended that the Knowledge Database is deployed jointly with the training module. In general, for avoiding network congestion, it is preferable that at least the API and the Network Classifier are instantiated in the same host as the SDN Controller (There is no need of splitting the components over different nodes).

Enabler component	***Traffic Classification API***
id	*T42E4_API*
Description and main functionality	*This component is needed for acting as a central proxy of the operations that are offered by the enabler. It is responsible of managing the API calls related to starting an inference (i.e., traffic classification) process, add new data to the data set and retrain the corresponding AI model with new data, among other actions.*
Target node/s	*High-tier Edge node (recommended along with SDN Controller)*
Candidate technologies	*Flask will be used for this purpose*

Enabler component	***Knowledge Database***
id	*T42E4_Database*
Description and main functionality	*This component will contain the training dataset, the model's data to be either updated by the Training Module or leveraged by the Network Classifier, and other information needed by the enabler to work. Given the potential large size of the data, it may be recommended to instantiate this enabler in the same node with the rest of components of the enabler.*
Target node/s	*High-tier Edge node. It would be recommended to be in the same host as the training module.*
Candidate technologies	*To be decided, possibility of just using a local persistent volume of k8s.*

Enabler component	***Training Module***
id	*T42E4_Training*
Description and main functionality	*This component is in charge of the training of the model with new traffic data. Data will come from .pcap files, however, the training module will receive just the extracted features and not the entire files (passed by another application of the SDN Controller or developed further on in the project if needed).*

Enabler component	*Training Module*
Target node/s	*High-tier Edge node. It would be recommended to be in the same host as the Knowledge Database.*
Candidate technologies	*To be decided. Preference towards Tensorflow with libraries such as Keras or scikit-learn depending on the final model choice. Regarding the model choice, ETSI ENI Proof of Concept #5[4] and the work from Pritom et al.[5] will be considered.*

Enabler component	*Data Stream Network Classifier*
id	*T42E4_Classifier*
Description and main functionality	*This component will receive the extracted features of a traffic data packet and will determine the class of traffic to which it belongs, among the available options.*
Target node/s	*High-tier Edge node. It would be recommended to be in the same host as the SDN Controller.*
Candidate technologies	*To be decided. Preference towards Tensorflow with Python and libraries like NumPy and Pandas, Spark or lightweight alternatives for data stream processing (this component may be split in two in the future)*

B.5 - Multi-link enabler

Table 8. General information of the Multi-link enabler

Enabler	*Multi-link enabler*
id	*T42E5*
Owner and support	*UPV, NEWAYS, OPL, PRO, TL*
Description and main functionalities	*Multi-link wireless network capabilities provide the possibility of sending video streams of data over different Radio Access Networks and different channels in each of them (for instance, regarding cellular, using more than 1 connection). Besides, it should provide reliability mechanisms: in case one channel is down, signal cannot be lost or at least it should be recovered almost in real time. The reliability mechanism will be integrated by means of redundancy rather than diversity (e.g., MPTCP), since it is better considering latency aspects, especially when switching.* *Regarding redundancy, signal will be sent through two channels, of (at least) two different RATs, so in case one is down the other can take over. In case of video streaming, the redundant channel will transmit at lower resolution, and in case it has to take over this resolution will dynamically increase. Since this enabler comes from a requirement of the port automation pilot, the involved RATs will be cellular, WiFi and fluidmesh (starting with two of them). When the main link is lost and the second takes over, a third channel should be initiated for acting as the redundant one.* *This enabler will be expanded in the future to support not just multi-link connections for video but also for other classes of low-latency traffic.*
Plane/s involved	*Smart Network and Control Plane*
Vertical, related capabilities and features	*Not Applicable*
Relation with other enablers	• *T43E8: Long-term data storage enabler* • *T44E1: Tactile Dashboard enabler*
Requirements mapping	• *R-P1-22: Multilink wireless network capabilities (self-explanatory).*

[4] https://eniwiki.etsi.org/index.php?title=PoC_05:_Intelligent_Traffic_Profiling
[5] P. Kumar Mondal, L. P. Aguirre Sanchez, E. Benedetto, Y. Shen, and M. Guo, "A dynamic network traffic classifier using supervised ML for a Docker-based SDN network", Connection Science, 2020.

Enabler	*Multi-link enabler*
	• *R-P1-21: Remote reliability capabilities (in case one network fails, another can take over, considering redundancy mechanisms)*
Use case mapping	• *UC-P1-6: Wireless remote RTG operation* • *UC-P1-7: Target visualisation during RTG operation*
Required components	*API Server, Network Agent, Network Selection Middleware, Configuration Database, Signalling Server, Video Origin Server, Video Destination Server, Video Client*

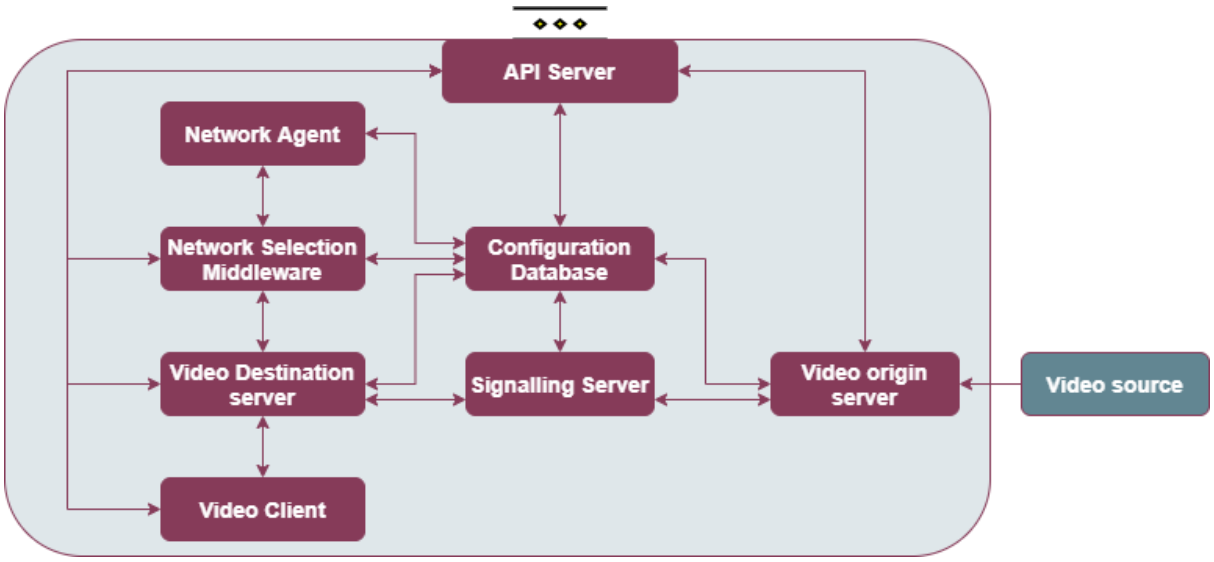


Figure 17. High-level diagram of the Multi-link enabler

Endpoints

Method	URL	Description
POST	**/video/server**	Selects the server to which pushing the video from the origin server.
POST	**/video/source**	Selects the IP address of the video source
GET	**/links/interfaces**	Returns the available, common radio interfaces for the hosts that contain both origin and destination servers
POST	**/link/interfaces**	Configures the interfaces allowed for video transmission
POST	**/link/initiate**	Initiates the signaling flow between the origin and the destination server, based on the better connection
POST	**/video/initiate**	Initiates the signaling flow and the video transmission from the origin to the destinations server, based on the better connection
POST	**/video/end**	Ends the signaling flow and the video transmission from the origin to the destinations server
POST	**/client/connect**	The video client connects to a video server to reproduce its content

Components

Finally, in the following tables are summarised the functionalities and candidate technologies of the components that are needed, shown in Figure 17, for implementing the enabler. Most components will be deployed together in the Edge node that will act as the "sink" of the wireless multilink enablers and will receive the video data sent by the video origin server to be presented to the end user. The latter is the only component that won't be placed along with the rest of components, since it will be located on the edge node that is physically connected (or can reach by wired interface) input from the camera/s and has to send it through the channels mandated by the network selection middleware (a secondary API may be needed at this point).

Enabler component	***API Server***
id	*T42E5_server*
Description and main functionality	*The high-level operations will be managed by API endpoints (e.g., peer origin and destination server, video quality demanded, etc.)*
Target node/s	*Any*
Candidate technologies	*Flask*

Enabler component	***Network Agent***
id	*T42E5_agent*
Description and main functionality	*The network agent will gather network metrics (latency, packet loss, bandwidth, etc.) from the established connections and send the data to the Network Selection Middleware for primary RAT selection.*
Target node/s	*Any*
Candidate technologies	*To be decided (IPerf, D-ITG, NetPerf, etc.)*

Enabler component	***Configuration database***
id	*T42E5_configuration*
Description and main functionality	*This component will register the information related to Edge nodes that will need the establishment of communication, including the available RATs. It will not store any data related to the "preferred" connection. That feature is managed by the network selection middleware once all the required connections are established.*
Target node/s	*Any*
Candidate technologies	*PostgreSQL (later on, it can be substituted by LTSE enabler)*

Enabler component	***Network Selection Middleware***
id	*T42E6_selection*
Description and main functionality	*This middleware will analyse the RTT of the available RATs and will decide which will be the main radio channel, the redundant and the backup ones (not transmitting). It should be highlighted that it won't be continuously changing among the main and the redundant connection unless detecting anomalous number of packet drops.*
Target node/s	*Any*
Candidate technologies	*Custom component in Python*

Enabler component	***Signaling Server***
id	*T42E5_signaling*
Description and main functionality	*This component will establish the connections between peers, according to the information stored in the configuration database. Since redundant mechanisms are to be implemented, the connections will be peer to peer, independent.*
Target node/s	*Any*
Candidate technologies	*Websockets, WebRTC, OvenMediaEngine*

Enabler component	***Video destination server***
id	*T42E5_server*

Enabler component	***Video destination server***
Description and main functionality	*It will receive the video streams for the different RATs and perform the typical operations such as video decoding to present results to the user. It will present to the user only the video from the primary connection choice.*
Target node/s	*Any*
Candidate technologies	*To be decided. Candidate technologies to leverage: WebRTC, openCV, H.264, OvenMediaEngine*

Enabler component	***Video client***
id	*T42E5_client*
Description and main functionality	*This component will be present in a computer user graphical interface, so user can observe the video stream in (almost) real time.*
Target node/s	*Any*
Candidate technologies	*To be decided. Candidate technologies to leverage: HTML5 Player, OvenPlayer*

Enabler component	***Video origin server***
id	*T42E5_origin*
Description and main functionality	*It will be connected to the video source and perform all the operations related to video encoding, transcoding, etc. Video will be sent by the connections (and quality) mandated by the Network Selection middleware.*
Target node/s	*Any*
Candidate technologies	*To be decided. Candidate technologies to leverage: webRTC, OvenMediaEngine engine, H.264*

