

DRAFT

D4.3

Enhancing IoT Tactile & Contextual Sensing/Actuating

WORKPACKAGE WP4

DOCUMENT D4.3

REVISION V1.0

DELIVERY DATE 31/07/2022

PROGRAMME IDENTIFIER H2020-ICT-2020-1

GRANT AGREEMENT ID 957246

START DATE OF THE PROJECT 01/10/2020

DURATION 3 YEARS

© Copyright by the IoT-NGIN Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 957246



DISCLAIMER

This document does not represent the opinion of the European Commission, and the European Commission is not responsible for any use that might be made of its content.

This document may contain material, which is the copyright of certain IoT-NGIN consortium parties, and may not be reproduced or copied without permission. All IoT-NGIN consortium parties have agreed to full publication of this document. The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the IoT-NGIN consortium as a whole, nor a certain party of the IoT-NGIN consortium warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, and does not accept any liability for loss or damage suffered using this information.

ACKNOWLEDGEMENT

This document is a deliverable of IoT-NGIN project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 957246.

The opinions expressed in this document reflect only the author's view and in no way reflect the European Commission's opinions. The European Commission is not responsible for any use that may be made of the information it contains.

PROJECT ACRONYM	IoT-NGIN
PROJECT TITLE	Next Generation IoT as part of Next Generation Internet
CALL ID	H2020-ICT-2020-1
CALL NAME	Information and Communication Technologies
TOPIC	ICT-56-2020 - Next Generation Internet of Things
TYPE OF ACTION	Research and Innovation Action
COORDINATOR	Capgemini Technology Services (CAP)
PRINCIPAL CONTRACTORS	Atos Spain S.A. (ATOS), ERICSSON GmbH (EDD), ABB Oy (ABB), INTRASOFT International S.A. (INTRA), Engineering-Ingegneria Informatica SPA (ENG), Robert Bosch Espana Fabrica Aranjuez SA (BOSCHN), ASM Terni SpA (ASM), Forum Virium Helsinki (FVH), Optimum Technologies Piroforikis S.A. (OPT), eBOS Technologies Ltd (EBOS), Privanova SAS (PRI), Synelxis Solutions S.A. (SYN), CUMUCORE Oy (CMC), Emotion s.r.l. (EMOT), AALTO-Korkeakoulusaatio (AALTO), i2CAT Foundation (I2CAT), Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), Sorbonne Université (SU)
WORKPACKAGE	WP4
DELIVERABLE TYPE	REPORT
DISSEMINATION LEVEL	PUBLIC
DELIVERABLE STATE	FINAL
CONTRACTUAL DATE OF DELIVERY	31/07/2022
ACTUAL DATE OF DELIVERY	29/07/2022
DOCUMENT TITLE	Enhancing IoT Tactile & Contextual Sensing/Actuating
AUTHOR(S)	A. Gonos (OPT), J. Escrig (I2CAT), M. Catalan (I2CAT), M. Montagud (I2CAT), T. Velivassaki (SYN), A. Voulkidis (SYN), M. Sophocleous (EBOS), A. Corsi (ENG), F. Bellesini (EMOT).
REVIEWER(S)	A. Corsi (ENG), J. Gorroñogoitia (ATOS)
ABSTRACT	SEE EXECUTIVE SUMMARY
HISTORY	SEE DOCUMENT HISTORY
KEYWORDS	Ambient Intelligence, Tactile Internet, Device Discovery, Device Access Control, Augmented Reality

Document History

Version	Date	Contributor(s)	Description
V0.1	05/04/2022	OPT	Table of Contents
V0.1.1	06/04/2022	OPT	Updates on ToC
V0.1.2	18/05/2022	I2CAT	Updates on industry LL requirements
V0.1.3	30/05/2022	OPT, SYN	Inputs to IoT Device Access Control, sec 3
V0.1.4	01/06/2022	I2CAT	Updates on industry LL requirements
V0.2	03/06/2022	EBOS, SYN, OPT	Updates on Section 3, device discovery & device indexing
V0.3	06/07/2022	OPT, SYN, I2CAT, EBOS	Updates on Sections 3, 4 and 5; computer vision based device discovery, device indexing and access control.
V0.4	18/07/2022	OPT, I2CAT, ENG, SYN	Fine-graining Smart Agriculture LL requirements; Updates on IoT Device Indexing and Access Control technical design descriptions; Updates on Device Discovery; Inputs to AR/MR module
V0.5	20/07/2022	OPT, I2CAT, SYN, EMOT	Updates on IoT Device Discovery, IoT Device Indexing and Access Control technical description and implementation of use cases; Overall enhancements
V0.6	21/07/2022	OPT, SYN, EBOS	Enhancements in Sections 2 and 4; Document consolidation for peer review
V0.6.1	25/07/2022	ENG	Peer review
V0.6.2	26/07/2022	ATOS	Peer review
V0.7	28/07/2022	OPT, I2CAT, SYN, EMOT	Addressing review comments; Consolidated version
V0.8	28/07/2022	CAP	Quality Check
V1.0	29/07/2022	CAP	Final version

Table of Contents

Document History	4
Table of Contents	5
List of Figures.....	7
List of Tables.....	9
List of Acronyms and Abbreviations.....	10
Executive Summary	11
1 Introduction.....	12
1.1 Intended Audience.....	13
1.2 Relations to other activities	13
1.3 Document overview	15
2 Ambient Intelligence requirements in IoT-NGIN Living Labs.....	16
2.1 Human-Centred Twin Smart Cities Living Lab.....	17
2.2 Smart Agriculture IoT Living Lab.....	18
2.3 Industry 4.0 Living Lab	20
2.4 Energy Grid Active Monitoring/Control Living Lab	24
3 Ambient Intelligence in IoT-NGIN.....	25
3.1 IoT Device Discovery	26
3.1.1 Description.....	27
3.1.2 Interfaces	34
3.2 IoT Device indexing	34
3.2.1 Description.....	34
3.2.2 Interfaces.....	40
3.3 IoT Devices access control.....	40
3.3.1 Description.....	41
3.3.2 Interfaces	46
3.4 AR/MR module.....	46
3.4.1 Description.....	46
3.4.2 Interfaces	50
4 Implementation for the IoT-NGIN Living Labs.....	52
4.1 IoT Device Discovery	52
4.1.1 Smart Agriculture	53
4.1.2 Industry 4.0	59
4.1.3 Smart Energy	61

4.2	IoT Device indexing	61
4.3	IoT Devices access control	64
4.4	AR/MR module	68
5	Installation and user guidelines	71
5.1	IoT Device Discovery	71
5.1.1	Installation guidelines	71
5.2	IoT Device indexing	73
5.2.1	Installation guidelines	73
5.2.2	User guidelines	74
5.3	IoT Devices access control	75
5.3.1	Installation guidelines	75
5.3.2	User guidelines	77
5.4	AR/MR module	82
6	Conclusions	83
7	References	84
Annex 1	IoT-NGIN Aml tools Interfaces	87
	IoT Device Discovery	87
	IoT Device Indexing	90
	IoT Device Access Control	100
	AR/MR module	102

List of Figures

Figure 1: Work packages structure.....	14
Figure 2: High-level architecture of IoT-NGIN ambient intelligence tools.....	25
Figure 3: Work: Detection and tracking using siammot for detecting vehicles in a highway at daytime with backlight (left) and at night time (right), small objects are not detected and the approach is not correctly working with poor light conditions.....	28
Figure 4: Figure cmp_yolo_efficientDet: Results comparison between EfficientDet (top) and Scaled-YOLO_v4 (bottom) approaches at daytime with backlight (left), and at night time (right). Better results are achieved using scaled-yolo_v4 in general.....	29
Figure 5: Extended Flow chart of VLP solution.....	30
Figure 6: The high-level architecture of the Code Scanning variant of IDD.....	32
Figure 7: Communication flow. UWB Anchor device.....	33
Figure 8: Communication flow. UWB Tag device.....	33
Figure 9: Integration of the UWB positioning solution with the Device Indexing module.....	34
Figure 10: Example of northbound communication with IoT Agent and Context Broker.....	36
Figure 11: Example of southbound communication with IoT Agent and Context Broker.....	37
Figure 12: IDI basic workflow overview.....	38
Figure 13: Overview of the IDI multitenancy feature.....	39
Figure 14: Updated basic IDI workflow.....	40
Figure 15: High-level architecture of the Devices Access Control module.	41
Figure 16: View of the Konga dashboard of the IoT Device Access Control module.....	42
Figure 17: Proximity Plugin Sequence Diagram.....	43
Figure 18: OpenID Connect Plugin Sequence Diagram.....	44
Figure 19: SSI Plugin Sequence Diagram.....	45
Figure 20: Kong open source plugins for authentication.....	45
Figure 21: Kong open source plugins for security.....	46
Figure 22. High-level sequence diagrams for AR presentation and actuation cases.....	51
Figure 23: Model Architecture [38].....	53
Figure 24: Directory Tree.	54
Figure 25: Samples of the training set.	55
Figure 26: Labelling Interface.....	56
Figure 27: The label of the object.....	56
Figure 28: The architecture of Fast R-CNN	57
Figure 29: Model results samples.	58
Figure 30: Prediction times.....	58

Figure 31: Loss characteristics.	59
Figure 32: IoT Device discovery preliminary results in the Industry 4.0 Living Lab.	60
Figure 33: Smart Energy charging station screenshots taken from the videos made by EV cameras.	61
Figure 34: Mobile device and Digital Twin relation.	62
Figure 35: IDI use case for Mobile Device.	64
Figure 36: Sequence Diagram for Access Control in accessing IoT data, applicable in Smart Agriculture LL UC4 and Smart Energy LL UC10.	65
Figure 37: IoT Devices Access Control Plugins' Logs (for the OpenID Connect Authentication plugin in the black terminal, while for the Proximity plugin in the blue terminal).	67
Figure 38: Retrieved Smart Agriculture Device data after successfully passing access control for both the Proximity and the OpenID Connect Authentication plugins.	68
Figure 39: Presentation of overlaid and contextual information to be applied for AR interfaces.	69
Figure 40: Macro blocks architecture for the AR/MR component.	70
Figure 41: Installation complete for the IoT Device Access Control module.	76
Figure 42: Login page for the IoT Device Access Control dashboard.	77
Figure 43: Welcome screen of the IoT Device Access Control dashboard to connect with its Admin service.	78
Figure 44: Services tab in the IoT Device Access Control dashboard.	78
Figure 45: Service creation in the IoT Device Access Control dashboard.	79
Figure 46: Accessing <i>Routes</i> for the <i>device-indexing</i> service.	79
Figure 47: Route creation in the IoT Device Access Control dashboard.	80
Figure 48: Adding plugins for <i>Route</i> entities.	80
Figure 49: Adding custom plugins.	81
Figure 50: Uninstalling the Access Control module.	82

List of Tables

Table 1: Relation of WP4 activities to other WPs and tasks.....	14
Table 2: Ambient Intelligence tools of IoT-NGIN within LL Use Cases.....	16
Table 3: Requirements analysis for Use case #1 "Traffic Flow Prediction & Parking prediction".	17
Table 4: Requirements analysis for Use case #2 "Crowd Management".	18
Table 5: Requirements analysis for Use case #4 "Crop diseases prediction, smart irrigation and precision aerial spraying".	19
Table 6: Requirements analysis for Use case #5 "Sensor aided crop harvesting" use case...20	
Table 7: Requirements analysis for Use case #6 "Human-centred safety in a self-aware indoor factory environment".....	21
Table 8: Requirements analysis for Use case #7 "Human-centred augmented reality assisted build-to-order assembly".	22
Table 9: Requirements analysis for Use case #8 "Digital powertrain and condition monitoring ".....	23
Table 10: Requirements analysis for Use case #10 "Driver-friendly dispatchable EV charging".	24
Table 11: Summary of methods included in the IoT Device Discovery module.	27
Table 12: Demonstration scenarios for AR use cases.	47
Table 13: Sensors to be detected via AR/MR module.	47
Table 14: AR Display Devices.	49
Table 15: Relevant requirements for selecting the development AR framework.	50
Table 16: Discovery methods implemented in each use case.....	52
Table 17: Multitenancy for use case #4.	63
Table 18: Response codes under different scenarios in the OpenID Connect Authentication (Keycloak) plugin.....	66
Table 19: Response codes under different scenarios in the <i>proximity</i> plugin.....	66
Table 20: Environmental variables for the IDI client configuration.	75
Table 21: IoT Device Discovery interfaces.....	87
Table 22: IoT Device Indexing interfaces.....	90
Table 23: IoT Device Access Control interfaces.....	100
Table 24: IoT AR module interfaces.....	102

List of Acronyms and Abbreviations

AAA	Authorization, Authentication and Accounting
AGLV	Automated Guided Land Vehicle
AGV	Automated Guided Vehicle
AI	Artificial Intelligence
Aml	Ambient Intelligence
AMPQ	Advanced Message Queuing Protocol
AR	Augmented Reality
CB	Context Broker
CNN	Convolutional Neural Network
LL	Living Lab
LSE	Least Square Error
GPS	Global Positioning System
HTTP	Hypertext Transfer Protocol
IDAC	IoT Device Access Control
IDI	IoT Device Indexing
IoT	Internet of Things
JWT	JSON Web Token
MR	Mixed Reality
MQTT	Message Queue Telemetry Transport
NGSI	Next Generation Service Interfaces
QR	Quick Response
SSI	Self-Sovereign Identity
UC	Use Case
UWB	Ultra-Wide Band
VLP	Visual Light Positioning
WP	Work Package

Executive Summary

Ambient Intelligence in Internet of Things (IoT) will be an essential part of the next-generation IoT. IoT-NGIN aims to provide a set of components, delivering innovative features and functionality for the next-generation Ambient Intelligence IoT and demonstrate them via diverse living labs and use cases. This deliverable reports on the contributions of IoT-NGIN in that direction, within the scope of its WP4 "Enhancing IoT Tactile & Contextual Sensing/Actuating". The main outcomes reported in this deliverable include:

- Updated analysis of ambient intelligence requirements arising from the Living Lab use cases claiming the use of WP4 tools
- Updated technical specifications of the intermediate version of the four components developed to enable the envisioned use cases, namely:
 - IoT Device Discovery (IDD), which is responsible for recognition and/or positioning of IoT Devices. This component features 4 variants: Computer-Vision based image recognition, Visual Light Positioning, code scanning and Ultra-Wide Band localization.
 - IoT Device Indexing (IDI), which manages device-related information and supports Digital Twin functionality.
 - IoT Device Access Control (IDAC), which intervenes communications among IoT-NGIN components and provides configurable multi-criteria access control.
 - IoT AR/MR Service, which delivers AR/MR interactions across several use cases, towards diverse sensors and diverse AR interfaces.
- Implementations of these tools in foreseen use cases, including integration among them.
- Installation and user guidance of the tools, referring to the intermediate versions of these components, available as open source on GitLab.

