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

Technical Support Documentation - Final

Deliverable No.	D6.6	**Due Date**	30-Apr-2023
Type	Report	**Dissemination Level**	Public
Version	1.0	**WP**	WP6
Description	Includes initial versions of all technical and supporting documentation of components developed in WP4 and WP5 along with feedback from Open Call participants.		

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
Paco Mahedero	P01 UPV	framabio@upv.es
Rafael Vañó	P01 UPV	ravagar2@upv.es
Raúl Reinosa	P01 UPV	rreisim@upv.es
Eduardo Garro	P02 PRO	egarro@prodevelop.es
Miguel Llacer	P02 PRO	mllacer@prodevelop.es
Jose Antonio ClementeNton	P02 PRO	jclemente@prodevelop.es
Katarzyna Wasielewska-Michniewska	P03 IBSPAN	katarzyna.wasielewska@ibspan.waw.pl
Georgios Stavropoulos	P04 CERTH	stavrop@iti.gr
Iordanis Papoutsoglou	P04 CERTH	ipapoutsoglou@iti.gr
Anastasia Blitsi	P04 CERTH	akblitsi@iti.gr
Evripidis Tzionas	P04 CERTH	tzionasev@iti.gr
Konstantinos Flevarakis	P04 CERTH	kostisfl@iti.gr
Konstantinos Fragkos	P06 INF	cfragkos@infolysis.gr
Nikolaos Vrionis	P06 INF	nvrionis@infolysis.gr
Vaios Koumaras	P06 INF	vkoumaras@infolysis.gr
Aggeliki Papaioannou	P06 INF	apapaioannou@infolysis.gr
Johan Schabbink	P09 NEWAYS	johan.schabbink@newayselectronics.com
Ron Schram	P09 NEWAYS	Ron.schram@newayselectronics.com
Fotios Konstantinidis	P10 ICCS	fotios.konstantinidis@iccs.gr
Tina Katika	P10 ICCS	tina.katika@iccs.gr
Thomas Papaioannou	P10 ICCS	thomas.papaioannou@iccs.gr
Aristeidis Dadoukis	P10 ICCS	aristeidis.dadoukis@iccs.gr
Konstantinos Routsis	P10 ICCS	konstantinos.routsis@iccs.gr
Oscar López	P13 S21SEC	olopez@s21sec.com
Jordi Blasi	P13 S21SEC	jblasi@s21sec.com
Josue Moret	P13 S21SEC	jmoret@s21sec.com
Jaroslaw Legierski	P15 OPL	Jaroslaw.Legierski@orange.com
Zbigniew Kopertowski	P15 OPL	Zbigniew.Kopertowski@orange.com

History

Date	Version	Change
26-Jan-2023	0.1	ToC creation
10-Feb-2023	0.2	Initial Contributions
24-Feb-2023	0.3	2nd round of contributions by all involved partners
03-Mar-2023	0.3.1	Contributions to Section 1 by INF
20-Mar-2023	0.3.2	Contributions to Section 1 and 2 by UPV
24-Mar-2023	0.4	3rd round of contributions by all involved partners
7-Apr-2023	0.5	Final contributions by all involved partners
12-Apr-2023	0.9	Internal review
27-Apr-2023	1.0	Final Version

Key Data

Keywords	Enablers, Technical Documentation, Support
Lead Editor	P06 INF – Nikolaos Vrionis
Internal Reviewer(s)	Johan Schabbink, P09 NEWAYS Fotios Konstantinidis, P10 ICCS

Executive Summary

This deliverable is written in the framework of WP6 – Testing, Integration and Support of **ASSIST-IoT** project under Grant Agreement No. 957258. This document is the second of a series that is devoted to release and distribution of the documentation material produced as part of Task 6.4 –Technical and Support Documentation that primarily represents the documentation of WP4 and WP5 technical developments. In parallel, this document explains the documentation strategy that is used throughout the task's duration, with the goal of creating compact and sufficient documentation of the project's technical outcomes while also considering the feedback of the first Open Call participants.

The documentation aims to offer information on how to deploy and use the ASSIST-IoT enablers of the horizontal planes and verticals of the ASSIST-IoT architecture. Due to ASSIST-IoT enablers are software that can be installed on any machine, there are some hardware requirements. These needs are defined according to the ASSIST-IoT deployment, which can be made up of one or more tiers, each with one or more nodes running a k8s installation. If indoor localization is needed, the project's tags and anchors can be used. The Kubernetes master plane that manages the division of the workload among the different tiers considers a set of minimum requirements and, more specifically, MicroK8s, k3s, and manual k8s installation are the Kubernetes distributions that are completely tested and thus supported by the ASSIST-IoT architecture.

Furthermore, this deliverable presents the essential enablers, which are a subset of all the designed enablers that are strictly required to be present in every particular deployment and are as follows: a) Smart Orchestrator enabler, b) VPN enabler, c) Edge data broker enabler, d) Long-term data storage enabler, e) Tactile dashboard enabler, f) OpenAPI management enabler, g) Basic security enablers, h) DLT logging and auditing enabler, and i) Manageability enablers.

D6.6 also refine the guidelines related to an ASSIST-IoT deployment, to help a potential administrator. They will be able to deploy the platform and its associated tools in order to realise a particular business scenario, following a specific set of steps: 1) preparation of the main top-tier node, including high availability strategy and a set of essential enablers, 2) provisioning of Kubernetes distribution on the rest of nodes (within the top and other tiers), 3) installation of the rest essential enablers, and 4) installation and configuration of other enablers to address the specifics of the business scenario. Currently, two scripts are in place for aiding in the deployment process.

Additionally, D6.6 provides guidance on how the potential user can login to the tactile dashboard of ASSIST-IoT through the web, to a) manage clusters in the ASSIST-IoT deployment, b) manage helm repositories for gettingenablers compliant with the ASSIST-IoT architecture, and c) manage enablers within the infrastructure.

Following the general installation instructions, each enabler goes through a particular documentation process, which is delivered using a specialized ASSIST-IoT wiki repository, which may be found at the following link: https://assist-iot-enablers-documentation.readthedocs.io/en/latest/index.html. The wiki is organized around the overall ASSIST-IoT architecture, following a general approach consisting of the following sections: Introduction, Features, Place in Architecture, User Guide, Prerequisites, Installation, Configuration options, Developer guide, Version control and Release, License, Notice. The current version of the wiki represents the state up to M30.

Finally, this document presents information pertaining to the first Open Callers' feedback based on a comprehensive questionnaire. The section encompasses a detailed overview of the questionnaire, along with valuable insights into the responses received. The primary objective of this section is to facilitate improvements in the Technical and Support Documentation as well as the GitLab Repository.

Table of contents

List of tables

List of figures

List of acronyms

Acronym	Explanation
AI	Artificial Intelligence
AMD	Advanced Micro Devices
API	Application Programming Interface
ARM	Advanced RISC Machines (related to architecture of processors)
AV	Audio/Video
BIM	Building Information Modelling
CAN	Controller Area Network
CAN FD	Controller Area Network Flexible Data-Rate
CNF	Cloud-native Network Function
CNI	Container Network Interface
CPU	Central Processing Unit
CSV	Comma Separated Value
DLT	Distributed Ledger Technology
DNS	Domain Name System
DoS	Denial of Service
Dx.y	Deliverable No y of Work Package x
FL	Federated Learning
GUI	Graphical User Interface
GWEN	GateWay/EdgeNodes
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
HW	Hardware
I/O	Input/Output
IDS	Intrusion Detection System
IMU	Inertial Measurement Unit
IoT	Internet of Things
IP	Internal Protocol
IT	Information Technology
JSON	JavaScript Object Notation
K8S	Kubernetes
KPI	Key Performance Indicator
LED	Light Emitting Diode
LTS	Long-Term Storage

MANO	Management and Orchestration
ML	Machine Learning
MQTT	MQ Telemetry Transport
MR	Mixed Reality
NFVO	Network Function Virtualisation Orchestrator
NGIoT	Next-Generation Internet of Things
OAM	Operations, Administration and Management (related to network traffic)
OC	Open Call
ONOS	Open Network Operating System
OS	Operating System
OSM	Open-Source MANO
OAuth	Open Authorization
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
PIP	Policy Information Point
PUD	Performance and Usage Diagnosis
RAM	Random Access Memory
RDF	Resource Description Framework
RKE	Rancher Kubernetes Engine
RS	Recommended Standard
RST	reStructuredText
SDN	Software-Defined Networking
SD-WAN	Software-Defined Wide Area Network
SQL	Structured Query Language
SSL	Secure Sockets Layer
TSN	Time-Sensitive Networking
UI	User Interface
USB	Universal Serial Bus
UWB	Ultra-Wide Band
VM	Virtual Machine
VoIP	Voice over Internet Protocol
VPN	Virtual Private Network
WAN	Wide Area Network
WiFi	Wireless Fidelity
WP	Work Package

XACML	eXtensible Access Control Markup Language
XML	Extensible Markup Language

1 About this document

The ASSIST-IoT Project introduces a blueprint architecture consisting of several enablers that can be further exploited by the industrial community to enhance already available applications or even introduce new ones. Every ASSIST-IoT advancement should be well recorded so that third-party stakeholders may track the advances and apply the results.

Towards this objective, this deliverable, as an updated version of its previous version, collects in a single, self-contained document all the necessary information, both conceptual and technical, to effectively support the aforementioned third parties. This document outlines the fundamental definitions and functions of the ASSIST-IoT enablers, as well as the general configuration processes, technical interactions, and settings to be configured when engaging with the ASSIST-IoT outputs, as well as licensing and version control. In general, this deliverable serves as the narrative of support documentation, laying the groundwork for a manual that will be written over the project's lifespan, portraying all relevant developments.

It should be noted that this deliverable corresponds to the 2nd release of a series of two documents, reporting mainly on the developments made until M30 (April 2023). Furthermore, this document consists of key overviews and components' information, while more thorough information on each enabler is provided by links to the ASSIST-IoT public online repository (https://assist-iot-enablers-documentation.readthedocs.io/en/latest/index.html).

1.1Deliverable context

Keywords	**Lead Editor**
Objectives	As an extent to T6.4 activities, which relate to generating technical documentation and support materials to reflect functionalities and characteristics of technical outputs of the action (mainly from WP4 and WP5), the main objective related to deliverable D6.6 is the following: • Developing and releasing supporting documentation for both 3rd parties participating in Open Calls and Stakeholders, together with the Open Source Community publishing
Work plan	This deliverable belongs to the set of deliverables of WP6, and is directly linked, specifically, to Task 6.4, forming the second document of a series of two deliverables (D6.5 and D6.6), as the second technical documentation report, that provides critical support for the proper deployment of WP4 and WP5 enablers. While the documentation offered by this deliverable will be used as a tool for technical information, its primary function is to assist technical personnel (developers, admins, etc.) and third parties (e.g., Open Call participants) integrate, use, or extend the capabilities of ASSIST-IoT technical outputs, while playing important role for rest WPs activities as well (e.g. WP7, WP8).
Milestones	This deliverable is indirectly connected with MS6 "Software structured finished" which is verified by the definition of the software structure of enablers.
Deliverables	D6.6 is the second version of the Documentation series documents of the ASSIST-IoT Project. D6.6 is also linked with D6.5 and other deliverables, that have already been submitted as part of WP4, WP5 and WP6 reporting. These previous documents serve as the main source of input for this deliverable, since the main technical output of the ASSIST-IoT Project originates mainly from WP4 and WP5, which are the two main technical development work packages of the project.

1.2 The rationale behind the structure

The deliverable D6.6 is divided into four sections with the first one (Section 1) to be the introductory section providing all the document's administrative information and the fourth one to provide the conclusion.

Comprehensive information regarding the documentation is provided in Section 2. The section is broken into six subsections. The first four of which (Sections 2.1, 2.2, 2.3, and 2.4) provide broad instructions and rules that apply to any third party aiming to deploying enablers, while the latter two (Sections 2.5, 2.6) contain a broad summary of all the ASSIST-IoT Project enablers as well as links to online wikis specifically established for the more thorough technical documentation of each enabler.

The final section of the document, Section 3, elucidates the questionnaire that was administered to all participants of the first Open Call. This section provides a detailed overview of the questionnaire, the answers received and an extensive analysis of the outcomes.

1.3 Version-specific notes

D6.6 represents the second report on the technical advancements of the ASSIST-IoT Project. This updated version comprises the latest status of the enablers and an analysis of first Open Caller's feedback on the GitLab Repository and ASSIST-IoT Technical and Support Documentation, derived from a thoroughly designed questionnaire.

2 Documentation

The documentation approach that is followed in the ASSIST-IoT Project, is being provided by this deliverable and is further updated and elaborated in this section (in comparison to D6.5 previous reporting). The approach is focused on the enablers and their documentation both with respect to each enabler, as well as overall instructions on how to deploy the ASSIST-IoT environment.

The approach that has been followed for the technical support documentation, conceives this deliverable as a general reference on installation prerequisites and steps on the ASSIST-IoT deployment, and also introduce the reader to the User Interface modules that have been designed and developed to support the interaction with the enablers. Additionally, as part of the work done in T6.4, the dedicated wikis introduced in D6.5 are updated for further documenting the enablers and are reported in this section. Finally, D6.6 presents the information in a self-contained manner, updating the content presented in D6.5.

2.1 ASSIST-IoT Installation Prerequisites

2.1.1 Hardware Requirements

An ASSIST-IoT deployment can be composed of one or more **tiers**, as specified in the deployment view of the reference architecture (see Figure 1). In turn, each tier will be comprised of one or more **nodes**, each of them running a **k8s distribution**.

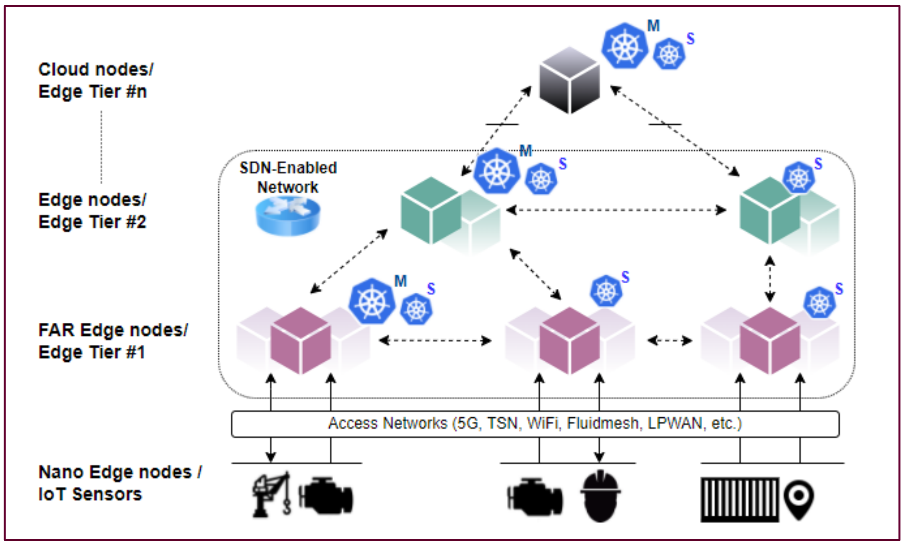

Cloud nodes/ Edge Tier #n
Edge nodes/ Edge Tier #2
FAR Edge nodes/ Edge Tier #1
Nano Edge nodes / IoT Sensors
SDN-Enabled Network
M
S
Access Networks (5G, TSN, WiFi, Fluidmesh, LPWAN, etc.)

Figure 1. Deployment view of ASSIST-IoT

Top-tier nodes could be deployed on premises or in the cloud. One **node** can be enough, however, a common convention is to follow high availability strategies in order to be prepared in case of node failure. The following **hardware requirements** are needed in order to support the essential enablers that should be installed on it (these should be increased in case of hosting a larger number of enablers and/or third-party applications/ services):

- AMD processor, with virtualisation capabilities. Minimum: 2 CPUs, recommended: >= 4 CPUs.
- RAM memory: Minimum: 8 GB, recommended: >= 16 GB.
- Storage: Minimum/recommended: 40 GB.
- At least one interface with Internet connection (100 Mbps or higher). In case of being on premises, a second interface must belong to the internal network.

In case of having more than one working node belonging to the tier, they should have similar hardware resources, so the Kubernetes master controlling them assigns similar amounts of workloads, avoiding unbalanced workload.

Regarding **nodes** from **low-level tiers**, it is hard to specify a set of requirements as it greatly depends on the use cases to be addressed. Again, it is possible to follow a high availability strategy, if not for all the enablers/applications at least for the critical ones and the k8s master controlling them (**NOTE:** A topology logically divided in tiers suggests having a master controlling the nodes belonging to each of them. However, it is possible to have one master controlling the whole topology, as long as all the nodes share the same k8s distribution). The **minimum requirements** to support a Kubernetes distribution (K3s recommended, see Section 2.1.2) and a few workloads are the following:

- CPU with virtualisation capabilities.
- At least 1 GB of RAM and 8 GB of storage.
- At least one interface connected to the top tier node/s.
- Additional wired/wireless interfaces (CAN, Serial, WiFi, 4G/5G, etc.) might be needed depending on the addressed use cases.