B.6 - SD-WAN Enabler

Table 9. General information of the SD-WAN enabler

Enabler	***SD-WAN enabler***
id	*T42E6*
Owner and support	*UPV, OPL*
Description and main functionalities	*The objective of this enabler is to provide access between nodes from different sites based on SD-WAN technology. This enabler will consist of three main virtualised elements, (i) an edge node in each one of the sites to be interconnected (SD-WAN edge), which will act as the endpoint between the site's network and the different access networks that connect them (MPLS, Internet, 4G, Satellite, etc.), creating/finalising the necessary tunnels; (ii) an SD-WAN controller, which includes, IP address management, and pushing down policies onto SD-WAN Edges, among other functionalities; and (iii) a configuration server, also known as WAN Service Configurator, that contains the application-based policies and functionalities (configuration of end-to-end WAN services, QoS/business optimisation, etc.).* *Besides, the SD-WAN edge will consist of two main components, a virtual switch and monitoring agents. The monitoring agents will collect network metrics that will be processed in the WAN monitoring module, getting outcomes that in turn will feed the WAN Service Orchestrator for applying the optimal configurations (in tunnel selection and application prioritisation).*
Plane/s involved	*Smart Network and Control Plane*
Relation with other enablers	• *T42E1: Smart Orchestrator*

Enabler	*SD-WAN enabler*
Requirements mapping	• *R-P3A-12: Edge Connectivity (OEM and Edge nodes may not be in the same network, and the connection among them should be secure and anonymous in case data travels through public networks. SD-WAN enabler can aid to apply QoS mechanisms to reduce E2E latency as well).*
Use case mapping	*This enabler will grant a secure and optimised connection for applications and services from different sites.* • *UC-P3B-1: Vehicle's exterior condition documentation* • *UC-P3B-2: Exterior defects detection support*
Required components	*Virtual Switch, Monitoring Agent, SD-WAN controller, WAN Monitoring, Monitoring Database, WAN Service Configurator*

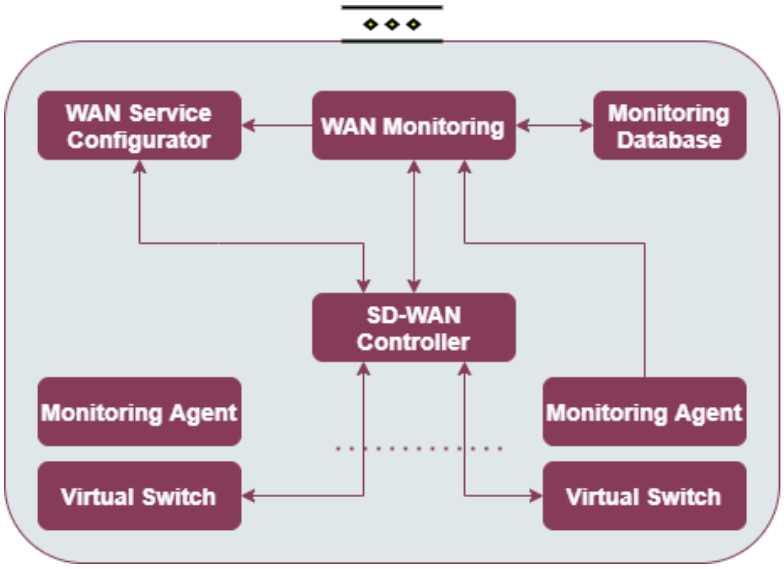


Figure 18. High-level diagram of the SD-WAN enabler

Components

Finally, in the following tables are summarised the functionalities and candidate technologies of the components that are needed, shown in Figure 18, for implementing the enabler. The components of this enabler will be instantiated in a decentralised manner. For each site that is expected to be connected to the WAN, at least a virtual switch and a monitoring agent has to be present in an SD-WAN edge node (this is, a node connected to the access networks, like the Internet). Besides, the SD-WAN Controller and the applications on top of it that control the logic of the WAN connections need to have connectivity with the SD-WAN edge nodes and have to be instantiated preferably together (the SD-WAN Controller, the WAN Monitoring, Monitoring Database and the WAN Service Configurator).

Enabler component	***Virtual Switch***
id	*T42E6_switch*
Rationale	*This component will be located in the ASSIST-IoT Edge nodes that act as SD-WAN edge. It will receive commands from the SD-WAN Controller to set-up the connections between sites through the (wired) access networks and apply the corresponding rules to ensure the QoS requirements of the involved applications.*
Node type/s*	*Edge node with direct access to the networks connecting the sites that want to be communicated (e.g., Internet, MPLS, etc.).*
Candidate technologies	*OpenVSwitch*

Enabler component	***Monitoring Agent***
id	*T42E6_agent*
Rationale	*This component will be located in the ASSIST-IoT Edge nodes that act as SD-WAN edge. It will gather network metrics (latency, packet loss, jitter, etc.) that will be send to the WAN Monitoring component.*
Node type/s*	*Edge node with direct access to the networks connecting the sites that want to be communicated (e.g., Internet, MPLS, etc.).*
Candidate technologies	*To be decided (IPerf, D-ITG, NetPerf, etc.)*

Enabler component	***SD-WAN controller***
id	*T42E6_controller*
Rationale	*The (SDN) controller will control the switches of the SD-WAN edge, based on the path policies mandated by the WAN Service configuration component.*
Node type/s*	*Cloud or Edge Node with access to the SD-WAN edge nodes of the sites to be connected to the WAN.*
Candidate technologies	*To be decided (probably ONOS)*

Enabler component	***WAN Monitoring***
id	*T42E6_monitoring*
Rationale	*This component will collect and analyse the information captured by the monitoring agents instantiated in the SD-WAN edge nodes in order to analyse if these metrics are still compliant to the QoS thresholds. If the collected metrics exceed the threshold of service policies, it communicates with the WAN Service Configurator in order to decide a reconfiguration.*
Node type/s*	*Cloud or Edge Node with access to the SD-WAN edge nodes of the sites to be connected to the WAN.*
Candidate technologies	*Custom component in Python. Since no large traffic is expected, stream technologies won't be used.*

Enabler component	***Monitoring Database***
id	*T42E6_database*
Rationale	*This component will aid the WAN monitoring component to store the retrieved monitoring parameters that could be useful for further analysis.*
Node type/s*	*Cloud or Edge Node with access to the SD-WAN edge nodes of the sites to be connected to the WAN.*
Candidate technologies	*To be decided. Mongo is the preferable option, especially since historic data could be used for better training or refinement. In case only context data is needed (last, updated data needed, not historic), FIWARE's Orion Context Broker could be used for it.*

Enabler component	***WAN Service Configurator***
id	*T42E6_config*
Rationale	*The main objective of this component is to select among the available tunnels (and underlying access networks) the optimal one according to QoS thresholds.*
Node type/s*	*Cloud or Edge Node with access to the SD-WAN edge nodes of the sites to be connected to the WAN*
Implementation technologies	*Custom component in Python*

B.7 - WAN Acceleration enabler

Table 10. General information of the WAN Acceleration enabler

Enabler	***WAN Acceleration enabler***
id	*T42E7*
Owner and support	*UPV, NEWAYS, OPL*
Description and main functionalities	*This enabler aims at increasing the efficiency of data transfer in Wide Area Network. This enabler will contain a set of independent, standalone VNFs with that purpose. These functions can be either chained (so data that requires of different techniques travels through the different functions) or selected for specific purposes. For now, three functions are expected to be provided:* • *Traffic shaping* • *Data compression*
Plane/s involved	*Smart Network and Control Plane*
Relation with other enablers	• *T42E1: Smart Orchestrator* • *T42E6: SD-WAN enabler* • *T55EX: (task has not started hence not formally documented yet) – Orchestration of enablers deployment*
Requirements mapping	• *R-P3A-12: Edge Connectivity (OEM and Edge nodes may not be in the same network, and the connection among them should be as optimal as possible).*
Use case mapping	*This enabler will accelerate the connections for applications and services from different sites. In theory it can be installed in any deployment with multi-site deployments, so they can be of utility in these use cases of the project:* • *UC-P3B-1: Vehicle's exterior condition documentation* • *UC-P3B-2: Exterior defects detection support*
Required components	*Network Stream Processor, Traffic Shaping module, Data Compression module, Data Decompression module*

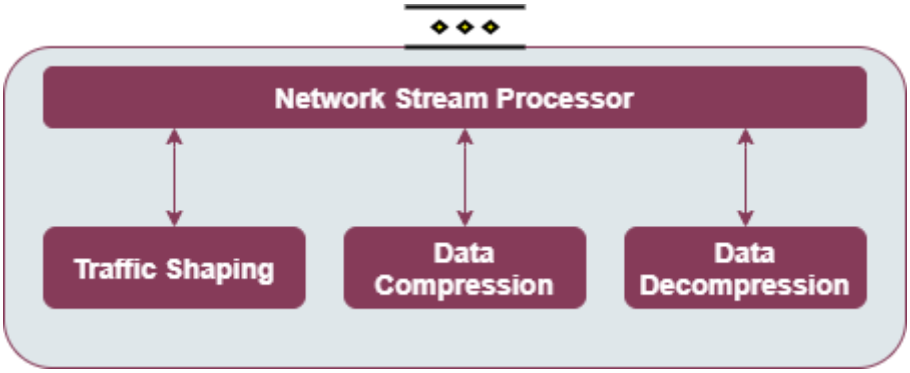


Figure 19. High-level diagram of the WAN Acceleration enabler

Components

Enabler component	***Network stream processor***
id	*T42E7_stream*
Description and main functionality	*This component will receive the data streams and execute the acceleration functions demanded via the enabler interface.*
Target node/s	*Cloud or Edge Node with access to the SD-WAN edge nodes of the sites to be connected to the WAN*
Candidate technologies	*To be decided (options: spark, Kafka, etc.)*

Enabler component	***Traffic shaping module***
id	*T42E7_shaping*
Description and main functionality	*This component implements a bandwidth management technique for delaying less-priority data packets to ensure that other applications fulfil their intended QoS, latency or performance requirements.*
Target node/s	*Cloud or Edge Node with access to the SD-WAN edge nodes of the sites to be connected to the WAN*
Candidate technologies	*Custom component in Python or Go.*

Enabler component	***Data Compression module***
id	*T42E7_compression*
Description and main functionality	*This component will be in charge of compressing network traffic to reduce bandwidth usage of the WAN network.*
Target node/s	*Cloud or Edge Node with access to the SD-WAN edge nodes of the sites to be connected to the WAN*
Candidate technologies	*Custom component in Python or Go.*

Enabler component	***Data Decompression module***
id	*T42E7_decompression*
Description and main functionality	*This component will be in charge of decompressing the network traffic that arrives from the WAN network and that has been comprised by the Data Compression counterpart.*
Target node/s	*Cloud or Edge Node with access to the SD-WAN edge nodes of the sites to be connected to the WAN*
Candidate technologies	*Custom component in Python or Go.*