This deliverable reports on the intermediate version of the technological contributions from WP4, so specific parts report on work-in-progress or even on planned developments (in the case of AR/MR module). This deliverable will be updated in D4.4 "Enhanced IoT Tactile & Contextual Sensing/Actuating (Final Version)", to be delivered in April 2023.

1 Introduction

The *Tactile Internet*, as defined by ITU in 2014 [1], “will enable haptic interaction with visual feedback, with technical systems supporting not just audiovisual interaction, but also that involving robotic systems to be controlled with an imperceptible time-lag”. IoT-NGIN supports the concept of ambient intelligence (Aml) and tactile internet, via a set of components, also referred as *Aml tools* in the rest of the document, which include:

- *IoT Device Discovery (IDD)*, which is responsible for identifying or recognizing an IoT device, based on conventional methods, such as QR scanning, or advanced visual methods such as combining computer vision based recognition, Ultra-Wide Band (UWB) based localization and Visual Light Positioning (VLP).
- *IoT Device Indexing (IDI)*, which manages information updates related to IoT devices' status and characteristics. IDI can be used to collect device information by the physical device and appropriately update its digital twin or on the opposite direction, to pass control commands to the physical device via its Digital Twin.
- *IoT Device Access Control (IDAC)*, which provides controlled access to the IoT devices, considering a multi-criteria based mechanism applied before any activity on the device by any user.
- *(Augmented Reality) AR / Mixed Reality (MR) module*, which supports AR/MR interactions with IoT devices, leading to accessing monitoring information or even pass control commands.

The present document, entitled “Enhancing IoT Tactile & Contextual Sensing/Actuating”, is the third deliverable of Work Package (WP) 4 (D4.3) and reports the activities of Tasks 4.2 “Enhancing IoT devices discovery, recognition and indexing”, Task 4.3 “Pervasive Security and Ambient Intelligence based Control” and Task 4.4 “AR enhanced personalized IoT sensing and actuating”. Moreover, the activities reported in D4.3 align with the objectives of WP4:

- Improve real things/objects recognition using visual and non-visual methods.
- Pervasive security and novel real-world implementation of an easy-to-manage access rights system based on spatial/ physical access, user rights/groups and IoT device ownership
- Designing mixed reality interface standards between machines and humans/employees
- Increase the flexibility, interoperability, and scalability of manufacturing through hypermedia-based systems
- Create and maintain a repository of AR/MR software components libraries and tools to support AR-IoT interaction.

The present document is a technical report which provides updates on the deliverable D4.2 “Enhancing IoT Ambient Intelligence”, as well as enhancements on the Aml tools of IoT-NGIN. The main updates related to D4.2 are summarized as:

- Updated analysis of the IoT-NGIN Living Lab (LL) Use Cases (UCs) in terms of the use of Aml tools.
- More mature technical design of the tools, compared to the initial version presented in D4.2.
- Use-case specific implementations of the Aml tools. These include updates on the device discovery part, including computer vision based model, Ultra-Wide Band and Visual Light Positioning methods for tracking of Automated Guided Vehicles (AGV) in

the Smart Industry LL, an image recognition model for identifying IoT devices in the Smart Agriculture Living Lab. Moreover, the IoT Device Indexing component has been finalized, enhanced with persistence, multi-tenancy and activity checking functionalities, while the IoT Device Access Control has been implemented, integrating three plugins for imposing different level or scope of access control.

- Initial version of the AR/MR module, providing comprehensive analysis and technical design for the implementation in the relevant LL Ucs.
- Installation and user guidance on the updated versions of the Aml tools implementation.

Last, but not least, the Aml tools developed in WP4 are provided are open source in the IoT-NGIN Gitlab group, available at https://gitlab.com/h2020-iot-ngin/enhancing_iot_tactile_contextual_sensing_actuating.

1.1 Intended Audience

The target audience includes mainly IoT system providers, ML engineers and AR developers, including technical staff, who would be interested in developing ambient intelligence in IoT systems. Through this report, they may identify their relevant ambient intelligence requirements in application domains covered by the IoT-NGIN Living Labs. Moreover, the audience may find the technical specifications and design of the IoT-NGIN tools for supporting ambient intelligence in IoT applications, as well specific implementations for the IoT-NGIN use cases. Furthermore, IT staff and developers have the chance to download, test and extend the current versions of these tools.

Moreover, the report is useful internally, to the members of the development and integration team of the IoT-NGIN consortium, involving mainly Work Package (WP) 3-WP6 partners, the Living Lab teams of WP7, but also to the whole Consortium and third parties, especially those participating via Open Calls, for integration, validation and exploitation purposes. Useful feedback could be also received from the Advisory Board, including both technical and impact creation comments.

1.2 Relations to other activities

The activities of WP4 are strongly connected to the rest IoT-NGIN activities, as indicated in Figure 1.

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

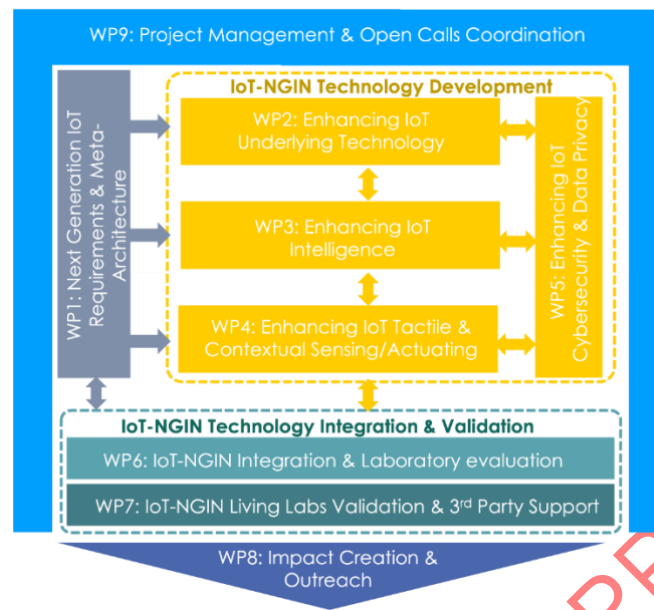


Figure 1: Work packages structure.

Considering this, WP4 is related to the work packages and tasks described in Table 1.

Table 1: Relation of WP4 activities to other WPs and tasks

WP	Relation to WP4 and D4.3
WP1	The definition of the Living Labs' use cases (Ucs) and elicitation of user/system requirements has served D4.3 for the (updated) definition of the IoT Ambient Intelligence components implemented in tasks T4.2, T4.3 and T4.4. Also, the role of the tools and modules deployed in WP4 shall be represented in the meta-architecture resulting by the activities in WP1.
WP2	No interaction is foreseen among WP4 and WP2.
WP3	The training and sharing of AI models of the IoT Device Discovery components can be conducted via the Machine Learning as a Service (MlaaS) platform developed in WP3. Moreover, deployment of AI models into physical devices may be passed via the Digital Twin functionality of the IoT Device Indexing module.
WP5	The IoT Device Indexing and the IoT Device Access Control modules deployed in WP4 interact with different components developed within the scope of WP5, such as the IoT Vulnerability Crawler and the MTD Honeypots Framework. Moreover, the IoT Device Access Control integrates the Self-Sovereign Identities module.
WP6	WP4 components will be integrated with the rest of project's technologies and frameworks within WP6, while some of the WP4

	components may be used in the development of application logic for the use cases or the Open Calls.
WP7	WP4 components will be implemented and used in several living labs and use cases as defined in Section 2.
	WP4 will support 3 rd parties by offering a specific set of functionalities (such as the IoT-AR repository).
WP8	WP4 will provide notable outcomes and results for supporting impact creation activities. Moreover, it will consider feedback (e.g. from the market analysis and business modelling tasks) which could be relevant for the WP4 design and development.

1.3 Document overview

The rest of the document is organized as follows.

Section 2 provides any updates on the requirements of the Living Lab use cases by the Ambient Intelligence tools developed in WP4.

Section 3 presents the technical design of these tools, while section 4 describes use case specific implementations for the Living Labs, where relevant.

Section 5 provides installation and user guidance for the ambient intelligence tools, while section 6 draws conclusions and next steps.

2 Ambient Intelligence requirements in IoT-NGIN Living Labs

This section provides updated analysis of the requirements arising from the use cases of the four living labs of the IoT-NGIN project, in order to update, as well, the four software modules developed within WP4. The analysis of the Living Labs (LLs) use cases from the Ambient Intelligence perspective has been thoroughly presented in D4.2, including preconditions of each use case of interest, mapping of those preconditions to Ambient Intelligent tools, use case requirements analysis, as well as a user story.

Table 2 summarizes the Aml tools claimed for each LL use case, identifying the variant of interest for each tool, while indicating the table which analyzes the relevant UC requirements. As shown in this table, 8 out of 10 LL UCs will integrate Aml tools in their IoT-NGIN deployment.

Table 2: Ambient Intelligence tools of IoT-NGIN within LL Use Cases.

Living Lab	Use Case	Device Discovery	Device Indexing	Device Access Control	AR/MR tools	Mapping table
Human-Centred Twin Smart Cities	#1	Computer vision	Yes	AAA SSI	No	Table 3
	#2	Computer vision	Yes	AAA SSI	No	Table 4
Smart Agriculture IoT	#4	<ul style="list-style-type: none"> Image recognition Code scanning 	Yes	AAA proximity	Yes	Table 5
	#5	Code scanning	Yes	AAA	No	Table 6
Employee Friendly Industry 4.0	#6	Computer Vision UWB VLP	Yes	AAA	Yes	Table 7
	#7	Code Scanning	Yes	No	Yes	Table 8
	#8	No	Yes	Yes	No	Table 9
Smart Energy	#10	Computer vision /Code scanning	Yes	AAA Proximity	Yes	Table 10

2.1 Human-Centred Twin Smart Cities Living Lab

Aml tools are planned to be used in UC#1 "Traffic Flow Prediction & Parking prediction" and UC#2 "Crowd Management", as indicated in Table 3 and Table 4, respectively. Both Ucs consider using computer vision based image recognition to identify moving vehicles or cars, respectively. Moreover, device indexing is foreseen for keeping IoT device data, as well as authentication and authorization services.

Table 3: Requirements analysis for Use case #1 "Traffic Flow Prediction & Parking prediction".

Use Case		#1 Traffic Flow Prediction & Parking prediction
Ambient Intelligence Application		Object detection
Ambient Intelligence Service		
Device discovery	Device recognized	Cars, tracks, bicycles
	Recognition method	Computer vision
Device Indexing	Registration method	Manual
	Data registered	device ID, device type, location
	Data Type(s)	JSON files
	Database type	NoSQL – structured
	Data size	<1MB per day
Device Access Control	User rights	Yes – for modifying Semantic Twin data
	Spatial proximity	No
Actuation – AR interaction	Actuation	Not needed
	AR interface	Not needed
	AR software	Not needed

Table 4: Requirements analysis for Use case #2 "Crowd Management".

Use Case		#2 Crowd Management
Ambient Intelligence Application		Object detection
Ambient Intelligence Service		
Device discovery	Device / Object recognized	People
	Recognition method	Computer Vision
Device Indexing	Registration method	Manual
	Data registered	device ID, device type, location
	Data Type(s)	JSON files
	Database type	NoSQL – structured
	Data size	<1MB per day
Device Access Control	User rights	Yes – for modifying Semantic Twin data
	Spatial proximity	No
Actuation – AR interaction	Actuation	Not needed
	AR interface	Not needed
	AR software	Not needed

2.2 Smart Agriculture IoT Living Lab

Within the Smart Agriculture IoT Living Lab, both UC#4 "Crop diseases prediction, smart irrigation and precision aerial spraying" and UC#5 "Sensor aided crop harvesting" express requirements towards the WP4 tools in Table 5 and Table 6, respectively. In UC#4, image recognition based on Computer Vision, combined with code scanning, will be employed to enable identification of individual SynField IoT devices. Moreover, the Device Indexing component will be used for tracking device related updates, while access to those data will be controlled by both authentication/authorization services and proximity constraints. Last, but not least, AR/MR functionality will be needed to facilitate data visualization and application of control commands to the devices.

Table 5: Requirements analysis for Use case #4 “Crop diseases prediction, smart irrigation and precision aerial spraying”.

Use Case		#4 Crop diseases prediction, smart irrigation and precision aerial spraying
Ambient Intelligence Application		IoT devices recognition, and indexing
Ambient Intelligence Service		
Device discovery	Device recognized	SynField devices [2]
	Recognition method	Computer vision & QR based
Device Indexing	Registration method	Manual/programmatic
	Data registered	For mobile phones: Device ID, device type, location For SynField devices: Device ID, device type, location, measurements For drones: Device ID, device type, location, images
	Data Type(s)	JSON files
	Database type	NoSQL – structured
	Data size	Depending on the number of devices and how often measurements/images are collected; Up to 300KB per day per SynField node Up to 500 MB per day per drone
Device Access Control	User rights	Yes – for reading data and applying actuation (e.g. irrigation)
	Spatial proximity	Yes – proximity considered for granting access to device data and controls
Actuation – AR interaction	Actuation	Yes, for information visualization and activating the irrigation using AR tools
	AR interface	Yes (smartphone)
	AR software	Unity + Vuforia

Table 6: Requirements analysis for Use case #5 “Sensor aided crop harvesting” use case.