To have an accurate number of processing, memory, and storage numbers, a previous study should be conducted to analyse the needs of the enablers and applications that they will host. In case of making use of the **ASSIST-IoT's GWEN as edge node**, 12V± 15% 5A power supply is required. An adapter from 230V is provided with it to generate the needed voltage. In case of making use of other edge nodes (RaspberryPis, SIMATICs, etc.), they will have their own power needs and adapters. Being modular, GWEN can support several use cases, allowing an expansion of its computing capabilities (e.g., RAM, CPU), connection interfaces (e.g., TSN) and on-purpose boards (e.g., GPU) via carrier boards with expansion modules. The expansion boards are tailored to the needs of each pilot and can be added or swapped if needed. For wireless connectivity 5G and Wi-Fi antenna's are needed.

2.1.1.1 Requirements for indoor localisation

The ASSIST-IoT project is designing and developing its own tags and anchors for indoor localisation. Apart from the hardware part, an associated (software) enabler to be installed in an edge node (e.g., a GWEN) is being developed, which should be properly configured in order to retrieve and use the gathered information properly. Regarding **tags**:

- The battery of the tags has to be charged during non-working hours.
- The monitored assets need to have a tag attached (e.g., in case of a worker, he/she has to wear a belt or helmet with it).
- Information about the correspondence between tag and asset must be noted, so they can be later on related. In case of a worker, information about whether it is connected to a belt or helmet should be noted as well.

Regarding **anchors:**

- 230V/50Hz mains outlet is needed to power the anchors. A mains adapter is delivered together with the anchor.
- At the construction site no outlets are in place yet, therefore a set of anchors has been made that run on battery. The battery is designed to last a week under normal circumstances and can be recharged via a USB charger. The master anchor is connected via USB to the edge node (e.g., a GWEN) and is power by the edge node.
- During the installation of the anchor, the position of the anchor has to be documented. This is needed to enable the associated localization enabler to determine the absolute position of the tag.

2.1.2 Installation of Kubernetes and Add-ons

As aforementioned, an ASSIST-IoT deployment can be logically distributed into different tiers. This logical distribution *suggests* a Kubernetes master plane governing the workloads of each of them. Before describing aspects related to K8s installation, some considerations for designing each one of the tiers are provided:

- A k8s master should control nodes of similar hardware resources. Otherwise, the assignment of resources to pods might be unbalanced.
- In case that the system administrator prefers having a single cluster to manage, a set of **actions for working with a cluster with heterogeneous nodes** can be performed: (i) defining the hardware request and limits in the pods descriptors, (ii) configuring node affinities, (iii) specifying pod affinities and anti-affinities, and (iv) enabling some k8s features such as accelerators for GPU support, or managers for constraining workloads to specific CPUs. It is still under analysis which of these configurations could be applied directly by the smart orchestrator.
- Nodes belonging to different networking sites (e.g., cloud vs pilot site) should not share a k8s master, as in case of network failure the nodes of the one of the sites would lose the control plane.
- Lastly, if new nodes are to be added in an existing deployment, it is preferable to include them as workers whenever possible, as masters devote more resources to control plane tasks.

Once the hardware topology of the site is in place, a k8s distribution must be installed in each of the nodes. Two routes can be followed: installation on bare metal vs installation on top of an Operating System (OS) or Virtual Machine (VM). The two options are perfectly valid, being the bare-metal route slightly more optimal but less straightforward (more manual configuration effort required). For the sake of simplicity, installation on top of OS/VM is followed.

In principle, most kubernetes distributions should be supported by the ASSIST-IoT orchestrator, however, only microK8s[1], k3s[2], and manual k8s installations (hereinafter referred to as *kubeadm* installation[3]) have been fully tested. It is important to highlight that **the master plane of a particular distribution cannot control worker nodes with other ones**. Hence, it is recommended that the nodes of a specific tier have the same distribution installed.

Regarding the **selection** of Kubernetes **distributions**, **K3s** is better **for constrained devices**, as its memory footprint is much lower than the other options (besides k0s[4]). It is also optimised for ARM32, ARM64 and ARMv7 platforms, hence better in case of leveraging nodes like RaspberryPis. **MicroK8s** has significantly lowered its RAM consumption in recent releases, being usable in nodes with even 1 GB of RAM, but it is still notably higher than K3s. In any case, its performance is outperformed by k3s, hence it is not recommended for production. Sill, being a very easy-to-install and to use distribution, it is a great option **for development environments**. **Kubeadm** installation is less straightforward, but gives the user total control of the tools and options being installed. It is recommended **for top-tier nodes**. Other alternatives (like k0s, RKE[5] or vendor distributions, like OpenShift[6]) could be studied as well.

Some basic installation guidelines on top of Linux OSs are given next, considering the set of minimum add-ons that are required. It is worth highlighting that, for the main top-tier node, **a script for deploying a set of must, minimum features is provided** (i.e., deploying a kubeadm distribution with the required add-ons, with smart orchestrator and manageability enablers, see Annex A) for facilitating an ASSIST-IoT deployment, as explained in Section 2.2.2, so **these examples are useful for the rest of the nodes** (as it will also install a kubeadm with the required addons).

2.1.2.1 K3s installation with required add-ons

This subsection provides the necessary commands for installing and configuring a K3s cluster, including the installation of worker nodes, add-ons, and a high availability strategy. An available dedicated script automates this process. The script allows for the installation of either a master or a worker node, with different installation processes for each, depending on the chosen flags and options. For a master node installation, three important flags must be considered:

- **t**: SERVER or AGENT (in this case SERVER).

[1] https://microk8s.io/
[2] https://k3s.io/
[3] https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

- **i**: Server IP. If the edge is behind a NAT and the Smart Orchestrator or the worker nodes are outside, the value is your Public IP.
- **p**: Pod CIDR Network (This MUST be different in each cluster. If you choose 10.216.0.0/16, the other cluster MUST be for instance 10.215.0.0/16).

```
sudo ./K3s.sh -t SERVER -i serverIP -p 10.213.0.0/16
```

Alternatively, to create a worker node for addition to an existing master node, the command would be:

```
sudo ./K3s.sh -t AGENT -i serverIP -k serverToken
```

Here, the “s” flag refers to:

- **s**: Server IP (Master Node IP).
- **k**: The server token can be found on the master node machine, located at the following path: */var/lib/rancher/K3s/server/node-token*

The corresponding command would appear as follows:

```
sudo ./K3s.sh -t AGENT -i serverIP -k serverToken
```

The aforementioned script is available in the **Annex B**: K3 Installation script.

2.1.2.2 kubeadm installation and required add-ons

This subsection outlines the necessary commands for the installation and configuration of a kubeadm cluster, including the installation of worker nodes, add-ons, and a high availability strategy. The K8s installation with kubeadm can be accessed if the machine has at least two CPUs and can be divided into four main steps:

1. Initial server preparation
2. Installation of required tools
3. Cluster creation
4. Node addition.

The provided script allows for the installation of either a master or a worker node, depending on the chosen flags and options, with different installation processes for each. For a master node installation, two important flags must be considered:

- **t**: for server or agent designation (in this case, server)
- **p**: for the pod CIDR network (which must differ for each cluster; for instance, if 10.216.0.0/16 is chosen, the other cluster must use a different CIDR network). The command would appear as follows:

```
sudo ./kubernetes.sh -t SERVER -p 10.216.0.0/16.
```

Alternatively, to create a worker node for addition to an existing master node, the command would be:

```
sudo ./kubernetes.sh -t AGENT.
```

Once the worker node is ready, switch to the main cluster (master node) and copy the output of the following command:

```
kubeadm token create --print-join-command
```

Then, switch back to the master node and run the command output as sudo. The aforementioned script is available in the **Annex C**: kubeadm installation script.

2.2 Steps on deploying ASSIST-IoT

2.2.1 Essential Enablers

ASSIST-IoT architecture is realized by enablers, which provide functionalities related to its horizontal planes (device, network, data management, and application and services) and its verticals capabilities and properties (self-* mechanisms, scalability, interoperability, manageability and security, privacy and trust). Among all the designed enablers, a set of them, so-called essential, is strictly required to be present in any given ASSIST-IoT deployment. On the contrary, other enablers (from the offered ones or custom-made) depend on the particular use cases to be addressed.

Some of the essential enablers will be pre-installed, whereas others should be installed by an ASSIST-IoT administrator (as some configuration is required and cannot be fully automatized). The following enablers are considered essential:

- **Smart orchestrator** (Smart network and control plane, pre-installed): This enabler is considered essential as it will be in charge of controlling the lifecycle (instantiation, communication, and termination) of the rest of enablers belonging to an ASSIST-IoT deployment. It will control not only network but also non-network virtualised functions, allocating them within the managed infrastructure (i.e., k8s clusters and nodes), assigning resources, and ensuring proper configuration also by configuring the required k8s add-ons (e.g., CNIs, service meshes, etc.).
- **VPN enabler** (Smart network and control plane): It facilitates the access of a node or a device from a different network to the site's, private one, using a public network (e.g., the Internet) or an external private network. This kind of VPN solution is considered essential to minimise networking attack surfaces.
- **Edge data broker** (Data management plane): This enabler oversees the distribution of data among nodes. Its role as a data router is based on a publish/subscribe schema (data demand and data supply from/to nodes) and takes into account load balancing criteria. It is considered essential in those kinds of deployments that require a dispatching element to move data (e.g., from sensors) from/to edge nodes upwards/downwards.
- **Long-term data storage enabler** (Data management plane, pre-installed): Enablers can incorporate databases as part of their design, however, in general their storage will be limited to their own (logical) scope. This enabler is essential as (i) it keeps information about enablers' context (e.g., volumes, historic, connections), crucial in case of shutdown, (ii) it allows enablers to rely on a "centric", "cloud" storage so they can save space, and (iii) it manages access to the platform and enablers, based on roles and user profiles. It will be compatible with relational and non-relational schemes.
- **Tactile dashboard enabler** (Application and services plane, pre-installed): Considered essential as it will be the central GUI of the system. Although ASSIST-IoT deployments will require system administrator interventions (console-based, primarily related to k8s maintenance), the goal of the solution is to allow user-friendly configuration, management of devices, network, services, results and, globally, consultation of parameters at many different levels via a UI.
- **OpenAPI management enabler** (Application and services plane): Although ASSIST-IoT has been conceived as a holistic NGIoT solution, covering most needs appearing in associated use cases, it will likely coexist with other apps and systems that require interacting with its enablers. Hence, this manager is considered essential as it will ease this interaction, working as an API proxy (with certain rules and configuration) to corresponding interfaces of underlying enablers so that external systems (e.g., Open Call winners' IT tools) can interact with them.
- **Basic security enablers** (Security, privacy and trust vertical): Security entails a set of functionalities that IoT systems must equip. In ASSIST-IoT, the essential security features will be provided by (i) the Identity Manager, for validating the identity -after a resource request- against a trusted central server (storing credentials based on OAuth2 protocol), and (ii) via the Authorisation enabler, that decides whether to grant access to such resource or not (based on XACML policies).

- **DLT logging and auditing enabler** (Security, privacy and trust vertical): Transparency, non-repudiation, and action accountability have been considered essential when designing an ASSIST-IoT solutions. Whereas passing all data transactions through blockchain in a large, scalable IoT deployment does not seem the best approach (due to constrained resources and energy consumption), the critical events happening throughout the system must be logged and properly registered via this enabler.
- **Manageability enablers** (Manageability vertical, pre-installed): Set of enablers that allow users to perform manageability operations leveraging GUIs, including: (i) registration of k8s clusters and devices in the system, (ii) registration and configuration of enablers, and (iii) conformation of (composite, more complex) services by facilitating the combination of enablers (either essential or non-essential), scattered through diverse nodes.

Specific installation order on the essential ASSIST-IoT enablers is not required since there is no enforced dependency between them. However, the configuration of each of the enablers is important since it could be influenced by identified interactions. Details on the different configurations are provided in the dedicated enabler wikis which are reported in Sections 2.5 and 2.6.

2.2.2 Installation Steps

With the topology for fulfilling a business scenario defined, and having a bird's-eye view of the essential enablers, an ASSIST-IoT administrator (possibly alongside a system administrator) can proceed to install the platform and its associated tools. A summary of the steps for having a successfully installation of ASSIST-IoT platform can be summarised into the following main ones (it is important to highlight that these steps will change over the evolution towards the second release of the action):

1. Preparation of the main top-tier node, including high availability strategy and a set of essential enablers.
2. Provisioning of kubernetes distribution on the rest of the nodes (within the top and other tiers).
3. Installation of the rest essential enablers.
4. Installation and configuration of use-case related enablers.

The full software stack has not been completely defined, therefore specialised tools such as Terraform[4] and/or Ansible[5] have not been leveraged yet to automate or facilitate the process (i.e., to provision hosts, operating systems, k8s distribution and addons, essential enablers). So far, for the **main top-tier node** (first step), a **script** has been developed to be installed in an Ubuntu 18.04 operating system. This script automates:

- Installation and configuration of kubeadm and required add-ons (Helm, Cilium, CoreDNS).
- Installation of OSM and the components of the smart orchestrator enabler.
- Installation of the tactile dashboard and manageability enablers.
- Installation of the OpenAPI management enabler.

In case that a high availability strategy is considered, an odd number of kubeadm nodes (>= 3) should join the top-tier cluster. To that end, an administrator should provision the nodes manually, following the instructions provided in Section 0. In the future, dedicated tools like the aforementioned ones will be considered for facilitate the process.

Apart from the top-tier node, or nodes, the rest of tiers should be provisioned to fulfil the **second step**. Currently, this should be done **manually**, installing the necessary kubernetes distributions and grouping them into clusters as defined per topological decisions (see Sections 2.1.1 and 2.1.2). Guidelines have been done to install them on top of Linux operating systems, being k3s and kubeadm the suggested ones for production environments. In the case of considering deployment tools in the future, it is still to be defined whether being a standalone from the first one or merged altogether.

[4] https://www.terraform.io/
[5] https://www.ansible.com/

As one can observe, not all the essential enablers are installed with the main script. There are different motivations behind this decision. On the one hand, as the script just works for a single machine, it would have all of them installed in the same top-tier node, which might not be desirable; on the other hand, because of the distributed nature of **the rest of the essential enablers**, installing them via script in a single machine might be largely insufficient, therefore being better to let an administrator to install and configure them **via the manageability enablers**. Hence, the **third step** is to be performed manually by an ASSIST-IoT administrator, via the latter enablers and interacting with their interfaces/APIs whenever needed.

Once the logical tiers are ready and the essential enablers installed, the administrator can use the system to deploy the rest of (optional) enablers to address their business scenarios (fourth step). It is possible as well to include/delete new clusters, or nodes within the existing ones, always considering that any workload is present before proceeding to their elimination.

2.3 Management User Interface for Enablers

Once the essential enablers are in place, an ASSIST-IoT manager can access to the tactile dashboard via a web browser. To that end, the manager must navigate to the IP address of the physical server hosting the K8s of the top-tier cluster and the configured NodePort during the Helm chart installation (by default, port 30080, see Figure 2), and then log in using the default administration credentials (user:*admin*, password: *admin*, which can be changed within the application) or the default user credentials (user:*assist*, password: *assistiot*).

