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

D6.5. Technical Support Documentation - Initial

Deliverable No.	D6.5	**Due Date**	30-Apr-2022
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.		

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

This document contains material, which is the copyright of certain ASSIST-IoT consortium parties, and may not be reproduced or copied without permission. This deliverable contains original unpublished work except where clearly indicated otherwise. Acknowledgement of previously published material and of the work of others has been made through appropriate citation, quotation or both.

The information contained in this document is the proprietary confidential information of the ASSIST-IoT Consortium (including the Commission Services) and may not be disclosed except in accordance with the Consortium Agreement. The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the Project Consortium as a whole nor a certain party of the Consortium warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, and accepts no liability for loss or damage suffered by any person using this information.

The information in this document is subject to change without notice.

The content of this report reflects only the authors' view. The Directorate-General for Communications Networks, Content and Technology, Resources and Support, Administration and Finance (DG-CONNECT) is not responsible for any use that may be made of the information it contains.

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
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
Theoni Dounia	P06 INF	tdounia@infolysis.gr
Nikolaos Vrionis	P06 INF	nvrionis@infolysis.gr
Vaios Koumaras	P06 INF	vkoumaras@infolysis.gr
Aggeliki Papaioannou	P06 INF	apapaioannou@infolysis.gr
Alex van den Heuvel	P09 NEWAYS	alex.van.den.heuvel@newayselectronics.com
Ron Schram	P09 NEWAYS	Ron.schram@newayselectronics.com
Fotios Konstantinidis	P10 ICCS	fotios.konstantinidis@iccs.gr
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
27-Dec-2021	0.1	ToC creation: Section contributors assigned
10-Jan-2022	0.2	Section contributors refined; Enabler template added

31-Jan-2022	0.3	1st round of contributions completed
14-Feb-2022	0.4	2nd round of contributions completed
28-Feb-2022	0.5	3rd round of contributions completed
21-Mar-2022	0.6	4th and final round of contributions completed
27-Mar-2022	0.7	Final adjustments before the IR process
11-Apr-2022	0.9	IR completed and comments integrated
20-Apr-2022	0.99	PIC review completed and consolidated final version created
30-Apr-2022	1.0	Final version uploaded to the EC portal

Key Data

Keywords	Enablers, Technical Documentation, Support
Lead Editor	P06 INF –Theoni Dounia, P06 INF – Nikolaos Vrionis
Internal Reviewer(s)	Alex van den Heuvel, P09 NEWAYS Fotios Konstantinidis, P10 ICCS

Executive Summary

This deliverable belongs to the framework of WP6 – Testing, Integration, and Support – of the ASSIST-IoT Project under Grant Agreement No. 957258. Through this deliverable, the project presents the documentation material created under the scope of Task 6.4 – Technical and Support Documentation – that reflects mainly the documentation of WP4 and WP5 technical developments. In parallel, this document introduces the documentation strategy that will be followed throughout the task's duration, aiming to create compact and sufficient documentation of the technical outcomes of the project.

More specifically, the documentation focuses on providing instructions on how to deploy and use the ASSIST-IoT enablers of the horizontal planes and verticals of the ASSIST-IoT architecture. Since the ASSIST-IoT enablers are pieces of software that can be installed on any machine, specific hardware requirements apply too. These requirements are defined according to the ASSIST-IoT deployment, that can be composed of one or more tiers, comprised of one or more nodes, each running a k8s installation. In case that indoor localization is required, the tags and anchors designed and developed in the project can be leveraged. The Kubernetes master plane that handles 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.

Additionally, this deliverable presents the essential enablers that are a specific subset of all the designed enablers, strictly required to be present in any given deployment and are the following: 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.

D6.5 also elaborates on the development process that can be aided by an ASSIST-IoT administrator which, when properly configured, will deploy the platform and its associated tools in order to realise a predefined 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 the nodes (within the top and other tiers), 3) Installation of the rest essential enablers, and 4) Installation and configuration of use-case related enablers. Currently, a script has already been created for the first step while the rest of them are manually performed. These steps could be altered as the developments progress.

It also provides direction on how the potential user can login to the tactile dashboard of ASSIST-IoT through the web, in order to a) manage clusters in the ASSIST-IoT deployment, b) manage helm repositories for getting enablers compliant with the ASSIST-IoT architecture, and c) manage enablers within the infrastructure.

Following the general installation instructions, each enabler follows different documentation processes, which, for facilitation reasons, are presented using a dedicated ASSIST-IoT wiki repository hosted at the following link: https://assist-iot-enablers-documentation.readthedocs.io/en/latest/index.html. The wiki is organised with respect to 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 status until M18 and will be updated following the advancements of the enablers.

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
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)
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 a number of 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 appropriately documented in order for third-party stakeholders to be able to follow the developments and make use of the outcomes.

Towards this objective, this deliverable collects in a single, self-contained document, all the necessary information, both conceptual and technical, to effectively support the aforementioned third parties. The document summarizes the basic definitions and functionalities of the ASSIST-IoT enablers, the general configuration steps, technical interactions, and parameters to be configured when interacting with the ASSIST-IoT outputs, while licenses and version control are included. In general, this deliverable acts in the narrative of support documentation setting the foundation of a manual that is to be created throughout the project's duration depicting all the relative developments.

It should be highlighted that this deliverable corresponds to the 1st release of a series of two documents, reporting mainly the developments made until M18. The content will be expanded and adapted in the upcoming final version of the document (D6.6), reporting any updates and new developments until M30. It should also be mentioned that this deliverable consists of key overviews and components' information, while more detailed information regarding each enabler is provided with links to the ASSIST-IoT public online repository (https://assist-iot-enablers-documentation.readthedocs.io/en/latest/index.html) specifically created for the documentation purposes.

1.1Deliverable context

Keywords	**Lead Editor**
Objectives	As an extent to T6.4 responsibilities, 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 D6.5 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 first document of a series of two deliverables (D6.5 and D6.6). Being the first technical documentation report that provides fundamental support for the correct deployment of WP4 and WP5 enablers. While the documentation provided through this deliverable will be used as a tool for technical information, its main purpose is to help 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 an important role for WP7 activities as well.
Milestones	No Milestones are directly related to this deliverable.
Deliverables	D6.5 is directly linked to one of the upcoming WP6 deliverable, D6.6, since it is the first version of the Documentation series documents, setting the path for the final documentation release of the final technical outputs of the ASSIST-IoT Project. D6.5 is also linked to three other deliverables, that have already been submitted (D4.1, D5.1, and D5.2). These three documents serve as the main source of input for this deliverable, since the main technical output of the ASSIST-IoT Project comes 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 D6.5 is organized into 3 main sections, the first of which (Section 1) is an introductory section containing all the administrative information related to the document.