Use Case		#5 Sensor aided crop harvesting
Ambient Intelligence Application		IoT indexing and access control
Ambient Intelligence Service		
Device discovery	Device recognized	AGLVs
	Recognition method	QR based
Device Indexing	Data registered	Device type, ID, location
	Registration method	Manual/programmatic
	Data Type(s)	JSON files
	Database type	NoSQL – structured
	Data size	Up to 500 MB per day per robot
Device Access Control	User rights	Yes – for reading/visualizing data
	Spatial proximity	N/A
Actuation – AR interaction	Actuation	No
	AR interface	No
	AR software	N/A

2.3 Industry 4.0 Living Lab

All 3 Ucs in the Industry 4.0 LL have declared interest in using Aml functionality, listed in Table 7, Table 8 and Table 9, respectively. These refer to computer vision based device discovery, localization, as well as device indexing and control, but also AR visualization and interaction.

Table 7: Requirements analysis for Use case #6 "Human-centred safety in a self-aware indoor factory environment".

Use Case		#6 Human-centred safety in a self-aware indoor factory environment
Ambient Intelligence Application		AGV – Humans collisions prevention
Ambient Intelligence Service		
Device discovery	Device recognized	Automated Guided Vehicles, human-guided vehicles (also humans)
	Recognition method	UWB – VLP– Computer Vision
Device Indexing	Registration method	Manual / <u>Automatic recognition (computer vision)</u> For UWB and VLP methods the first registration will be manual
	Data registered	Device type [E.g., 'AGV'], D [integer], Location method ['UWB', 'VLP', 'CV'] Position [x, y], timestamp (positioning instant)
	Data Type(s)	JSON files
	Database type	Local.
	Data size	Depending on the number of devices and how often they are detected <500 Mb per day
Device Access Control	User rights	Yes – for device data access
	Spatial proximity	Not needed
Actuation – AR interaction	Actuation	If there is a potential collision detected, the workers will receive a notification from the system. Other actuations are not needed
	AR interface	AR headsets (and maybe smartphones)
	AR software	Unity + Vuforia

Table 8: Requirements analysis for Use case #7 "Human-centred augmented reality assisted build-to-order assembly".

Use Case		#7 Human-centred augmented reality assisted build-to-order assembly
Ambient Intelligence Application		AR assistance on assembly steps
Ambient Intelligence Service		
Device discovery	Device recognized	Drive cabinet
	Recognition method	Computer Vision – AR Model target
Device Indexing	Registration method	Manual
	Data registered	List of assembly steps, status of assembly
	Data Type(s)	-
	Database type	-
	Data size	-
Device Access Control	User rights	Not needed
	Spatial proximity	Not needed
Actuation – AR interaction	Actuation	Completion of assembly steps (wiring). Automatic recognition of completed step, or manual interaction by user
	AR interface	Smartphone
	AR software	Unity + Vuforia

Table 9: Requirements analysis for Use case #8 "Digital powertrain and condition monitoring ".

Use Case		#8 Digital powertrain and condition monitoring
Ambient Intelligence Application		Access control to the powertrain
Ambient Intelligence Service		
Device discovery	Device recognized	Powertrain ensemble (Drive + motor + sensors)
	Recognition method	Manual
Device Indexing	Registration method	Manual
	Data registered	Drive operating data (motor speed, voltage, torque, current ...). External sensors (temperature, thermal camera data, accelerometer data).
	Data Type(s)	The various sensor and device signals are gathered to an intermediary gateway device running node-red. The gateway device serves/forwards mostly JSON payloads, but can adapt to other formats as well
	Database type	Local, timeseries database
	Data size	TBD
Device Access Control	User rights	Access is restricted by default, only authorized applications and users should have access to the data.
	Spatial proximity	Not needed
Actuation – AR interaction	Actuation	Not needed
	AR interface	Not needed
	AR software	Not needed

2.4 Energy Grid Active Monitoring/Control Living Lab

Ambient Intelligence is also foreseen in UC#10 "Driver-friendly dispatchable EV charging" in the Smart Energy domain via all 4 tools of IoT-NGIN. ML based image recognition will be used to recognize charging stations, with which AR interaction will enable visualization of those stations data. To this, device data will be persisted via device indexing, while access to those data is required to be controlled for authorized users close to the device. The relevant requirements are listed in Table 10.

Table 10: Requirements analysis for Use case #10 "Driver-friendly dispatchable EV charging".

Use Case		Driver-friendly dispatchable EV charging
Ambient Intelligence Application		Charging station – AR interaction
Ambient Intelligence Service		
Device discovery	Device recognized	Charging Stations
	Recognition method	Visual method – Computer Vision / Code scanning
Device Indexing	Registration method	<u>Automatic recognition</u>
	Data registered	Device type ['charging stations'], ID [integer], Position [x, y], timestamp
	Data Type(s)	JSON files
	Database type	NoSQL – structured
	Data size	500 Mb per day
Device Access Control	User rights	Authentication with username and password
	Spatial proximity	Less than 2 meters
Actuation – AR interaction	Actuation	Charging session management
	AR interface	Smartphones
	AR software	Unity + Vuforia

3 Ambient Intelligence in IoT-NGIN

The IoT-NGIN ambient intelligence tools can be combined in integrated scenarios to support a wide range of use cases. The interaction among such tools is represented in the high-level architecture of Figure 2, which updates the relevant diagram of D4.2 – Figure 12.

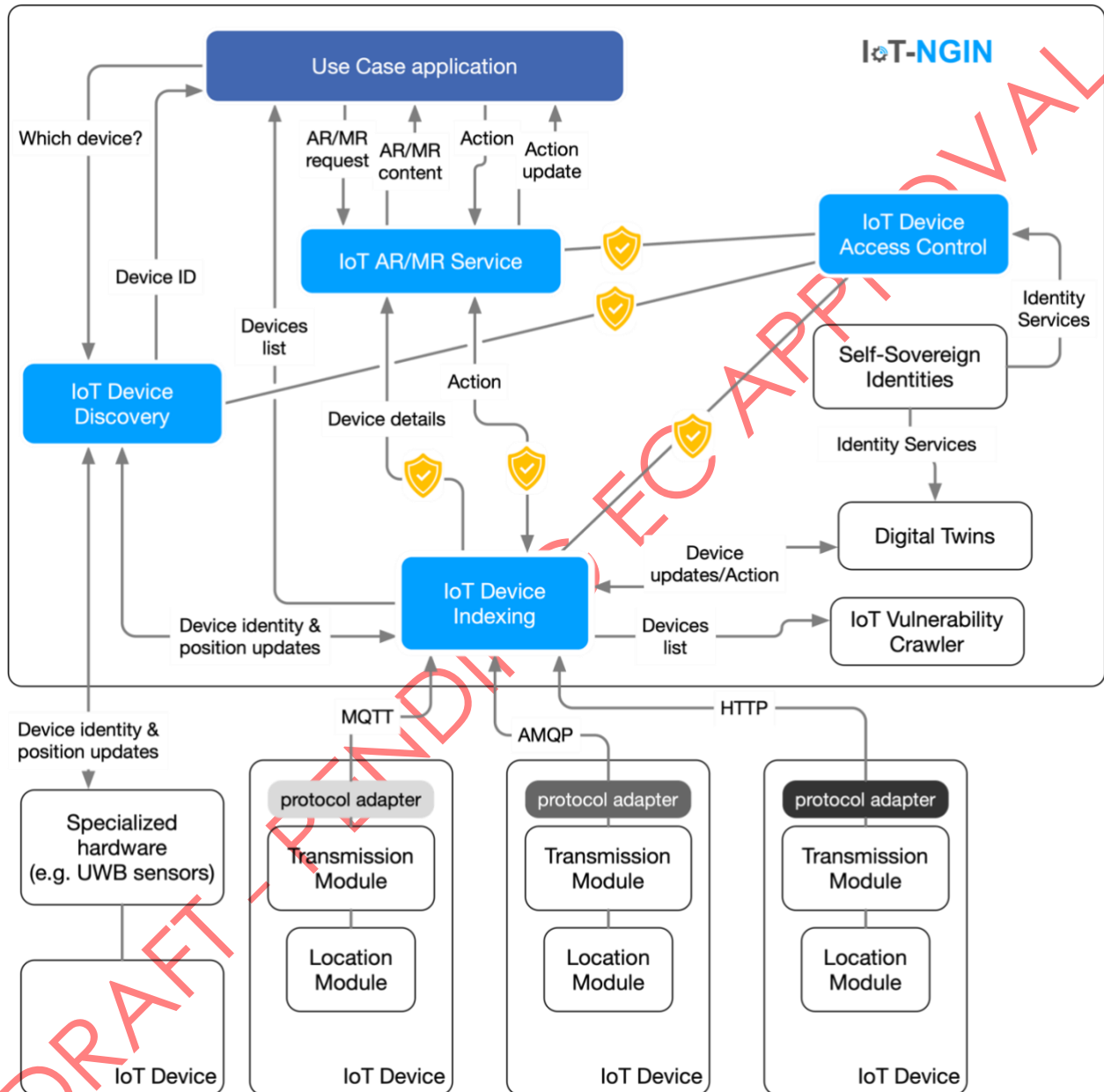


Figure 2: High-level architecture of IoT-NGIN ambient intelligence tools.

This architecture presents the logic for advanced interaction services to physical IoT devices, appearing on the lower part of the figure, via a *Use Case application* on a user's device. The 'background' services include:

- IoT Device Discovery (IDD), which is a service allowing for identification or recognition of IoT Devices. As this functionality is quite dependent on the underlying IoT devices involved in the use case of interest, IoT-NGIN delivers 4 variants, which could be used

independently or in some combination to deliver differentiated levels of accuracy, which may span from plain recognition of an image type as to full tracking of specific devices.

- IoT Device Indexing (IDI), which acts as a 'glue' component between IoT-NGIN services and physical devices, providing Digital Twin functionality up to the level of an Informative Twin, in the sense described in [3].
- IoT Device Access Control (IDAC), which intervenes communications among IoT-NGIN services and provides configurable multi-criteria access control, in a plugins-based approach via a flexible API Gateway.
- IoT AR/MR Service, which delivers AR/MR functionality across several use cases. As this is also highly dependent on the use case, the foreseen AR interface and the target device, on which AR interactions are going to be applied, implementations are foreseen in 4 Use Cases in diverse environments and different mobility characteristics. With respect to D4.2, this service has been updated to integrate the *IoT Device Actuation* functionality, which thus does not appear as a separate component.

In the following, the technical specifications of the Aml tools are provided.

3.1 IoT Device Discovery

The IoT Device Discovery component has three main functions:

- Recognize objects that are contained inside a specific space from the signal received from different types of sensors.
- Position the objects recognized in a known coordinates system
- Record the position of the objects together with the ID and the type of the object in the IoT Device Indexing module

The module supports four different methods for recognition and positioning of objects. Three of them are visual; Computer Vision, Code Positioning and Visual Light Positioning. The last one is the Ultra-Wide Band positioning which is non-visual but based on radio signals. These four methods were selected after reviewing the state of the art of the recognition and positioning methods in the deliverable D4.2 (sections 2.1 and 2.2). The selection was done according to the needs of the use cases of the project.

Additionally, one of the objectives of the WP4 is to improve the existing visual/non-visual methods for object recognition. For this purpose, three of the methods included in the IoT Device Discovery module present some progress beyond the state of the art in terms of accuracy, robustness or latency.

The following table classifies the discovery methods included in the IoT Device Discovery module in visual and non-visual and also specifies what type of recognition and positioning method is used in each case. It also indicates what type of specific hardware is required for each method and the main progresses beyond the state of the art that have been achieved in the present project as the result of the research conducted in WP4.

Table 11: Summary of methods included in the IoT Device Discovery module.

IoT Device Discovery		Recognition Method	Positioning Method	Specific Hardware Required	Main progress beyond the SoTA
Visual Methods	Computer Vision	Convolutional Neural Network	Homo-graphy	Camera	Tracking and detection accuracy improvement
	Visible light positioning	Visual Light Communication	Trilateration	LED Lamps Camera	Positioning based on frequency identification and clustering
	QR Codes	Code Scanner	-	Camera	The component is provided to cover the QR scanning needs of Ucs, based on SoTA methods and considering logging for security or other purposes
Non Visual Methods	Ultra Wide Band positioning	Ultra Wide Band	Trilateration	UWB beacons UWB tag	Positioning algorithm in NLOS (Non Line-Of-Sight) situations or noisy scenarios

More details describing each of the discovery method included in the IoT Device Discovery module and the interfaces available to interact with them are explained in the next subsections.

3.1.1 Description

3.1.1.1 Computer Vision

In the section 2.1.1 of the deliverable D4.2 some methods based on computer vision for object detection were review. As a conclusion, it was found that the Convolutional Neural Networks (CNN) would achieve the best performance in terms of accuracy and present a better chance to go beyond the state of the art in computer vision for object detection. For this reason, it was decided that the CNN would be employed as a core solution for the discovery method based on computer vision of IoT-NGIN. In the section 4.1.1.1 of the same deliverable, it was explained in detail how the CNN can detect different types of objects in images and how they can be trained to detect new classes of objects that are identified in the use cases of the IoT-NGIN project.

In this deliverable, we introduce the need of not only detecting objects but also **tracking** them. This is to assign an ID to all the objects identified in and image of a video, and then re-identify the same objects in the following images of the video and assign to them the same IDs that they had in the previous images. This need is a clear requirement of the use case #6 "Human-centred safety in a self-aware indoor factory environment" in which the trajectory and velocity of the objects is needed to anticipate potential collisions between vehicles and humans working in a factory. Furthermore, the tracking functionality of the IoT Device Discovery module can enable multiple applications that would not be possible only with the detection of the objects.

In this section we review and test different trackers on the literature in a real scenario (vehicles in highway with poor light conditions) and finally **we proposed a new tracker** that combines two existing neural networks and that improves the tracking and detection accuracy of the existing trackers in the literature.

Multiple trackers on the literature were reviewed [4] [5] [6] [7] [8] [9] which are based on different approaches. The selected tracker follows the tracking-by-detection paradigm. It is a multi-object tracker that is part of the new family of trackers that learns the object detection task and appearance, embedding the task simultaneously (JDE) in a shared neural network [10]. It is based on a siamese network for object tracking which has been employed on visual and multi-object tracking with very promising results [11] [12] [13]. The siamese tracker [13] models the motion of instances across frames and it is used to temporally link detection in online multi-object tracking. Detection is performed using an end-to-end detector (Faster-RPN [14]) with a motion model and adds a region-based Siamese tracker to model instance-level motion. The tracker is a region-based multi-object tracking network that detects and associates object instances simultaneously and uses deep features as visual cues for data association. The tracker improves the motion model of trackers such as [15] [16].