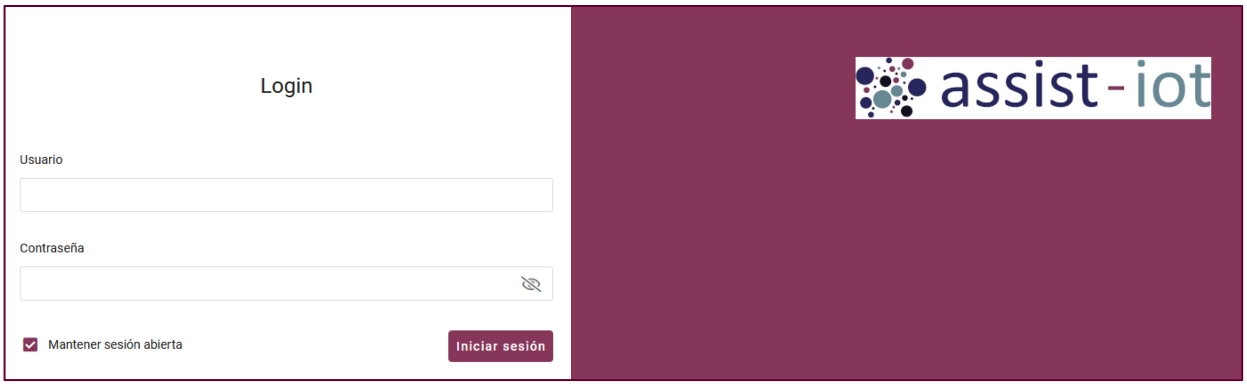
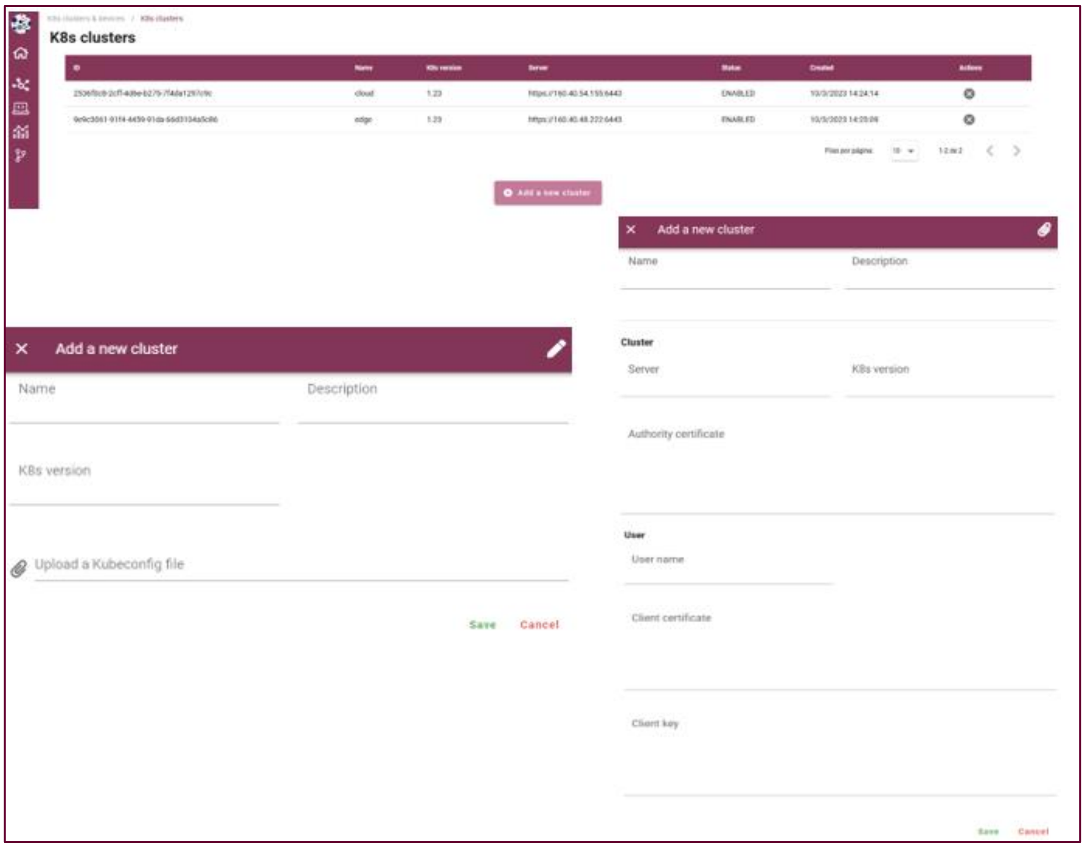


Figure 2. Login screen

If credentials are correct, the user can interact graphically with the tactile dashboard interfaces, including the ones related to manageability. The following actions can be performed in the current release of the manageability enablers (v0.1.0):

- To manage K8s clusters in the ASSIST-IoT deployment.
- To manage Helm chart repositories for getting enablers compliant with the ASSIST-IoT architecture.
- To manage enablers within the infrastructure.

Figure 3 one can observe some screenshots of the graphical interface developed for managing clusters. With it, a user can register (and delete) k8s clusters that can be used for instantiating enablers, as well as getting some information about them. To that end, apart from a reachable IP address, information from their corresponding *kubeconfig* files must be indicated. Currently, the interface supports k8s and k3s clusters, although in the future additional distributions will be included.


K8s clusters & devices / K8s clusters

K8s clusters

ID	Name	K8s version	Server	Status	Created	Actions
2536f0c8-2cff-4dbe-b279-7f4da1297c9c	cloud	1.23	https://160.40.54.155:6443	ENABLED	10/3/2023 14:24:14	⊗
9e9c3061-91f4-4459-91da-66d3134a5c86	edge	1.23	https://160.40.48.222:6443	ENABLED	10/5/2023 14:25:08	⊗

Figure 3. Cluster management interface

An additional interface has been developed to visualize in a more user-friendly way the K8 clusters registered with the Smart Orchestrator. In this interface is displayed a topology graph of these registered K8 clusters, including all the K8 nodes (differentiating between master and worker nodes) that make up each cluster. Furthermore, the user can see a list with the deployed enablers in a cluster and deploy an enabler in a specific K8s node (all the components included in the Helm chart will be deployed in the selected node).

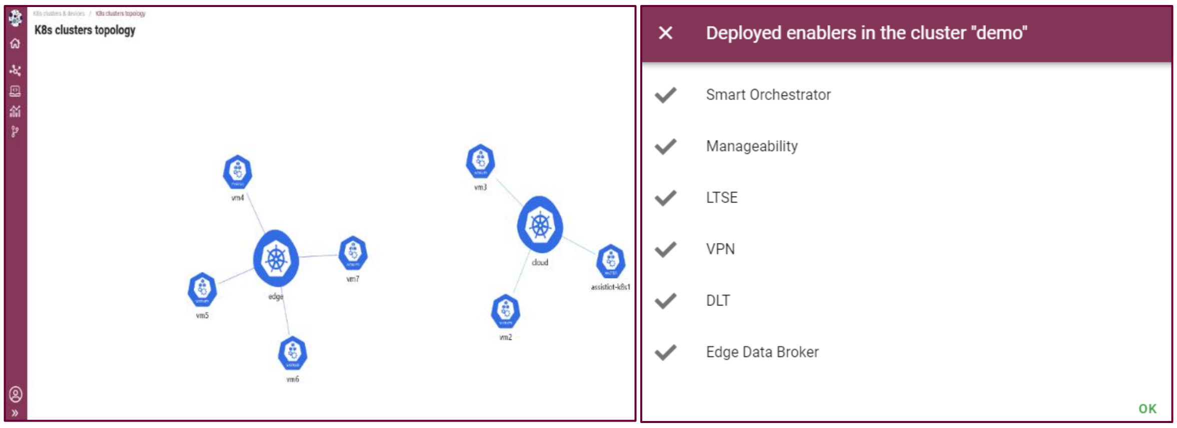
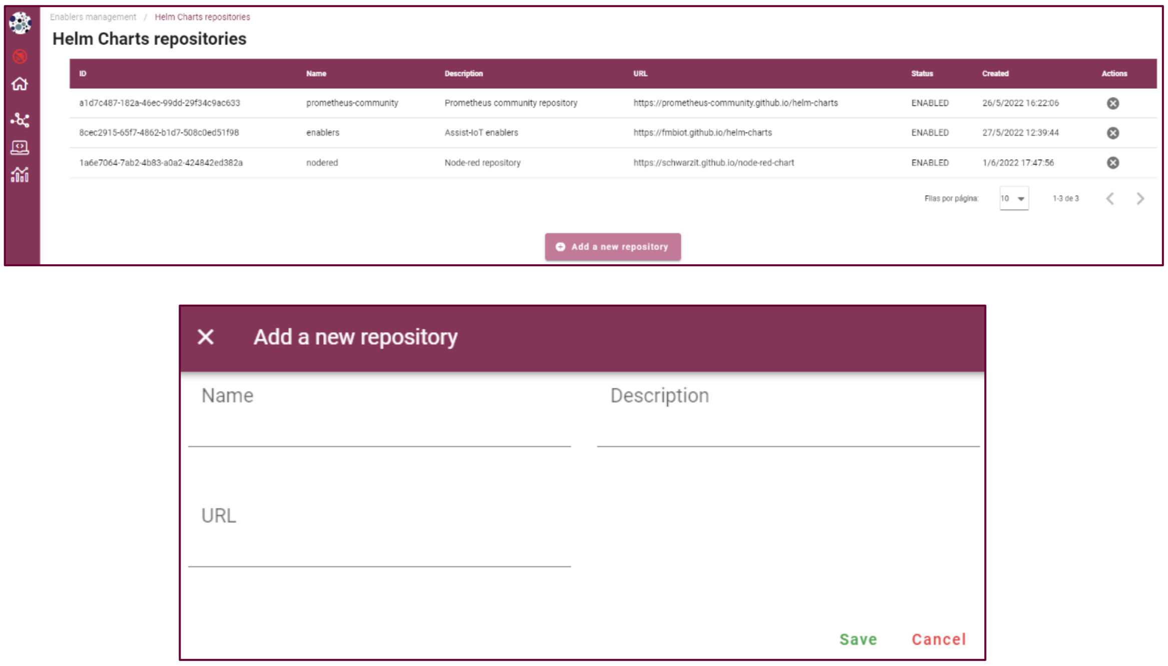


Figure 4. K8s clusters topology management interface

Before launching an enabler, Helm chart repositories should be registered so the Chart packages included can be later on downloaded and consumed. In principle, any HTTP server that houses an index.yaml describing the published Charts in the repo and the charts compressed in tar.gz files can be registered (i.e., GitHub/GitLab repositories, cloud or local repositories with Chartmuseum, etc.). Repositories have to be accessible from the cluster that hosts the Smart Orchestrator enabler. A snapshot of the graphical interface used for managing repositories can be seen in Figure 5.

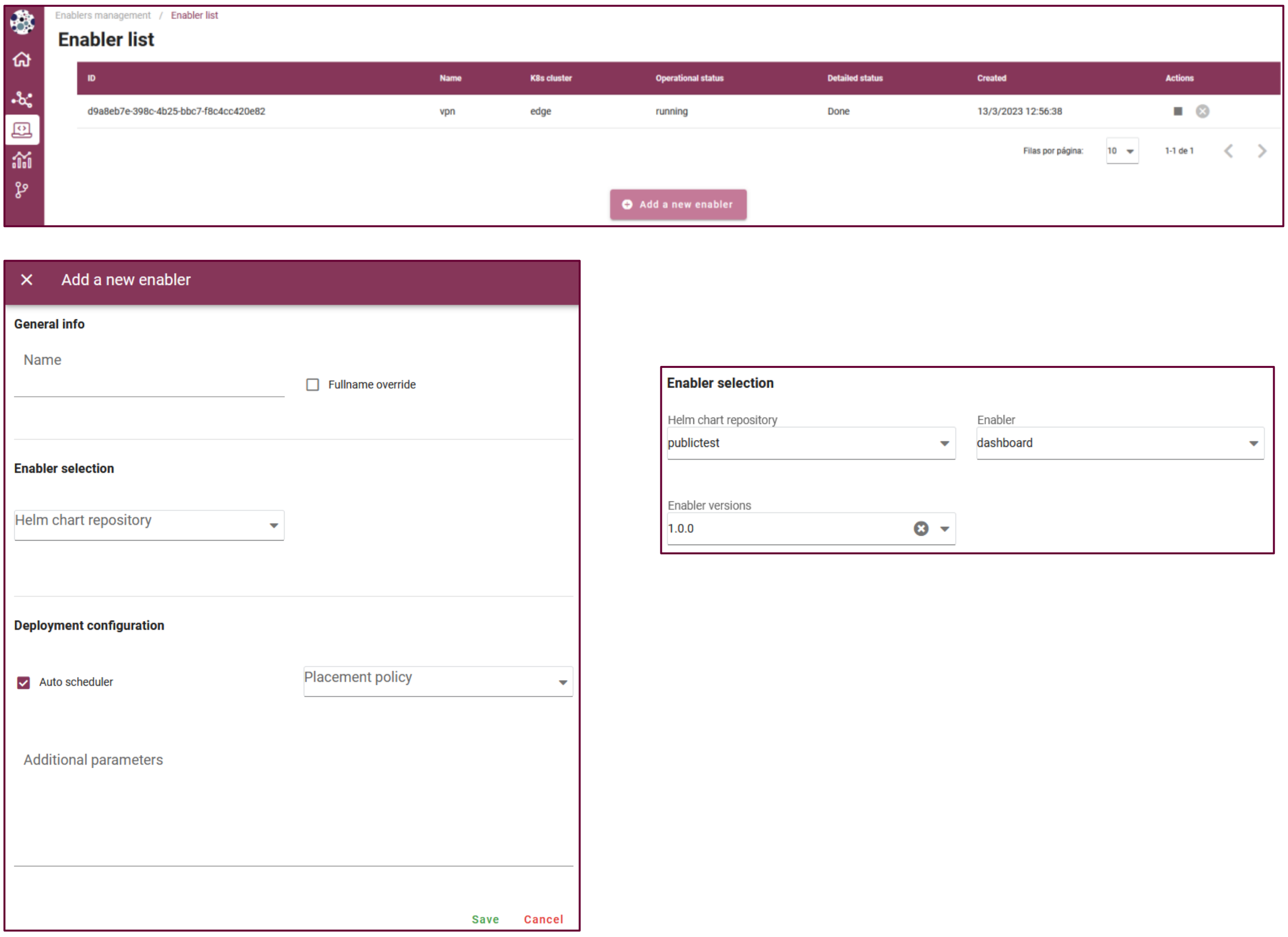


Helm Charts repositories

ID	Name	Description	URL	Status	Created	Actions
a1d7c487-182a-46ec-99dd-29f34c9ac633	prometheus-community	Prometheus community repository	https://prometheus-community.github.io/helm-charts	ENABLED	26/5/2022 16:22:06	⊗
8cec2915-65f7-4862-b1d7-508c0ed51f98	enablers	Assist-IoT enablers	https://fmbiot.github.io/helm-charts	ENABLED	27/5/2022 12:39:44	⊗
1a6e7064-7ab2-4b83-a0a2-424842ed382a	nodered	Node-red repository	https://schwarzit.github.io/node-red-chart	ENABLED	1/6/2022 17:47:56	⊗

Figure 5. Helm repositories management interface

The next management interface that has been implemented is related to enablers (see Figure 6). With it, the user can instruct the system to deploy or terminate an enabler. In the current release, for indicating the enabler that will be deployed, the user has to select in order: (i) the Helm chart repository, (ii) the chart and (iii) the chart version of the enabler using the displayed selectables. When selected the enabler, the user has to indicate where will be deployed: (i) manually selection of the K8s cluster or (ii) automatic selection of the K8s cluster using the *auto scheduler* feature of the Smart Orchestrator. Finally, the user can pass a JSON object in the *additional parameters* field in order to modify any default value of the Chart values manifest, which is very useful for customizing the configuration of an enabler for a particular environment.


Enablers management / Enabler list

Enabler list

ID	Name	K8s cluster	Operational status	Detailed status	Created	Actions
d9a8eb7e-398c-4b25-bbc7-f8c4cc420e82	vpn	edge	running	Done	13/3/2023 12:56:38	■ ⊗

Filas por página: 10 1-1 de 1

Add a new enabler

Add a new enabler

General info

Name

☐ Fullname override

Enabler selection

Helm chart repository

Deployment configuration

☑ Auto scheduler

Placement policy

Additional parameters

Save Cancel

Enabler selection

Helm chart repository: publictest

Enabler: dashboard

Enabler versions: 1.0.0

Figure 6. Enablers management interface

Finally, the interface related with the management of services and enablers' workflow is still under development. At this moment, a frontend based on Node-RED can be used to define data flows or pipelines involving the LTSE and EDBE enablers with the main purpose of obtaining data from the MQTT protocol and moving it to the HTTP protocol and vice versa. In the near future the plan is to add more enablers to these pipelines and use the semantic related enablers to integrate more complex data pipelines.

2.4 Enablers Technical Documentation

Enablers offer a range of functionalities that span various planes and verticals, resulting in significant technological disparities in their development and internals. Furthermore, enablers exhibit varying levels of development and readiness. To facilitate this multidimensional approach, specific wikis have been established to document each enabler, and these wikis will be continuously updated to reflect evolving developments.

It is imperative to emphasize that the Read the Docs Platform, which has been selected as the host for the documentation of ASSIST-IoT, facilitates continuous and dynamic integration. As a result, it serves as the primary public repository that guides the documentation plan and reflects the progress of technical developments. Specifically, the wikis presented in this deliverable document the maturity level and developments of the enablers up to M30. Given that the project is still ongoing and additional developments are underway for some of the enablers, it is possible that updates to certain wikis may be required.

2.4.1 Structure of the Wiki

Since the purpose of this deliverable is to generate detailed documentation of the ASSIST-IoT enablers, aiming at facilitating its reading process, the documentation is provided using links that lead to wikis with respect to each of the developed enablers.

More specifically, the Read the Docs Platform is used to create a public repository that hosts the documentation. Among the available options for hosting a technical documentation wiki, DevDocs[6], Wiki.js[7], Gitbook[8], and MkDocs[9] were also considered. Being Read the Docs one of the most widely used platforms and also considering the provided simplification of software documentation by automating building, versioning, and hosting the docs, it was considered the best choice. It supports Sphinx (the standard Python documentation system) docs written in ReStructuredText (RST) format, and can pull from Subversion, Bazaar, Mercurial, and Git repositories, being Git the selected option in the case of ASSIST-IoT. RST is a lightweight markup language that emphasizes plain-text readability, widely used for API documentation, and provides standard extension mechanisms, called directives, and roles, which can make remarkable difference on the final product.

The ASSIST-IoT wiki can be found in the following link: https://assist-iot-enablers-documentation.readthedocs.io/en/latest/index.html.

In terms of how the wiki is organized, the repository is divided into sections (as shown in **Figure 7**) corresponding to the classification of the enablers according to the ASSIST-IoT approach. Each section and subsection are clickable and linked to a new page dedicated to the respective content.