Section 2 is devoted to providing all the necessary documentation information. In specific, the section is divided in six subsections from which the first four (Sections 2.1, 2.2, 2.3, 2.4) will provide general instructions and guidelines that apply to any 3rd party that aims at deploying enablers, while the last two (Sections 2.5, 2.6) provide a general description of all the enablers developed within the ASSIST-IoT Project along with links to online wikis specifically created for the more detailed technical documentation of each enabler.

The document concludes with Section 3 by referring to the future activities towards the next release of the deliverable (D6.6 – April 2023).

1.3 Version-specific notes

Being M18, and as an extend D6.5, the first reporting of the technical developments of the ASSIST-IoT Project, any additional developments and updates will be included and documented in the next deliverable of the series, D6.6. Since not all the documented enablers have reached the same maturity level and/or are not in the final stage of the development, it should be expected that in the next iteration the potential differences between the current status and the future one along with any new additions will be reported.

2 Documentation

The documentation approach that will be followed in the ASSIST-IoT Project, is being introduced by this deliverable and is elaborated in this section. 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, dedicated wikis have been created for documenting the enablers and are reported in this section.

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**.

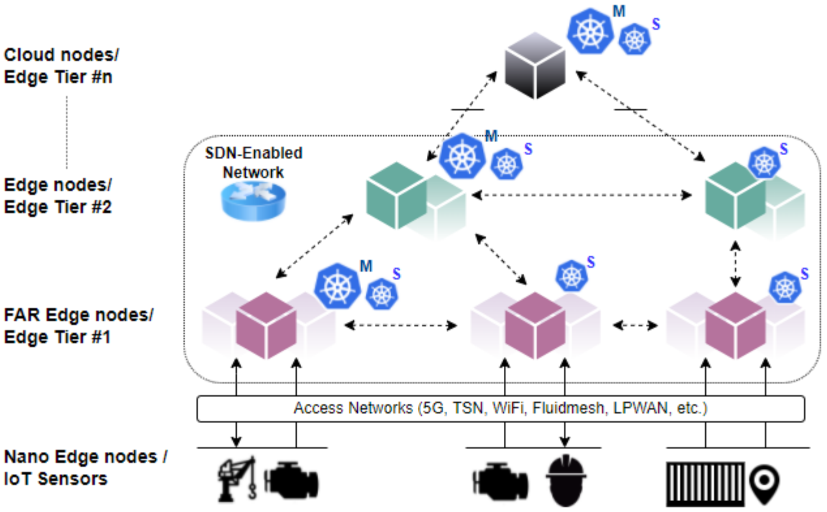


Figure 1. Deployment view of ASSIST-IoT

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% 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.

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 to retrieve and use the gathered information properly. Regarding **tags**:

- The battery of the tags must be charged during non-working hours.
- The monitored assets need to have a tag attached (e.g., in case of a worker, they must 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.
- During the installation of the anchor, the position of the anchor must be documented. This is needed to allow 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.

[1] https://microk8s.io/

[2] https://k3s.io/

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

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 operating systems 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) 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

In this subsection are presented the commands required for installing and configuring a k3s cluster, considering the installation of worker nodes as well, add-ons and a high availability strategy. Dedicated scripts for automating the actions are still under development and will be reported in detail in future deliverables.

Installation

For installing k3s in Raspbian or Raspberry Pi OS, some actions must be performed first:

- Enable *cgroups* by appending *cgroup_memory=1 cgroup_enable=memory* to the *file /boot/cmdline.txt*
- Configure the usage of *iptables* instead of *nftables*:
  - In Raspbian Buster:

    ```
    sudo iptables -F
    sudo update-alternatives --set iptables /usr/sbin/iptables-legacy
    sudo update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy
    sudo reboot
    ```

  - In the newest versions of the Raspberry Pi OS:

    ```
    sudo apt install iptables
    ```

K3s can be installed using one single command:

```
curl -sfL https://get.k3s.io | sh -
```

If the cluster must be accessible through a public IP and SSL is used, use this command:

```
curl -sfL https://get.k3s.io | INSTALL_K3S_EXEC="server --tls-san <public_ip_address>" sh -
```

Add-ons and other tools

In order to be managed by the smart orchestrator and facilitating, clusters require having CoreDNS and Cilium CNI installed. Guidelines for installing them will be provided in the wikis and in the next version of the deliverable.

Adding worker nodes

[4] https://k0sproject.io/
[5] https://rancher.com/docs/rke/latest/en/
[6] https://www.redhat.com/es/technologies/cloud-computing/openshift

A k3s worker node can be created and added to the cluster by running a single command in the new node machine, after following the previous steps:

```
curl -sfL https://get.k3s.io | K3S_URL=https://<master_ip>:6443 K3S_TOKEN=mynodetoken sh -
```

The k3s token is stored at */var/lib/rancher/k3s/server/node-token* on the master node machine.

An important requirement is that each machine in the cluster must have a unique hostname (**cat /etc/hostname**). If a hostname is repeated, add the *K3S_NODE_NAME* to the installation command with a unique value.

High availability with embedded DB

K3s needs a cluster with an odd number of server (master) nodes to achieve high availability. First, create a server node as described in the installation section, but add a custom token that will be shared across the server nodes of the cluster and the flag *--cluster-init*:

```
curl -sfL https://get.k3s.io | K3S_TOKEN=SECRET k3s server --cluster-init sh -
```

Once the first server node is launched, create and join the additional server nodes to the cluster:

```
curl -sfL https://get.k3s.io | K3S_TOKEN=SECRET k3s server --server https://<master_ip>:6443 sh –
```

2.1.2.2 MicroK8s installation with required add-ons

In this subsection are presented the commands required for installing and configuring a MicroK8s cluster, considering the installation of worker nodes as well, add-ons and a high availability strategy. Dedicated scripts for automating the actions are still under development and will be reported in detail in future deliverables.

Installation

MicroK8s requires an operating system which supports *snapd*. It is recommended to use Ubuntu 20.04 LTS or 18.04 LTS because both have *snapd* pre-installed and MicroK8s is developed by the same maintainer company.