B.8 - VPN enabler

Table 11. General information of the VPN enabler

Enabler	***VPN enabler***
id	*T42E8*
Owner and support	*UPV*
Description and main functionalities	*This enabler will facilitate the access to a node or device from a different network to the site's private network using a public network (e.g., the Internet) or a non-trusted private network. VPN enabler must work in High Availability (HA) mode, with VPN servers distributed in different nodes, and failover. As a first step, the site's network will be considered trusted, so VPNs will not be needed to connect nodes or devices that belong to it.* *It should be highlighted that SD-WAN enabler will be the primarily choice for connecting sites' networks while VPN will (primarily) connect particular external elements to the site's network since (1) VPN lacks both network and application-level performance optimisation, and (2) it requires extensive manual effort to add different sites to the entire WAN.*
Plane/s involved	*Smart Network and Control Plane*
Relation with other enablers	*No interactions have been identified at this point. To evaluate potential case of interaction with T51E5 – Automated device connection and configuration.*
Requirements mapping	• *R-P3A-1: Monitored Data channels (in case the car is on the road, the communication between its far edge node and the system must be private).*
Use case mapping	• *UC-P3A-1: Fleet in-service emissions verification*

Enabler	*VPN enabler*
	This enabler can be of utility in "generic" use cases, for instance when users (mostly administrators) require access to dashboards and applications (without strict require latency requirements) from outside the site's private network.
Required components	*Load balancer, VPN Server, Health VPN Client*

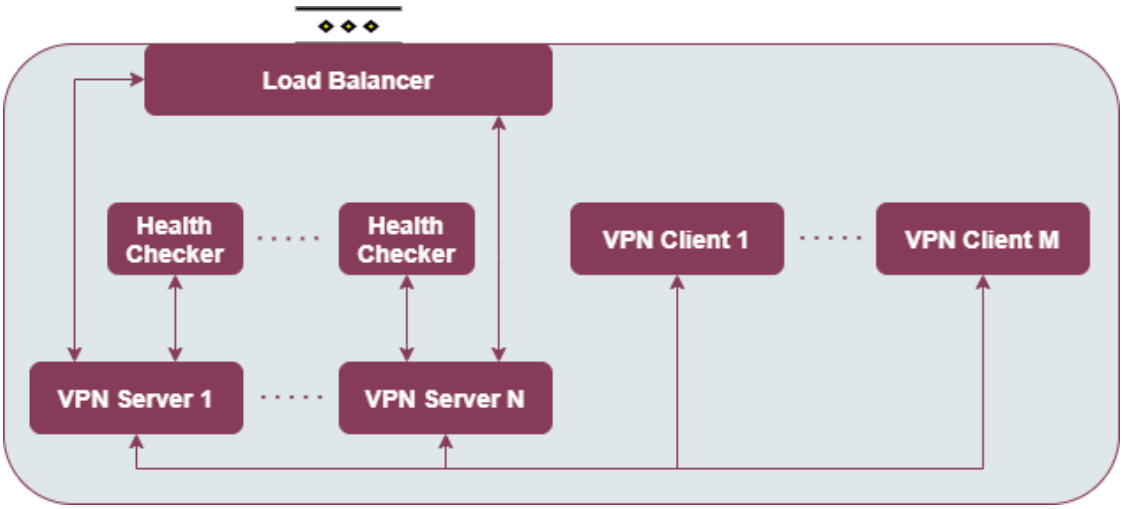


Figure 20. High-level diagram of the VPN enabler

Endpoints

Method	URL	Description
POST	**/server/start**	This command will start the VPN service with predefined interfaces and secure keys.
GET	**/server/key**	Returns the public key
POST	**/server/peer**	Adds a peer to the servers and return the information required for its connection
GET	**/server/peers**	Returns all the peers
DELETE	**/server/peer**	Deletes a specific peer
GET	**/server/status**	Returns the status of the VPN servers
POST	**/client/start**	Initiates a VPN connection to a server, using the public key, address other required options (only for containerised clients, for pc desktop and mobile phones use a client available in marketplace)
POST	**/client/stop**	Closes a VPN connection to a server (only for containerised clients, for pc desktop and mobile phones use a client available in marketplace)

Components

Finally, in the following tables are summarised the functionalities and candidate technologies of the components that are needed, shown in Figure 20, for implementing the enabler. Being a key enabler for granting private, protected access to external users, High Availability mode (HA) is a convenient feature. To support it, different VPN servers should be instantiated in different nodes. Regarding additional considerations, health check capabilities must be present in the same host of each VPN server to supervise its correct performance. Finally, VPN clients have to be instantiated in hosts that belong to other networks.

Enabler component	***Load Balancer***
id	*T42E8_balancer*
Description and main functionality	*This component will be in charge of distributing the traffic load among the VPN server instances available in the site's network. A failover mechanism will be implemented through it, so in case one server is down an extra instance is initiated and configured during its recovery.*
Target node/s	*High-tier Edge node*
Candidate technologies	*HAProxy (other alternatives such as ngingx and Traefik will be evaluated)*

Enabler component	***VPN Server***
id	*T42E8_server*
Description and main functionality	*This component allows clients to connect to a secure, private network. All data is encrypted by the VPN protocols the server is configured with, so the connection is protected (not even Internet Service Providers can monitor it).*
Target node/s	*High-tier Edge node*
Candidate technologies	*Wireguard server (in case it lacks a required feature, other alternatives will be studied)*

Enabler component	***Health Checker***
id	*T42E8_checker*
Description and main functionality	*This component will validate the health of each server in response to periodic queries from the node that contains the load balancer.*
Target node/s	*High-tier Edge node*
Candidate technologies	*Custom Python component*

Enabler component	***VPN Client***
id	*T42E8_client*
Description and main functionality	*This component is installed in each host that wants to connect to the private network. It has the role to initiate the connection with the VPN server, and it is able to decrypt the data that is send by the VPN server over the public (or non-trusted private) network.*
Target node/s	*(External) Edge node or Smart IoT Device*
Candidate technologies	*Wireguard client (in case it lacks a required feature, other alternatives will be studied)*

B.9 - Semantic Repository enabler

Table 12. General information of the Semantic Repository enabler

Enabler	***Semantic Repository Enabler***
id	*T43E1*
Owner and support	*SRIPAS, MOW, PRODEVELOP, Konecranes, Ford-Werke*
Description and main functionalities	This enabler offers a "Nexus" for data models and ontologies, that can be uploaded in different file formats, and served to users with relevant documentation. This enabler is aimed to support files that describe data models or support data transformations, such as ontologies, schema files, semantic alignment files etc. Additionally, human-readable documentation for the models will be served. Offered files, their metadata and documentation are, in principle, public, so that this enabler

Enabler	***Semantic Repository Enabler***
	could be used as support for a shared semantic ecosystem. The secure access restrictions may be more tightly controlled with an API manager in future versions. In short, the core function of this enabler is to be a database of data models and ontologies, with public read access. In summary, it supports: • Versioning: different versions of data models • Ownership: only the data model owner may update a data model • Provision & search: Data models are public and browseable • Documentation: Documentation provided by data model owner is served
Plane/s involved	*Data Management Plane*
Relation with other enablers	• *T43E8: Long-term data storage* • *T53E1: Identity Manager Enabler* • *T53E2: Authorisation Enabler*
Requirements mapping	• *R-C-2: Data governance (manages data models for different data sources)* • *R-P2-15: BIM data models and interoperability compliance (stores relevant data models)* • *R-P3A-2: Data models' compliance (stores relevant standard data model)*
Use case mapping	*N/A*
Required components	*API Server, Persistent Storage, User Management*

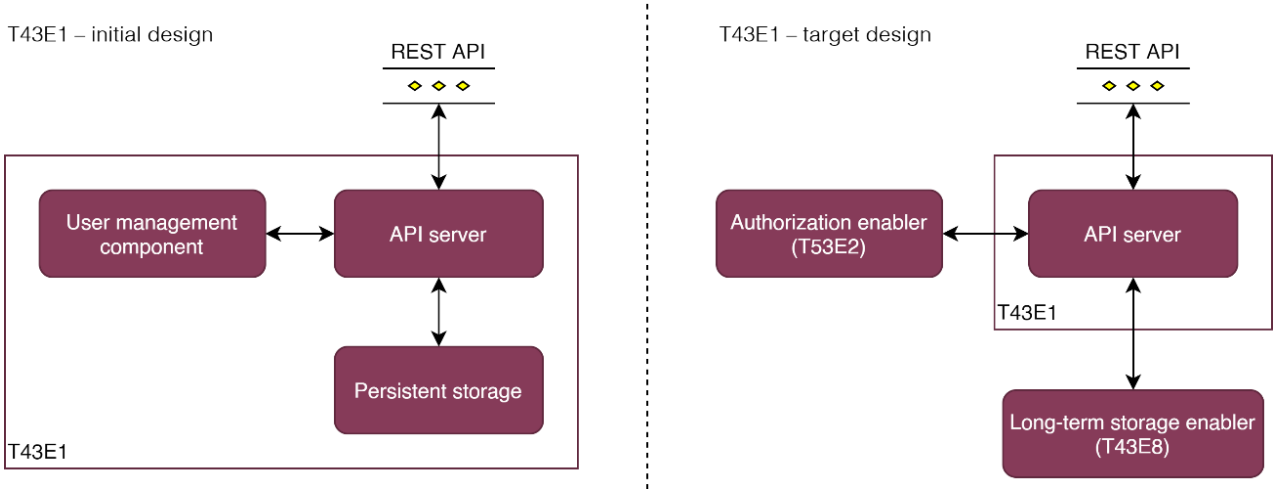


Figure 21. High-level diagram of the Semantic Repository enabler

Endpoints

Many of the endpoint URLs contain the version id fragment, which is expected to be numeric with dots separating the components, of which there may be three at most. For example: '1', '1.1', '1.2.3'. To specify the latest available version, 'latest' should be used as the version id.

Method	URL	Payload (if needed)	Description	Response format
GET	**[model endpoint] /**		Lists available repositories	JSON
POST/ PUT/ DELETE	**[model endpoint] /{repo id}**	JSON (PUT, POST)	Creates (POST), updates (PUT), or removes (DELETE) a specified repository and its settings	
GET	**[model endpoint] /{repo id}**		Returns the settings of the repository and lists models with versions and formats in it	JSON

Method	URL	Payload (if needed)	Description	Response format
GET	**[model endpoint] /{repo id} /{model id}**		Returns the metadata of the model and lists the available versions and formats of the given model	JSON
POST/ PUT/ DELETE	**[model endpoint] /{repo id} /{model id} /{version id}**	JSON (PUT, POST)	Creates (POST), updates (PUT), or removes (DELETE) a version of a model with its metadata (version, creation data, modification date, description etc.)	
GET	**[model endpoint] /{repo id} /{model id} /{version id} /meta**		Returns the metadata of the given model version	JSON
POST/ PUT/ DELETE	**[model endpoint] /{repo id} /{model id} /{version id} ?format={data format}**	Flat file (PUT, POST)	Sets (POST), updates (PUT), or removes (DELETE) a specified file from the server	
GET	**[model endpoint] /{repo id} /{model id} /{version id} ?format={data format}**		Returns the specified version of a data model in a given format. e.g. /raul/saref/1.1/content?format=rdfxml returns a 'saref' model from repository 'raul' in version 1.1 in file format RDF/XML	Flat file in a given file format
POST/ PUT/ DELETE	**[model endpoint] /{repo id} /{model id} /{version id} /doc**	Markdown documentation file (PUT, POST)	Sets (POST), updates (PUT), or removes (DELETE) markdown documentation	
GET	**[model endpoint] /{repo id} /{model id} /{version id} /doc**		Returns the documentation for a model	Markdown documentation file

Starting with an empty database, we want to create a repository named 'tea' and store in it a data model named 'pu-erh' in two formats. The process to do this is as follows:

1. **POST** /tea with JSON specifying the settings for the new repository.
2. **POST** /tea/pu-erh/1.0/ with JSON specifying the metadata of the data model.
3. **POST** /tea/pu-erh/1.0/content?format=rdfxml with the raw XML content.
4. **POST** /tea/pu-erh/1.0/content?format=ttl with the raw Turtle content.