We tested the join detector with tracker in a real scenario with adverse conditions. We decided to track vehicles moving in a highway with some of them being very far from the camera and with poor lighting conditions. We have seen that the joint detector with tracker was not enough to cope with small objects in the scene and it was not working correctly with low light conditions, as can be seen in the Figure 3. So, we decided to perform a previous object detection of the objects of the scene as input to the tracker. Several object detectors were analyzed and based on their performance and accuracy on the Coco [17] database were selected for. Scaled-YOLO_v4 [18] from the Yolo family is the one that has provided better results in general as it can be seen in Figure 4 compared with EfficientDet family detectors [19].



Figure 3: Work: Detection and tracking using siamot for detecting vehicles in a highway at daytime with backlight (left) and at night time (right), small objects are not detected and the approach is not correctly working with poor light conditions.



Figure 4: Figure cmp_yolo_efficientDet: Results comparison between EfficientDet (top) and Scaled-YOLO_v4 (bottom) approaches at daytime with backlight (left), and at night time (right). Better results are achieved using scaled-yolo_v4 in general.

Using the detection outputs as the inputs for the tracker has improved the accuracy considerably, especially for the small objects. However, we have detected some general problems, which are briefly discussed next. Motorbikes were not detected when they were far from the camera, but cars with the same size were detected. Also, performance is affected by the lighting conditions. In general, at night the detector is still working when there is some minimum light on the scene. The traffic lamps on the highway for the evaluated interurban scenes are enough to have some detections but the accuracy ratio has decreased considerably compared with the daytime, whereas the size of the vehicles to be detected at night needs to be bigger than at daytime. Regarding the tracker, the main observed problems were caused by occlusions and the switching of the objects' ids.

3.1.1.2 Visual Light Positioning

The Visual Light Positioning solution was initially presented in D4.2. Herein, we provide an extended flow chart for the VLP device that is being developed.

The device is composed by a Raspberry Pi 4 main board and a camera peripheral. The software runs on Linux and is programmed in C++ language for an optimized performance. The detection of the VLP Leds and the positioning is based on OpenCV libraries.

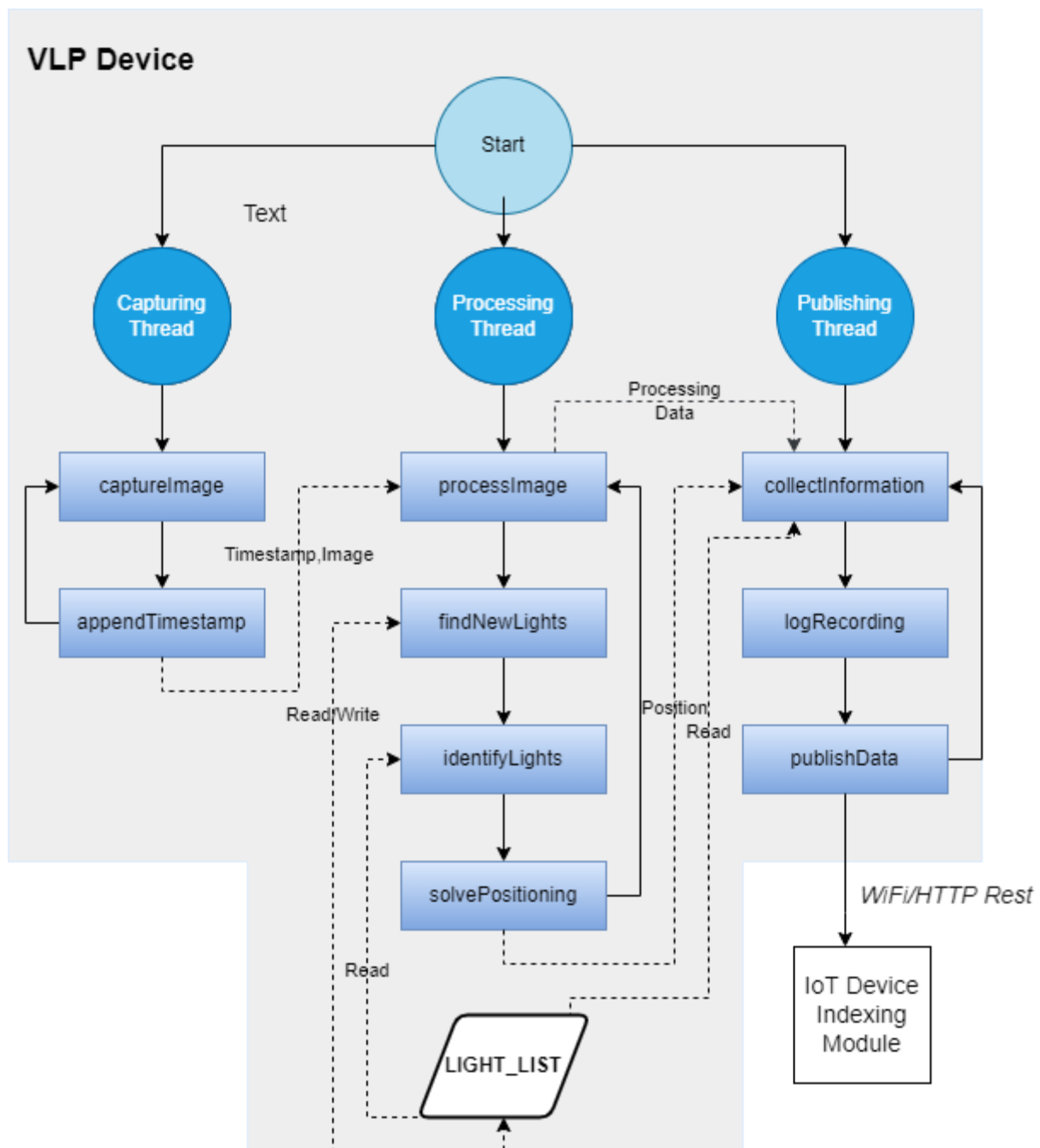


Figure 5: Extended Flow chart of VLP solution.

As shown in Figure 5, the system is based on three independent threads to assure real-time traceability. The multi-thread approach separates the capture of the images (Capture Thread), the identification of the lights and the calculation of the positioning (Processing Thread), as well as the publication of the data in the Device Indexing Module (Publishing Thread).

In the *Capture Thread*, images are obtained from the camera at a stable frame rate. A timestamp is added to the image, which is then queued for further processing. This thread is mostly idle in the meantime between two consecutive frames.

The *Processing Thread* carries out several tasks. First, frames are processed to identify potential lights (VLP Leds). This involves the binarization of frames, the application of an algorithm to find the contour of areas with a higher intensity (which could correspond to lights) and an efficient discarding of non-VLP (not modulated) light sources (e.g. conventional lights deployed in the same area or solar reflections).

The solution requires information about the accurate location of the VLP Leds that are used as a reference for the positioning algorithm. This data is provided in a configuration file (referred as `LIGHT_LIST` in Figure 5). Once at least four VLP Leds are identified, a Perspective-n-Point algorithm implemented by the OpenCV `solvePNP` function is run to calculate the location of the VLP device in respect to the known position of the identified Leds.

Finally, the *Publishing Tread* is responsible for the transmission of the location data using a WiFi interface to the IoT Device Indexing Module. Also, it is used to store and visualize information about the processed frames for debug purposes.

3.1.1.3 Code scanning

The Code Scanning variant of IDD is realized as a server-side code scanner, exposing RESTful interfaces to initiate scanning operations in a stateless manner. The technical design of this sub-component has been already presented in D4.2. The high-level architecture is depicted in Figure 6, indicating the interactions with the rest Aml tools of IoT-NGIN. The *Code Scanner* is composed of a core part, as well as a RESTful API enabling interaction with the component. The core part (Scan service) performs the scanning operation on the designated code images, communicated as an image per se or via its url through the API consumed by another IoT-NGIN or third-party service, or even a mobile/web application (consumer). In order to access the API, the consumer service must be authenticated, as required by the *Code Scanner* and configured in the *IoT Device Access Control*. In its simplest form, Authorization, Authentication and Accounting (AAA) services are imposed, while additional access control rules might be in place, depending on the use case. Indicatively, proximity constraints might be imposed for granting consumer IoT devices access to scan results associated with specific IoT devices.

In addition, logging the code scan activity associated with a consumer IoT device or a scanned IoT device is possible via the integration of this component with the IoT Device Indexing module. Monitoring the relevant activity enables maintaining the digital footprint of IoT devices, which can be useful in various scopes, such as security, energy profile or business-wise monitoring.

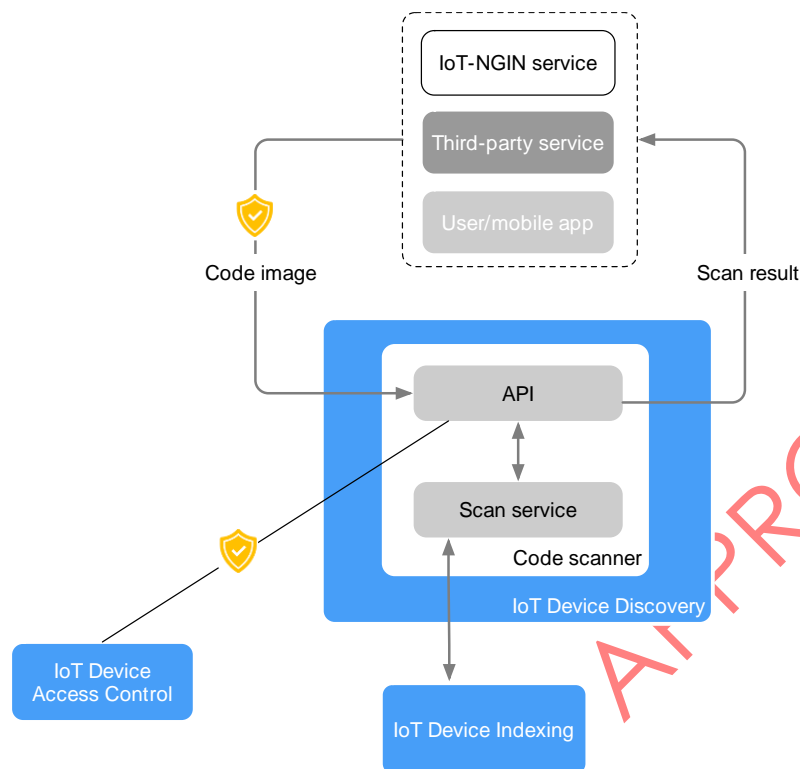


Figure 6: The high-level architecture of the Code Scanning variant of IDD.

The updated REST interfaces of this component are described in Table 21 in Annex 1.

3.1.1.4 Ultra-Wide Band Positioning

The implementation of the Ultra-Wide Band Positioning (UWB) solution and the details of the initial prototype were already presented in Section 4.1.1.4 of deliverable D4.2.

The initial implementation has been improved and complemented with the implementation of an API that facilitates:

- Remote hardware actuation. Concretely, reset actions can be sent to the anchor devices without on-site intervention if it is needed during the deployment setup or the system's execution, in case UWB positioning malfunctioning is detected.
- Anchor positioning updates. Once started, the locating device (UWB tag installed in the AGV) can remotely receive the position of new anchors installed; so that it can use this information to update its location.
- Remote configuration. Main configuration parameters of the UWB positioning system can be remotely applied via API to the deployed devices.
- Integration with the IoT Device Indexing module. The UWB tag implements an HTTP REST interface to send location updates to the IoT-NGIN IoT Device Indexing module.

Different algorithms have been considered for the estimation of the position in the UWB tag. Finally, the Gauss-Newton [20] method has been implemented in the ESP32 device. The effect of different parameters, such as the maximum amount of iterations of the algorithm or its expected precision has been characterized. Also, an averaging method has been implemented to minimize the impact of outliers in a real deployment. The final

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

implementation has focused in achieving a trade-off between accuracy, computing time and data freshness.

The following diagrams in Figure 7 and Figure 8 represent the communication flow in the anchor and tag devices, respectively.

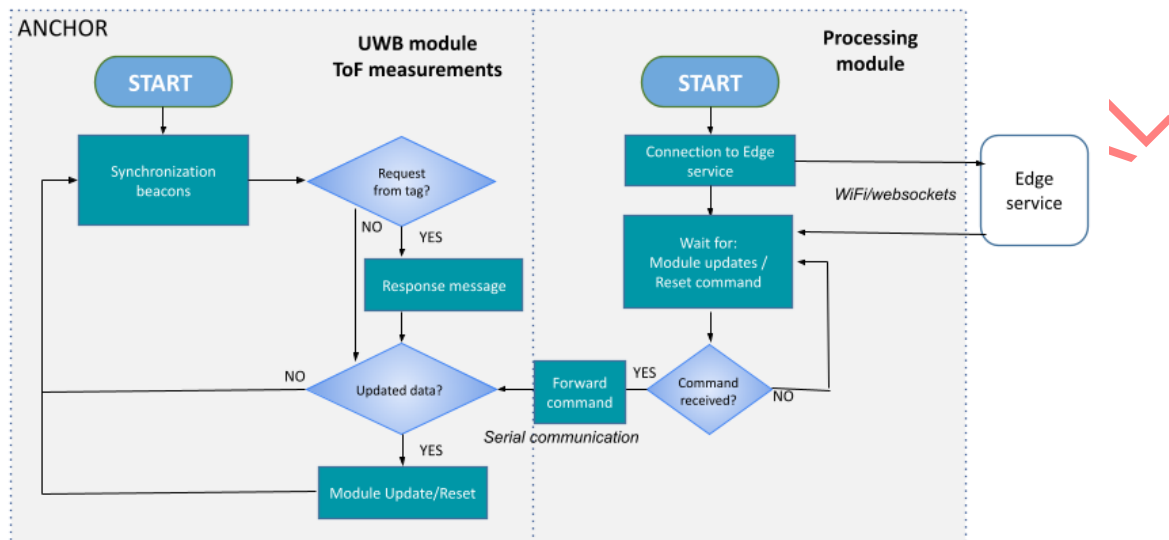


Figure 7: Communication flow. UWB Anchor device.