MicroK8s can be installed using one single command:

```
snap install microk8s --classic
```

The *channel* (e.g., like the version) can be specified adding the *flag --channel* to the installation command (e.g., **--channel=1.21/stable).** For listing the available channels, run the **snap info microk8s** command.

Once the installation is completed, the current user must be added to the microk8s user group.

```
sudo usermod -a -G microk8s $USER
sudo chown -f -R $USER ~/.kube
```

Then, re-enter the session for completing the installation process.

```
su - $USER
```

In MicroK8s, additional features like *core-dns*, *storage* or *helm*, can be added to Kubernetes using its built-in add-ons. First, run the MicroK8s status command for listing the enabled and the available (disabled) add-ons for the current MicroK8s distribution. An add-on can be installed by running the command:

```
microk8s enable <add-on name>
```

The uninstallation command is similar:

```
microk8s disable <add-on name>
```

Add-ons and other tools

In order to be managed by the smart orchestrator and facilitating, clusters require having CoreDNS and Cilium CNI installed. In the wikis and in the next version of the deliverable, there will be instructions on how to put them in place.

Adding worker nodes

Adding worker nodes is only possible from the 1.23 release. Before this release, it is only possible to add master nodes to the cluster. In the master node, run the following command:

```
microk8s add-node
```

The execution of this command will return some instructions for joining to this cluster using the join command (e.g., **microk8s join 192.168.1.230:25000/92b2db237428470dc4fcfc4ebbd9dc81/2c0cb3284b05 --worker**). Finally, run the appropriate join command with the **--worker** flag in the machine where is installed the MicroK8s instance that will be added as a worker node to the cluster.

High availability

From the 1.19 release of MicroK8s, the high availability feature (*ha-cluster* add-on) is enabled by default. MicroK8s needs a cluster of at least three worker nodes for achieving high availability.

First, create two MicroK8s instances and add them as nodes to the master node following the steps described in the last point, but without including the --worker flag in the commands, as the added nodes must be master nodes instead of worker nodes.

Once the three master nodes have been added to the cluster, the final step is to make MicroK8s failure domain aware. For achieving it, associate an integer to each failure domain and update *the /var/snap/microk8s/current/args/ha-conf* with it. Then, restart MicroK8s.

```
echo "failure-domain=42" > /var/snap/microk8s/current/args/ha-conf

microk8s.stop

microk8s.start
```

An important consideration is that the add-ons only work in on the node the add-on was enabled from.

2.1.2.3 kubeadm installation and required add-ons

In this subsection are presented the commands required for installing and configuring a kubeadm cluster, considering the installation of worker nodes as well, add-ons and a high availability strategy. Dedicated scripts for automating the actions are still under development and will be reported in detail in future deliverables.

The K8s installation with kubeadm can be reached if the machine has at least 2 CPU. It is divided in four main steps:

1. Initial server preparation
2. Installation of the needed tools
3. Cluster creation
4. Adding nodes

Initial server preparation

Working as a root user, the swap has to be disabled by executing the command:

```
swapoff -a
```

Followed by the modification of the */etc/fstab,* commenting the line where the word **swapfile** appears. Secondly, the last file to be modified is the */etc/sysctl.conf* by adding the lines:

```
net/bridge/bridge-nf-call-ip6tables = 1

net/bridge/bridge-nf-call-iptables = 1

net/bridge/bridge-nf-call-arptables = 1
```

Finally, some tools must be downloaded:

```
apt-get install ebtables ethtool
```

Tools installation

The first step is to update the system and then to install Docker:

```
apt-get update
apt-get install -y docker.io
```

Then, the installation can be checked by executing:

```
systemctl docker status
docker version
```

Secondly, the HTTPS support package is installed with:

```
apt-get update
apt-get install -y apt-transport-https
```

Thirdly, install curl if you don't have it already installed:

```
apt-get install curl
```

Finally, to get the Kubernetes repository key and add the repository to the system:

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add
cat <<EOF>/etc/apt/sources.list.d/kubernetes.list
deb http://apt.kubernetes.io/ kubernetes-xenial main
EOF
```

The three components needed are: kubeadm, kubelet and kubectl:

```
apt-get update apt-get install -y kubelet kubeadm kubectl
```

Cluster creation

Regarding the cluster creation, the command is the following one:

```
kubeadm init
```

Finally, execute the next command to load the configuration:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

To check the cluster is operating correctly:

```
kubectl get pods --all-namespaces
```

Add-ons and other tools

In order to be managed by the smart orchestrator and facilitating, clusters require having CoreDNS and Cilium CNI installed. Guidelines for installing them will be provided in the wikis and in the next version of the deliverable.

As a final requirement, to be able to add the cluster to the smart orchestrator, OpenEBS is needed:

```
kubectl apply -f https://openebs.github.io/charts/openebs-operator.yaml
```

Addition of nodes

After following the steps in the tools installation section, the nodes can be added to our cluster:

```
kubeadm join --token {token} {IP-master-node}:port
```

The token is displayed once que cluster is created or by typing the command on the master node of our cluster:

```
kubeadm token list
```

Check the nodes are added to the cluster:

```
kubectl get nodes
```

High availability

The instructions for achieving high availability with kubeadm can be found in the following link.

2.2 Steps on deploying ASSIST-IoT

2.2.1 Essential Enablers

ASSIST-IoT architecture is realised 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[7] and/or Ansible[8] 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

[7] https://www.terraform.io/
[8] https://www.ansible.com/

provided in Section 2.1.2.3. In the future, dedicated tools like the aforementioned ones will be considered for facilitate the process.

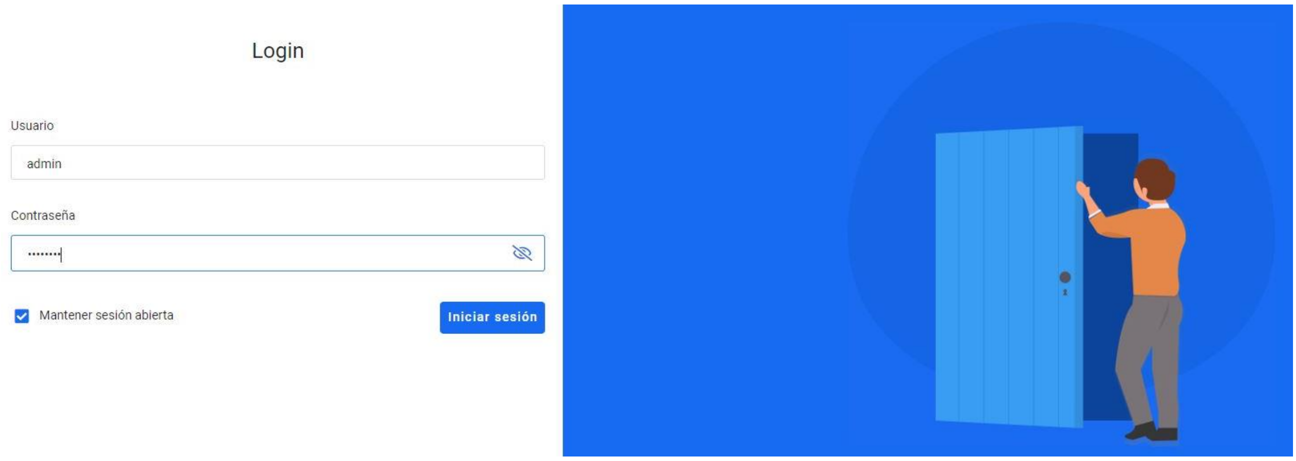
Apart from the top-tier node, or nodes, the rest of tiers should be provisioned to fulfil the **second step**. Currently, this should be done **manually**, installing the necessary kubernetes distributions and grouping them into clusters as defined per topological decisions (see Sections 2.1.1 and 2.1.2). 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.

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 (by default, port 8080, see Figure 2) and log in using the default administration credentials (user: *admin*, password: *admin*, which can be changed within the application).



Login

Usuario

admin

Contraseña

••••••••

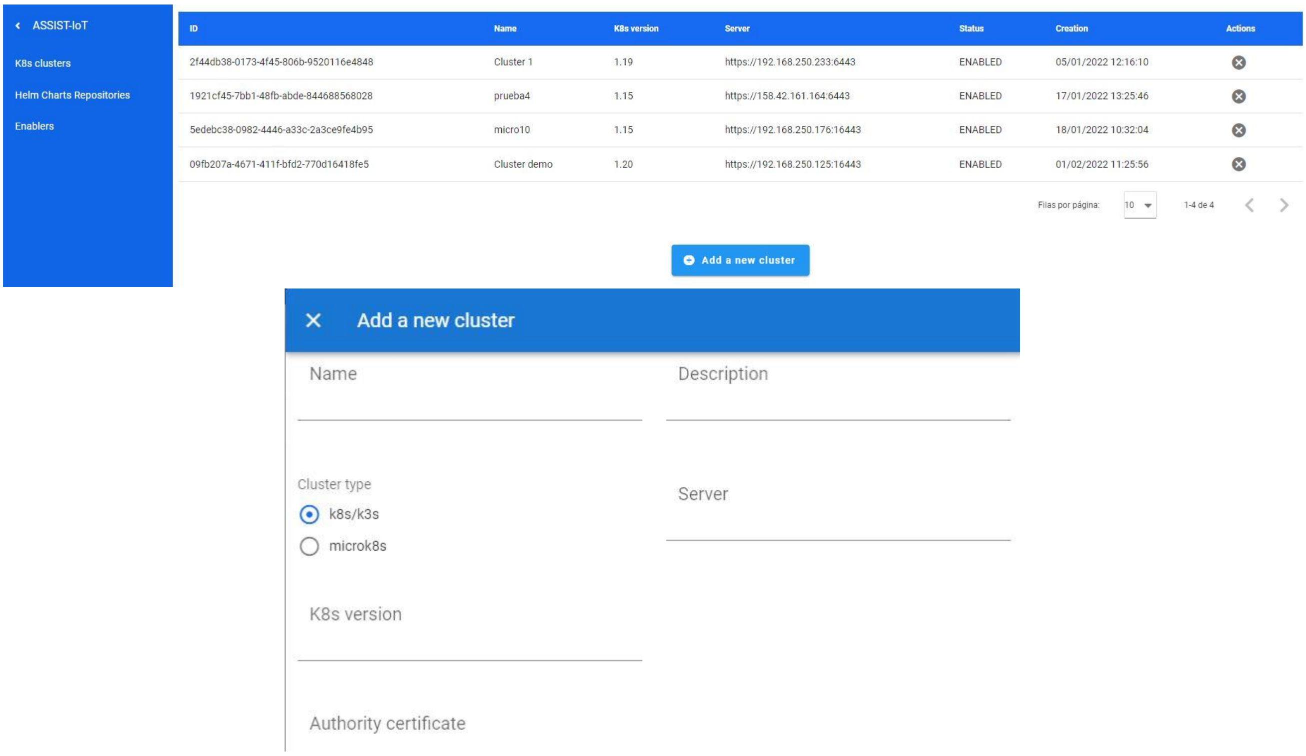
☑ Mantener sesión abierta

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

In 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.



ID	Name	K8s version	Server	Status	Creation	Actions
2f44db38-0173-4f45-806b-9520116e4848	Cluster 1	1.19	https://192.168.250.233:6443	ENABLED	05/01/2022 12:16:10	☒
1921cf45-7bb1-48fb-abde-844688568028	prueba4	1.15	https://158.42.161.164:6443	ENABLED	17/01/2022 13:25:46	☒
5edebc38-0982-4446-a33c-2a3ce9fe4b95	micro10	1.15	https://192.168.250.176:16443	ENABLED	18/01/2022 10:32:04	☒
09fb207a-4671-411f-bfd2-770d16418fe5	Cluster demo	1.20	https://192.168.250.125:16443	ENABLED	01/02/2022 11:25:56	☒

Add a new cluster

Name

Description

Cluster type
● k8s/k3s
○ microk8s

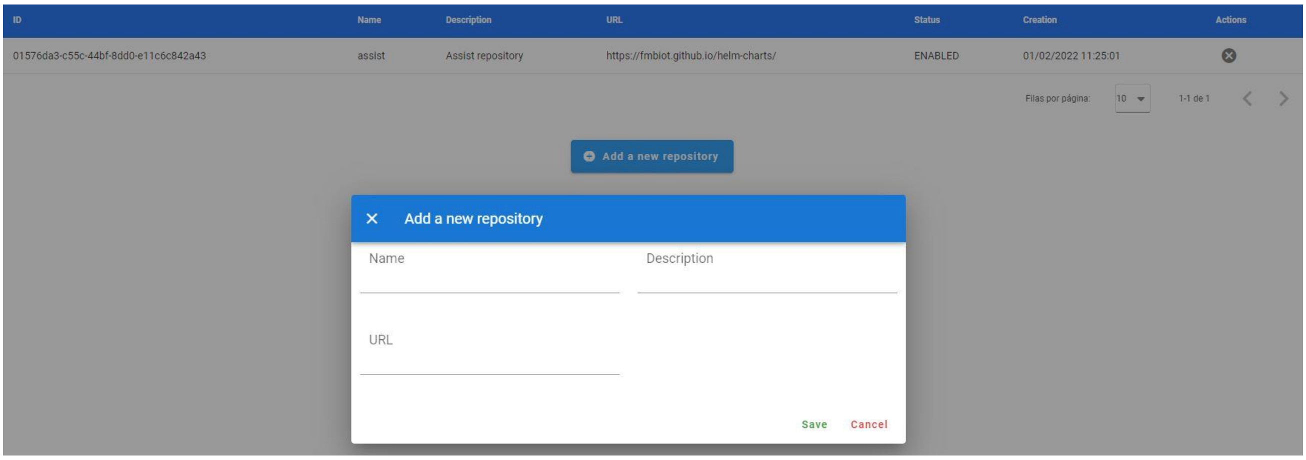
Server

K8s version

Authority certificate

Figure 3. Cluster management interface

Before launching an enabler, Helm 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.html where Charts are published 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 4.



ID	Name	Description	URL	Status	Creation	Actions
01576da3-c55c-44bf-8dd0-e11c6c842a43	assist	Assist repository	https://fmbiot.github.io/helm-charts/	ENABLED	01/02/2022 11:25:01	☒

Add a new repository

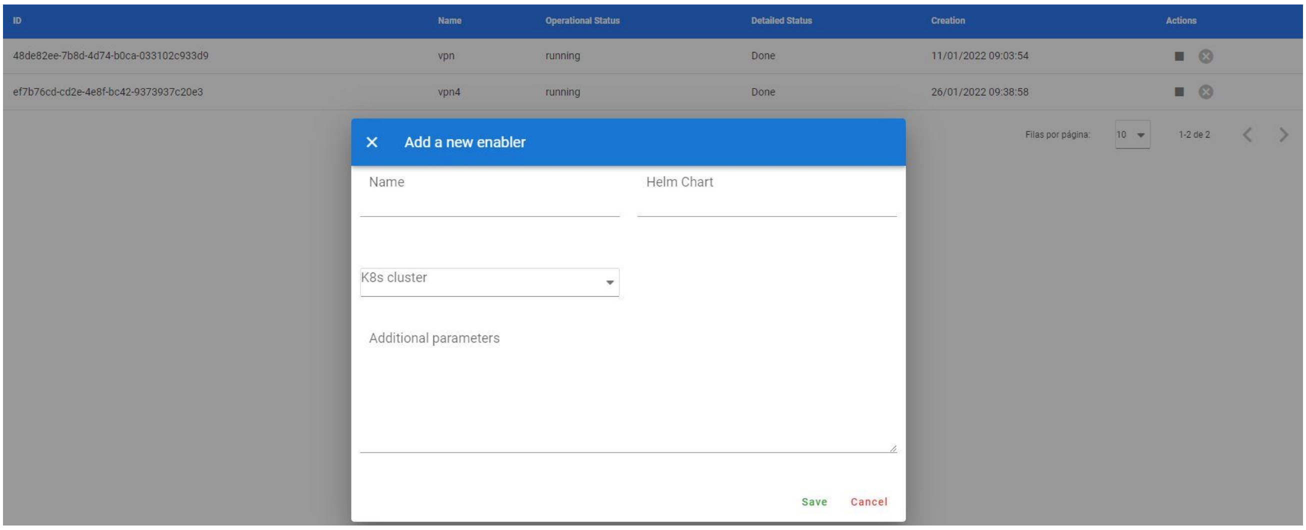
Name

Description

URL

Figure 4. Helm repositories management interface

Lastly, the last management interface that has been implemented is related to enablers (see Figure 5). With it, the user can instruct the system to deploy or terminate an enabler. In the current release, the user has to indicate manually the enabler as well as the cluster in which its components will be deployed, as the intelligence for selecting it among the available ones is not yet in place. Besides, when registering an enabler, the user can pass an 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. In future releases, key values will be configured by means of dedicated form, in order to facilitate user interaction, among other functionalities.



ID	Name	Operational Status	Detailed Status	Creation	Actions
48de82ee-7b8d-4d74-b0ca-033102c933d9	vpn	running	Done	11/01/2022 09:03:54	■ ⊗
ef7b76cd-cd2e-4e8f-bc42-9373937c20e3	vpn4	running	Done	26/01/2022 09:38:58	■ ⊗

Filas por página: 10 ▾ 1-2 de 2

× Add a new enabler

Name

Helm Chart

K8s cluster ▾

Additional parameters

Save Cancel

Figure 5. Enablers management interface

Manageability enablers and their interfaces are expected to be upgraded in the next releases, once the smartness of the orchestrator enabler, additional key k8s add-ons for supporting multi-clustering features, as well as the logging and the monitoring stacks are in place. They will include the enabler in charge of facilitating the realisation of data pipelines (easing the interaction of enablers, graphically), which is under development. These new enhancements and additional features will be indicated in the next iteration of the present deliverable, i.e., D6.6, apart from in their dedicated wikis.

2.4 Enablers Technical Documentation

Enablers provide functionalities that belong to different planes and verticals and hence, development and internals can be technologically very different among them. In addition, the development and readiness status differ among the enablers. To support that multidimensional approach, dedicated wikis, for documenting each enabler, have been created and will continue to be updated as the developments evolve.