Access to the endpoints in the initial version will be based on simple authentication. This can be later delegated to the authentication and authorisation enablers. Additionally, roles may be defined to control which users can access what functionalities. For example, repository and model creation may be restricted to only selected users.

Components

Enabler component	*API Server*
id	*T43E1_API*
Description and main functionality	*The API Server that manages requests on different endpoints. It is fully stateless.*
Node type/s*	*Any*
Candidate technologies	*Akka HTTP, HTTPS*

Enabler component	*Persistent Storage*
id	*T43E1_storage*
Description and main functionality	*Persists model and model documentation files.*
Target node/s	*Any*
Candidate technologies	*Postgres, MySQL*

Enabler component	*User Management*
id	*T43E1_management*
Description and main functionality	*Manages access rights to secured endpoints. May be needed for testing alpha versions. Will be eventually replaced by functions offered by authentication and authorisation enablers.*
Target node/s	*Any*
Candidate technologies	*Java, Scala*

B.10 - Semantic Translation enabler

Table 13. General information of the Semantic Translation enabler

Enabler	*Semantic Translation Enabler*
id	*T43E2*
Owner and support	*SRIPAS, UPV*
Description and main functionalities	Semantic Translation enabler offers a configurable service to change the contents of semantically annotated data in accordance with translation rules – so called "alignments", or alignment files. The core use-case, around which this enabler is designed, is to move data between semantic ontologies (which can be thought of as data schemas or vocabularies) that can express the same information, without changing the meaning of the information. Flexibility of design and expressivity of configuration files allow for other use-cases, such as semantic reduction (removing selected information, e.g. because of privacy reasons), further annotation (adding additional information based on data content and possibly external variables), or even encoding or encrypting selected data items into a serialised form. The Semantic Translator supports RDF as the only modern standard for semantic data. By design it supports and promotes the "core ontology" design, in which data transformations are always unidirectional and done to, or from a central ontology, and paired into "translation channels" to achieve bidirectional transformations. In this manner, n-to-n translations can be easily implemented, and the cost of including a new data model in existing deployments does not grow exponentially.

Enabler	***Semantic Translation Enabler***
	Translation services are offered as a "static" API for batch data, or through a publish-subscribe broker for streaming data.
Plane/s involved	*Data Management Plane*
Relation with other enablers	• *T53E1: Identity Manager Enabler* • *T53E2: Authorisation Enabler*
Requirements mapping	• *R-C-1: Data sovereignty (provides semantic interoperability for data sources)* • *R-P2-15: BIM data models and interoperability compliance (provides interoperability)* • *R-P3A-1: Monitored Data channels (provides interoperability, if needed)*
Use case mapping	*N/A*
Required components	*API Server, Streaming broker, Translation channel manager, Alignment Application Core, Storage*

Semantic translation has a supporting role in any data transfer that requires the addition of semantic interoperability. It includes any use-case, in which sender and receiver of data use different semantics. In such cases, the semantic translation enabler can transform the data and deliver it with semantics compatible with the receiver. Application of the semantic translation enabler within the scope of concrete use-cases depends not only on the use-case description, but also on the heterogeneity of data sources and consumers present. The semantics required by systems and devices used in the pilots is not defined at this time. It is therefore not possible to authoritatively define a closed list of pilot requirements and use-cases, in which the Semantic Translation Enabler will for sure be involved.

For example: Requirement BS-P2-4: (Health and safety inspection support) and use-case UC-P2-7: (Health and safety inspection support) may require semantic translation (or semantic annotation) to present relevant data in format and semantics understood by the system, that an OSH inspector uses. In case of semantic and syntactic compatibility (e.g. by virtue of using a common data standard), the semantic enablers will not be needed. Similarly, in R-P2-11 (Geofencing) semantic transformation of geoposition and geofencing data may be required.

For this reason, the semantic enablers (at this point) address mostly common requirements, and not pilot-specific ones.

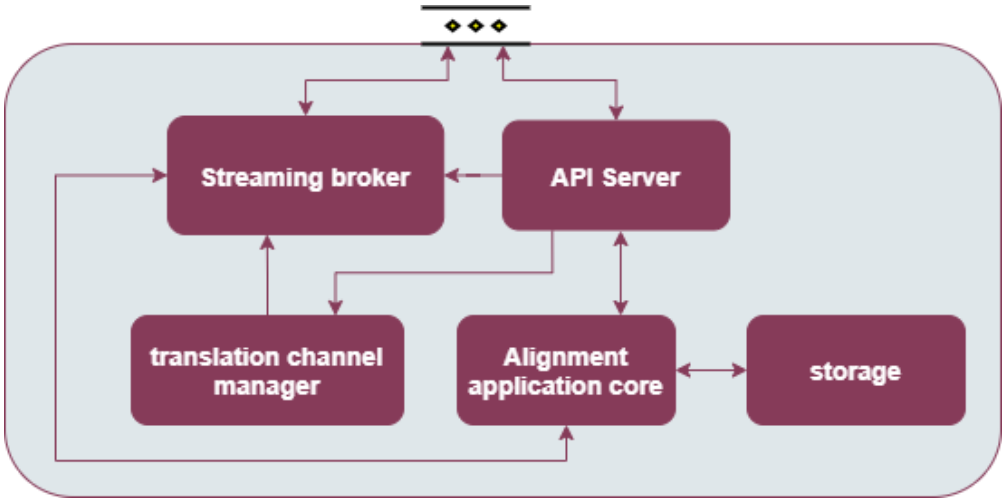


Figure 22. High-level diagram of the Semantic Translation enabler

Endpoints

The following endpoints are expected to be improved during the next phase, which will be documented in D4.2.

Translation and configuration REST API.

Method	URL	Payload (if needed)	Description	Response format
POST	**/alignments**	IPSM-AF 2.0 Alignment file	Upload an alignment	
GET	**/alignments**		Retrieves list of available alignments	JSON
GET, DELETE	**/alignments /{name}/ {version}**		Retrieves or deletes a single alignment file	IPSM-AF 2.0
POST	**/channels**	JSON Channel configuration	Creates a translation channel using existing alignments	
GET	**/channels**		Lists available translation channels	
DELETE	**/channels /{channel id}**		Removes a translation channel	
POST	**/translate /{alignment id} /{alignment 2 id} /{alignment 3 id}**	RDF	Translates a single piece of data through a chain of alignments. The length of the chain may be limited for performance and security reasons.	
POST	**/translate/ {channel id}**	RDF	Translates a single piece of data through alignments that are used by a translation channel. The data is returned directly, without actually entering the translation channel.	

Responsive streaming interface.

Translation channels are mini data pipelines implemented within the enabler, that expose two broker topics – one for input, another one for output. When clients write to the input topic, translated data appears on the output topic. The broker supports the reactive streaming manifesto, so more complicated streaming pipelines may be built by combining Semantic Translator topics to other streams.

Method	URL	Payload (if needed)	Description	Response format
Subscribe/unsubscribe	**{output topic id}**		Subscribe to the output topic of a translation channel to receive messages.	
Send	**{input topic id}**	RDF	Send data to a translation channel. It will be translated and appear on the output topic for that channel.	RDF
Pushback	**{output topic id}**		Implements the reactive streaming principles, allowing clients enough time to process all messages.	

Components

Enabler component	***API Server***
id	*T43E2_API*
Description and main functionality	*Main entry point for configuration of the enabler, and for translation via REST API.*
Target node/s	*Any*
Candidate technologies	*Akka http*

Enabler component	***Streaming broker***
id	*T43E2_broker*
Description and main functionality	*Manages data flow between topics from translation channels and internal stream processors.*
Target node/s	*Any, preferably close to data source or sink (edge or far edge)*
Candidate technologies	*Apache Kafka, Akka streams*

Enabler component	***Translation channel manager***
id	*T43E2_translation*
Description and main functionality	*Manages lifecycle of translation channels*
Target node/s	*Any*
Candidate technologies	*Scala*

Enabler component	***Alignment application core***
id	*T43E2_alignment*
Description and main functionality	*The core component that performs semantic translation by applying alignment files to data. Will be used by the REST API, as well as by the streaming broker. Once configured, becomes stateless, so can be scaled and deployed in multiple nodes to increase performance.*
Target node/s	*Any, preferably close to data source or sink (edge or far edge)*
Candidate technologies	*Scala, Apache Jena*

Enabler component	***Storage***
id	T43E2_storage
Description and main functionality	Stores loaded alignment files and enabler configuration
Target node/s	Any
Candidate technologies	Scala, PostgreSQL

B.11 - Semantic Annotation enabler

Table 14. General information of the Semantic Annotation enabler

Enabler	***Semantic Annotation Enabler***
id	*T43E3*
Owner and support	*SRIPAS, PRODEVELOP, CERTH*
Description and main functionalities	This enabler offers a syntactic transformation service, that annotates data in various formats and lifts it into RDF. Full list of formats is yet to be decided and the first version will support JSON. The enabler is lightweight and stateless, so that it may have many independent deployments. The core functionality is designed to be integrated into a pipeline before the Semantic Translation enabler, which requires the use of RDF. In essence, using Semantic Annotator enables usage of the Semantic Translator with formats other than RDF.
Plane/s involved	*Data Management Plane*

Enabler	*Semantic Annotation Enabler*
Relation with other enablers	*Probably can be entirely standalone*
Requirements mapping	• *R-C-1: Data sovereignty (provides syntactic interoperability for data sources)* • *R-P2-15: BIM data models and interoperability compliance (provides interoperability)* • *R-P3A-1: Monitored Data channels (provides interoperability, if needed)*
Use case mapping	*N/A*
Required components	*API Server, Transformer*

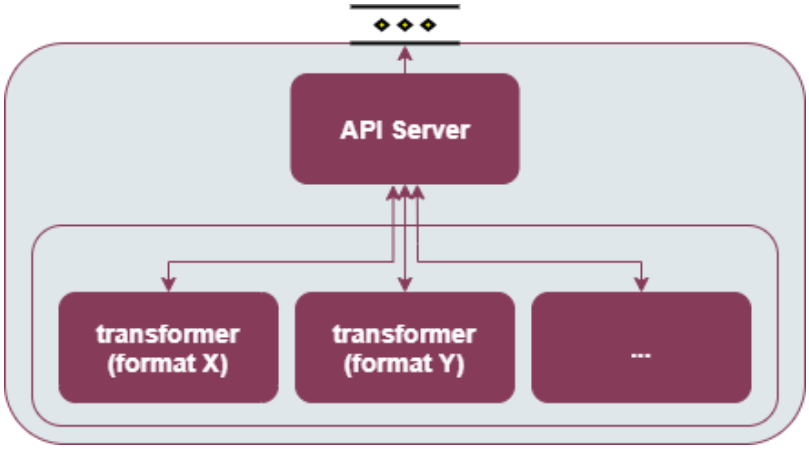


Figure 23. High-level diagram of the Semantic Annotation enabler

The planned implementation will include a "generic annotator" as an implementation of the annotator template (delivered as a separate product). The generic annotator will naively transform data to and from RDF annotated with a generic "syntactic" ontology. Support is planned for JSON, XML and CSV. At least the path to RDF will be supported. Bidirectional transformation is still researched.