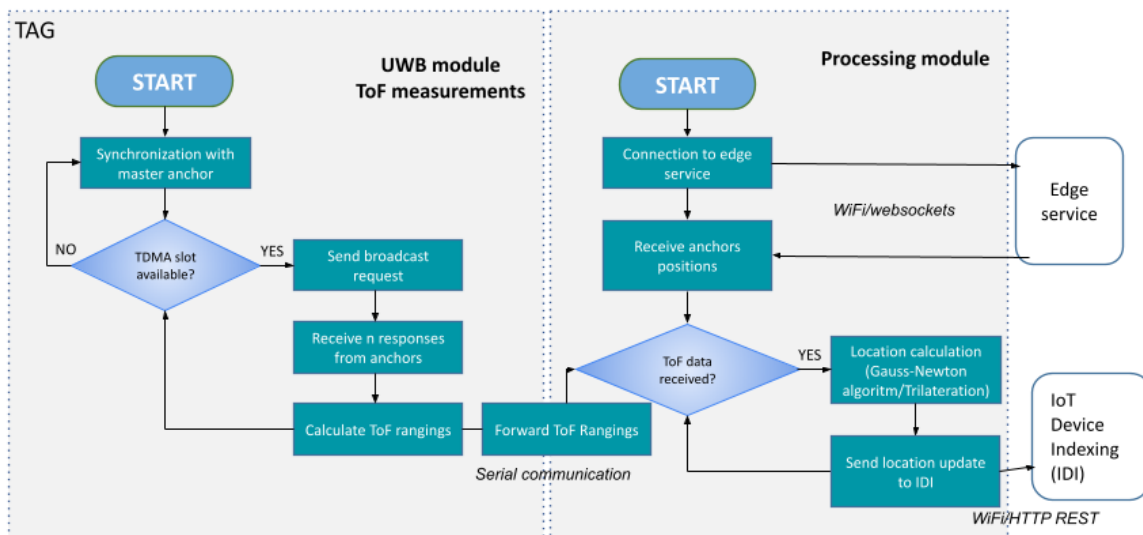


Figure 8: Communication flow. UWB Tag device.

The screenshot in Figure 9 shows the successful integration of the UWB positioning solution with the IoT Device Indexing module.

As it can be seen for each tracked position, the following data are sent in JSON format:

- Location: x, y in meters (relative position to the pre-established reference coordinates system)

- Location source: Device Discovery method (in this case, UWB)
- Location timestamp: Tracking timestamp (here, the tracking is performed approximately each second).

In this log, as seen in the last line, the id of the device is "1344" and it corresponds to an AGV device (previously registered).

```
Measurements: {'device_data': [{'location': '20.944521, 2.601307', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:48.931000'}]}
Batch 251 was received successfully!
Object: {'location': '21.802029, 2.517698', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:50.006000'}
Measurements: {'device_data': [{'location': '21.802029, 2.517698', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:50.006000'}]}
Batch 252 was received successfully!
Object: {'location': '22.721222, 2.361671', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:50.974000'}
Measurements: {'device_data': [{'location': '22.721222, 2.361671', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:50.974000'}]}
Batch 253 was received successfully!
Object: {'location': '21.988852, 3.907974', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:51.942000'}
Measurements: {'device_data': [{'location': '21.988852, 3.907974', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:51.942000'}]}
Batch 254 was received successfully!
Object: {'location': '24.174204, 3.226782', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:53.017000'}
Measurements: {'device_data': [{'location': '24.174204, 3.226782', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:53.017000'}]}
Batch 255 was received successfully!
Object: {'location': '25.321184, 3.296211', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:53.985000'}
Measurements: {'device_data': [{'location': '25.321184, 3.296211', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:53.985000'}]}
Batch 256 was received successfully!
Object: {'location': '22.243738, 2.718638', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:55.061000'}
Measurements: {'device_data': [{'location': '22.243738, 2.718638', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:55.061000'}]}
Batch 257 was received successfully!
Loading details of entity: urn:ngsi-ld:Device:1344
-----
{'id': 'urn:ngsi-ld:Device:1344', 'type': 'AGV', 'timeInstant': {'type': 'DateTime', 'value': '2022-06-03T10:41:28.278Z', 'metadata': {}}, 'device_data': {'type': 'StructuredValue', 'value': [{'location': '22.243738, 2.718638', 'location_source': 'UWB', 'location_timestamp': '2022-05-27T14:15:55.061000'}], 'metadata': {'timeInstant': {'type': 'DateTime', 'value': '2022-06-03T10:41:28.278Z'}}}}
```

Figure 9: Integration of the UWB positioning solution with the Device Indexing module.

Some preliminary tests of the system have been performed in a real scenario in the Bosch factory with adequate results.

On the other hand, a software module is being also developed to bring the location computation and position update functionalities out of the hardware device. In this way, these functionalities could run in any Linux-based device (e.g, an edge server) with independence of the hardware and technology used for retrieving the location information. Concretely, the module will receive distance information from different anchor devices through a serial interface or a CSV file. This information will be forwarded to the location computation module. Once the position is calculated, the positioning update module will send it to the Device Indexing module. Currently, the location computation module supports the Least Square Error (LSE) and Gauss-Newton algorithms; but will allow the incorporation of other algorithms in the future to study better approaches to improve the location accuracy.

3.1.2 Interfaces

The interfaces of the Device Discovery variants are listed in Table 21 in Annex 1.

3.2 IoT Device indexing

3.2.1 Description

The IoT Device Indexing (IDI) module is a mechanism to store, keep track of and query a device's Digital Twin. A Digital Twin, while it does not have a definitive term in the literature, can be defined as the digital representation of a device's information. This copy can be used to access the device's data without having to come in direct contact with the device itself.

The current implementation of IDI, as has been described in deliverable D4.2 [21], relies heavily on FIWARE technologies. More specifically, the IDI module consists of the FIWARE Orion Context Broker, the IoT Agent component and the Historic Data Registry.

A brief description of the IDI components of is as follows:

- **FIWARE Orion Context Broker:** The Context Broker is responsible for brokering information related to constructing the Digital Twin of a device. Orion Context Broker allows managing the entire lifecycle of context information including updates, queries, registrations and subscriptions. It is an NGSIv2 [22] server implementation to manage context information and its availability. Context information consists of entities (e.g. a car) and their attributes (e.g. the speed or location of the car), as well as accompanying metadata [23].
- **IoT Agent:** The IoT Agent component acts as a middleware between a device and the FIWARE Context Broker. It translates the information sent by a device in its native communication protocol (HTTP, MQTT, AMQP), in an NGSI model during data retrieval, while it is also used to pass commands to the devices.
- **Historic Data Registry:** Since the Context Broker component does not support data persistence, the Historic Data Registry was developed to keep a record of all the changes occurring on a device's information.

Context Broker & IoT Agent

Both the FIWARE Context Broker (CB) and the IoT Agent expose several RESTful HTTP Application Programming Interfaces (APIs), which allow for the interaction with the Context Broker, particularly managing and accessing updates, queries, registrations, and subscriptions.

The communication with the devices which are accessible via one or more IoT Agent instances can be bidirectional; *Southbound traffic* handles the communication of control commands to the devices, while *Northbound traffic* allows data retrieval from the devices. Information about both Northbound and Southbound traffic can be found in FIWARE tutorials for the IoT Agent [24].

Northbound Traffic

A basic flow of how a device interacts with the IoT Agent and the Context Broker can be described as follows:

1. A device must first create a *Service* in IoT Agent. *Services* are described as a means to authenticate a device when it is sending a measurement.
2. Then the device should register itself to the IoT Agent, in order to become identifiable.
3. A device can, then, send its data, through its selected communication protocol to IoT Agent.
4. The IoT Agent receives the information, translates it into an NGSI-compatible request and pushes it to the Context Broker.

Now that the Context Broker has the device's Digital Twin successfully instantiated, an external system or a user can take advantage of its API endpoints and gain access to the available information. The above flow is graphically represented in Figure 10.

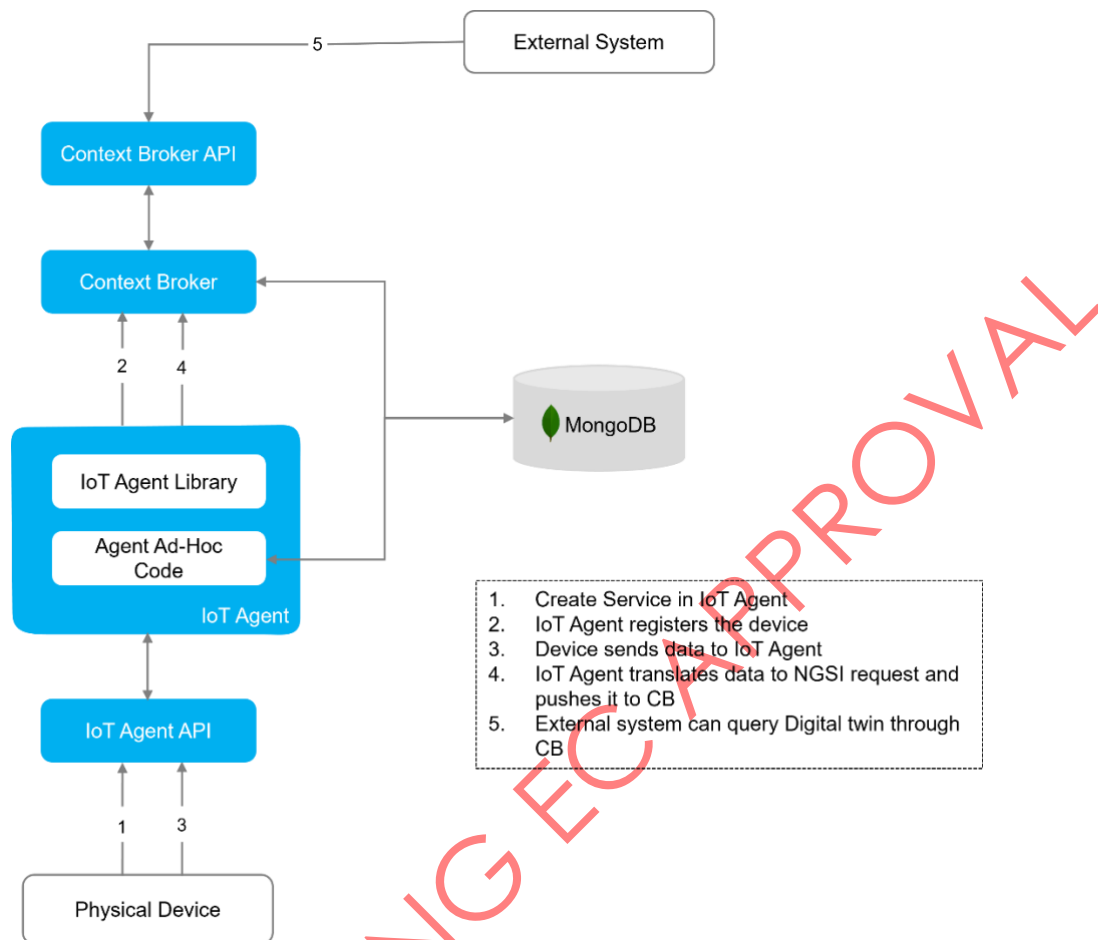


Figure 10: Example of northbound communication with IoT Agent and Context Broker.

Southbound Traffic

Apart from having a device send data to the IoT Agent and Context Broker, the two components can, also, interact with it. A basic workflow is as follows:

1. An external system or user requests from the Context Broker to modify the Digital Twin's data.
2. The Context Broker forwards the request to the IoT Agent.
3. The IoT Agent translates the request and sends it to the device.
4. The device receives the request and informs the IoT Agent about the status of the update.

The IoT Agent forwards the response of the device to the Context Broker and updates the Digital Twin. A visual representation of the above workflow is depicted in Figure 11.

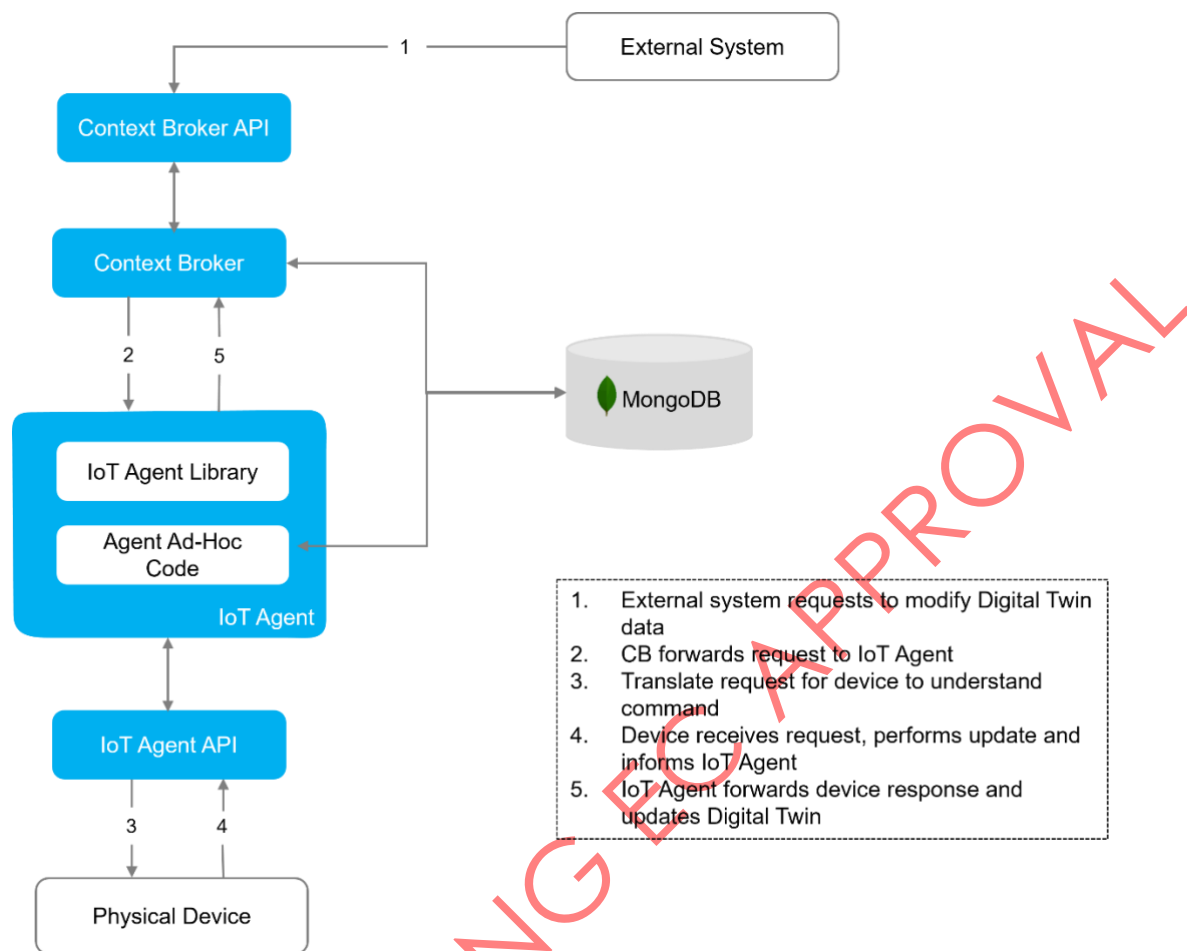


Figure 11: Example of southbound communication with IoT Agent and Context Broker.