It is important to stress that the Read the Docs Platform, chosen to host the documentation of ASSIST-IoT, supports continuous and dynamic integration and thus it will be the main public repository that will guide the documentation plan and reflect the progress of the technical developments. More specifically, it is important to highlight that the wikis presented in this deliverable reflect the enablers' maturity level and developments performed until M18. Since the project is ongoing and additional developments continue to take place in almost all enablers, an updated version of the wikis, depicting the final status of all ASSIST-IoT enablers, will be provided in D6.6 (M30).

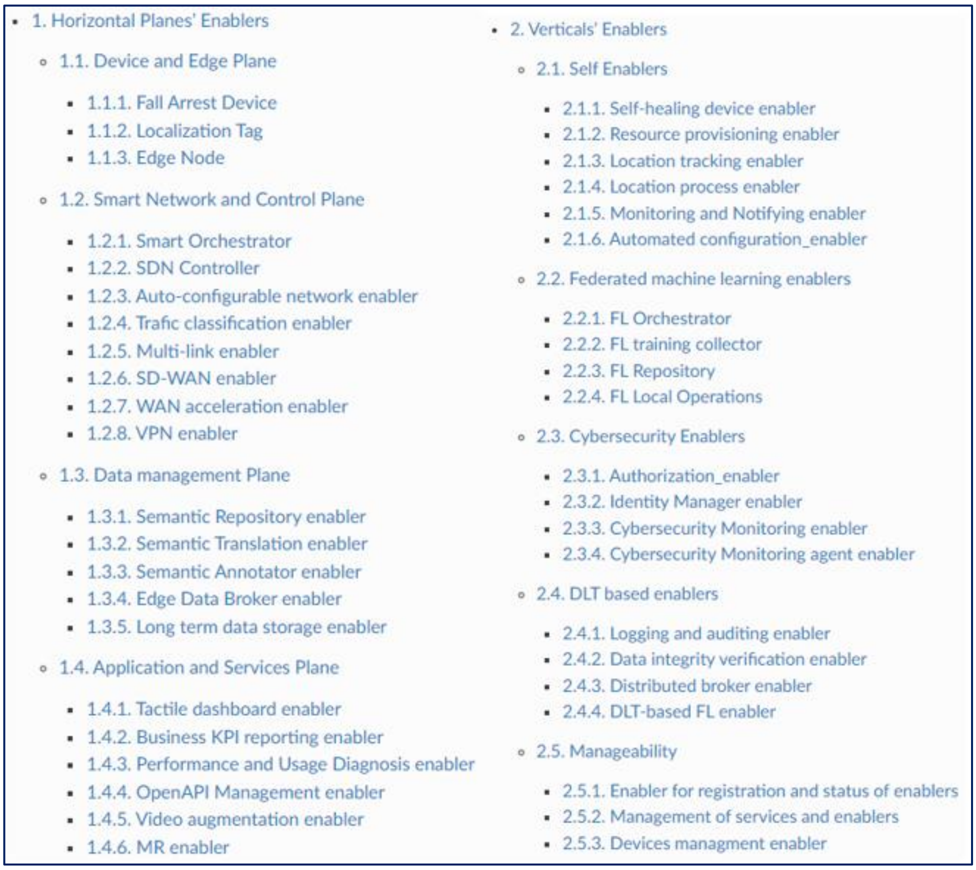
2.4.1 Structure of the Wiki

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 was used to create a public repository that will host the documentation. Among the available options for hosting a technical documentation wiki, DevDocs[9], Wiki.js[10], Gitbook[11], and MkDocs[12] 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 also 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 6) 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 6. ASSIST-IoT Wiki Documentation structure

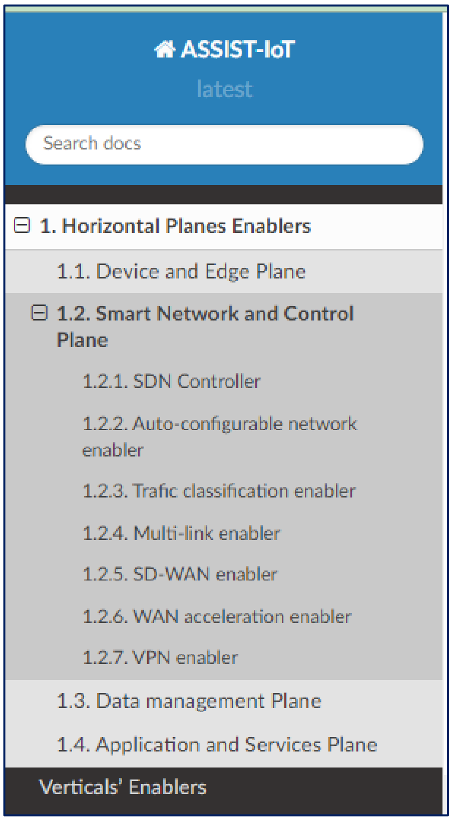
The navigation between the different wiki pages is achieved by using the side bar that provides both a search and manual selection options. The navigation bar is presented in Figure 7.

[9] https://devdocs.io/

[10] https://js.wiki/

[11] https://www.gitbook.com/

[12] https://www.mkdocs.org/



ASSIST-IoT

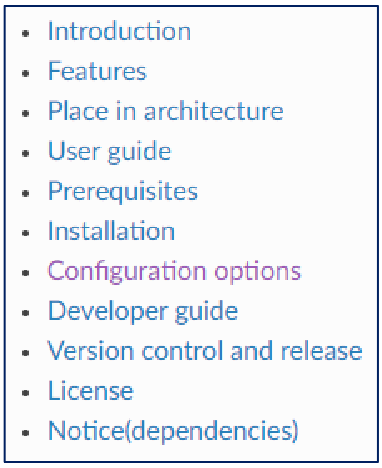
latest

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 7. 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 8.



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

Figure 8. 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 aforementioned 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

According to the level of maturity of each enabler, the aforementioned sections may differ in terms of content. As the developments progress the content of the sections may be altered, and potential incomplete sections will be filled. As mentioned before, any new additions or changes will be reflected in the new version of the wiki, released with D6.6.

2.4.2 Documentation Release Plan

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. Since most enablers are still under development and will continue to be integrated as the project evolves, an indicative table with future releases has been created for the purpose of this deliverable. The tentative next and final release dates act only as indications and could change according to the project's needs and advancement. The purpose of Table 2 is to inform the potential third user about the release plan. For more information regarding the exact status of each enabler's current version, please refer to the respective wiki, for which links are provided in Sections 2.5 and 2.6.

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

Code	Enabler Name	Tentative Next Release	Tentative Final Release
T41E1	Localization Tag	September 2022	November 2022
T41E2	Fall Arrest Device	September 2022	November 2022
T41E3	GWEN	September 2022	February 2023
T42E1	Smart Orchestrator	April 2022	December 2022
T42E2	SDN Controller	March 2022	September 2022
T42E3	Auto-configurable Network	June 2022	December 2022
T42E4	Traffic Classification	July 2022	December 2022
T42E5	Multi-link	October 2022	February 2023
T42E6	SD-WAN	October 2022	February 2023
T42E7	WAN Acceleration	December 2022	March 2023
T42E8	VPN	July 2022	
T43E1	Semantic Repository	February 2022	December 2022
T43E2	Semantic Translation	February 2022	September 2022
T43E3	Semantic Annotation	February 2022	June 2022
T43E4	Edge Data Broker	March 2022	January 2023
T43E5	Long-term Data Storage	April-2022	April 2023
T44E1	Tactile Dashboard	April 2022	December 2022
T44E2	Business KPI Reporting	April 2022	December 2022
T44E3	Performance and Usage Diagnosis (PUD)	March 2022	January 2023
T44E4	OpenAPI Management	March 2022	December 20211
T44E5	Video Augmentation	April 2022	April 2023
T44E6	MR	March 2022	January 2023
SELF11	Self-healing Device	March 2022	October 2022
SELF12	Resource Provisioning	April 2022	October 2022
SELF13	Location Tracking	August 2022	December 2022
SELF16	Location Processing	April 2022	December 2022
SELF14	Monitoring and Notifying	April 2022	December 2022
SELF15	Automated Configuration	April 2022	April 2023
T52E1	FL Orchestrator	April-2022	April 2023
T52E2	FL Training Collector	April 2022	December 2022
T52E3	FL Repository	April 2022	December 2022
T52E4	FL Local Operations	April 2022	December 2022
T53E1	Authorisation	April 2022	April 2023
T53E2	Identity Manager	April 2022	April 2023
T53E3	Cybersecurity Monitoring	April2022	April 2023
T53E4	Cybersecurity Monitoring Agent	April 2022	April 2023
T54E1	Logging and Auditing	April 2022	December 2022