A template for the annotator will be a set of interfaces, that must be individually implemented for deeper support of semantic annotation of specific data. The enabler is designed to be fully stateless.

Endpoints

Method	URL	Payload (if needed)	Description	Response format
POST	**/annotate /{format id}**	Message to annotate	Semantically annotates a message	RDF
POST	**/annotate /{format id}/custom**	Message to annotate and custom annotation rules	Semantically annotates a message overriding existing transformers with custom annotation rules	RDF
GET	**/annotate**		Retrieves the list of supported formats	
GET	**/annotate /{format id}**		Retrieves the annotation rules for a given format	

Components

Enabler component	*API Server*
id	T43E3_API
Rationale	A lightweight component to direct requests to and from transformers.
Target node/s	Any
Candidate technologies	Akka http

Enabler component	*Transformer*
id	T43E3_transformer
Rationale	Serves as a module to translate between a specific set of messages (e.g. JSON for a given schema) and RDF.
Target node/s	Any
Candidate technologies	Scala, Apache Jena

B.12 - Edge data broker

Table 15. General information of the Edge data broker

Enabler	***Edge Data Broker***
id	*T43E7*
Owner and support	*ICCS, UPV, PRODEVELOP, NEWAYS, CERTH*
Description and main functionalities	*It enables the efficient management of data demand and data supply from/to the Edge Nodes. It optimally distributes data where it is needed for application, services and further analysis. Data distribution is based on reported demand and available resources at the Edge Nodes. It provides: subscriptions and messages between the broker and the Edge Nodes; management of message scheduling, routing and delivery; common interfaces for searching and finding information*
Plane/s involved	*Data Management Plane*
Relation with other enablers	• *T44E2: Semantic translation enabler* • *T44E4: OpenAPI Management* • *T43E8: Long-term Storage enabler* • *T44E6: MR enabler* • *SELF13: (geo) Localisation* • *T54E4: DLT-based Federated Learning enabler* • *T55E1: Devices management* • *T55E3: Workflow between enablers based on events, messaging exchange, or others*
Requirements mapping	• *R-C-2 Data governance (This enabler provides effective and efficient use of data)*
Use case mapping	*All*
Required components	*Distributed Load Balancing, Data Management, Integration with other Broker, Data Routing*

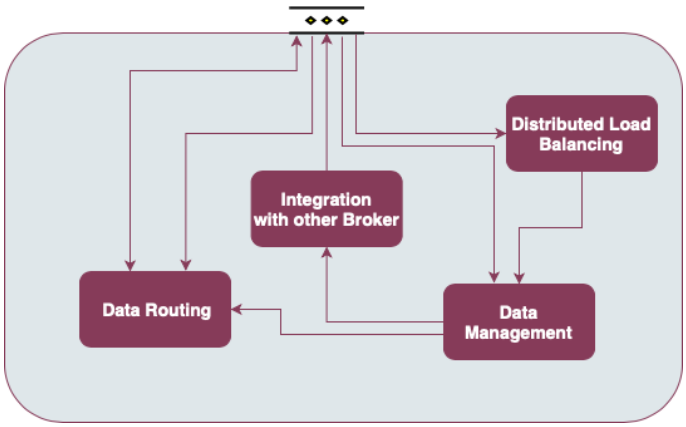


Figure 24. High-level diagram of the Edge data broker

Endpoints

Method	URL	Description
GET	**/health**	The health check will return: 200 when the Edge Data Broker is accepting connections and is joined with the cluster (for clustered setups). 503 will be returned in case any of the above two conditions are not met.
GET	**/metrics**	Configurable endpoint that provides multiple metrics relevant to monitoring and alerting
Pub Sub	**Multiple topics**	The Pub/Sub interface of the Broker can be used to send/receive data among devices and applications

Components

Enabler component	***Load Balancer***
id	*T43E7_LoadBalancer*
Description and main functionality	*The load balancer component is used to ensure that client connections are distributed among the nodes so that each node has the same number of connections. Distribution load balancing strategies such as random, source hashing etc. are used to declare which node will be used to route the incoming connection.*
Target node/s	*Gateway*
Candidate technologies	*MQTT*

Enabler component	***Data Routing***
id	*T43E7_DataRouting*
Description and main functionality	*This component allows the edge data broker to dynamically choose the routing map using customised plugins that were initiated during start up. This component also provides the mechanism to distribute messages to a set of subscribers (i.e. application or devices that consumes data) in a shared subscription topic, such that each message is received by only one subscriber (Shared Subscription).*
Target node/s	*Gateway*
Candidate technologies	*MQTT*

Enabler component	***Data Management***
id	*T43E7_DataManagement*
Description and main functionality	*The data management component provides local intelligence capabilities to the Edge Data Broker, improving the efficient management of the data. The received data is filtered in real time and routed based on conditional events such that the subscriber will receive only meaningful data.*
Target node/s	*Gateway*
Candidate technologies	*MQTT, Lua Scripts*

Enabler component	***Integration with other brokers***
id	*T43E7_Integrationwithotherbrokers*
Description and main functionality	*This component enables integration capabilities with other brokers, by connecting two different MQTT brokers to each other. It allows for example that a topic tree of a remote broker becomes part of the topic tree on the Edge Data Broker.*
Target node/s	*Gateway*
Candidate technologies	*MQTT*

B.13 - Long-term data storage enabler

Table 16. General information of the Long-term Data Storage enabler

Enabler	***Long-term Storage Enabler***
id	*T43E8*
Owner and support	*PRODEVELOP, SRIPAS, UPV, CERTH*
Description and main functionalities	*The role of this enabler is to serve as a secure and resilient storage, offering different storage sizes and individual storage space for other enablers (which could request back when they are being initialising in Kubernetes pods). It also guarantees that the data will be kept safe, in face of various kinds of unauthorised access requests, or hardware failures, by only allowing access to the data once the Identity Manager and the Authorisation enablers have confirmed their access rights.*
Plane/s involved	*Data Management Plane*
Relation with other enablers	• *T53E1: Identity Manager Enabler* • *T53E2: Authorisation Enabler* • *T43E1: Semantic Repository* • *T44E1: Business KPI reporting enabler* • *T44E2: Performance and usage diagnosis enabler* • *T44E4: OpenAPI management* • *T43E7: Edge data broker*
Requirements mapping	• *R-C-2: Data governance* • *R-C-3: Compliance with legal requirements on data protection* • *R-C-6: Data persistence and trust* • *R-P3A-1: Monitored Data channels* • *R-P3A-3: OEM fleet data storage* • *R-P3A-6: Active monitoring mode initiation by the OEM software engineer capability* • *R-P3B-20: Information pre-fetching*
Use case mapping	*All*
Required components	*LTSE Gateway, LTS noSQL cluster, LTS noSQL Index, LTS noSQL Shard, LTS noSQL Document, LTS SQL Server, LTS SQL Database*

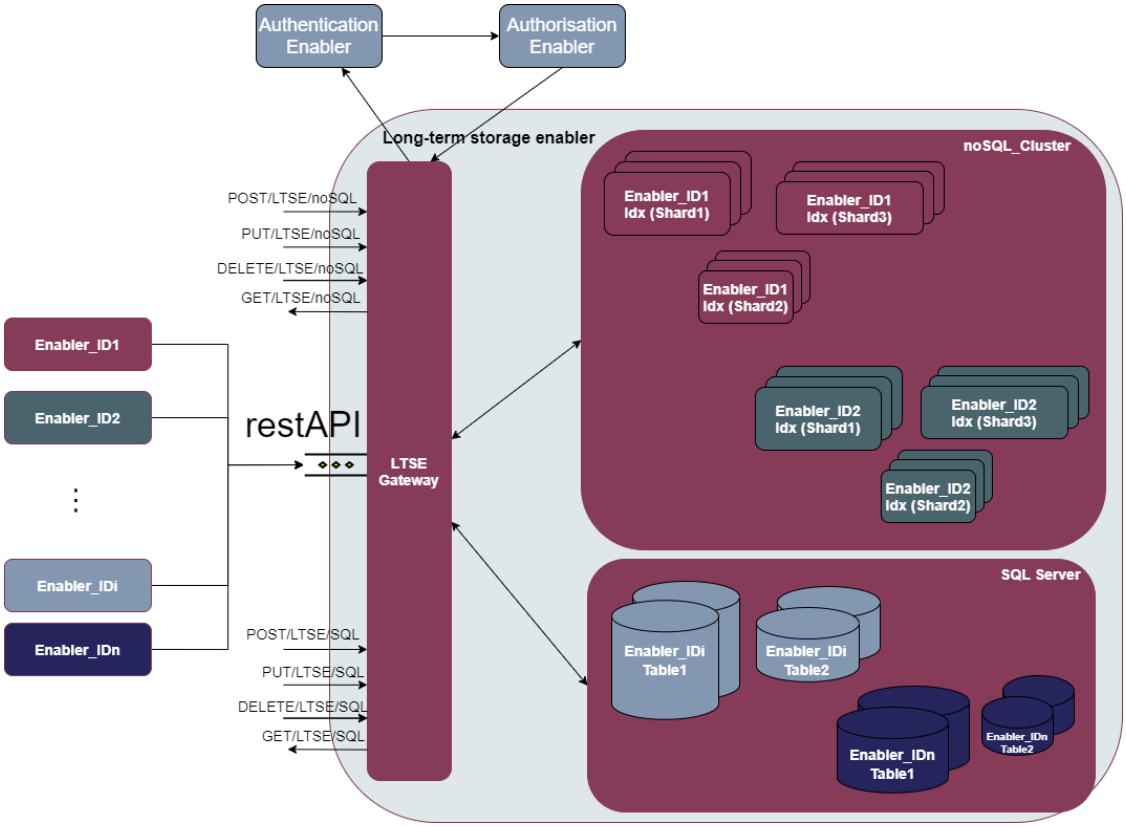


Figure 25. High-level diagram of the Long-term Data Storage enabler

Endpoints

Method	URL	Description
POST	**/LTSE/noSQL/<enabler_ID>**	Creates a new <enabler_ID> NoSQL_node in the node_SQL_Cluster
PUT	**/LTSE/noSQL/<enabler_ID>**	Updates a <enabler_ID> NoSQL_node in the LTS_SQL_Cluster
DELETE	**/LTSE/noSQL/<enabler_ID>**	Removes an <enabler_ID> NoSQL_node from the LTS_SQL_Cluster
GET	**/LTSE/noSQL/<enabler_ID>**	Retrieves persistent storage information from the LTS_NoSQL_node of the specific <enabler_ID>
POST	**/LTSE/SQL/<enabler_ID>**	Creates a new <enabler_ID> SQL database in the LTS_SQL_Server
PUT	**/LTSE/SQL/<enabler_ID>**	Updates the <enabler_ID> SQL database of the LTS_SQL_Server
DELETE	**/LTSE/SQL/<enabler_ID>**	Removes an <enabler_ID> SQL database from the LTS_SQL_Server
GET	**/LTSE/SQL/<enabler_ID>**	Retrieves persistent storage information from the LTS_SQL_server of the specific <enabler_ID>

* The inner noSQL APIs of the Cluster, Node, Index and Document as well as a security API will be also exposed.