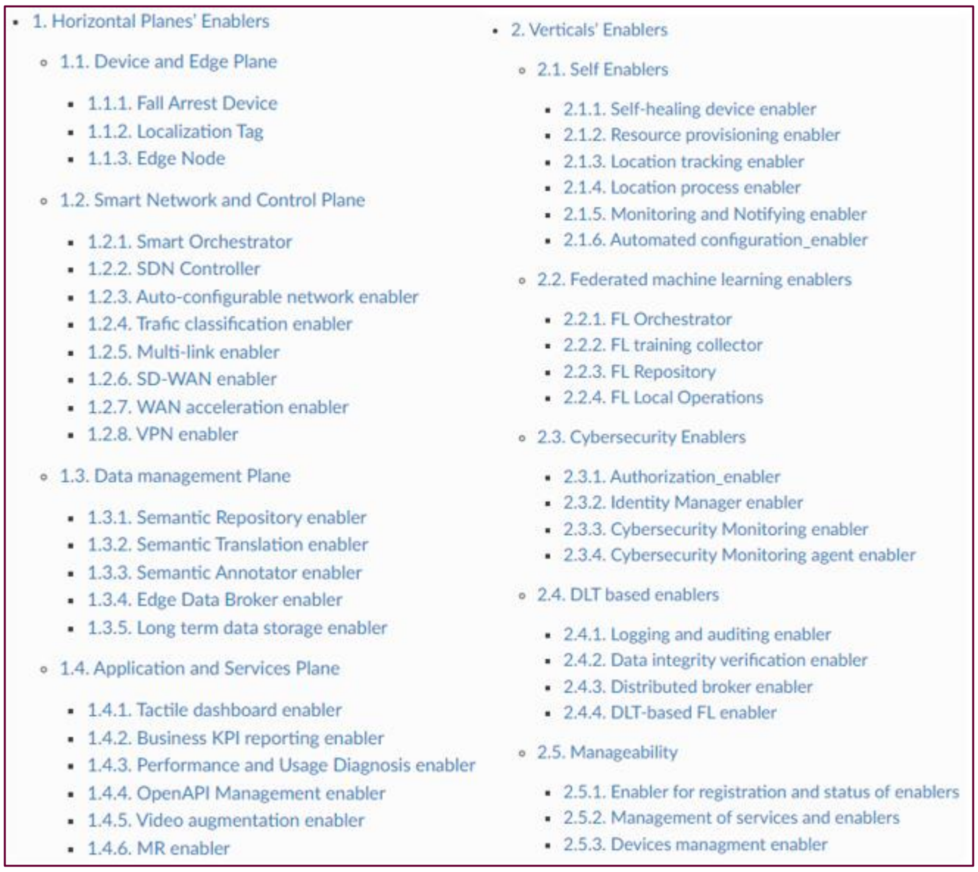
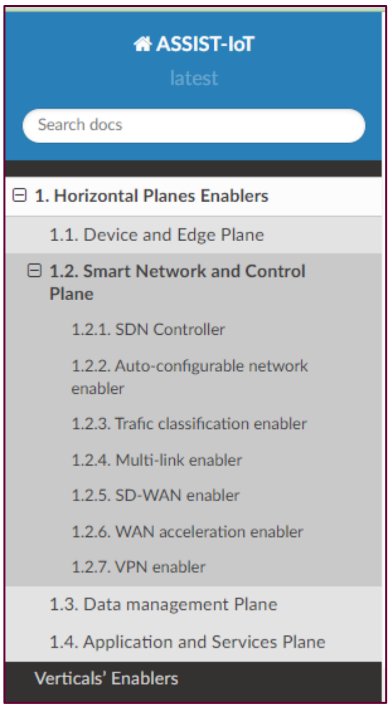

- 1. Horizontal Planes' Enablers
  - 1.1. Device and Edge Plane
    - 1.1.1. Fall Arrest Device
    - 1.1.2. Localization Tag
    - 1.1.3. Edge Node
  - 1.2. Smart Network and Control Plane
    - 1.2.1. Smart Orchestrator
    - 1.2.2. SDN Controller
    - 1.2.3. Auto-configurable network enabler
    - 1.2.4. Trafic classification enabler
    - 1.2.5. Multi-link enabler
    - 1.2.6. SD-WAN enabler
    - 1.2.7. WAN acceleration enabler
    - 1.2.8. VPN enabler
  - 1.3. Data management Plane
    - 1.3.1. Semantic Repository enabler
    - 1.3.2. Semantic Translation enabler
    - 1.3.3. Semantic Annotator enabler
    - 1.3.4. Edge Data Broker enabler
    - 1.3.5. Long term data storage enabler
  - 1.4. Application and Services Plane
    - 1.4.1. Tactile dashboard enabler
    - 1.4.2. Business KPI reporting enabler
    - 1.4.3. Performance and Usage Diagnosis enabler
    - 1.4.4. OpenAPI Management enabler
    - 1.4.5. Video augmentation enabler
    - 1.4.6. MR enabler
- 2. Verticals' Enablers
  - 2.1. Self Enablers
    - 2.1.1. Self-healing device enabler
    - 2.1.2. Resource provisioning enabler
    - 2.1.3. Location tracking enabler
    - 2.1.4. Location process enabler
    - 2.1.5. Monitoring and Notifying enabler
    - 2.1.6. Automated configuration_enabler
  - 2.2. Federated machine learning enablers
    - 2.2.1. FL Orchestrator
    - 2.2.2. FL training collector
    - 2.2.3. FL Repository
    - 2.2.4. FL Local Operations
  - 2.3. Cybersecurity Enablers
    - 2.3.1. Authorization_enabler
    - 2.3.2. Identity Manager enabler
    - 2.3.3. Cybersecurity Monitoring enabler
    - 2.3.4. Cybersecurity Monitoring agent enabler
  - 2.4. DLT based enablers
    - 2.4.1. Logging and auditing enabler
    - 2.4.2. Data integrity verification enabler
    - 2.4.3. Distributed broker enabler
    - 2.4.4. DLT-based FL enabler
  - 2.5. Manageability
    - 2.5.1. Enabler for registration and status of enablers
    - 2.5.2. Management of services and enablers
    - 2.5.3. Devices managment enabler

Figure 7. ASSIST-IoT Wiki Documentation structure

[6] https://devdocs.io/
[7] https://js.wiki/
[8] https://www.gitbook.com/
[9] https://www.mkdocs.org/

The navigation between the different wiki pages is achieved by using the side bar that provides both a search and manual selection options. The navigation bar is presented in **Figure 8**.



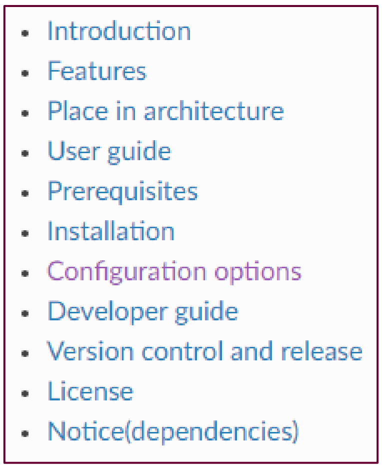
ASSIST-IoT
latest
Search docs

1. Horizontal Planes Enablers
   - 1.1. Device and Edge Plane
   - 1.2. Smart Network and Control Plane
     - 1.2.1. SDN Controller
     - 1.2.2. Auto-configurable network enabler
     - 1.2.3. Trafic classification enabler
     - 1.2.4. Multi-link enabler
     - 1.2.5. SD-WAN enabler
     - 1.2.6. WAN acceleration enabler
     - 1.2.7. VPN enabler
   - 1.3. Data management Plane
   - 1.4. Application and Services Plane

Verticals' Enablers

Figure 8. Wiki's side bar options

Additional to the general section organization, the section dedicated to each enabler follows a specific table of contents, especially created to reflect all the key information needed for the documentation of the enablers. The structure of this table of contents is depicted **Figure 9**.



- Introduction
- Features
- Place in architecture
- User guide
- Prerequisites
- Installation
- Configuration options
- Developer guide
- Version control and release
- License
- Notice(dependencies)

Figure 9. Section organization for every enabler

In specific, the core sections under each enabler are *Introduction*, *Features*, *Place in Architecture*, *User Guide*, *Prerequisites*, *Installation*, *Configuration options*, *Developer guide*, *Version control and Release*, *License*, and *Notice*. These sections are just a general guide and can be further adapted according to the needs of each enabler. A more detailed description of the section is provided in Table 1.

Table 1. Description of the wiki main sections

Title	Description
Introduction	This section specifically refers to a high-level description of the reported enabler, giving an overview of the respective enabler and its functionalities.
Features	Operational and intelligent functionalities of the enabler are to be described and detailed in this section.
Place in the architecture	Since most enablers may not have a clear place in the whole ASSIST-IoT architecture, this section is dedicated to providing the necessary description for the understanding of how this enabler is placed/located in the ASSIST-IoT concept and how it can interact with other components.
User Guide	The User Guide section is dedicated in documenting and describing the potential UIs that will be created to support the user-enabler interaction. Additionally, this section will include examples and guidelines on how the users can interact with the respective enabler, according to the enabler's needs.
Prerequisites	This section lists all the required installation that is required to take place before the respective enabler can be installed and configured. Hardware and Software prerequisites should also be listed here.
Installation	The installation section is a complete guide on how to install the enabler and perform the configurations on the installation process.
Configuration process	In the Configuration Process, the configuration steps that should take place before the enabler is ready for use are listed.
Developer Guide	Since each enabler exposes a number of APIs for the interaction of an app with the enabler, this section serves as a centralized guide on what are the available APIs and how a developer can interact with them.
Version control and release, License, Notice	These three sections are dedicated in listing context information related to each enabler. First, the Version control and release that is concerned, with identifying and keeping track of the different versions of the respective enabler, are documented. Following, the License section provides the legally binding guidelines for the use and distribution of the respective enabler. Finally, the Notice section provides any legally required notifications/instructions in addition to the License documents

Please note that the content of the aforementioned sections in this final deliverable has been fully developed and is reflective of the maturity level of each enabler at the time of completion. Any incomplete sections have been filled, and modifications or updates to the content have been included as necessary. This final version of the wiki is being released alongside this deliverable.

2.4.2 Documentation Potential Enhancements

Reporting the ASSIST-IoT enablers in a form that acts as a guide to third parties and stakeholders that want to utilize the ASSIST-IoT technical outputs, is one of the main reporting goals of the ASSIST-IoT documentation and, as an extend, of this deliverable. Most of the enablers have been completed and maybe some will be enhanced or integrated until the end of the project, an indicative table with potential future releases of the documentation has been created for the purpose of this deliverable. The Potential Enhancements/Integrations dates act only as indications and could change according to the project's needs and advancement. The fundamental objective of Table 2 is to enlighten prospective third-party users regarding potential releases of some enablers. To obtain further insights into the current version status of each enabler, please refer to the respective wiki, which is hyperlinked in Sections 2.5 and 2.6.

Table 2. Indicative release plan for the ASSIST-IoT enablers documentation

Code	Enabler Name	Current Status of Enablers' Documentation	Potential Enhancements/Integrations
T41E1	Localization Tag	Ongoing	To be completed by October 2023
T41E3	GWEN	Ongoing	To be completed by October 2023
T42E1	Smart Orchestrator	Completed	Potential enhancements by October 2023
T42E2	SDN Controller	Ongoing	N/A
T42E3	Auto-configurable Network	Ongoing	To be completed by October 2023
T42E4	Traffic Classification	Ongoing	To be completed by October 2023
T42E5	Multi-link	Ongoing	To be completed by October 2023
T42E6	SD-WAN	Completed	N/A
T42E7	WAN Acceleration	Completed	N/A
T42E8	VPN	Completed	To be completed by October 2023
T43E1	Semantic Repository	Completed	N/A
T43E2	Semantic Translation	Completed	To be completed by October 2023
T43E3	Semantic Annotation	Completed	N/A
T43E4	Edge Data Broker	Completed	N/A
T43E5	Long-term Data Storage	Completed	Potential enhancements by October 2023
T44E1	Tactile Dashboard	Completed	Potential enhancements by October 2023
T44E2	Business KPI Reporting	Completed	N/A
T44E3	Performance and Usage Diagnosis(PUD)	Completed	N/A
T44E4	OpenAPI Management	Completed	Potential enhancements by October 2023
T44E5	Video Augmentation	Completed	N/A
T44E6	MR	Ongoing	To be completed by October 2023
SELF11	Self-healing Device	Completed	N/A
SELF12	Resource Provisioning	Ongoing	To be completed by October 2023
SELF16	Location Processing	Completed	N/A
SELF14	Monitoring and Notifying	Completed	N/A
SELF15	Automated Configuration	Ongoing	To be completed by October 2023
T52E1	FL Orchestrator	Ongoing	To be completed by October 2023
T52E2	FL Training Collector	Completed	N/A
T52E3	FL Repository	Completed	N/A
T52E4	FL Local Operations	Completed	N/A
T53E1	Authorisation	Completed	N/A
T53E2	Identity Manager	Completed	N/A
T53E3	Cybersecurity Monitoring	Completed	N/A
T53E4	Cybersecurity Monitoring Agent	Completed	N/A

T54E1	Logging and Auditing	Ongoing	To be completed by October 2023
T54E2	Data Integrity Verification	Ongoing	To be completed by October 2023
T54E3	Distributed Broker	Ongoing	To be completed by October 2023
T54E4	DLT-based FL	Ongoing	To be completed by October 2023
T55E1	Enablers manager	Completed	Potential enhancements by October 2023
T55E2	Composite services manager	Completed	Potential enhancements by October 2023
T55E3	Clusters and topology manager	Completed	Potential enhancements by October 2023

2.5 Horizontal Planes Enablers

2.5.1 Device and Edge Plane

2.5.1.1 Smart Devices

Enabler: *Localization Tag*	**Id:** *T41E1*
Owner and Support: *NEWAYS, Pilot Stakeholders (MOSTOSTAL, FORD)*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Ongoing*	
Enabler Description	
The ASSIST-IoT localisation tag is a Smart IoT device used for people's localisation purposes, especially in indoor cases. This device has tag functionality, and it contains a buzzer and red LED. The buzzer is used to indicate to the person that he/she is in a restricted area. The button is used to alert the system when the worker detects an accident and immediate help is needed.	
Keywords / Key components	
UWB, low power, indoor localisation	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/device/localization_tag.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A	

Suggestion and comments: N/A

2.5.1.2 GWEN

Enabler: GWEN	**Id:** *T41E3*
Owner and Support: *NEWAYS, Pilot Stakeholders (MOSTOSTAL, FORD)*	
Related Deliverable/s: *D3.5, D4.1, D4.2*	
Status of Enabler: *Ongoing*	
Enabler Description	
The GateWay/EdgeNode (GWEN) is a device used as and edge computing interface between sensors & actuators and or systems on one side and a communication network on the other side. Sensors actuators and systems can be connected through wired and/or wireless interfaces. The interface with a network can also be wired or wireless. *Available wired interfaces are: Ethernet, CAN & CAN FD, USB2 and USB3, PCIe, HDMI, Camera interface (CSI) SD card* *Available wireless interfaces are: WiFi, Bluetooth and 3G/4G/5G. In addition an UWB interfaces via USB is available for localisation purposes.* *The GWEN also contains computational power to be able to operate AI algorithm, while Docker in combination with Kubernetes is used as container runtime on top of Linux as OS, together with several apps for specific purposes within pilots.*	
Keywords / Key components	
Ethernet, WIFI, Bluetooth, 5G, UWB, CAN, USB, Docker, Linux	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/device/edge_node.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A **Suggestion and comments: N/A**	

2.5.2 Smart Network and Control Plane

2.5.2.1 Smart Orchestrator

Enabler: *Smart Orchestrator*	**Id:** *T42E1*

Owner and Support: *UPV*
Related Deliverable/s: *D4.1, D4.2*
Status of Enabler: *Developed (first functional version ready; to enhance until October 2023)*
Enabler Description
This enabler facilitates the interaction with the main components of theMANO 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 wholelifecycle of Containerised Functions, network and not-network related, from their instantiation to their termination, allowing their deployment in any K8s cluster available. It also controls the underlying network, implementing specific network policies to allow/deny traffic according to ASSIST-IoT networking specifications.
Keywords / Key components
Orchestrator, MANO, CNF, Intent
Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/smart/smart_orchestrator.html
Used by Open Callers, suggestions and comments
Open Caller name: N/A   **Suggestion and comments: N/A**

2.5.2.2 SDN Controller

Enabler: *SDN controller*	**Id:** *T42E2*
Owner and Support: *UPV*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Ongoing*	
Enabler Description	
The SDN Controller is the key element of an SDN-enabled network, where the main functionalities are related to network management, operation and maintenance, allowing topology management, network configuration, network control and network operations, among other features. Two solutions are investigated based on open source implementation: µONOS and Tungsten.	
Keywords / Key components	
SDN, network configuration, network management, network monitoring, topology	

Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/smart/sdn_controller.html
Used by Open Callers, suggestions and comments
Open Caller name: N/A **Suggestion and comments: N/A**

2.5.2.3 Auto-configurable network enabler

Enabler: *Auto-configurable network*	**Id:** *T42E3*
Owner and Support: *OPL, UPV*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Ongoing*	
Enabler Description	
This enabler provides solution for network configuration using the SDN Controller of an ASSIST-IoT ecosystem. The policy based solution using the northbound APIs of the SDN Controllers that improves the performance of selected KPIs of the network (required by use case applications). The strategies are under specification based on requirements of network performance and quality for use cases applications. Solution for network resources optimisation are under investigation.	
Keywords / Key components	
Network configuration, policy based management, KPI, network performance, network quality	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/smart/auto_configurable_network_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A **Suggestion and comments: N/A**	

2.5.2.4 Traffic classification enabler

Enabler: *Traffic classification*	**Id:** *T42E4*
Owner and Support: *UPV*	
Related Deliverable/s: *D4.1, D4.2*	