T54E2	Data Integrity Verification	October 2022	December 2022
T54E3	Distributed Broker	October 2022	December 2022
T54E4	DLT-based FL	October 2022	December 2022
T55E1	Management of enablers existence in a deployment	April 2022	December 2022
T55E2	Management of services and enablers workflow	July 2022	February 2022
T55E3	Management of devices in an ASSIST-IoT deployment	April 2022	December 2022

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*	
Status: *Initiated*	
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 they are 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	

Enabler: *Fall Arrest Device*	**Id:** *T41E2*
Owner and Support: *NEWAYS, Pilot Stakeholders (MOSTOSTAL, FORD)*	
Related Deliverable/s: *D4.1*	
Status: *Initiated*	
Enabler Description	
The ASSIST-IoT fall arrest device is a localisation tag with an Inertial Measurement Unit (IMU) and a push button. The push button is used by the person wearing the tag, to indicate that this person is in an emergency situation and needs immediate help. The fall arrest device uses the IMU to determine if the person has fallen or not. If an emergency situation is detected, a message is transmitted to the anchors immediately. An enabler can pick up this message and act accordingly.	
Keywords / Key components	
IMU, UWB, low power, indoor localisation	

Link to wiki
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/horizontal_planes/device/fall_arrest.html

2.5.1.2 GWEN

Enabler: GWEN	**Id:** *T41E3*
Owner and Support: *NEWAYS, Pilot Stakeholders (MOSTOSTAL, FORD)*	
Related Deliverable/s: *D3.5, D4.1*	
Status: *Under development*	
Enabler Description	
The GateWay/EdgeNode (GWEN) is a device used as interface between sensors & actuators on one side and a communication network on the other side. Sensors and actuators can be connected through wired and wireless interfaces. The interface with a network can also be wired or wireless. *Available wired interfaces are: Ethernet, RS232/485, CAN & CAN FD, USB2 and USB3* *Available wireless interfaces are: WiFi, Bluetooth and 3G/4G/5G. In addition an UWB interfaces is available for localisation purposes.* *The GWEN also contains compute power to be able to operate AI algorithm, while Docker is used as container runtime on top of Linux as OS.*	
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	

2.5.2 Smart Network and Control Plane

2.5.2.1 Smart Orchestrator

Enabler: *Smart Orchestrator*	**Id:** *T42E1*
Owner and Support: *UPV, NEWAYS, OPL*	
Related Deliverable/s: *D4.1, D4.2*	
Status: *Under development*	
Enabler Description	

This enabler facilitates the interaction of user interfaces and other enablers with the main components of the MANO framework, namely the Network Function Virtualisation Orchestrator (NFVO) and the Kubernetes clusters, exposing only the required inherent functionalities. In particular, this enabler will control the whole lifecycle of Containerised Functions, network and not-network related, from their instantiation to their termination, allowing their deployment in any k8s cluster available.
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

2.5.2.2 SDN Controller

Enabler: *SDN controller*	**Id:** *T42E2*
Owner and Support: *OPL, UPV*	
Related Deliverable/s: *D4.1*	
Status: *Under development*	
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	

2.5.2.3 Auto-configurable network enabler

Enabler: *Auto-configurable network*	**Id:** *T42E3*
Owner and Support: *OPL, UPV*	
Related Deliverable/s: *D4.1*	

Status: *Initiated*
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

2.5.2.4 Traffic classification enabler

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

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: *Initiated*
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

2.5.2.6 SD-WAN enabler

Enabler: *SD-WAN*	**Id:** *T42E6*
Owner and Support: *UPV, OPL*	
Related Deliverable/s: *D4.1, D4.2*	
Status: *Initiated*	
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	

2.5.2.7 WAN Acceleration enabler

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

2.5.2.8 VPN enabler

Enabler: *VPN*	**Id:** *T42E8*
Owner and Support: *UPV*	
Related Deliverable/s: *D4.1, D4.2*	
Status: *Under development*	
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	

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*	
Status: *Under development*	
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	

2.5.3.2 Semantic Translation enabler

Enabler: *Semantic Translation*	**Id:** *T43E2*
Owner and Support: *SRIPAS, UPV*	
Related Deliverable/s: *D4.1*	
Status: *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/semantic_translation_enabler.html

2.5.3.3 Semantic Annotation enabler

Enabler: *Semantic annotation*	**Id:** *T43E3*
Owner and Support: *SRIPAS, PRO, CERTH*	
Related Deliverable/s: *D4.1*	