Components

Enabler component	***LTSE Gateway***
id	*T43E8_gateway*
Description and main functionality	The LTSE gateway acts as a proxy in order to identify if the external enabler data is collected whether as SQL or noSQL format. To do so, restAPI calls with append SQL/noSQL format are available. The LTSE Gateway will, based on the restAPI request, communicate with the SQL or noSQL cluster accordingly. Additionally, the LTSE Gateway will be the responsible of guaranteeing the quotes of the enablers not surpassing an agreed SLA size. Clauses such as maximum storage per enabler, or the maximum period of storing persistent enabler data are also configured in the LTSE gateway, which in turn are managed via the Orchestration Enablers deployment enabler. The LTSE gateway is

Enabler component	*LTSE Gateway*
	also the entrance gate to the LTSE from external enablers, given that the administration authorisation policies have been confirmed through the Identity manager and the Authorisation enabler.
Target node/s	Any
Candidate technologies	Scala, Apache Jena

Enabler component	*LTS noSQL Cluster*
id	*T43E8_noSQL_cluster*
Description and main functionality	A group of one or more LTS_noSQL nodes instances that are connected together, and carries out the distribution of tasks, searching and indexing, across all the noSQL nodes. Any time an instance of LTS noSQL is started from the LTS gateway, a noSQL node is started. Every noSQl node in the LTE_noSQL_Cluster can handle HTTP and transport traffic by default with the external enablers through the LTS gateway. The transport layer is used exclusively for communication between nodes; the HTTP layer is used by REST clients. New noSQL_Nodes can be added to the LTS_noSQL_cluster in order to increase capacity. noSQL_Nodes can have different roles (which can be user restricted by setting the *node.roles* field in LTS_noSQL_Cluster config YML file): • Master: elected as responsible for handling LTS_noSQL_cluster management and state • Data: hold the spaces (shards) that indexed the data documents, and handle data related operations like CRUD, search, and aggregations.
Target node/s	*Cloud-server*
Candidate technologies	*ElasticStack*

Enabler component	*LTS noSQL Index*
id	*T43E8_noSQL_index*
Description and main functionality	An LTS_noSQL_Index is an optimised collection of documents, and each document is a collection of fields, which are the key-value pairs that contain the associated noSQL enabler data. An LTS_noSQL_index is a logical grouping of one or more physical shards, where each shard is actually a self-contained LTS_noSQL_index. By default, once the LTS gateway has connected the noSQL_enabler with the LTS_noSQL_Cluster, the noSQL_enabler indexes all data in every field and each indexed field has a dedicated, optimised data structure. It is also envisioned that the noSQL_LTS enabler is also schema-less, which means that documents can be indexed without explicitly specifying how to handle each of the different fields that might occur in its document.
Target node/s	*Cloud-server*
Candidate technologies	*ElasticStack*

Enabler component	*LTS noSQL Shard*
id	*T43E8_noSQL_shard*
Description and main functionality	By distributing the documents in a LTS_noSQL_Index across multiple shards, and distributing those shards across multiple nodes, LTS enabler can ensure redundancy, which both protects against hardware failures, and increases query capacity as noSQL_Nodes are added to a cluster. As the LTS_noSQL_Cluster grows (or shrinks), it automatically migrates shards to rebalance the cluster. There are two types of shards: primaries and replicas. Each document in a LTS_noSQL_Index belongs to one primary shard. A replica shard is a copy of a primary shard. Replicas provide redundant copies of the indexed data to protect against hardware failure and increase capacity to serve read requests like searching or retrieving a document. The number of primary shards in a

Enabler component	*LTS noSQL Shard*
	LTS_noSQL_Index is fixed at the time that it is created, but the number of replica shards can be changed at any time, without interrupting indexing or query operations. However, the larger the shard size, the longer it takes to move shards around when rebalancing the LTS_noSQL_Cluster. On the other hand, having lots of small shards makes the processing per shard faster, but more queries means more overhead.
Target node/s	*Cloud-server*
Candidate technologies	*ElasticStack*

Enabler component	*LTS noSQL Document*
id	*T43E8_noSQL_document*
Description and main functionality	Complex data structures that have been serialised as JSON documents. When there are multiple LTS_noSQL_Nodes in an LTS_noSQL_Cluster, stored documents are distributed across the cluster and can be accessed immediately from any node.
Target node/s	*Cloud-server*
Candidate technologies	*ElasticStack*

Enabler component	*LTS SQL Server*
id	*T43E8_SQL_server*
Description and main functionality	It manages the SQL database cluster, which is a collection of database files from accepted enablers in a single instance. It performs database actions on behalf of the enablers as well. The SQL_Server can handle multiple concurrent connections from external enablers via the LTSE_Gateway. In general, the full hierarchy is: SQL_Cluster, SQL_Database, schema, table. For High Availability, a master database with one or more standby servers can be setup, but a master – master architecture can also be deployed. The standby servers' databases can remain synchronised with the master server's database. If the main server fails, the standby contains almost all of the data of the main server and can quickly be turned into the new master database server.
Target node/s	*Cloud-server*
Candidate technologies	*PostgreSQL, PostgREST/psql-api*

Enabler component	*LTS SQL Database*
id	*T43E8_SQL_DB*
Description and main functionality	Inside the SQL_Server are multiple databases, which are isolated from each other but can access cluster-level objects. Inside each database are multiple schemas, which contain objects like tables and functions. When connecting to the SQL_database, an enabler must specify the specific name in its connection request. It is not possible to access more than one database per connection. Database-level security has two components: access control managed via the Identity Manager through LTSE_Gateway, and authorisation control, managed via the Authorisation Enabler through LTSE_Gateway.
Target node/s	*Cloud-server*
Candidate technologies	*PostgreSQL*

B.14 - Tactile Dashboard enabler

Table 17. General information of the Tactile Dashboard enabler

Enabler	***Tactile dashboard enabler***
id	*T44E1*
Owner and support	PRODEVELOP, UPV, SRIPAS
Description and main functionalities	The Tactile Dashboard enabler has the capability of representing data stored in the ASSIST-IoT pilots, through meaningful combined visualisations in real time. It also provides (aggregates and homogenises) all the User Interfaces for the configuration of the different ASSIST-IoT enablers, and associated components. The tactile dashboard is divided into two components: Frontend, Backend (described in the Dashboard components tables).
Plane/s involved	Application and services plane
Relation with other enablers	• *T44E2: Business KPI reporting Enabler* • *T44E3: Performance and usage diagnosis enabler* • *T44E4: OpenAPI management enabler* • *T53E1: Authorisation enabler* • *T53E2: Identity Manager* • *T55EX: (task has not started, so that not formally documented yet) – Orchestration of enablers deployment*
Requirements mapping	• *R-C-3: Compliance with legal requirements on data protection* • *R-P1-6: Terminal data access* • *R-P1-15: Alignment exposure* • *R-P2-11: Geofencing* • *P-P2-14: Evacuation instructions* • *R-P3A-7: Active monitoring mode initiation by the aftersales service technician capability* • *R-P3A-8: Active monitoring initiation by the driver capability* • *R-P3A-10: Vehicle dashboard notifications* • *R-P3B-23: Business frontend* • *R-P3B-24: Interactive image annotation support*
Use case mapping	*This enabler is inherent to the ASSIST-IoT ecosystem and, therefore, it should be present at all pilots, otherwise it would not be possible to neither configure the ASSIST-IoT platform nor expose the ASSIST-IoT platform to end-users.*
Required components	*Dashboard frontend, Dashboard backend*

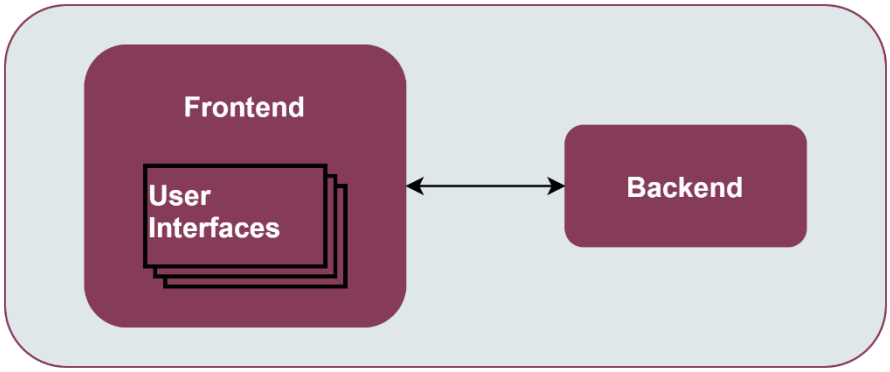


Figure 26. High-level diagram of the Tactile Dashboard enabler

Components

Enabler component	***Dashboard frontend***
id	*T44E1_frontend*
Description and main functionality	*Offers a web application that exposes the UI through which the user interacts with the ASSIST-IoT platform.*
Target node/s	*Cloud server*
Candidate technologies	*Custom Javascript front-end framework based on Vue.js*

Enabler component	***Dashboard backend***
id	*T44E1_backend*
Description and main functionality	*REST API created to interact from the frontend with the external entities offering different functionalities of ASSIST-IoT platform. To do so, API requests will be available by providing a single-entry point in the dashboard, over which all enablers' APIs must be exposed for the sake of platform's management (e.g., SDN controller API endpoints, LTSE API endpoints, etc.). Furthermore, before allowing access to end-users to the ASSIST-IoT platform, the dashboard backend will connect to the Identity Manager and Authorisation Enabler in order to ensure that the access rights and associated roles are properly authorised.*
Target node/s	*Cloud server*
Candidate technologies	*Custom JavaScript framework based on Vue.js*

B.15 - Business KPI reporting Enabler

Table 18. General information of the Business KPI reporting enabler

Enabler	***Business KPI reporting Enabler***
id	*T44E2*
Owner and support	*PRODEVELOP, UPV, SRIPAS*
Description and main functionalities	*All valuable Key Performance Indicators (KPIs) desired by the end-user should be available for representation in graphs, reports, etc. This enabler will allow to embed them as User Interfaces within the tactile dashboard. It will facilitate the visualisation and combination of charts, tables, maps and other visualisation graphs in order to search for hidden insights. The enabler is composed of a server component containing the business logic engine, accompanied with a UI component that defines the graphical UI that users interact with, and a Command Line Interface (CLI) tool especially designed for developers.*
Plane/s involved	*Application and services plane*
Relation with other enablers	• *T43E4: Edge data broker Enabler* • *T43E8: Long-term Storage Enabler* • *T44E1: Tactile dashboard Enabler* • *T55EX: (task has not started, so that not formally documented yet) – Orchestration of enablers deployment*
Requirements mapping	• *R-P1-6: Terminal data access* • *R-P2-1: Personal location tracking* • *R-P2-2: Construction plant location tracking* • *R-P2-3: Smart wristband for construction workers* • *R-P2-4: Continuous authentication for wristband* • *R-P2-7: Monitoring the weather conditions at the construction site* • *R-P2-8: Personal cooling system* • *R-P2-10: Motion pattern monitoring and analysis*