Historic Data Registry

The Historic Data Registry takes advantage of the *Subscription* mechanism of the FIWARE Context Broker. Whenever a device sends an update, the Context Broker issues a notification to the subscribed Registry. The Registry exposes different endpoints that an external system or user can take advantage of to access the information.

A basic workflow that better illustrates the connections and interactions with the Historic Data Registry is as follows:

1. The Data Registry should, first, subscribe to the Context Broker, in order to receive notifications whenever a device updates its data. The information from that notification is stored appropriately.
2. In order for the device to be able to send its data, it should first be registered by the Agent to the Context Broker.
3. The device sends, in its native communication protocol, its data to the IoT Agent.
4. The IoT Agent formats the data in a NGSI -compatible request and forwards it to the Context Broker. The Broker provides the updates to the Digital Twin.
5. The subscription mechanism of the Broker is, then, activated and a notification with the updated values of the device is sent to the Data Registry.

Figure 12 describes graphically the steps of the workflow.

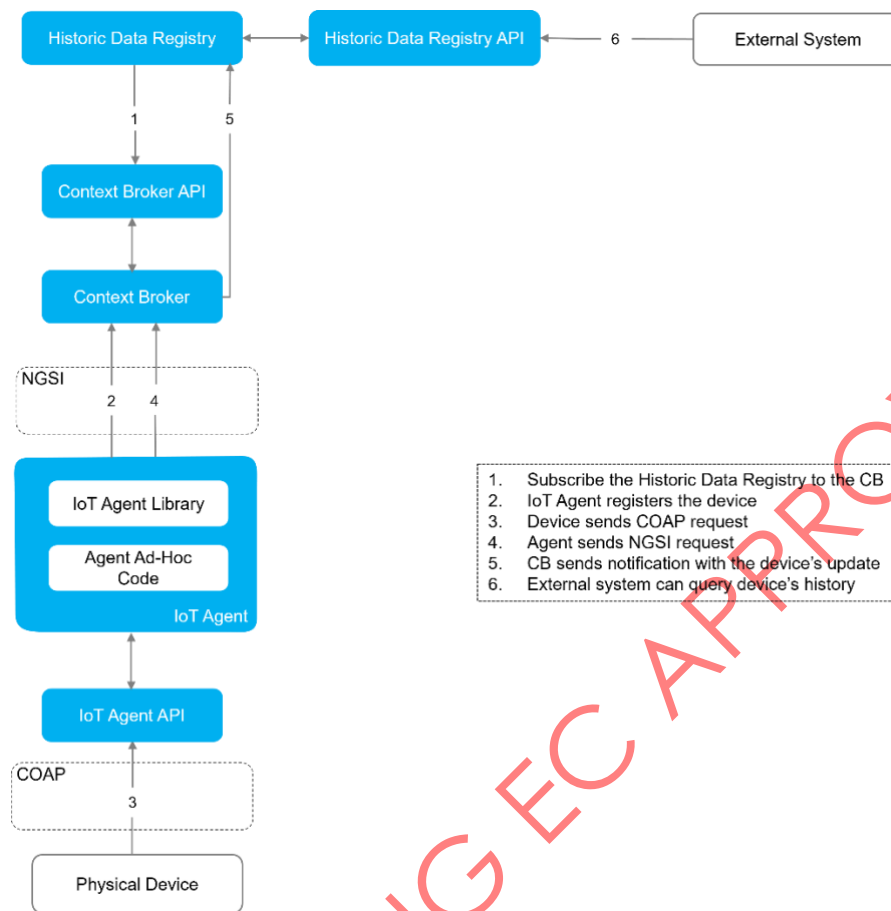


Figure 12: IDI basic workflow overview.

IDI Multi-tenancy

Multitenancy, in general and especially in cloud infrastructures, allows for the usage of a fixed set of resources to be allocated to multiple tenants and ensures the tenants' isolation from one another, meaning that each instance of deployed software gets executed without having knowledge of the activity of the other instances. This allows for a more flexible method of managing the available infrastructure, while serving a large number of requests. Information about FIWARE multitenancy can be found in FIWARE IoT stack documentation [25].

In the case of IDI, multitenancy is mainly used as a means for isolation, authentication and authorization.

Isolation

The Digital Twins of the devices can be stored in sub-databases inside the main MongoDB database and have no access or knowledge outside of their context, that is the database they are stored in.

Authentication & Authorization

Depending on the sensitivity of the data, special restrictions can be put on who can access the data or what permissions are given (read/write permissions) for a set of Digital Twins kept in a sub-database.

The FIWARE components of IDI have this feature embedded in their implementation. In order to support multitenancy across the entirety of IDI, the Historic Data Registry has been designed to mirror the separation occurring in the MongoDB of the FIWARE components.

A graphical representation of the above can be found in Figure 13. Depending on the type of the device (e.g. Drones, Mobile phones, Synfield IoT devices), there are different sub-databases that hold the corresponding Digital Twins.

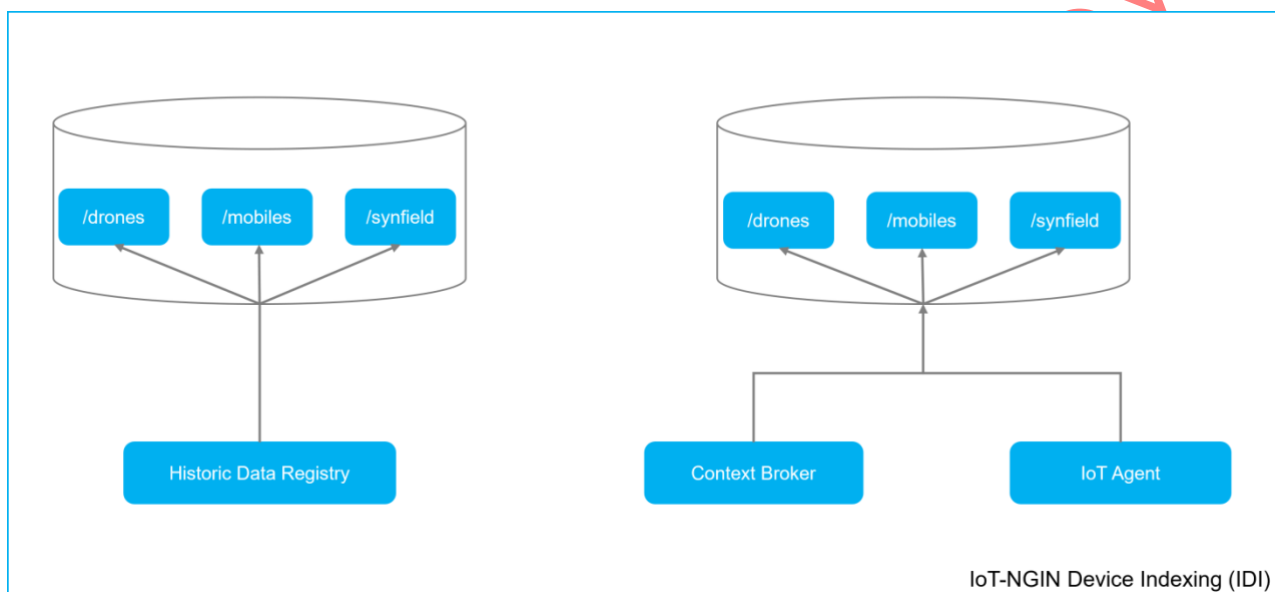


Figure 13: Overview of the IDI multitenancy feature.

Newness and Activeness of Device

The IDI implementation includes an additional component, in order to provide information about the state of the devices. In the context of the IoT-NGIN project, two states were necessary to be described: whether a device has newly joined the network and its activity status.

The newness can be determined as follows: when a device first becomes available, the first step is to register to the Context Broker. The registration process effectively creates the digital representation of the device, which is described by a set of features. The Context Broker keeps track of the dates when these features were created and modified. Taking advantage of this capability, the component checks the creation and modification dates of the features and, if they are identical, the device is acknowledged as 'new'. The moment an update measurement is sent, its state is appropriately altered. As for the activeness of a device, the component checks the last modification date of the Digital Twin's context. In the case a certain period has passed without the device updating its data, it is marked as 'inactive'. The component periodically checks the state of the devices' Digital Twin context and makes the necessary updates.

Therefore, taking this addition into account, Figure 12 could be updated as shown in Figure 14.

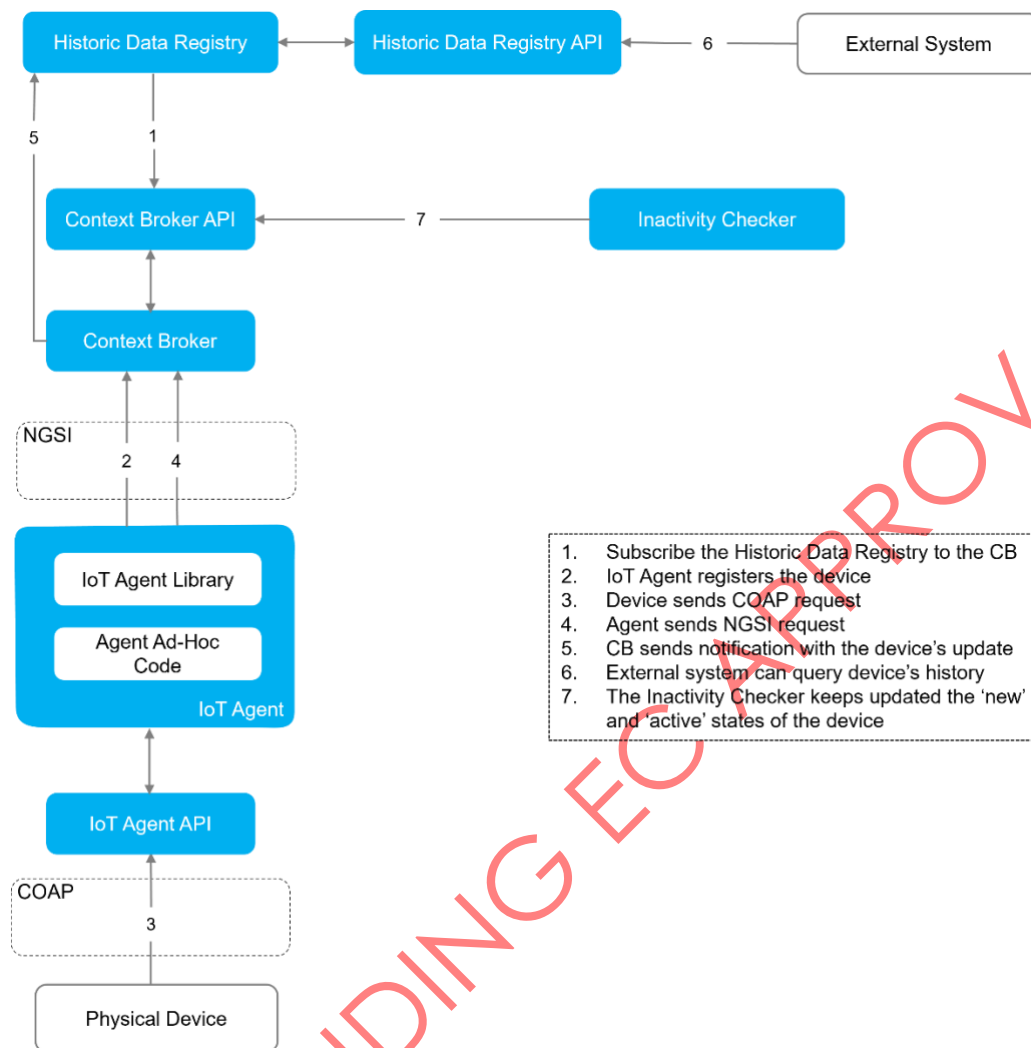


Figure 14: Updated basic IDI workflow.

Step 7 of Figure 10 represents the '*Inactivity Checker*' component interacting with the Context Broker API to update the 'new' and 'active' states in a device's Digital Twin.

While the '*Inactivity Checker*' component is part of the IDI, its inclusion is optional and contingent on the specifics of the use case the IDI is used for.

3.2.2 Interfaces

The interfaces of this component are listed in Annex 1, Table 22.

3.3 IoT Devices access control

Access control is of utmost importance in large IoT systems, with multiple services to be protected, a variety of users and different levels of access. The need for a highly efficient and transparent mechanism that allows multiple methods of authorization and authentication as per use case, is mandatory. The *IoT Devices Access Control (IDAC)* module

of IoT-NGIN has been implemented to handle the access to the resources of the IoT-NGIN framework, in a manner that does not imply the direct involvement of the clients or the devices per se. Through a single gateway URL, multiple services can be exposed in different paths and be protected according to the needs of the application. Users, instead of accessing the services directly, will just now need to make use of the IDAC API.