Status: *Under development*	
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/semantic_annotator_enabler.html	

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*	
Status: *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.htm*l*

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: *Initiated*	
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	

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: *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

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: *Under development*	
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 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	

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*	
Status: *Testing Phase*	
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

2.5.4.4 OpenAPI Management enabler

Enabler: *OpenAPI Management*	**Id:** *T44E4*
Owner and Support: *UPV, SRIPAS, PRO*	
Related Deliverable/s: *D4.1*	
Status: *Under development*	
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	

2.5.4.5 Video Augmentation enabler

Enabler: *Video Augmentation*	**Id:** *T44E5*
Owner and Support: *PRO*	
Related Deliverable/s: *D4.1, D4.2*	
Status: *Under development*	

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

2.5.4.6 MR enabler

Enabler: *MR*	**Id:** *T44E6*
Owner and Support: *ICCS*	
Related Deliverable/s: *D4.1*	
Status: *Testing Phase*	
Enabler Description	
The MR enabler receives data and transforms it in a format suitable for visualisation through head-mounted MR devices. Data, which may come from long-term storage or real-time data streams, are requested according to its relevance to the user. Information is displayed to the user, according to their authorisation/access rights, via an MR device. The enabler supports user interaction with the virtual content and view customisation.	
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	

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*	
Status: *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	

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*	
Status: *Under development*	
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

2.6.1.3 Location tracking enabler

Enabler: *Location Tracking*	**Id:** *SELF13*
Owner and Support: *NEWAYS*	
Related Deliverable/s: *D5.1*	
Status: *Under development*	
Enabler Description	
This enabler communicates the position of the tags relative to a fixed point. X, Y and Z dimensions allow to discover the 3D position. This enabler also controls indicators at the tag to alert the person wearing the tag.	
Keywords / Key components	
Location, indicators	
Link to wiki	
https://assist-iot-enablers-documentation.readthedocs.io/en/latest/verticals/self/location_tracking_enabler.html	

2.6.1.4 Location processing enabler

Enabler: *Location Processing*	**Id:** *SELF16*
Owner and Support: *SRIPAS*	
Related Deliverable/s: *D5.1*	
Status: *Under development*	
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

2.6.1.5 Monitoring and Notifying enabler

Enabler: *Monitoring and Notifying*	**Id:** *SELF14*
Owner and Support: *SRIPAS, CERTH*	
Related Deliverable/s: *D5.1, D5.2, D5.3*	
Status: *Under development*	
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	

2.6.1.6 Automated configuration enabler

Enabler: *Automated configuration*	**Id:** *SELF15*
Owner and Support: *SRIPAS*	
Related Deliverable/s: *D5.1, D5.2, D5.3*	
Status: *Under development*	
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

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*	
Status: *Under development*	
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	

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*	
Status: *Under development*	
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

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*	
Status: *Under development*	
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	

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*	
Status: *Under development*	
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

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*	
Status: *Under development*	
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	

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*	
Status: *Under development*	

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

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*	
Status: *Under development*	
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	

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*	

Status: *Under development*
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

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*	
Status: *Under development*	
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. If any requirement of filtering prior to logging, a filtering module will be considered to be deployed. The DLT API is the candidate component for performing any filtering.	