Status of Enabler: *Testing phase*
Enabler Description
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 an 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.
Keywords / Key components
Network Traffic, Classifier, AI/ML
Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/smart/traffic_classification_enabler.html
Used by Open Callers, suggestions and comments
Open Caller name: N/A   **Suggestion and comments: N/A**

2.5.2.5 Multi-link enabler

Enabler: *Multi-link*	**Id:** *T42E5*
Owner and Support: *UPV, PRO, TL*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Ongoing*	
Enabler Description	
Multi-link wireless network capabilities provide the possibility of sending IP-based 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 and redundancy mechanisms: in case one channel is down, signal cannot be lost or at least it should be recovered almost in real time (to achieve it, data will be sent via more than one wireless network).	
Keywords / Key components	
Reliability, Redundancy, Connectivity	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/smart/multi_link_enabler.html	

Used by Open Callers, suggestions and comments
Open Caller name: N/A
Suggestion and comments: N/A

2.5.2.6 SD-WAN enabler

Enabler: *SD-WAN*	**Id:** *T42E6*
Owner and Support: *UPV*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Completed*	
Enabler Description	
The objective of this enabler is to provide access between nodes from different sites based on SD-WAN technology. In particular, this enabler will implement mechanisms to connect K8s clusters via private tunnels, facilitating (i) the deployment and chaining of virtual functions to secure connections between them and/or towards the Internet and (ii) the implementation of functions to optimise WAN traffic.	
Keywords / Key components	
SD-WAN, Controller, Connectivity	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/smart/sd_wan_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A	
Suggestion and comments: N/A	

2.5.2.7 WAN Acceleration enabler

Enabler: *WAN acceleration*	**Id:** *T42E7*
Owner and Support: *UPV*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Completed*	
Enabler Description	

This enabler aims at increasing the efficiency of data transfer in Wide Area Network. This enabler will contain a set of independent, standalone CNFs 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.
Keywords / Key components
Traffic shaping, Compression, Traffic optimisation
Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/smart/wan_acceleration_enabler.html
Used by Open Callers, suggestions and comments
Open Caller name: N/A **Suggestion and comments: N/A**

2.5.2.8 VPN enabler

Enabler: *VPN*	**Id:** *T42E8*
Owner and Support: *UPV*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Finished*	
Enabler Description	
This enabler facilitates 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, by establishing a dedicated encrypted tunnel.	
Keywords / Key components	
VPN, Tunnelling, Connectivity	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/smart/vpn_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: ATHEMS **Suggestion and comments: N/A**	

2.5.3 Data management Plane

2.5.3.1 Semantic Repository enabler

Enabler: *Semantic Repository*	**Id:** *T43E1*
Owner and Support: *SRIPAS, MOW, PRODEVELOP, KONECRANES, FORD*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Completed*	
Enabler Description	
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.	
Keywords / Key components	
Data Models, Ontologies, Repository	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/datamanagement/semantic_repository_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A **Suggestion and comments: N/A**	

2.5.3.2 Semantic Translation enabler

Enabler: *Semantic Translation*	**Id:** *T43E2*
Owner and Support: *SRIPAS, UPV*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Testing Phase*	
Enabler Description	
Semantic Translation enabler offers a configurable service to change the contents of semantically annotated data in accordance with translation rules – so called "alignments". Data can be translated in batch, or through persistent streams.	
Keywords / Key components	

Semantic translation, streaming, ontologies, RDF
Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/datamanagement/se-mantic_translation_enabler.html
Used by Open Callers, suggestions and comments
Open Caller name: N/A **Suggestion and comments: N/A**

2.5.3.3 Semantic Annotation enabler

Enabler: *Semantic annotation*	**Id:** *T43E3*
Owner and Support: *SRIPAS, ICCS*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Completed*	
Enabler Description	
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, with CSV and XML to follow Annotation can be done in batch (in first release) or through persistent configurable streams.	
Keywords / Key components	
Semantic annotation, RDF, streams	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/datamanagement/se-mantic_annotator_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A **Suggestion and comments: N/A**	

2.5.3.4 Edge Data Broker enabler

Enabler: *Edge Data Broker*	**Id:** *T43E4*
Owner and Support: *ICCS, UPV, PRO, NEWAYS, CERTH*	

Related Deliverable/s: *D4.1, D4.2*
Status of Enabler: *Testing Phase*
Enabler Description
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.
Keywords / Key components
edge data broker, distributed, clustered, MQTT, middleware, data analytics
Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/datamanagement/edge_data_broker_enabler.html
Used by Open Callers, suggestions and comments
Open Caller name: BREATHE **Suggestion and comments: N/A**

2.5.3.5 Long-term Data Storage enabler

Enabler: *Long-term Data Storage*	**Id:** *T43E5*
Owner and Support: *PRO, UPV*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Completed*	
Enabler Description	
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.	
Keywords / Key components	
Long-Term Storage, noSQL, SQL, resilient, centralized	
Link to wiki	

https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/datamanagement/long_term_data_storage_enabler.html
Used by Open Callers, suggestions and comments
Open Caller name: BREATHE, HAIR, RAZOR, SMART SONIA, ATHEMS, ADDICTIVE
Suggestion and comments: N/A

2.5.4 Application and Services Plane

2.5.4.1 Tactile Dashboard enabler

Enabler: *Tactile Dashboard*	**Id:** *T44E1*
Owner and Support: *PRO, UPV, SRIPAS*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Finalized*	
Enabler Description	
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.	
Keywords / Key components	
Frontend, dashboard, VUE.js, responsive, webpage	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/application/tactile_dashboard_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: ATHEMS	
Suggestion and comments: N/A	

2.5.4.2 Business KPI Reporting enabler

Enabler: *Business KPI Reporting*	**Id:** *T44E2*
Owner and Support: *PRO, UPV, SRIPAS*	
Related Deliverable/s: *D4.1, D4.2*	

Status of Enabler: *Completed*
Enabler Description
This enabler will illustrate valuable KPIs within Graphical User Interfaces embedded into 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.
Keywords / Key components
pie charts, bar graphs, KPIs
Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/application/business_kpi_reporting_enabler.html
Used by Open Callers, suggestions and comments
Open Caller name: ATHEMS   **Suggestion and comments: N/A**

2.5.4.3 Performance and Usage Diagnosis enabler

Enabler: *Performance and Usage Diagnosis*	**Id:** *T44E3*
Owner and Support: *PRO, UPV, SRIPAS*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Completed*	
Enabler Description	
Performance and Usage Diagnosis (PUD) 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 tune machine resources.	
Keywords / Key components	
monitoring, metrics collection, targets, status alerting	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/application/performance_and_usage_diagnosis_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: ATHEMS	

Suggestion and comments: N/A

2.5.4.4 OpenAPI Management enabler

Enabler: *OpenAPI Management*	**Id:** *T44E4*
Owner and Support: *UPV, SRIPAS, PRO*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Completed*	
Enabler Description	
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.	
Keywords / Key components	
API, open calls, swagger, swagger-json	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/application/open-api_management_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A **Suggestion and comments: N/A**	

2.5.4.5 Video Augmentation enabler

Enabler: *Video Augmentation*	**Id:** *T44E5*
Owner and Support: *PRO*	
Related Deliverable/s: *D4.1, D4.2*	
Status of Enabler: *Completed*	
Enabler Description	
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).	
Keywords / Key components	

Object detection, Camera software, AV, ML
Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/application/video_augmentation_enabler.html
Used by Open Callers, suggestions and comments
Open Caller name: SPINE **Suggestion and comments: More frequent updates on GitLab Repository**

2.5.4.6 MR enabler

Enabler: *MR*	**Id:** *T44E6*
Owner and Support: *ICCS*	
Related Deliverable/s: *D4.1 ,D4.2*	
Status of Enabler: *Testing Phase*	
Enabler Description	
The novel interface that is used in the MR enabler offers a human-centric interaction through better cooperation of the end-users with the IoT environment. Through the MR enabler, the human effort and decisions are introduced in the loop of every critical action, whenever needed. The MR enabler aids human-friendly haptics and the end-user can receive and provide tactile, real-time and visual feedback as well as data capable of identifying critical improvements, preventions and triggers in long-, short-term, or real-time. Through reporting functions, the MR enabler gathers reliable data to extract information and perform analytics. Decision-making is improved as human flexibility, creativity and expertise, interact with IoT platforms and devices. The functionalities of the MR enabler are summarized as follow: • *Identifying assets (along with relevant data) at close proximity,* • *Visualizing rendered (3D) models through the head-mounted MR devices, along with highlighted zones of the same model. The models and all related data come from the long-term storage,* • *Shows location and information of the workers of the construction site,* • *Receiving alert messages from real-time data streams and displaying them to the user, and* • *Capturing and storing media files in order to include them in a report.*	
Keywords / Key components	
Mixed reality, BIM visualisation, real-time data, IoT, alerting	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/application/mr_enabler.html	
Used by Open Callers, suggestions and comments	

Open Caller name: N/A
Suggestion and comments: N/A

2.6 Verticals' Enablers

2.6.1 Self-* Enablers

2.6.1.1 Self-healing device enabler

Enabler: *Self-healing device*	**Id:** *SELF11*
Owner and Support: *PRO, SRIPAS, UPV*	
Related Deliverable/s: *D5.1, D5.3, D5.4*	
Status of Enabler: *Under testing*	
Enabler Description	
This enabler aims at providing to IoT devices with the capabilities of actively attempting to recover themselves from abnormal states, mainly divided in three categories: 1) security (jamming, DoS), 2) dependability (data corruption, network protocol violation), and 3) long-term (HW's end-of-life, HW unsupported capabilities), based on a pre-established routine schedule.	
Keywords / Key components	
IDS, RAM monitoring, CPU monitoring. self-healing	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/self/self_healing_device_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A **Suggestion and comments: N/A**	

2.6.1.2 Resource provisioning enabler

Enabler: *Resource Provisioning*	**Id:** *SELF12*
Owner and Support: *SRIPAS, UPV*	
Related Deliverable/s: *D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Ongoing*	

Enabler Description
This enabler will be able to horizontally scale (up or down) the resources devoted to a specific enabler (inside a node) in a dynamic fashion, based on time series inference and custom logic.
Keywords / Key components
Self-configuration, Time Series, Horizontal Pod Autoscaler
Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/self/resource_provisioning_enabler.html
Used by Open Callers, suggestions and comments
Open Caller name: N/A **Suggestion and comments: N/A**

2.6.1.3 Location processing enabler

Enabler: *Location Processing*	**Id:** *SELF16*
Owner and Support: *SRIPAS*	
Related Deliverable/s: *D5.1, D5.4*	
Status of Enabler: *Ongoing*	
Enabler Description	
This enabler will provide spatial data storage and processing capabilities. It will be able to integrate spatial information from various sources and process it in a streaming fashion.	
Keywords / Key components	
Location, Database, Query, Data Streaming	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/self/location_process_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A **Suggestion and comments: N/A**	

2.6.1.4 Monitoring and Notifying enabler

Enabler: *Monitoring and Notifying*	**Id:** *SELF14*
Owner and Support: *SRIPAS, CERTH*	
Related Deliverable/s: *D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Testing phase*	
Enabler Description	
This enabler could be viewed as a general purpose by representing it as a combination of high-level monitoring module (which would allow to monitor devices, logs, etc.) and notifying a module that could send custom messages to predefined system components.	
Keywords / Key components	
Monitoring, Notifying, Data Streaming, Message Queue	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/self/monitoring_and_notifying_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: ATHEMS **Suggestion and comments: N/A**	

2.6.1.5 Automated configuration enabler

Enabler: *Automated configuration*	**Id:** *SELF15*
Owner and Support: *SRIPAS*	
Related Deliverable/s: *D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Ongoing*	
Enabler Description	
Automated Configuration Enabler keeps heterogenous devices and services synchronised with their configurations. User can update configuration and define fallback configurations in case of errors. Self- component will be responsible for reacting to changing environment and updating configuration as necessary.*	
Keywords / Key components	
Configuration, Self-Management, Synchronization	

Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/self/automated_configuration_enabler.html
Used by Open Callers, suggestions and comments
Open Caller name: ATHEMS **Suggestion and comments: N/A**

2.6.2 Federated machine learning enablers

2.6.2.1 FL Orchestrator

Enabler: *FL Orchestrator*	**Id:** *T52E1*
Owner and Support: *PRO, SRIPAS, UPV*	
Related Deliverable/s: *D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Ongoing*	
Enabler Description	
The FL orchestrator is responsible of specifying details of FL workflow(s)/pipeline(s). This includes FL job scheduling, managing the FL life cycle, selecting, and delivering initial version(s) of the shared algorithm, as well as modules used in various stages of the process, such as training stopping criteria. Finally, it can specify ways of handling different "error conditions" that may occur during the FL process.	
Keywords / Key components	
Orchestrator, Federated Learning, Lifecycle.	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/federated/fl_orchestrator.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A **Suggestion and comments: N/A**	

2.6.2.2 FL Training Collector

Enabler: *Fl Training Collector*	**Id:** *T52E2*
Owner and Support: *SRIPAS UPV, PRO*	

Related Deliverable/s: *D5.1, D5.2, D5.3, D5.4*
Status of Enabler: *Ongoing*
Enabler Description
The FL training process involves several independent parties that commonly collaborate in order to provide an enhanced ML model. In this process, the different local update suggestions shall be aggregated accordingly. This duty within ASSIST-IoT will be tackled by the FL Training Collector, which will also be in charge of delivering back the updated model. The FL training collector will consist of two components: (i) the combiner responsible of providing updates with respect to the shared averaged model, and (ii) the I/O component which will carry out the input and output communications of the enabler.
Keywords / Key components
Federated Learning, Model Update, Aggregator, Model Enhancement
Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/federated/fl_training_collector.html
Used by Open Callers, suggestions and comments
Open Caller name: N/A **Suggestion and comments: N/A**

2.6.2.3 FL Repository

Enabler: *FL Repository*	**Id:** *T52E3*
Owner and Support: *SRIPAS, PRO, UPV*	
Related Deliverable/s: *D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Ongoing*	
Enabler Description	
The FL repository will be a set of different databases, including initial ML algorithms, already trained ML models suitable for specific data sets and formats, averaging approaches, and auxiliary repositories for other additional functionalities that may be needed, and are not specifically identified yet.	
Keywords / Key components	
Federated Learning, Repository, Model Storage	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/federated/fl_repository.html	

Used by Open Callers, suggestions and comments
Open Caller name: N/A **Suggestion and comments: N/A**

2.6.2.4 FL Local Operations

Enabler: *FL Local Operations*	**Id:** *T52E4*
Owner and Support: *SRIPAS, PRO, UPV*	
Related Deliverable/s: *D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Ongoing*	
Enabler Description	
FL Local Operations enabler is an embedded enabler within each FL involved party/device of the FL systems. The FL Local Operation enabler will consist of four components: Local Data Transformer component (that will be in charge of guaranteeing that data is appropriately formatted for the FL model in use), Local Model Training component, Local Model Inference component, and Communication component (to enable in and out communications between involved local parties and FL orchestrator and FL collector).	
Keywords / Key components	
Federated Learning, Local Training, Local Inferencing, FL Party	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/federated/fl_local_operations.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A **Suggestion and comments: N/A**	

2.6.3 Cybersecurity enablers

2.6.3.1 Authorisation enabler

Enabler: *Authorisation*	**Id:** *T53E1*
Owner and Support: *S21SEC*	
Related Deliverable/s: *D4.1, D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Ongoing*	

Enabler Description
Authorisation server offers a decision-making service based on XACML policies. It has different modules that interact and can be deployed independently such as, PEP (Policy Enforcement Point), PAP (Policy Administration Point), PIP (Policy Information Point) and PDP (Policy Decision Point).
Keywords / Key components
XACML, PEP, PAP, PIP, PDP, policies
Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/cybersecurity/authorization_enabler.html
Used by Open Callers, suggestions and comments
Open Caller name: N/A
Suggestion and comments: N/A

2.6.3.2 Identity Manager enabler

Enabler: *Identity Manager*	**Id:** *T53E2*
Owner and Support: *S21SEC*	
Related Deliverable/s: *D4.1, D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Finished (potential enhancements until October 2023)*	
Enabler Description	
Using OAuth2 protocol, it will offer a federated identification service where service requester and provider will be able to establish a trusted relation without previously knowing each other. This way a secure identification process is completed without the service provider having received the requester credentials.	
Keywords / Key components	
OAuth2, federated, trusted, secure, credentials	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/cybersecurity/identity_manager_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A	
Suggestion and comments: N/A	

2.6.3.3 Cybersecurity Monitoring enabler

Enabler: *Cybersecurity Monitoring*	**Id:** *T53E3*
Owner and Support: *S21SEC*	
Related Deliverable/s: *D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Finished (Packaging the enabler)*	
Enabler Description	
Cybersecurity monitoring enabler, provides security awareness, visibility and infrastructure monitoring. Having raw data as input, the enabler will set a series of processing steps that will enable the discovery of cybersecurity threats, going through a sequence step: (i) collecting, parsing, and normalizing input events, (ii) enriching normalized events, (iii) correlating events for detecting cybersecurity threats.	
Keywords / Key components	
Security, agentless, monitoring, discovery, threats, normalizing, cybersecurity, detecting	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/cybersecurity/cybersecurity_monitoring_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A **Suggestion and comments: N/A**	