3.3.1 Description

The IoT Device Access Control module is implemented as a flexible Ingress Gateway enforcing chained access control methods, following different access control mechanisms which are implemented as plugins. The technical design of the IoT Devices Access Control (IDAC) module of IoT-NGIN has been described in D4.2 [2]. The high-level architecture is provided in Figure 15 of this document for convenience.

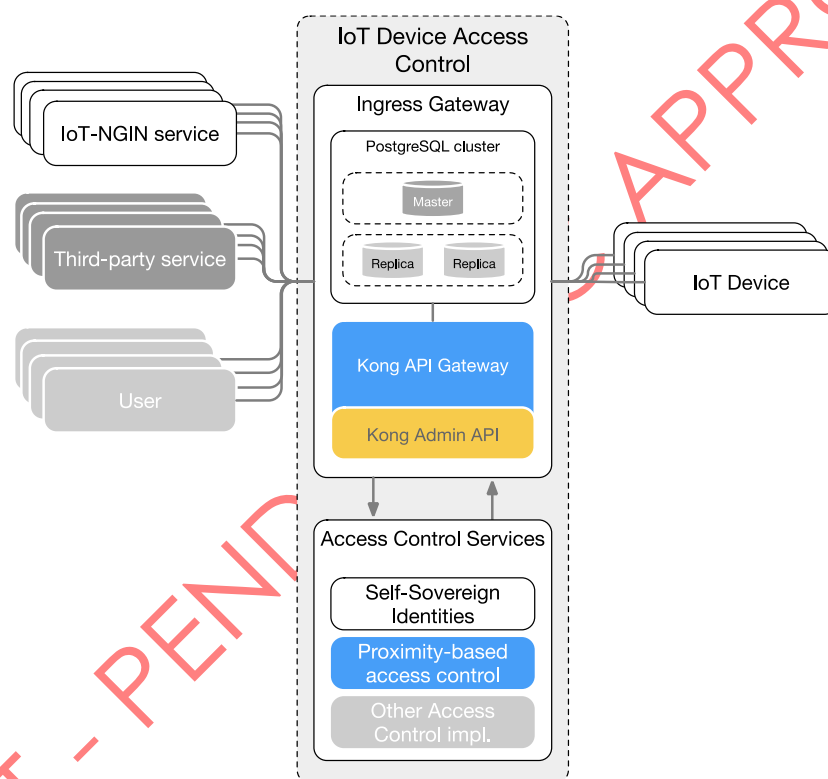


Figure 15: High-level architecture of the Devices Access Control module.

In IoT-NGIN, the Kong [7] open-source API Gateway is employed as an Ingress/API Gateway. Every request performed by an IoT-NGIN or third-party service or user, will be, first, evaluated by the Gateway against a set of authentication and authorization plugins.

Kong has been proved appropriate for the IDAC Gateway implementation, as it is a powerful, cloud-native and language agnostic API, that allows traffic management and transformations, security and governance and observability over services through various plugins. It also allows for the development and deployment of custom plugins according to the needs of each use case. Also, it is backed up by a database, PostgreSQL in the current implementation, that is used mainly to store configuration information about services, routes, consumers and plugins. In order to be able to manage the Gateway in a quick and simple

way, an open-source management tool for Kong is employed, namely Konga [8]. It provides an interface to overview the overall status of Kong and the database and the number of requests reached and handled, as well as other functionalities, as adding services and routes and configuring plugins. A view of the Konga Dashboard is depicted in Figure 16.

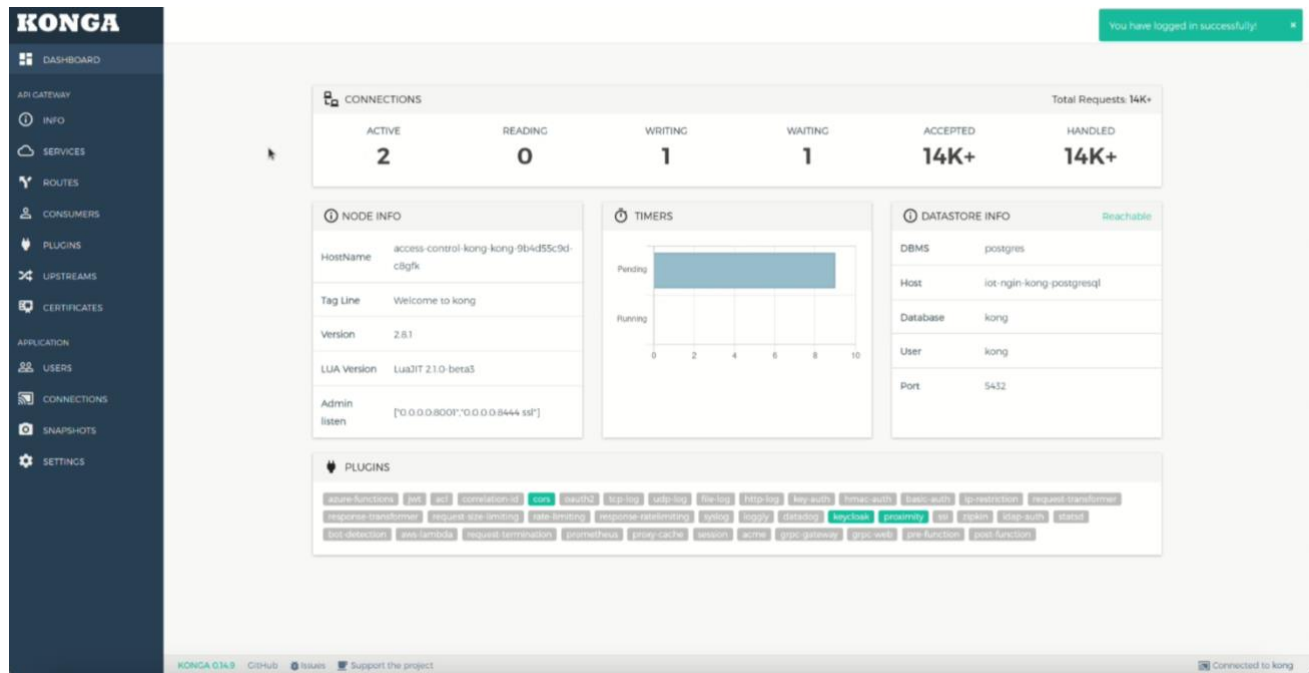


Figure 16: View of the Konga dashboard of the IoT Device Access Control module.

IDAC through Kong supports a number of ways to secure resources, which can be quite application-specific, by using authentication and security plugins. Within the scope of IoT-NGIN, the following custom plugins have been developed in Go [9]:

- Proximity plugin: It implements the ambient intelligence based authorization to IoT devices. In order to support this type of applications, in which the IoT ecosystem comprehends its components and environment, the system often needs to be aware of how close the requester is to the device attempted to be accessed, so that access to remote devices is forbidden. Through this plugin, the proximity of the requester to the device is checked based on an (admin-)user defined maximum allowed threshold. Moreover, the plugin supports checking the device type for each of the supported access requests, before granting access to them. In IoT-NGIN, the enhanced control mechanisms of this plugin have been integrated with the Digital Twin functionality, provided by the IDI component, thus securing access to the Digital Twin of devices registered in IDI. The plugin will then make requests to IDI in order to gain information about the latest location of the device requested, its type, the current location of the user and whether the device is active. First it checks that the type of device requested is allowed for this user and request, and it ensures that the requester's device is active, so that the location is as accurate as possible. Then it calculates their Euclidean distance; access to the resource is granted only if this distance is lower than the configured threshold. The communication flow as described above is depicted in Figure 17, where upstream API refers to the service protected by the IDAC component using the plugin.

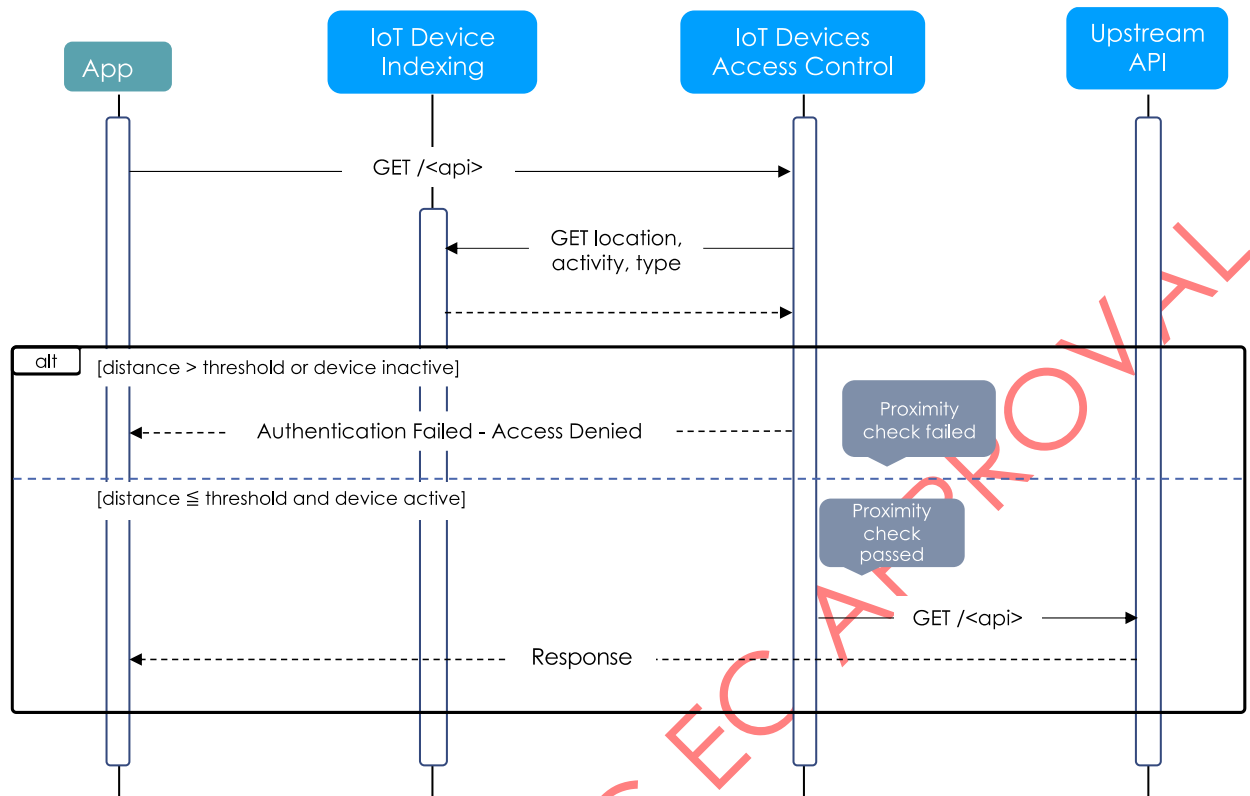


Figure 17: Proximity Plugin Sequence Diagram.

- OpenID Connect Authentication plugin: It allows securing applications and services, based OpenID Connect [10] (an extension to OAuth 2.0). Although Kong provides open-source plugins for basic authentication and OAuth authentication services implemented by Kong, as shown in Figure 20, it does not support an open source plugin for Open ID Connect, offered by a third-party provider. However, this would allow to integrate state-of-the-art solutions for the provision of Authentication Authorization Accounting (AAA) services. This plugin enables this capability, while for the IoT-NGIN purposes, Keycloak [11] has been used as the OpenID Connect Provider; however other providers could be possible, as well. The plugin supports both private and public Keycloak clients, while for its configuration the Open ID Connect *Client ID*, *Client Secret* and *endpoint* are required. In Keycloak protected services, the user must send a valid token from that same client along with the request. The plugin then communicates directly with the Keycloak instance and checks if the token is valid and has not expired in order to authorize access to the user as depicted in Figure 18.

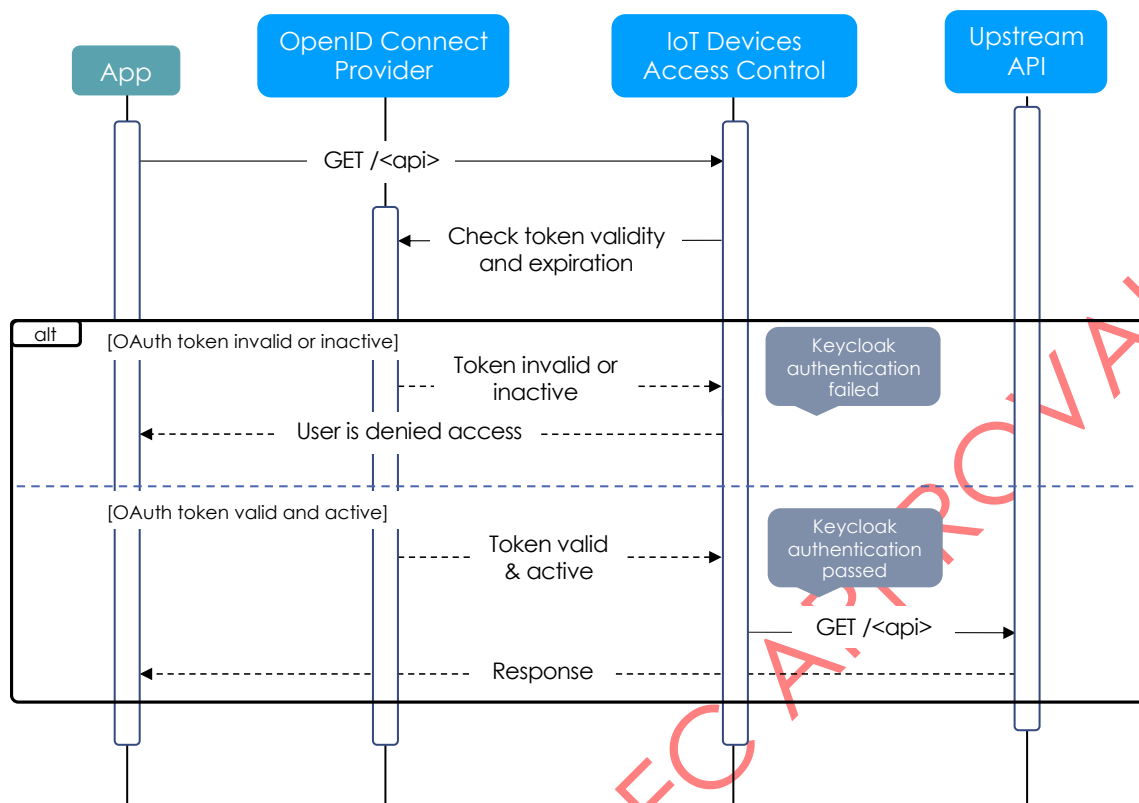


Figure 18: OpenID Connect Plugin Sequence Diagram.