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	

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*	
Status: *Under development*	
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	

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*	
Status: *Under development*	
Enabler Description	
This enabler will provide a mechanism that will facilitate data sharing between different heterogeneous IoT devices belonging to various edge domains and/or between different enablers of the architecture. In coordination with other enablers that will ensure trust between data sources (i.e., Identity and Authorisation providers), it will deal with data source metadata management and provide trustable, findable, and retrievable metadata for the data sources.	
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

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*	
Status: *Under development*	
Enabler Description	
This enabler will foster the use of DLT-related components to exchange the local, on-device models (or model gradients) in a decentralised way. The DLT can act as a component to manage AI contextual information and prevent any alteration to the data. The alteration of data is a threat to the Federated Learning approach and the DLT can help in mitigating the threat. Moreover, the enabler will allow mitigating single-point of failures. Finally, the enabler can be charged with validating the individually trained models to rule out malicious updates that can harm the global model.	
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	

2.6.5 Manageability

2.6.5.1 Enabler for registration and status of enablers

Enabler: *Management of enablers existence in a deployment*	**Id:** *T55E1*
Owner and Support: *UPV, SRIPAS, CERTH*	
Related Deliverable/s: *D5.2, D5.3*	
Status: *Under development*	
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 log collection capabilities is in place), (d) facilitate the deployment of standalone enablers (mostly for those that must 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

2.6.5.2 Enabler for management of services and enablers' workflow

Enabler: *Management of services and enablers workflow*	**Id:** *T55E2*
Owner and Support: *UPV, PRO, SRIPAS*	
Related Deliverable/s: *D5.2, D5.3*	
Status: *Initiated*	
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 predefined 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	

2.6.5.3 Devices management enabler

Enabler: *Management of devices in an ASSIST-IoT deployment*	**Id:** *T55E3*
Owner and Support: *UPV, NEWAYS*	
Related Deliverable/s: *D5.2, D5.3*	
Status: *Under development*	
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	

3 Future Work

This deliverable presents the first documentation release of the technical developments reached so far in the ASSIST-IoT Project. Since the main technical output of the project is created under WP4 and WP5 with the development and advancement of the enablers, D6.5 focuses on documenting the work done in these WPs, forming it in a way that reports the outputs and gives guidelines on how to use them. Hence, the document's focus is on the enablers and the general guidelines on the use of the infrastructure built around them.

To that direction, the document itself, acts as a general guide on how to complete all the necessary steps before any enabler can be installed, configured, and used. On the other hand, dedicated wikis with instructions on the installation and use of each enabler have also been created and reported in this document.

As the project advances the following actions are expected to be performed within T6.4 and finally reported in the next iteration of the deliverable (D6.6):

- The finalization of the enablers will bring to the surface all the necessary configuration steps required for the proper deployment of the ASSIST-IoT environment, leading to the final and complete documentation. T6.4 will closely monitor the advancements of WP4 and WP5 for the proper display of information in the following deliverable.
- The ASSIST-IoT wiki will continue to be updated, as the project advances, reflecting the enablers' progress until it finally forms a complete guide on the final version of the ASSIST-IoT enablers.

Apart from the already identified enablers, any new additions, driven either from the Pilots or the Open Calls that are to be performed, will be reflected in this task and its output. In parallel, the T6.4, through its deliverables, will continue developing and releasing standalone supporting ASSIST-IoT documentation, not only for the consortium members but also for 3rd parties participating in the Open Calls, interested Stakeholders and Open-Source Communities.