2.6.3.4 Cybersecurity Monitoring Agent enabler

Enabler: *Cybersecurity Monitoring Agent*	**Id:** *T53E4*
Owner and Support: *S21SEC*	
Related Deliverable/s: *D4.1, D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Finished (potential enhancements until October 2023)*	
Enabler Description	
Perform functions of an endpoint detection and response system, monitoring and collecting activity from end points that could indicate a threat. Security agent runs at a host-level, combining anomaly and signature-based technologies to detect intrusions or software misuse.	
Keywords / Key components	

Endpoint, response, detection, collecting, host-level, anomaly, intrusions, monitoring
Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/cybersecurity/cybersecurity_monitoring_agent_enabler.html
Used by Open Callers, suggestions and comments
Open Caller name: N/A **Suggestion and comments: N/A**

2.6.4 DLT-based enablers

2.6.4.1 Logging and auditing enabler

Enabler: *Logging and Auditing*	**Id:** *T54E1*
Owner and Support: *CERTH*	
Related Deliverable/s: *D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Finished (Packaging the enabler)*	
Enabler Description	
This enabler will log critical actions that happen during the data exchange between ASSIST-IoT stakeholders to allow for transparency, auditing, non-repudiation and accountability of actions during the data exchange. It will also log resource requests and identified security events to help to provide digital evidence and resolve conflicts between stakeholders, when applicable.	
Keywords / Key components	
Logging, Auditing, DLT-based	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/dlt/logging_and_auditing_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A **Suggestion and comments: N/A**	

2.6.4.2 Data integrity verification enabler

Enabler: *Data Integrity Verification*	**Id:** *T54E2*
Owner and Support: *CERTH, ICCS, KONECRANES, S21SEC*	
Related Deliverable/s: *D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Finished (Packaging the enabler)*	
Enabler Description	
This is an enabler responsible for providing DLT-based data integrity verification mechanisms that allow data consumers to verify the integrity of any data at question. Network peers host smart contract (chaincode) which includes the data integrity business logic. It stores hashed data in a data structure and it compares it with the hashed data of the queries made by clients in order to verify their integrity.	
Keywords / Key components	
Verification, DLT-based	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/dlt/data_integrity_verification_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A **Suggestion and comments: N/A**	

2.6.4.3 Distributed broker enabler

Enabler: *Distributed Broker*	**Id:** *T54E3*
Owner and Support: *CERTH, ICCS, KONECRANES, S21SEC*	
Related Deliverable/s: *D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Finished (Packaging the enabler)*	
Enabler Description	
This enabler will provide a mechanism that will facilitate data sharing between heterogeneous IoT devices belonging to various edge domains and/or between different enablers of the architecture. Using the security features of DLT, it will deal with data source metadata management and provide to the data consumers trustable, findable and retrievable metadata for the data sources.	
Keywords / Key components	

Sharing, DLT-based, metadata, Management
Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/dlt/distributed_broker_enabler.html
Used by Open Callers, suggestions and comments
Open Caller name: N/A **Suggestion and comments: N/A**

2.6.4.4 DLT-based FL enabler

Enabler: *DLT-based FL*	**Id:** *T54E4*
Owner and Support: *CERTH, KONECRANES, S21SEC*	
Related Deliverable/s: *D5.1, D5.2, D5.3, D5.4*	
Status of Enabler: *Under development*	
Enabler Description	
This enabler is a system that uses blockchain technology to provide a secure reputation mechanism for local operators in a federated learning (FL) system. The reputation mechanism prevents free-riders and malicious adversaries from accessing the global model without contributing to it. The system calculates reputation scores for each local operator instance and stores them on a permissioned blockchain network, ensuring privacy and security. The FL training collector uses the scores to decide on penalties or incentives for the FL local operations. The FL-DLT enabler consists of three components: the DLT communicator, the reputation score calculator, and the DLT storage.	
Keywords / Key components	
FL, DLT-based, Decentralised, Models, Validation, Exchange	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/dlt/dlt_based_fl_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: N/A **Suggestion and comments: N/A**	

2.6.5 Manageability

2.6.5.1 Enablers manager

Enabler: *Enablers manager*	**Id:** *T55E1*
Owner and Support: *UPV, SRIPAS, CERTH*	
Related Deliverable/s: *D5.2, D5.3, D5.4*	
Status of Enabler: *Finished (potential enhancements until October 2023)*	
Enabler Description	
This enabler will serve as a registry of enablers and, in case they are deployed, the retrieval of their status. In particular, it will: (a) Allow the registration of an enabler (this is, from an ASSIST-IoT repository). Essential enablers will be pre-registered, (b) Retrieve a list of currently-running enablers, (c) Depict the status and the specific logs of an enabler (the latter only if the enabler with log collection capabilities is in place), (d) facilitate the deployment of standalone enablers (mostly for those that have to be present at any deployment).	
Keywords / Key components	
Enablers registration, Enablers status, Helm repository, GUI	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/manageability/registration_and_status_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: ATHEMS	
Suggestion and comments: N/A	

2.6.5.2 Composite services manager

Enabler: *Management of services and enablers workflow*	**Id:** *T55E2*
Owner and Support: *UPV*	
Related Deliverable/s: *D5.2, D5.3, D5.4*	
Status of Enabler: *Finished (potential enhancements until October 2023)*	
Enabler Description	
This enabler will present a graphical environment where ASSIST-IoT administrators can instantiate the enablers required to work, and also to connect them to compose a composite service (i.e., a workflow). Having information about the physical topology and available k8s nodes/clusters, it will allow the user to	

decide whether to select the proper node or cluster for deploying an enabler, or let the system decide based on pre-defined architectural rules.
Keywords / Key components
Service composition, GUI
Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/manageability/registration_and_status_enabler.html
Used by Open Callers, suggestions and comments
Open Caller name: N/A **Suggestion and comments: N/A**

2.6.5.3 Devices management enabler

Enabler: *Clusters and topology manager*	**Id:** *T55E3*
Owner and Support: *UPV*	
Related Deliverable/s: *D5.2, D5.3, D5.4*	
Status of Enabler: *Finished (potential enhancements until October 2023)*	
Enabler Description	
The main functionality of this enabler will be to register: (i) a smart IoT device in a deployment, and (ii) a cluster in an ASSIST-IoT deployment, including in the latter case all the necessary messages to notify it to the smart orchestrator. It will also execute all the required actions related to networking for enabling connectivity among isolated/independent clusters, including those that have been added via VPN/SD-WAN technology. Besides, It will allow monitoring any registered node and device in the deployment, including its status (i.e., available and used resources) and current instantiated enablers' components.	
Keywords / Key components	
K8s, Cluster registration, IoT device registration, GUI	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/manageability/devices_management_enabler.html	
Used by Open Callers, suggestions and comments	
Open Caller name: ATHEMS **Suggestion and comments: N/A**	

3 Open Callers’ Feedback

In order to obtain a comprehensive understanding of Open Callers' experience with the ASSIST-IoT Technical and Support Documentation and GitLab Repository, an online questionnaire was developed. This survey enabled us to gain valuable insights that were instrumental in enhancing the documentation and making it more user-friendly. Additionally, the questionnaire allowed us to identify areas of ambiguity or difficulty that may have hindered users' experience, enabling us to address these issues and make the Documentation and GitLab Repository more intuitive or incorporate requested features. In conclusion, the questionnaire proved to be a valuable tool for improving the enablers’ functionality and delivering superior documentation for users.

In a distinct section of the survey, Open Callers were queried regarding their overall impressions of GitLab Repository. Based on the feedback obtained from their responses, we aim to evaluate the ease of accessing and managing artifacts, identify areas for improvement, and glean recommendations for optimization. Ultimately, we will utilize the information gathered through this process to refine GitLab Repository, ensuring its seamless integration into users' workflows.

The present survey was created using Microsoft Forms, an online survey and quiz production tool developed by Microsoft Corporation. This software is designed to facilitate the effortless creation of forms, surveys, quizzes, and polls, which can be easily disseminated among targeted audiences. Microsoft Forms offers a broad range of question types, facilitating the swift development of surveys and quizzes while also allowing for seamless collaboration with other users. Moreover, this tool provides robust analytics and reporting capabilities, enabling us to extract meaningful insights from the collected data.

3.1 Questionnaire Context and Target Audience

The Table 3 encompasses the questions that were used to gather data from Open Callers. The questions are divided into three categories, including questions about ASSIST-IoT Documentation, technical scope of ASSIST-IoT and GitLab Repositories. This questionnaire was provided online at Microsoft Forms to all 7 projects of the first ASSIST-IoT Open call at end of February 2023 (upon completion of their projects). Each question is designed to provide an in-depth understanding of the Open Callers’ experience and recommendations. The responses to all these questions will be provided in the next section, and they will be utilized to acquire insights into the survey topic and reach conclusions about the survey results.

Table 3. Questionnaire Context

Question	**Answer Type**
ASSIST-IoT Documentation Section:	
1) Open Caller-Project Title	Free Text
2) Did you use the technical documentation provided here: *https://assist-iot-enablers-documentation.readthedocs.io/en/latest/*	Yes/No
3) In case you were not aware of the documentation, how did you manage to learn about the enablers?	Free Text
4) Did the documentation provide you with enough information to understand and utilize the enablers?	Rating Scale 1-5
5) Do you have any suggestions to improve or update the documentation?	Free Text
6) How user-friendly was the documentation?	Free Text
7) Which enablers did you use?	Multiple Choice

8) How good did you find the documentation in our deliverables when preparing your proposal?	Free Text
Technical Scope of ASSIST-IoT Section:	
9) How different did you feel about the technical scope of ASSIST-IoT enablers and architecture comparing before and after working within ASSIST-IoT?	Free Text
10) How did you managed to fulfil any unexpected gap between the promised ASSIST-IoT technical scope in comparison with what you faced in the process.	Free Text
GitLab Section:	
11) Did you find the GitLab Repository useful?	Rating Scale 1-5
12) What was your experience using and managing GitLab artifacts?	Rating Scale 1-5
13) How was the accessibility process at GitLab repositories?	Rating Scale 1-5
14) Based on your experience using GitLab, what recommendations would you suggest?	Free Text

3.2 Questionnaire Results

This section's objective is to present the questionnaire's findings, draw attention to how the comments were addressed and highlight any improvements that were accomplished. The responses are also presented extensively in **Annex D: Open Callers'** Questionnaire Responses**.**

The chart **Enablers used by Open Callers** depicts the enablers that were used by Open Callers. This statistical breakdown enables us to identify the most favored enablers. Furthermore, the feedback obtained from the questionnaire is based on the Open Callers' experience with the above enablers.

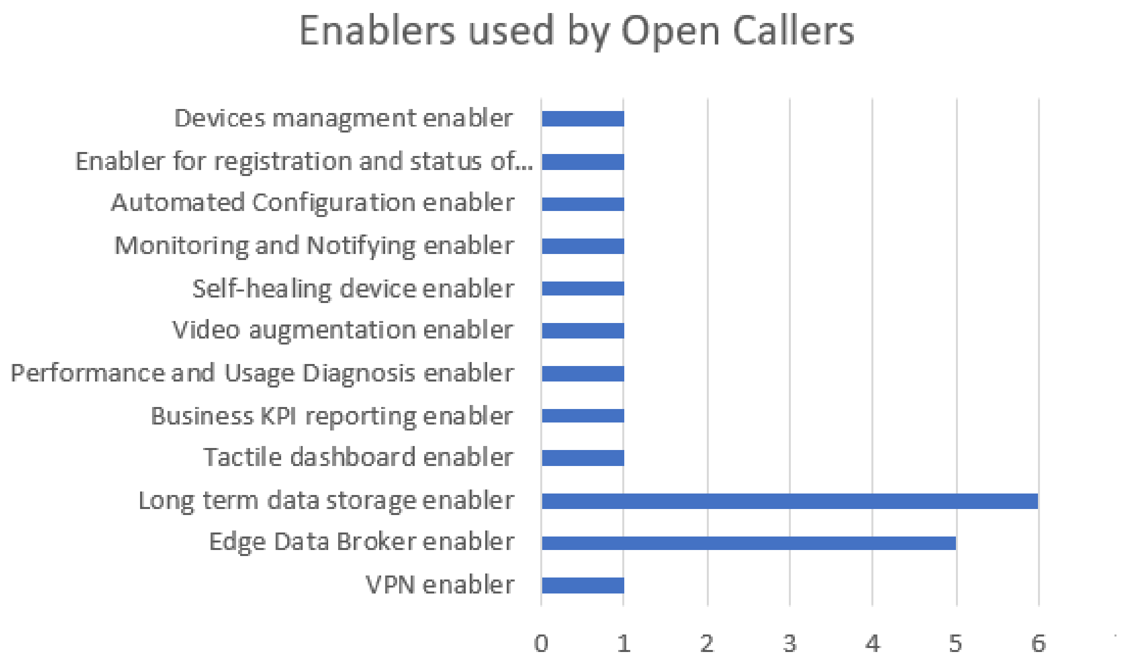


Figure 10. Enablers used by Open Callers

3.2.1 ASSIST-IoT Documentation Section Feedback

In order to continuously improve and refine the Technical and Support Documentation, it is essential to understand the level of satisfaction of our users. Therefore, we have collected and analysed data on Open

Callers' satisfaction and feedback to gain insights into how well the documentation is meeting their needs. In this section, we will present the results of our analysis, including statistics on user satisfaction and verbatim feedback from Open Callers. This information helped us to identify areas for improvement and enhance the overall quality and usability of our documentation.

The pie chart below (**Technical Documentation Usage**) illustrates how many Open Callers used the ASSIST-IoT Technical and Support Documentation. It demonstrates that six out of seven Open Callers used the documentation, indicating a high level of engagement with the Enablers. This indicates that the documentation was clear, providing users with the information they needed to properly use the Enablers. The Open Caller who did not use the documentation was able to learn about the Enablers through frequent interactions and communication with Pilot 3a Host.

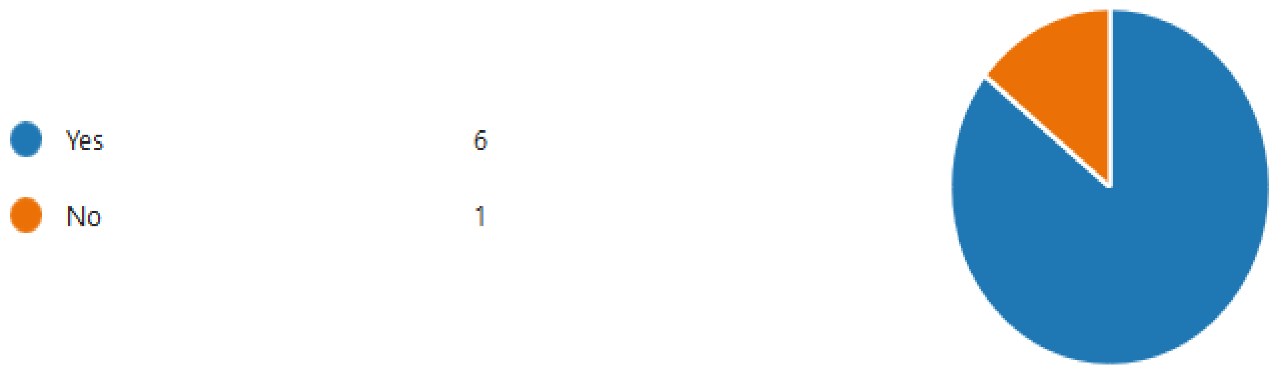


Figure 11. Technical Documentation Usage

The chart **Understanding and Utilizing Enablers** displays the feedback from Open Callers about their ability to comprehend and apply the enablers based on the documentation provided to them. The purpose of this chart is to showcase the effectiveness of the documentation in helping Open Callers understand and utilize the enablers. The rating of 4.33/5 for the question indicates that the Open Callers found the documentation to be helpful in understanding and utilizing the enablers. This is a very positive indication that the documentation is comprehensive and useful.

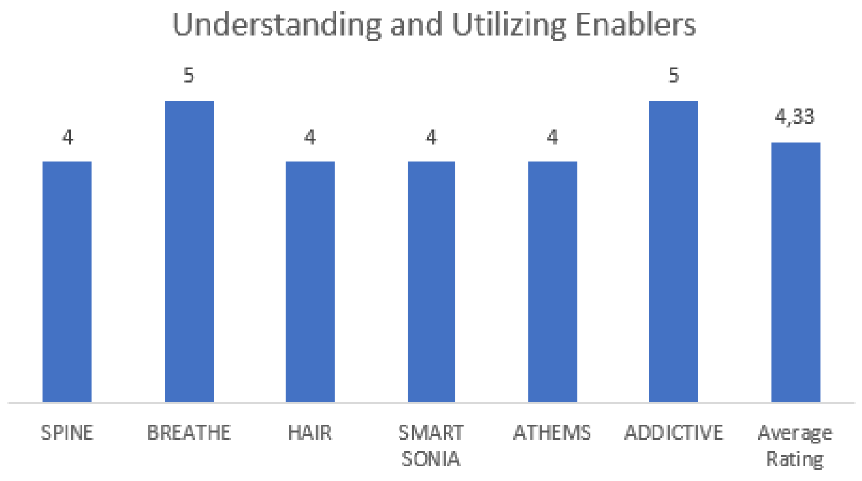


Figure 12. Understanding and Utilizing Enablers

Based on the responses received from questionnaire (chart: **Documentation User-Friendliness),** we are pleased to report that ASSIST-IoT Technical and Support Documentation has been rated as very user-friendly, with an average rating of 4.5 out of 5. This statistic indicates that the majority of Open Callers find the documentation easy to use and navigate, which is a crucial aspect of any successful documentation. It also indicates that the documentation's design was very effective.