Enabler	***Business KPI reporting Enabler***
	• *R-P2-11: Geofencing* • *R-P2-14: Evacuation instructions* • *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-10: Vehicle dashboard notifications* • *R-P3B-15: Automatic Defect Detection* • *R-P3B-20: Information pre-fetching* • *R-P3B-23: Business frontend*
Use case mapping	*This enabler fits all those use cases in which business analysis and reports are needed. The following use cases have been in principle identified:* • *UC-P1-2: Container Handling Operations reporting* • *UC-P2-1: Workers' health and safety assurance* • *UC-P2-7: Health and safety inspection support* • *UC-P3A-1: Fleet in-service emissions verification* • *UC-P3B-1: Vehicle's exterior condition documentation* • *UC-P3B-2: Exterior defects detection support*
Required components	*Business Server, Business UI, Business CLI*

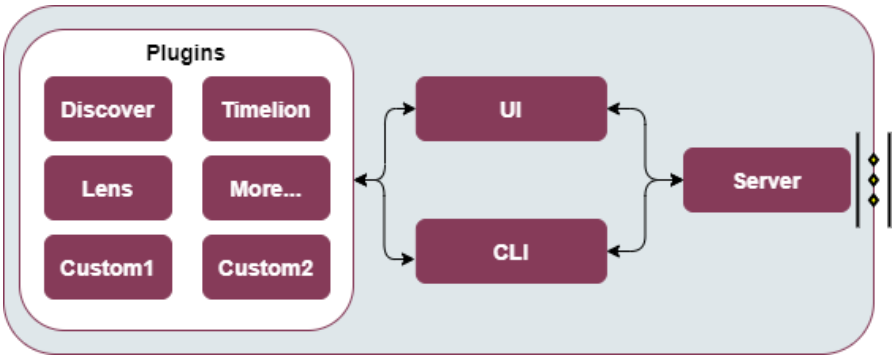


Figure 27. High-level diagram of the Business KPI reporting enabler

Components

Enabler component	***Business KPI Server***
id	*T44E2_Server*
Description and main functionality	*The Server collects data that have been passed through the Semantics flow from data collectors (either Long-Term Storage or Edge Data Broker enablers) into a dedicated database and provides access to it to the UI and CLI components via an internal REST API. Functionality is implemented through modular plugins (Discover, Tag, Lens, Maps, Timelion, etc.), which contain the business logic and communicate with the UI and CLI components. Custom plugins can also be easily integrated if needed, thanks to having a modular approach*
Target node/s	*Cloud server*
Candidate technologies	*Grafana, Kibana*

Enabler component	***Business KPI UI***
id	*T44E2_UI*
Description and main functionality	*When the end-user accesses the Business KPI enabler via the Dashboard Graphical User Interface, the UI component loads all server plugins that comprise the core functionalities of the Business KPI enabler. It also loads the utils module, which is a collection of helper*

Enabler component	***Business KPI UI***
	functions and objects to ease the developer work. Hence, the UI component provides an editor to create and explore interactive visualisations and a set of functionalities to allow end-users (industrial administrators) arrange the visualisations according to their goals.
Target node/s	*Cloud server*
Candidate technologies	*Grafana, Kibana*

Enabler component	***Business KPI CLI***
id	*T44E2_CLI*
Description and main functionality	*The CLI component enables custom plugins built by third party developers to interact with the Business Server, so that it is reachable from the UI to e.g., provide new data aggregation methods, or to visualise new chart types, colour palettes, etc.*
Target node/s	*Cloud server*
Candidate technologies	*Grafana, Kibana*

B.16 - Performance and usage diagnosis enabler (PUD)

Table 19. General information of the Performance and usage diagnosis enabler (PUD)

Enabler	***Performance and usage diagnosis enabler (PUD)***
id	*T44E3*
Owner and support	*PRODEVELOP, UPV, SRIPAS*
Description and main functionalities	*Performance and Usage Diagnosis enabler aims at collecting performance metrics from monitored targets by scraping metrics HTTP endpoints on them, and highlighting potential problems in the ASSIST-IoT platform, so that it could autonomously act in accordance or to notify to the platform administrator to fine tuning machine resources. The PUD enabler consists of multiple components, many of which are optional:* • *The main PUD server, which stores time series data.* • *WebUI used to visualise the collected data.* • *A push gateway for supporting short-lived jobs.* • *An alert manager to handle alerts.*
Plane/s involved	*Application and services plane*
Relation with other enablers	• *T44E1: Tactile dashboard Enabler* • *T44E4: OpenAPI management Enabler* • *SELF11: Self-healing device Enabler* • *SELF14: Monitoring and notifying Enabler* • *T53E4: Security monitoring Enabler* • *T55EX: (task has not started, so that not formally documented yet) – Orchestration of enablers deployment*
Requirements mapping	• *R-C-7: Edge-oriented deployment* • *R-P1-5: Container ID tracking system* • *R-P1-16: Open/Accessible Remote capabilities* • *R-P2-18: Temporary storage* • *R-P2-12: Alerts and notifications minimisation*
Use case mapping	*This enabler is inherent to the ASSIST-IoT platform and, therefore, it should be present at all pilots, otherwise it would not be possible to monitor and analyse the deployed infrastructure and carry out the required countermeasures.*
Required components	*PUD server, PUD WebUI, PUD Push Gateway, PUD AlertManager*

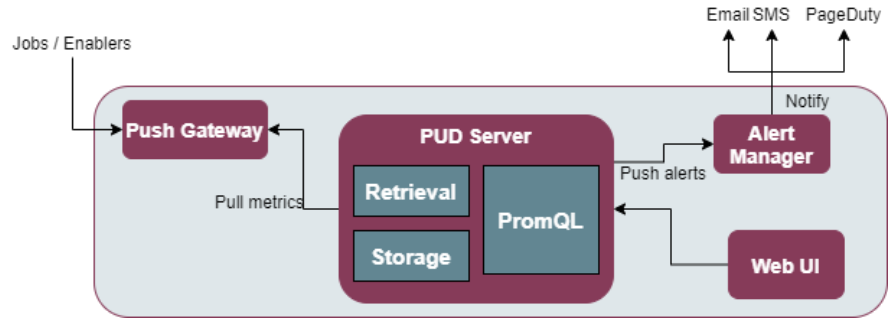


Figure 28. High-level diagram of the Performance and usage diagnosis enabler (PUD)

Components

Enabler component	***PUD Server***
id	*T44E3_Server*
Description and main functionality	*PUD Server scrapes metrics from instrumented enablers on K8S Pods, either directly or via the intermediary PUD Push Gateway (the latter, for short-lived jobs). It stores all scraped samples locally and runs rules over these data to either aggregate and record new time series from existing data or generate alerts. More than one PUD server can be run for the sake of a High Availability monitoring cluster. These servers will be independently of each other, relying only on their local time series database stored data in a custom, highly efficient format local storage for their core functionalities: scraping, rule processing, and alerting. If PUD agrees on using the Prometheus technology, it will provide the query language called PromQL that lets the user select and aggregate time series data in real time. The result of an expression can either be shown as a graph, viewed as tabular data in WebUI, or consumed by external systems via a REST API, such as Instant queries, Range queries, Match queries, etc.*
Target node/s	*High-tier edge node (it has to be able to communicate with all the K8s clusters of the site)*
Candidate technologies	*Prometheus, InfluxDB*

Enabler component	***PUD WebUI***
id	*T44E3_WebUI*
Description and main functionality	*An API consumer that can be used to visualise the collected PUD data via queries. By default, WebUI will be listening on localhost with "admin" / "admin" login credentials.*
Target node/s	*High-tier edge node (it has to be able to communicate with all the K8s clusters of the site)*
Candidate technologies	*Grafana*

Enabler component	***PUD Push Gateway***
id	*T44E3_PushGateway*
Description and main functionality	*The PUD Push Gateway allows batch jobs and enablers to expose their metrics to the PUD Server, since these kinds of jobs may not exist long enough to be scrapped.*
Target node/s	*High-tier edge node (it has to be able to communicate with all the K8s clusters of the site)*
Candidate technologies	*Prometheus Gateway*

Enabler component	***PUD AlertManager***
id	*T44E3_AlertManager*
Description and main functionality	*Alerting with PUD is separated into two parts. Alerting rules in PUD servers send alerts to the Alert Manager, which in turn manages those alerts, sending out notifications via*

Enabler component	*PUD AlertManager*
	methods such as email, on-call notification systems, and chat platforms. Therefore, it takes care of deduplicating, grouping, inhibiting, silencing (if needed), and routing alerts to the correct receiver integrations: • *"Grouping" categorises alerts of similar nature into a single notification. This is especially useful during larger outages when many enablers fail at once and several alerts may be firing simultaneously.* • *"Inhibition" is a concept of suppressing notifications for certain alerts if certain other alerts are already firing. Silences are a straightforward way to simply mute alerts for a given time. A silence is configured based on matchers, just like the routing tree. Incoming alerts are checked whether they match all the equality or regular expression matchers of an active silence. If they do, no notifications will be sent out for that alert.* *PUD Alert Manager is configured via command-line flags and a configuration YAML file. While the command-line flags configure immutable system parameters, the configuration file defines inhibition rules, notification routing and notification receivers.*
Target node/s	*High-tier edge node (it has to be able to communicate with all the K8s clusters of the site)*
Candidate technologies	*Prometheus Alert Manager*

B.17 - OpenAPI management enabler

Table 20. General information of the OpenAPI management enabler

Enabler	*OpenAPI management enabler*
id	*T44E4*
Owner and support	*UPV, SRIPAS, PRODEVELOP*
Description and main functionalities	*The OpenAPI management enabler will be an API Manager that allows enablers that publish their APIs, to monitor the interfaces lifecycles and also make sure that needs of external third parties (including granted open callers), as well as applications that are using the APIs, are being met. Hence, the OpenAPI manager will provide competencies for ensuring successful API usage in developer environment, but also could help end-users for business insights analytics, as well as ASSIST-IoT admins for preserving platform's security and protection. To do so, all ASSIST-IoT enablers should document their API in a common API specification format, which in principle has been identified to be the Swagger-JSON format.* *The provision of a unified API access through the OpenAPI manager consists of several steps. First, an API design document for each ASSIST-IoT enabler is provided through API Specification tool (e.g., OpenAPI definitions). Next, the definitions will be imported to the OpenAPI publisher interface. The user should then subscribe to the APIs through the API subscription web GUI. In principle, the following types of API users with the corresponding set of access and management privileges have been identified:* • *ASSIST-IoT admins: users will full access to all ASSIST-IoT enablers.* • *ASSIST-IoT end-users: users with restricted set of access rights necessary to execute API calls exposed through the front-end interface.* • *ASSIST-IoT External users.*
Plane/s involved	*Application and services plane*
Relation with other enablers	• *T44E2: Tactile Dashboard Enabler (for exposing the enabler in the ASSIST-IoT GUI)* • *T53E1: Authorisation enabler (for roles permission)* • *T53E2: Identity Manager (for access permission)* • *All the rest of enablers (for API management and exposure to external users)*
Requirements mapping	• *R-C-7: Edge-oriented deployment* • *R-P1-6: Terminal data access* • *R-P1-16: Open/Accessible remote capabilities*

Enabler	*OpenAPI management enabler*
	• *R-P1-17: Customisable remote desktop* • *R-P2-5: Wristband pairing with other devices capability* • *R-P3A-11: Connectivity between OEM and fleet*
Use case mapping	*This enabler is inherent to the ASSIST-IoT ecosystem and, therefore, it should be present at all pilots without a specific use case in mind yet. Otherwise, it would not be possible to allow external granted Open Callers to integrate and communicate their developments with ASSIST-IoT platform.*
Required components	*API Gateway, API Portal, API Publisher*

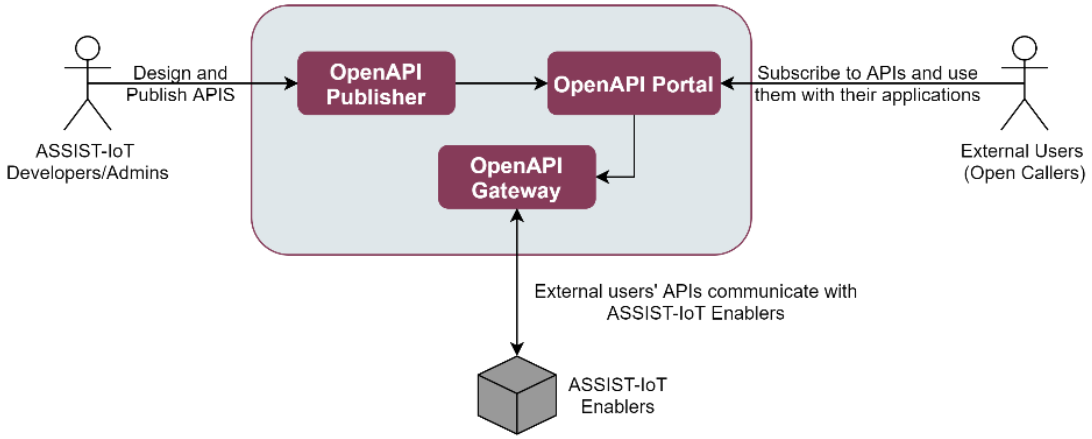


Figure 29. High-level diagram of the OpenAPI management enabler

Components

Enabler component	*OpenAPI Publisher*
id	*T44E4_Publisher*
Description and main functionality	*A collection of tools that API providers use to define APIs. They also generate API documentation, manage access and usage policies, and can as well be used for testing and debug purposes*
Target node/s	*Cloud server*
Candidate technologies	*Swagger*

Enabler component	*OpenAPI Portal*
id	*T44E4_Portal*
Description and main functionality	*A community site that enables the interested users to access to documentation, tutorials, sample code, software development kits, etc. It can also allow to manage subscription keys and obtain support from the API provider if needed.*
Target node/s	*Cloud server*
Candidate technologies	*WSO2 API Manager + WSO2 Identity Server, Swagger UI*

Enabler component	*OpenAPI Gateway*
id	*T44E4_Gateway*
Description and main functionality	*A server that acts as an API front-end, which receives API requests and passes requests to the back-end. It then passes the responses back to the requester. It can modify the requests and responses on the fly, and it can also provide the functionality to support authentication, authorisation, security, audit and caching.*

Enabler component	*OpenAPI Gateway*
Target node/s	*Cloud server*
Candidate technologies	*WSO2, APIgee, 3Scale, IBM API Management, Kong*

B.18 - Video augmentation enabler

Table 21. General information of the Video augmentation enabler

Enabler	*Video Augmentation enabler*
id	*T44E5*
Owner and support	*PRODEVELOP*
Description and main functionalities	*This enabler receives data (mainly images or video streams) captured either from ASSIST-IoT Edge nodes, or from ASSIST-IoT databases, and by means of Machine Learning Computer Vision functionalities, it provides object detection/recognition of particular end-user assets (e.g., cargo containers, cars' damages). To carry out the proper object recognition in operation, an appropriate annotated dataset should be previously used for training and testing. The Computer Vision functionalities will communicate with the ML algorithms repository in order to select the most appropriate ML algorithm according to the objected data. Once the ML algorithm has been trained and validated, it is used in the Inference Engine in order to overlap new unseen data as input, so that providing an answer it was trained to output in the User Application.*
Plane/s involved	*Application and services plane*
Relation with other enablers	• *T43E8: Long-term data storage* • *T44E4: OpenAPI management enabler* • *T52E3: ML Algorithms repository*
Requirements mapping	• *R-P1-5: Container ID tracking system* • *R-P1-16: Open/Accessible remote capabilities* • *R-P1-17: Customisable remote desktop* • *R-P1-23: AR support* • *R-P3A-9: Edge Intelligence* • *R-P3A-13: Augmented Reality support at the garage* • *R-P3B-15: Automatic defect detection* • *R-P3B-21: Automatic recognition* • *R-P3B-22: Augmented Reality support*
Use case mapping	• *UC-P1-7: Target visualisation during RTG operation*
Required components	*Data Pre-Processor, ML trainer, Inference engine*

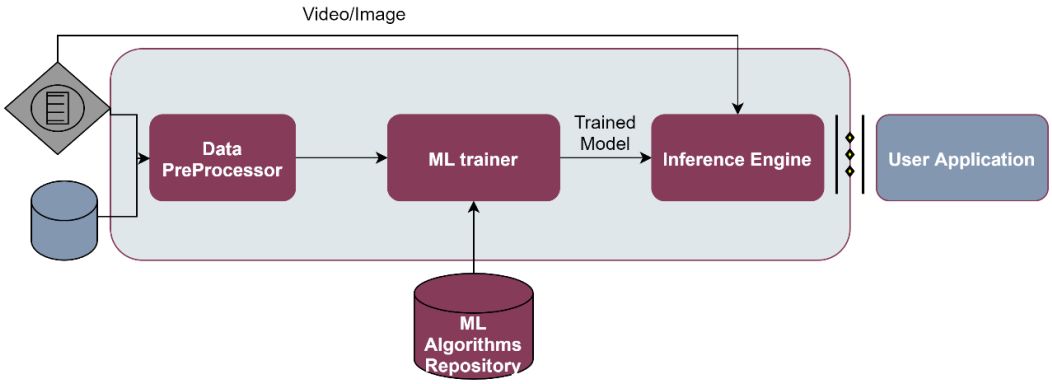


Figure 30. High-level diagram of the Video augmentation enabler

Components

Enabler component	***VA Data Pre-Processor***
id	*T44E5_DataPreProcessor*
Description and main functionality	*Since the data set can be collected from various sources such as a Cameras or Databases, but the collected data cannot be used directly for performing ML analysis process (e.g., there might be a lot of missing data, extremely large values, unorganised or noisy data), a data pre-processing should be done. First, the data should be split between training and validation sets. Next, Data pre-processor will also provide tools for cleaning the raw data such as taking care of missing values, categorical features, and normalisation*
Target node/s	*Cloud server*
Candidate technologies	*Scikit-learning, LabelEncoder*

Enabler component	***VA ML trainer***
id	*T44E5_Trainer*
Description and main functionality	*A ML model is a function with learnable parameters that maps an input to a desired output. The optimal parameters are obtained by training the model on data. ML Trainer will carry out the process of feeding the network with millions of training data points so that it systematically adjusts the knobs close to the correct values. The selected ML model to be used for training can be obtained from the ML Algorithms repository. The training process may be computationally intensive, because the data can be passed through Neural Network with several training rounds and it is recommended to be performed on a GPU.*
Target node/s	*Cloud server or High-tier edge nodes*
Candidate technologies	*OpenCV+Tensorflow (or +Caffee, or +Pytorch)*

Enabler component	***VA Inference Engine***
id	*T44E5_InferenceEngine*
Description and main functionality	*Inference Engine will be a set of C++ libraries providing a common API to deliver inference solutions for ASSIST-IoT: CPU, GPU, or VPU. The Inference Engine API can be used to read ML model, set the input and output formats, and execute the model on devices.*
Target node/s	*High-tier edge nodes*
Candidate technologies	*OpenVINO, OpenCV*

B.19 - MR enabler

Table 22. General information of the MR enabler

Enabler	***MR enabler***
id	*T44E6*
Owner and support	*ICCS*
Description and main functionalities	*The MR enabler receives data and transforms it in a format suitable for visualisation through head-mounted MR devices. Data, which may come from long-term storage or real-time data streams, are requested according to its relevance to the user.* *Information is displayed to the user, according to their authorisation/access rights, via an MR device. The enabler supports user interaction with the virtual content and view customisation.*
Plane/s involved	*Applications and services*
Relation with other enablers	• *T43E1: Semantic repository*

Enabler	*MR enabler*
	• *T43E7: Edge data broker* • *T43E8: Long-term data storage* • *T44E1: Business KPI reporting enabler* • *T53E2: Authorisation Enabler* • *SELF14: Monitoring and Notifying*
Requirements mapping	*N/A*
Use case mapping	• *UC-P2-7: Health and safety inspection support*
Required components	*Data Integration, Data Publication*

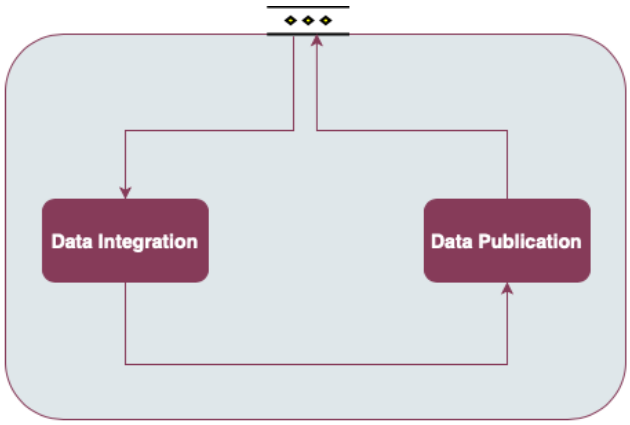


Figure 31. High-level diagram of the MR enabler

Components

Enabler component	*Data Integration*
id	*T44E3.2_DataIntegration*
Description and main functionality	*The Data Integration component integrates the data came from long-term storage, real-time data stream, other services and user input, correlates them according to their meaning and request to forwards them to the Data visualisation component.*
Target node/s	*Head-Mounted MR Device (HMD)*
Implementation technologies	*Unity*

Enabler component	*Data Publication*
id	*T44E3.2_Dapublication*
Description and main functionality	*The Data Publication component is responsible to visualise the information to the user via the MR glasses.*
Target node/s	*Head-Mounted MR Device (HMD)*
Implementation technologies	*Unity*