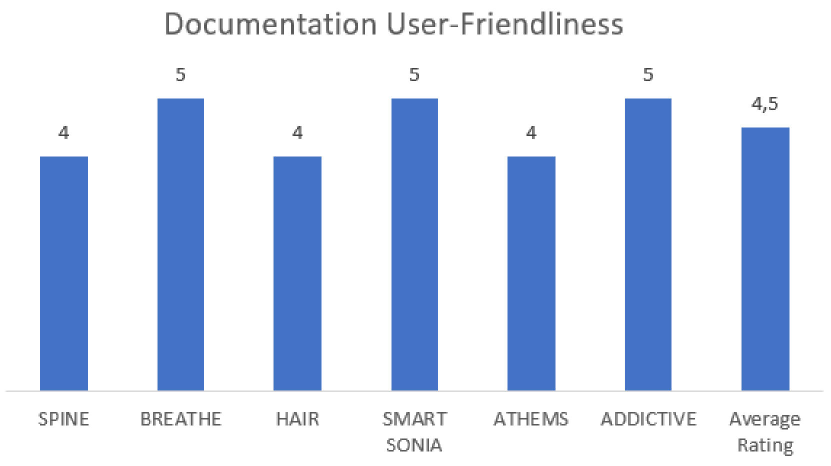


Figure 13. Documentation User-Friendliness

3.2.1.1 Suggestions and Improvements made by ASSIST-IoT team

The suggestions and improvements that have been implemented in response to Open Callers' feedback are summarized in this section. It includes the different ideas that have been identified to improve the user experience, as well as the steps that are being taken to put them into action.

In response to the survey question, users provided valuable insights on how to enhance the documentation further.

- An Open Caller noted that the documentation should be updated and contain more detailed information at the beginning of the pilot. **Related updates were performed.**
- Most Open Callers acknowledged that the documentation was updated during the project with more images and new information, making it more useful. Currently, **the documentation has been finalized for the fully developed enablers.**
- An Open Caller recommended that the payload formatting be included in a column in the REST API endpoints section. As a response, **we have updated the documentation, accordingly, including the payload formatting in a separate column to make it more accessible and user-friendly.**
- Overall, the feedback given by Open Callers was positive and has proven to be of great significance in enhancing the ASSIST-IoT documentation, thereby meeting the requirements and expectations of the users.

3.2.2 Technical Scope-Section of ASSIST-IoT Feedback

The purpose of this section of the questionnaire was to gather feedback on Open Callers' perceptions of the technical scope of the ASSIST-IoT enablers and architecture, and how those perceptions changed over the course of the project. Specifically, participants were asked to reflect on how their understanding of the project's technical scope differed before and after working within the project. Additionally, Open Callers were asked to describe how they managed to bridge any unexpected gaps between the promised technical scope and the reality of their experience.

The feedback gathered in response to these questions will be used to gain a better understanding of how well the technical scope of the ASSIST-IoT project was communicated and understood by Open Callers, as well as to identify any areas for improvement.

By analysing the responses to these questions, we can gain insight into the factors that influenced participants' perceptions of the technical scope of the project, as well as how effectively they were able to navigate any unexpected challenges. This information can be utilized to enhance the project and guarantee that Open Callers have a comprehensive understanding of the technical scope of the project and are prepared to handle any unexpected challenges that may arise.

3.2.2.1 Open Callers' Perception of Technical Scope and Enabler Architecture

Overall, the responses indicate that **Open Callers generally had a positive perception of the technical scope of the ASSIST-IoT project**. Many Open Callers reported feeling that the technical scope was ambitious but achievable, and that **their confidence increased over the course of the project**. Others noted that **they gained a clearer understanding of how the enablers worked and the kind of services they could offer**, which helped to reduce the amount of work necessary to complete certain tasks.

One Open Caller noted that the **enablers were well-documented**, which enabled them to start development before the enablers were fully ready. This suggests that **the documentation provided was effective in helping participants to navigate any unexpected challenges** that arose during the development process. Another Open Caller reported that **the project's technical scope met their initial expectations**, which is a positive indication that the project was well-designed and executed.

Furthermore, it appears that **Open Callers gained a lot of valuable knowledge about cloud technologies** such as HELM and Kubernetes, as well as the overall project architecture. This knowledge will be useful for future projects and may help to inform the development of similar projects in the future.

It is worth noting that one Open Caller did not have any remarks to make about the technical scope of the project. While this response does not provide much detail, it is still useful to note that at least one participant did not have any significant issues with the project's technical scope and everything had met his expectations.

In summary, the responses to the first question suggest that the technical scope of the ASSIST-IoT project was generally well-received by Open Callers, and that the project provided valuable learning opportunities for Open Callers to develop their technical skills and knowledge.

3.2.2.2 Managing Unexpected Gaps in ASSIST-IoT Technical Scope, Open Callers Experiences – Suggestions and Improvements

Open Callers encountered a few unexpected gaps in the technical scope of the ASSIST-IoT project but were generally able to manage them effectively. The subsequent bullet points represent the issues that have been reported by the Open Callers and how they managed those issues:

- **SPINE**: The initial state of the Video augmentation enabler was not developed as expected, but they were able to solve this problem by improving the enabler structure.
- **BREATHE**: Some enablers and functions were not available at the beginning of the project execution but were provided later. This is understandable given that the project was in its early stages and suggests that the project team was able to respond to feedback and make adjustments as necessary to ensure that the project met its goals.
- **HAIR**: They were able to integrate the enablers as expected, and that any small issues that arose were targeted quickly with prompt solutions. This suggests that ASSIST-IoT partners were well-prepared to manage unexpected technical gaps, and that they had effective communication and collaboration with Open Callers in the pilot phase.
- **RAZOR**: They received strong interaction and support from the Pilot Host, which likely contributed to their ability to manage any unexpected technical gaps that arose. This indicates that ASSIST-IoT partners were committed to providing effective support and guidance to Open Callers throughout the project.
- **SMART SONIA**: Reported that they researched possible solutions to any unexpected gaps and discussed them with the partners of the ASSIST-IoT project before choosing the most relevant solution for the project's goals. This suggests that the ASSIST-IoT partners were proactive in seeking out

solutions to any unexpected challenges, and that they were able to work collaboratively with partners to find the best solutions.

- **ADDICTIVE**: They did not find any unexpected gaps in the technical scope of the project. While this response does not provide much detail, it is a positive indication that at least one Open Caller did not encounter any significant challenges during the project.

In conclusion, **the responses suggest that while Open Callers encountered a few unexpected gaps in the technical scope of the ASSIST-IoT project, they were generally able to manage these gaps effectively. The ASSIST-IoT partners were flexible, adaptable, and committed to providing effective support to Open Callers, which likely contributed to the project's and open calls overall success.**

3.2.3 GitLab Repository Section of the Questionnaire

Based on the responses provided by the Open Callers who used the GitLab Repository of the project, it seems that **the repository was moderately useful to them, with an average rating of 3.5/5**. While this is not a high rating, **it is still a positive indication that the repository was able to provide valuable assistance to the Open Callers.**

The average rating of 3.5 out of 5 for the question on experience using and managing GitLab artifacts is also indicative of a moderate level of satisfaction. This suggests that while the users may have been able to use the repository effectively, they may have encountered some challenges or difficulties in the process.

On a more positive note, the high average rating of 4.25 out of 5 for the question on accessibility process at GitLab repositories is a strong indication that **the repository was easy to access and navigate**. This suggests that users were able to quickly and efficiently locate the information they needed, which is a key aspect of user satisfaction. The following chart GitLab Repository Ratings provides an overview of the ratings given by Open Callers who used the GitLab Repository of the project.

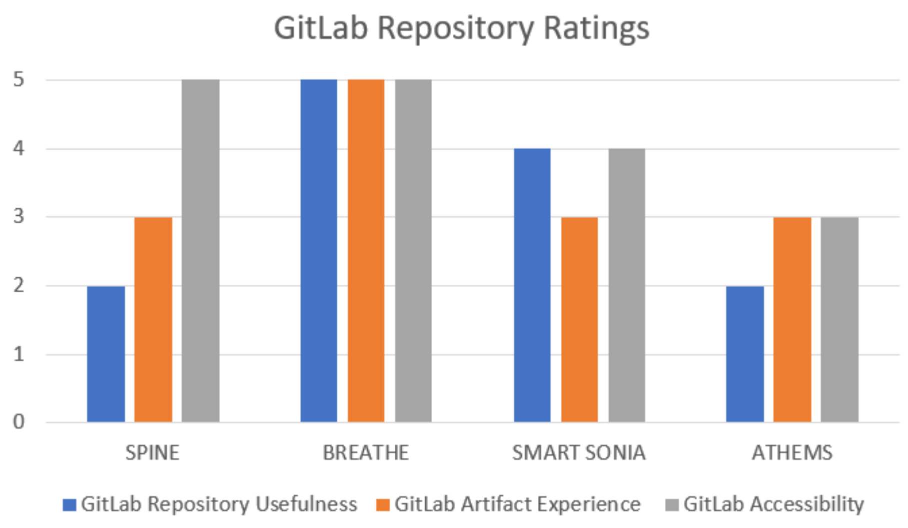


Figure 14. GitLab Repository Ratings

3.2.3.1 Suggestions and Improvements

Based on the feedback provided by Open Callers who used the GitLab Repository of the project, there are few potential suggestions and improvements that was implemented in order to enhance the user experience and increase the value provided by the repository.

- **Add Documentation to Repository:** It was proposed by Open Callers to include a link in the GitLab repository to the external Technical and Support Documentation, this was achieved by including the link to GitLab Repository.

- **More frequent updates:** Open Callers expressed a desire for more frequent updates to the GitLab repository to better understand the state of the project. The enablers now reached its final version, and all the features and functionality that intended to include have been implemented.
- **Easier landing to ASSIST-IoT technology:** where to start? How the different components fit my structure? Which are the main customizations required? How to reach interoperability?
- **Installation should be more automated**, and some recovery mechanisms should be put in place.
- **GWEN was not able** to be used yet, so they had to use their own hardware providers (it an IoT gateway was needed).
- **OpenAPI** integration was not mature enough, therefore shortcuts were needed to be found. For the 2nd round of Open Calls, this should be more straightforward.
- **Overall understanding and deployment time** was not agile enough to cover the necessities of the Open Call projects, that in some cases opted for generic images rather for utilizing the full Helm chart of enablers and ASSIST-IoT stack.

Overall, these suggestions and improvements could help to enhance the user experience and make it easier for users to understand the state of the project and use GitLab more effectively. By addressing these concerns, GitLab repository can continue to provide a valuable service to its users and improve its overall functionality and usability.

3.3 Documentation Relevancy for Open Callers' proposals

Based on the feedback received from Open Callers, it appears that the documentation included in the deliverables was generally well-received and beneficial for their proposals. The Open Callers found the documentation to be pertinent, advantageous, easily accessible, and well-organized. Furthermore, some Open Callers indicated that the documentation was highly detailed, implying that it furnished a comprehensive comprehension of the project. It is important to note that a few users were unaware of the Technical and Support Documentation during the proposal phase, which hindered their ability to locate the information they required.

In conclusion, the feedback implies that the documentation provided was successful in equipping users with the requisite knowledge to formulate their proposals. The responses of each Open Callers are provided in detail in Table 4.

Table 4. Responses on the effectiveness of documentation in proposal developments

Question: How good did you find the documentation in our deliverables when preparing your proposal?	
SPINE	Relevant information, used for the architecture definition
BREATHE	It was useful to know the general structure of the project and the way the different functionality were deployed.
HAIR	The documentation was easily accessible and complete at a very technical level that made the preparation of the proposal easier.
RAZOR	Highly detailed
SMART SONIA	All the documentation is well structured and can be easily understood.
ATHEMS	When preparing the proposal we found the available deliverables good enough to get an idea of how the overall ASSIST-IoT architecture looked like and what were how we were expected to interoperate with its components.
ADDICTIVE	Since I wasn't aware of the readthedocs page in the proposal phase, it was harder to find the information than it's currently.

4 Conclusion

This document presents the latest release of technical documentation that encapsulates the achievements of the ASSIST-IoT Project up to M30 (April 2023). The project's primary technical accomplishments have been realized through the successful execution of Work Packages 4 and 5, which were responsible for the creation and progression of enablers. The current document, D6.6, primarily focuses on documenting the work completed in these two work packages. It outlines the resulting outputs, while also providing guidance on how to effectively utilize them. Furthermore, the document highlights the valuable feedback received from the first Open Callers through a questionnaire, and how this feedback has contributed to the improvement of the Technical and Support Documentation and the GitLab Repository. Any additional information or updates regarding the documentation and the Open Call No2 participants will be made available through the next WP6 deliverable.

Annex A: Top-tier script (smart orchestrator)

```
#!/bin/bash

set -eux

function install_prerequirements() {
    sudo apt-get install curl
    sudo apt-get install git
    sudo mkdir -p /mnt/data
    sudo touch /mnt/data/targets.json
}

function install_osm() {
    cd osm/installers/
    chmod +x install_osm.sh
    ./install_osm.sh -D ../
}

function install_prometheus_federate() {
    helm repo add public-repo https://gitlab.assist-iot.eu/api/v4/projects/85/packages/helm/stable
    helm repo update
    helm install assist-prometheus public-repo/prometheus-federate
}

function install_prometheus_helm() {
    helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
    helm repo update
    kubectl create ns monitoring
    helm install prometheus-stack prometheus-community/kube-prometheus-stack --set grafana.service.type=NodePort --set prometheus.service.type=NodePort --set prometheus.prometheusSpec.scrapeInterval="5s" --namespace monitoring
}
```

```
function install_api(){
    helm install smartorchestrator public-repo/smartorchestrator
}

function install_dashboard(){
        helm    install    dashboard    public-repo/manageability-dashboard    --set
frontend.service.ports.dashboard.nodePort=30080

}
function create_namespacejob(){
    kubectl create ns job
}

# UNINSTALL
function uninstall_osm() {
    cd osm/installers/
    chmod +x uninstall_osm.sh
    ./uninstall_osm.sh -D ../
}

function uninstall_smart() {
    sudo rm /mnt/data/targets.json
    sudo kubeadm reset
}

function usage() {
    echo                       # Function: Print a help message.
    echo "usage: $0 [-t types]"
    echo "Must run as root"
    echo "options:"
    echo "  -u      uninstall SmartOrchestrator"
    echo "  -i      install SmartOrchestrator"
    exit 1
}
function exit_abnormal() {                        # Function: Exit with error.
```

```
  usage
  exit 1
}

while getopts "ui" options; do

  case "${options}" in
    u)
      uninstall_osm
      uninstall_smart
      ;;
    i)
      install_prerequirements
      install_osm
      install_prometheus_helm
      install_prometheus_federate
      install_api
      install_dashboard
      create_namespacejob
      ;;
  esac
done

if [ $OPTIND -eq 1 ]; then
    echo "No options were passed";
    exit_abnormal
fi
```

Annex B: K3 Installation script

```
#!/bin/bash


function install_cilium() {
    # Check the node architecture (arm64 or amd64)
    CILIUM_CLI_VERSION=$(curl -s https://raw.githubusercontent.com/cilium/cilium-cli/master/stable.txt)
    CLI_ARCH=amd64
    if [ "$(uname -m)" = "aarch64" ]; then CLI_ARCH=arm64; fi
    curl -L --fail --remote-name-all https://github.com/cilium/cilium-cli/releases/download/${CILIUM_CLI_VERSION}/cilium-linux-${CLI_ARCH}.tar.gz{,.sha256sum}
    sha256sum --check cilium-linux-${CLI_ARCH}.tar.gz.sha256sum
    sudo tar xzvfC cilium-linux-${CLI_ARCH}.tar.gz /usr/local/bin
    rm cilium-linux-${CLI_ARCH}.tar.gz{,.sha256sum}


    # cilium install --helm-set-string ipam.operator.clusterPoolIPv4PodCIDR=$POD_CIDR --helm-set-string disableEnvoyVersionCheck=true
    cilium install --helm-set-string ipam.operator.clusterPoolIPv4PodCIDR=$POD_CIDR
    cilium clustermesh enable --service-type NodePort
}


function install_helm() {
    # sudo snap install helm --classic
    curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
    chmod 700 get_helm.sh
    ./get_helm.sh
}


function install_k3s_agent() {
    curl -sfL https://get.k3s.io | INSTALL_K3S_VERSION="v1.23.10+k3s1" K3S_URL="https://$SERVER_IP:6443" K3S_TOKEN=$SERVER_TOKEN sh -
}
```

```
function install_k3s_server() {

    curl -sfL https://get.k3s.io | INSTALL_K3S_VERSION="v1.23.10+k3s1" sh -s - server
--flannel-backend=none --disable-network-policy --cluster-init --token 4ss1st_10t --
tls-san $SERVER_IP --node-external-ip $SERVER_IP --cluster-cidr $POD_CIDR

    export KUBECONFIG=/etc/rancher/k3s/k3s.yaml

    # TODO --write-kubeconfig-mode "0644"

}


function restart_cilium_unmanaged_pods() {

               kubectl      get      pods      --all-namespaces      -o      custom-
columns=NAMESPACE:.metadata.namespace,NAME:.metadata.name,HOSTNETWORK:.spec.hostNetw
ork --no-headers=true | grep '<none>' | awk '{print "-n "$1" "$2}' | xargs -L 1 -r
kubectl delete pod

}


function dynamic_cluster_installer() {

  echo 'Dynamic cluster installer fired'

  read -p 'name (Car model and ID, ex: 1234PED): ' NAME

  read -p 'description: ' DESCRIPTION

  read -p 'group: ' GROUP


  cat <<EOF | kubectl apply -f -

  apiVersion: batch/v1

  kind: Job

  metadata:

    name: installer

  spec:

    ttlSecondsAfterFinished: 100

    template:

      spec:

        containers:

        - name: installer

          image: fmbiot/clusterinstaller

          imagePullPolicy: Always

          env:

          - name: CLUSTER_DATA
```

```
            value: '{"name": "$NAME", "description": "$DESCRIPTION","group": "$GROUP"
}'
          - name: OS_IP
            value: "158.42.161.177"
          - name: OS_PORT
            value: "5007"
        restartPolicy: Never
    backoffLimit: 1
EOF
}

function helm_api_install(){
  helm repo add assist-tools https://fmbiot.github.io/helm-charts/
  helm install helm-api assist-tools/helm-api

   kubectl  wait  --for=condition=ready  pod  -l  app.kubernetes.io/name=helm-api  --
timeout=100s &
  completion_pid=$!

  # wait for failure as background process - capture PID
   kubectl  wait  --for=condition=failed  pod  -l  app.kubernetes.io/name=helm-api  --
timeout=100s &
  failure_pid=$!

  # capture exit code of the first subprocess to exit
  wait -n $completion_pid $failure_pid

  # store exit code in variable
  exit_code=$?

  if (( $exit_code == 0 )); then
    echo "Helm API completed"
  else
    echo "Helm API failed with exit code ${exit_code}, exiting..."
    exit $exit_code
  fi

}

# Function: Print a help message.
function usage() {
    echo
    echo "usage: $0 [-t types]"
    echo "Server mode must run as root"
    echo "options:"
    echo "  -h          Prints this information"
    echo "  -c          Install Cilium CLI"
    echo "  -t          Type of K3s' component (AGENT , SERVER or DYNAMIC SERVER)"
    echo "  -i          (Only in Server mode) K3s server machine IP"
    echo "  -p          (Only in Server mode) K3s server POD CIDR"
    echo "  -s          (Only in Agent mode) K3s server url"
    echo "  -k          (Only in Agent mode) K3s server token"
    exit 1
}

# Function: Exit with error.
function exit_abnormal() {
  usage
  exit 1
}

while getopts "t:cdi:k:p:h:" options; do

  case "${options}" in
    t)
      K3STYPE=${OPTARG}
      if [ "$K3STYPE" != "AGENT" ] && [ "$K3STYPE" != "SERVER" ] && [ "$K3STYPE" !=
"DYNAMIC-SERVER" ]; then
        echo "Error: TYPE must be AGENT , SERVER or DYNAMIC SERVER"
        exit_abnormal
        exit 1
      fi
      ;;
    c)
      echo "Installing Cilium CLI"
      install_cilium_cli
      # CLI=${OPTARG}
      ;;
    d)
      echo "Installing dynamic cluster"
      dynamic_cluster_installer
      ;;
    i)
      SERVER_IP=${OPTARG}
      ;;
    k)
      SERVER_TOKEN=${OPTARG}
      ;;
    p)
      POD_CIDR=${OPTARG}
      ;;
    h)
      usage
      ;;
    :)
      echo "Error: -${OPTARG} requires an argument."
      exit_abnormal
      ;;
    *)
      exit_abnormal
      ;;
  esac
done
```

```
if [ "$EUID" -ne 0 ]; then
  echo "Must run as sudo";
  exit_abnormal
fi

#No options were passed to the script
if [ $OPTIND -eq 1 ]; then
    echo "No options were passed";
    exit_abnormal
fi

#Depending on the type, the installation changes
if [ "$K3STYPE" == "AGENT" ]; then
    install_k3s_agent
elif [ "$K3STYPE" == "SERVER" ]; then
    install_k3s_server
    install_helm
    install_cilium
fi
```

Annex C: kubeadm installation script

```
#!/bin/bash

function install_prerequirements() {
    swapoff -a
    sudo apt-get install ebtables ethtool
    sudo apt-get update -y
    #Install docker
    sudo apt-get install -y curl
    sudo apt-get install -y docker.io
    sudo docker version
}

function install_kubeadm() {
    K8S_VERSION=1.23.3-00
    sudo apt-get update && sudo apt-get install -y apt-transport-https
    sudo curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
    add-apt-repository "deb https://apt.kubernetes.io/ kubernetes-xenial main"
    sudo apt-get update -y
    echo "Installing Kubernetes Packages ..."
    sudo apt-get install -y kubelet=${K8S_VERSION} kubeadm=${K8S_VERSION} kubectl=${K8S_VERSION}
    cat << EOF | sudo tee -a /etc/default/kubelet
    KUBELET_EXTRA_ARGS="--cgroup-driver=cgroupfs"
EOF
    sudo apt-mark hold kubelet kubeadm kubectl
}

function init_kubeadm() {
    sudo swapoff -a
    sudo sed -i.bak '/.*none.*swap/s/^\(.*\)$/#\1/g' /etc/fstab
    cat <<EOF > /tmp/kubeadm-init-args.conf
apiVersion: kubeadm.k8s.io/v1beta3
```

```
kind: ClusterConfiguration
apiServerExtraArgs:
  service-node-port-range: 80-32767
networking:
  podSubnet: $POD_CIDR
EOF
    sudo kubeadm init --config /tmp/kubeadm-init-args.conf
    sleep 5
}

function kube_config_dir() {
    K8S_MANIFEST_DIR="/etc/kubernetes/manifests"
    mkdir -p $HOME/.kube
    sudo cp /etc/kubernetes/admin.conf $HOME/.kube/config
    sudo chown $(id -u):$(id -g) $HOME/.kube/config
}

function install_helm() {
    HELM_VERSION="v3.7.2"
    if ! [[ "$(helm version --short 2>/dev/null)" =~ ^v3.* ]]; then
        # Helm is not installed. Install helm
        echo "Helm3 is not installed, installing ..."
          curl  https://get.helm.sh/helm-${HELM_VERSION}-linux-amd64.tar.gz  --output
helm-${HELM_VERSION}.tar.gz
        tar -zxvf helm-${HELM_VERSION}.tar.gz
        sudo mv linux-amd64/helm /usr/local/bin/helm
        rm -r linux-amd64
        rm helm-${HELM_VERSION}.tar.gz
    else
        echo "Helm3 is already installed. Skipping installation..."
    fi
}

function install_k8s_storageclass() {
    echo "Installing open-iscsi"
    sudo apt-get update -y
    sudo apt-get install open-iscsi -y
    sudo systemctl enable --now iscsid
    OPENEBS_VERSION="3.1.0"
    echo "Installing OpenEBS"
    helm repo add openebs https://openebs.github.io/charts
    helm repo update
    helm install --create-namespace --namespace openebs openebs openebs/openebs --version ${OPENEBS_VERSION}
    helm ls -n openebs
    local storageclass_timeout=400
    local counter=0
    local storageclass_ready=""
    echo "Waiting for storageclass"
    while (( counter < storageclass_timeout ))
    do
        kubectl get storageclass openebs-hostpath &> /dev/null

        if [ $? -eq 0 ] ; then
            echo "Storageclass available"
            storageclass_ready="y"
            break
        else
            counter=$((counter + 15))
            sleep 15
        fi
    done
    [ -n "$storageclass_ready" ] || FATAL "Storageclass not ready after $storageclass_timeout seconds. Cannot install openebs"
    kubectl patch storageclass openebs-hostpath -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
}

function taint_master_node() {
    K8S_MASTER=$(kubectl get nodes | awk '$3~/master/'| awk '{print $1}')
    kubectl taint node $K8S_MASTER node-role.kubernetes.io/master-
    sleep 5
}

function install_prometheus(){
    helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
    helm repo update
    kubectl create ns monitoring;
    helm install prometheus-community/kube-prometheus-stack --generate-name --set grafana.service.type=NodePort --set prometheus.service.type=NodePort --set prometheus.prometheusSpec.scrapeInterval="5s" --namespace monitoring
}

function install_cilium(){
    CILIUM_CLI_VERSION=$(curl -s https://raw.githubusercontent.com/cilium/cilium-cli/master/stable.txt)
    CLI_ARCH=amd64
    if [ "$(uname -m)" = "aarch64" ]; then CLI_ARCH=arm64; fi
    curl -L --fail --remote-name-all https://github.com/cilium/cilium-cli/releases/download/${CILIUM_CLI_VERSION}/cilium-linux-${CLI_ARCH}.tar.gz{,.sha256sum}
    sha256sum --check cilium-linux-${CLI_ARCH}.tar.gz.sha256sum
    sudo tar xzvfC cilium-linux-${CLI_ARCH}.tar.gz /usr/local/bin
    rm cilium-linux-${CLI_ARCH}.tar.gz{,.sha256sum}

    cilium install --helm-set-string ipam.operator.clusterPoolIPv4PodCIDR=$POD_CIDR
    cilium clustermesh enable --service-type NodePort
}

function usage() {
    echo                        # Function: Print a help message.
    echo "usage: $0 [-t types]"
    echo "Must run as root"
    echo "options:"
    echo "  -t      Type of Kubernetes' component (AGENT or SERVER)"
    echo "  -p      Pod CIDR Network (Only SERVER Mode)"
    exit 1
}
function exit_abnormal() {                          # Function: Exit with error.
  usage
  exit 1
}

while getopts "t:p:" options; do

  case "${options}" in
    t)
      KUBETYPE=${OPTARG}
      if [ "$KUBETYPE" != "AGENT" ] && [ "$KUBETYPE" != "SERVER" ]; then
        echo "Error: TYPE must be SERVER or AGENT"
        exit_abnormal
        exit 1
      fi
      ;;
    p)
      POD_CIDR=${OPTARG}
      ;;
    :)
      echo "Error: -${OPTARG} requires an argument."
      exit_abnormal
      ;;
    *)
      exit_abnormal
      ;;
  esac
done

if [ "$EUID" -ne 0 ]; then
  echo "Must run as sudo";
  exit_abnormal
fi

#No options were passed to the script
if [ $OPTIND -eq 1 ]; then
    echo "No options were passed";
    exit_abnormal
fi

#Depending on the type, the installation changes
if [ "$KUBETYPE" == "AGENT" ]; then
    install_prerequirements
    install_kubeadm
else
    install_prerequirements
    install_kubeadm
    init_kubeadm
    kube_config_dir
    taint_master_node
    install_helm
    install_k8s_storageclass
    install_cilium
    install_prometheus
fi
```

Annex D: Open Callers' Questionnaire Responses

1) Open Caller – Project Title:

1. SPINE
2. BREATHE
3. HAIR
4. RAZOR
5. SMART SONIA
6. ATHEMS
7. ADDICTIVE

2) Did you use the technical documentation provided here:https://assist-iot-enablers-documentation.readthedocs.io/en/latest/ (If you didn't use it, the questions 4-6 can be skipped):

1. SPINE: YES
2. BREATHE: YES
3. HAIR: YES
4. RAZOR: NO
5. SMART SONIA: YES
6. ATHEMS: YES
7. ADDICTIVE: YES

3) In case you were not aware of the documentation, how did you manage to learn about the enablers?

- RAZOR: Through continuous interaction (e-mail exchanges and regular/on-demand telcos) with Pilot 3a Host

4) Did the documentation provide you with enough information to understand and utilize the enablers?

1. SPINE: 4/5
2. BREATHE: 5/5
3. HAIR: 4/5
4. RAZOR: N/A
5. SMART SONIA: 4/5
6. ATHEMS: 4/5
7. ADDICTIVE: 5/5

5) Do you have any suggestions to improve or update the documentation?

1. SPINE: Opinion at the beginning of the pilot -> Should be updated and maybe contains more detailed info. Final opinion -> we realized that the documentation was updated during the project with more images and new information - now it is more useful; each enabler's details should contain his development state, related documentation, documentation date, and documentation version.
2. SMART SONIA: It would be helpful to have the payload formatting in a column at the REST API endpoints section.

6) How user-friendly was the documentation?

1. SPINE: 4/5

2. BREATHE: 5/5
3. HAIR: 4/5
4. RAZOR: N/A
5. SMART SONIA: 5/5
6. ATHEMS: 4/5
7. ADDICTIVE: 5/5

7) Which enablers did you use?

1. SPINE: Video augmentation enabler
2. BREATHE: Edge Data Broker enabler, Long term data storage enabler
3. HAIR: Edge Data Broker enabler, Long term data storage enabler
4. RAZOR: Edge Data Broker enabler, Long term data storage enabler
5. SMART SONIA: Long term data storage enabler
6. ATHEMS: VPN enabler, Edge Data Broker enabler, Long term data storage enabler, Tactile dashboard enabler, Business KPI reporting enabler, Performance and Usage Diagnosis enabler, Self-healing device enabler, Monitoring and Notifying enabler, Automated Configuration enabler, Enabler for registration and status of enablers, Devices management enabler
7. ADDICTIVE: Edge Data Broker enabler, Long term data storage enabler

8) How good did you find the documentation in our deliverables when preparing your proposal?

1. SPINE: relevant information, used for the architecture definition
2. BREATHE: It was useful to know the general structure of the project and the way the different functionality were deployed.
3. HAIR: The documentation was easily accessible and complete at a very technical level that made the preparation of the proposal easier.
4. RAZOR: Highly detailed
5. SMART SONIA: All the documentation is well structured and can be easily understood.
6. ATHEMS: When preparing the proposal we found the available deliverables good enough to get an idea of how the overall ASSIST-IoT architecture looked like and what were how we were expected to interoperate with its components.
7. ADDICTIVE: Since I wasn't aware of the readthedocs page in the proposal phase, it was harder to find the information than it's currently.

9) How different did you feel about the technical scope of ASSIST-IoT enablers and architecture comparing before and after working within ASSIST-IoT?.

1. SPINE: technical scope seems to be ambitious; at the end of the project we are more confident that the scop is possible to execute
2. BREATHE: Now it is more clear how enablers work and the kind of service they can offer to reduce the amount of work necessary to cover certain tasks (instead of self-implementing them).
3. HAIR: The enablers were well documented, which enabled us to start the development before the enablers were fully ready. Onde the enablers got complete, we had very few issues achieving a full integration.
4. RAZOR: Reached the initial expectations
5. SMART SONIA: We gained a lot of helpful knowledge for the future regarding the cloud (HELM, Kubernetes) and the overall project architecture.

6. ATHEMS: Nothing to remark
7. ADDICTIVE: We liked the many examples provided in the documentation and there are lots of enablers that can be integrated.

10) How did you managed to fulfill any unexpected gap between the promised ASSIST-IoT technical scope in comparison with what you faced in the process.

1. SPINE: Regarding VAE: the initial state was not developed as we were expecting; to solve this problem we decide to improve the enabler structure
2. BREATHE: There were some enablers and functions not available at the beginning of the project execution, but later they were provided. It is understandable, given we were within the first open call.
3. HAIR: Technically, we were able to integrate as expected. Small issues within the implementation of the enablers were targeted very fast and solutions came promptly. Having frequent meetings with our point of contact in the pilot also made the integration process of HAIR much easier.
4. RAZOR: Strong interaction with and support from Pilot Host
5. SMART SONIA: We researched any possible solution and after discussion with the partners of the Assist-IoT, we agreed to choose the one that seemed more relevant to the project's goals.
6. ATHEMS: Not applicable
7. ADDICTIVE: We didn't find any unexpected gaps

11) Did you find the GitLab Repository useful?

1. SPINE: 2/5
2. BREATHE: 5/5
3. HAIR: 3/5
4. RAZOR: N/A
5. SMART SONIA: 4/5
6. ATHEMS: 3/5
7. ADDICTIVE: N/A

12) What was your experience using and managing GitLab artifacts?

1. SPINE: 3/5
2. BREATHE: 5/5
3. HAIR: 3/5
4. RAZOR: N/A
5. SMART SONIA: 3/5
6. ATHEMS: 3/5
7. ADDICTIVE: N/A

13) How was the accessibility process at GitLab repositories?

1. SPINE: 5/5
2. BREATHE: 5/5
3. HAIR: 3/5
4. RAZOR: N/A
5. SMART SONIA: 4/5
6. ATHEMS: 3/5

7. ADDICTIVE: N/A

14) Based on your experience using GitLab, what recommendations would you suggest?

1. SPINE: more frequent updates, we understand easily the state of the project in the project documentation than in the GitLab repository; this one seems to not have much activity
2. BREATHE: Nothing else to be said.
3. HAIR: In the scope of Hair we did not had the need of using Gitlab.
4. RAZOR: Replies 11-13 not applicable (didn't use GitLab repository), we worked directly with the Pilot artifacts
5. SMART SONIA: The documentation could be added there also as a link or as a README file
6. ATHEMS: Nothing to remark
7. ADDICTIVE: We didn't use the Gitlab repository.