- SSI plugin: It adds protection based on the Privacy Preserving Self Sovereign Identities (SSI) component of IoT-NGIN, presented in D5.3 “Enhancing IoT Data Privacy & Trust” [12]. Users must send two tokens along with the request in the form of headers. The first token is the authorization token and the second is a Demonstrating Proof-of-Possession (DpoP) token. The DpoP is a JSON Web Token (JWT) [30] token that acts as a sender constraining mechanism, needing proof that the sender is the owner of a private key to allow or deny access. A detailed description, as well as the source code, of the IoT-NGIN SSI component can be found on the project's GitLab group [13]. The complete sequence of operations for this plugin can be seen in Figure 19, where SSI refers to the Identity, Authentication, and Authorisation (IAA) Proxy of the SSI Component, which is a proxy server intercepting communications between clients and resource servers and applying authentication and authorization using the tokens described above.

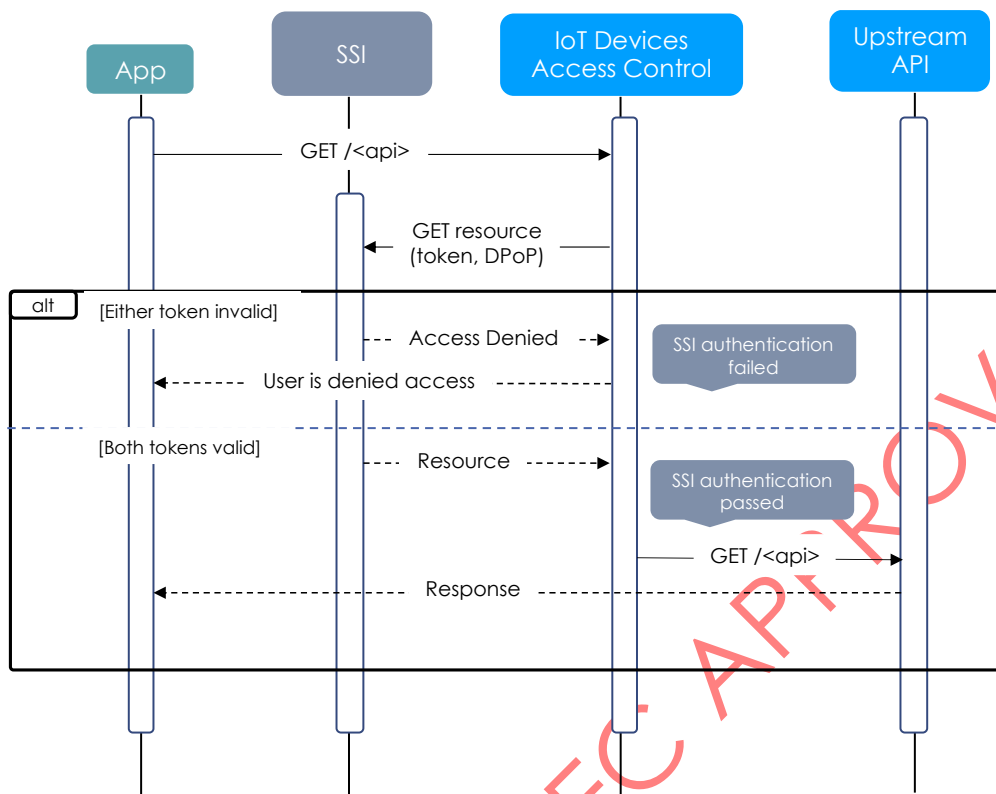


Figure 19: SSI Plugin Sequence Diagram.

Apart from these plugins developed within the context of IoT-NGIN, a number of other plugins for authentication, security, traffic control etc., already supported by Kong can be viewed in Figure 20 and Figure 21.

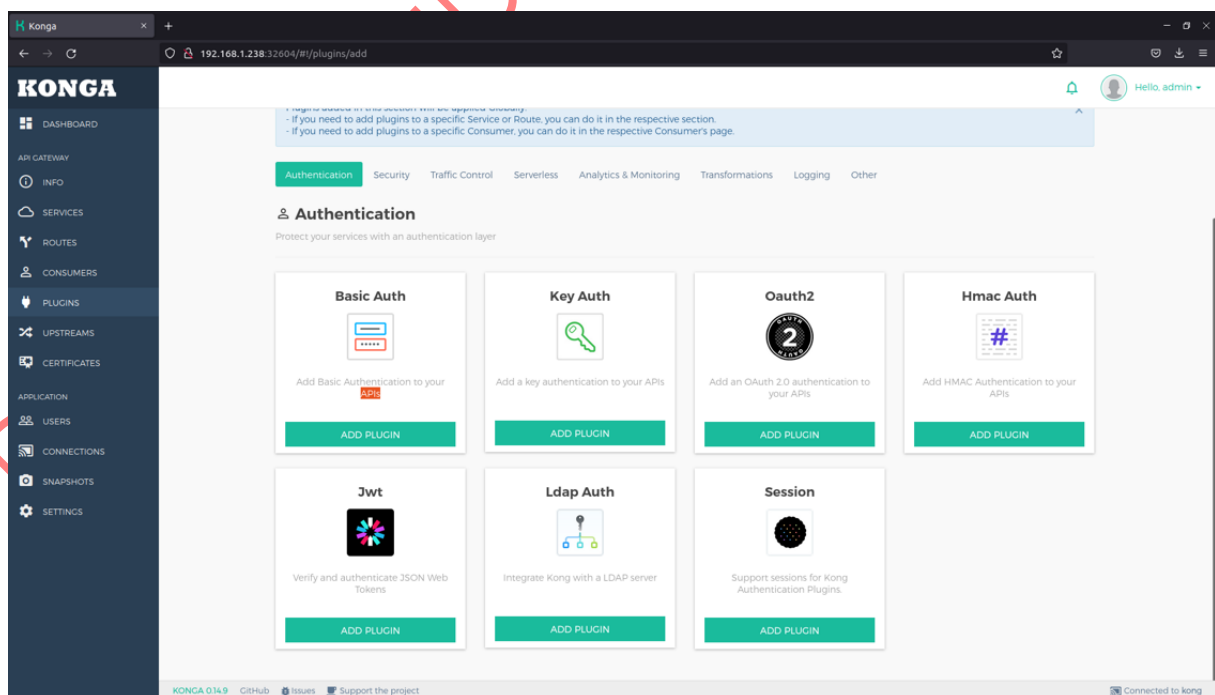


Figure 20: Kong open source plugins for authentication.

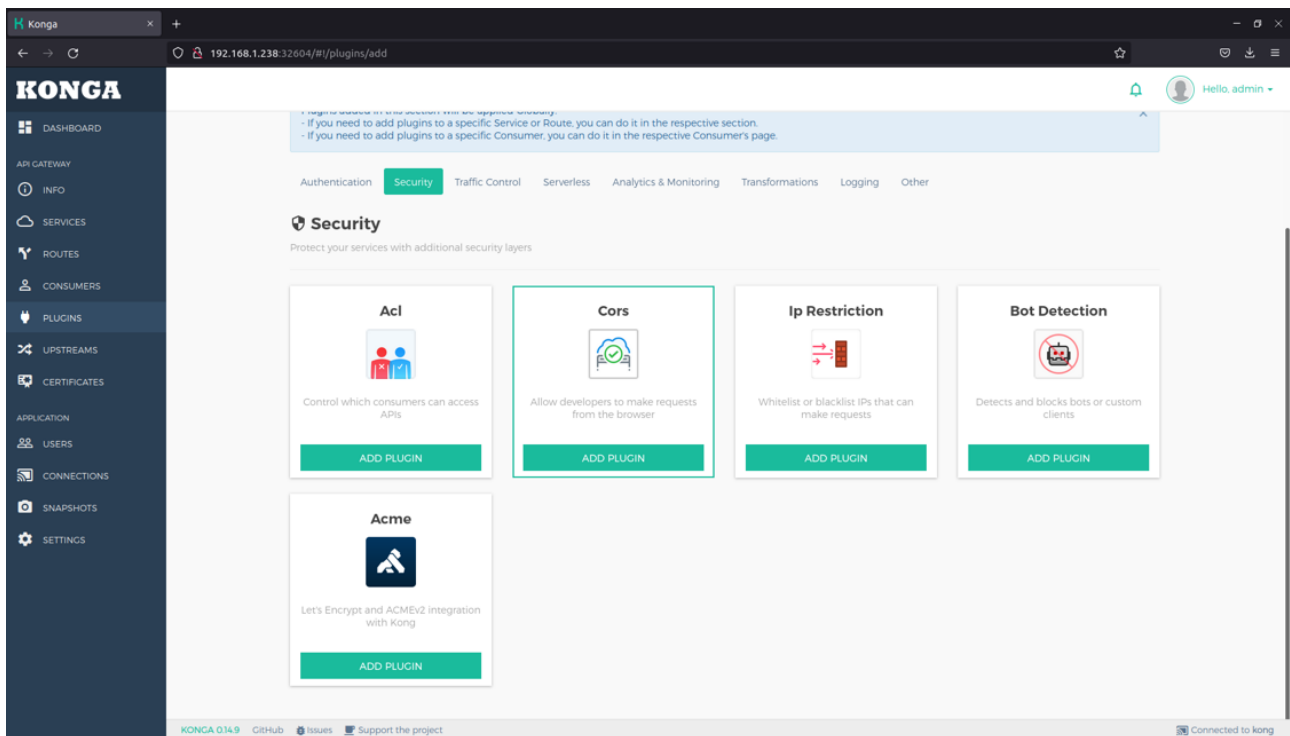


Figure 21: Kong open source plugins for security.

3.3.2 Interfaces

The interfaces of the IDAC component are provided in Annex 1 - Table 23.

3.4 AR/MR module

As discussed in D4.2 [21], Augmented and Mixed Reality (AR and MR) become promising solutions to visualize data from a rich variety of IoT sensors while keeping the focus on the associated real scenarios, and even to interact and actuate such sensors.

Section 2 has provided an overview of the use cases that require the use of AR for IoT interaction: Smart Agriculture IoT (UCs #4); Employee Friendly Industry 4.0 (UCs #6 #7); and Smart Energy (UCs #10). These use cases were described with further details in D4.2.

This section firstly provides a description of the requirements and needs for AR, along with the associated scenarios, development frameworks and devices that will be used in each of the involved use cases. Next, it provides an overview of the interactions between the IoT AR module and other modules of the IoT-NGIN architecture, along with the required interfaces, protocols and messages to be used to accomplish the targeted features / requirements.

3.4.1 Description

Upon having preliminary details about the use cases requiring AR technology, task forces were initiated on:

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

- (i) determining with further details the requirements and needs, and potential restrictions, for each use case in terms of AR presentation and interaction features
- (ii) exploring the required communication, interaction and integration interfaces with other components and modules of the IoT-NGIN architecture
- (iii) exploring the most adequate AR development frameworks and devices (e.g., mobile devices, headsets) to implement and run, respectively, the required AR apps.

Next, the main insights from these task forces to drive the actual development and integration efforts are summarized.

Demonstration Scenarios for AR Use Cases

Table 12: Demonstration scenarios for AR use cases.

UC	Area Size	Internet Requirements	Outdoor / Indoor	AR device static or mobile	Sensors fixed or mobile
#UC4: Farm	~300 ha	Needed. Wireless, but no special bandwidth and latency requirements	Outdoor	User wearing it will be static	Fixed (SynField node)
#UC6: Bosch Factory	Bosch Factory Region		Indoor	User wearing it will move (e.g. walking)	Mobile (sensors to be detected will be moving; e.g. persons, AGVs...)
#UC7: ABB Factory Floor	ABB Factory Region		Indoor	User wearing it will be static	Fixed (cabinet)
#UC10: EV Charging Stations	~5000 m2		Outdoor	User wearing it will be static (or walk at most)	Fixed (charging station)

Sensors to be detected

Table 13: Sensors to be detected via AR/MR module.

UC	Sensor	Quantity	Discovery Method	Access Control	Parameters	Granularity	Actuation
#UC4	SynField node	1	Computer Vision	Yes	- temperature - soil moisture - wind speed - leaf wetness - irrigation prediction	~ every 5 min	Yes (start / stop irrigation)
#UC6	AGV	≤ 2			- Vehicle id		

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

			Computer Vision, VLP, UWB	Yes (but no strict req.)	<ul style="list-style-type: none"> - Position (x, y, z) - Localization Method - Speed - Status - Timestamp 	High granularity, ~ every 1s, or even less	Receive alerts of potential collisions.
	Person	≤ 2			<ul style="list-style-type: none"> - Person id - Position - Speed - Timestamp 		
	Cart (full or empty)	≥ 2			<ul style="list-style-type: none"> - Cart id - Position - Speed - Timestamp - Full or empty 		
#UC7	No sensor, but physical cabinet	2	QR code	Yes (but no strict req.)	<ul style="list-style-type: none"> - 3D model of the cabinet - Optional: Info about the cabinet 	Once. No dynamic info	No
#UC10	EV Charging Station v1	2	Computer Vision (potentially: QR code)	Yes	<ul style="list-style-type: none"> - Model - Manufacturer - Power Output - Connector Type - Connector Availability - Energy Cost 	~ every 5s	Yes (start the charge, and maybe stop it)

AR Display Devices

Table 14: AR Display Devices.

UC	Type	# Devices	Need to be Geopositioned	Need to Provide Image / Video to a Cloud Computer Vision Server	Need to Overlay Info Next / Close to the real object
#UC4	Smartphone	1	Yes (outdoor: GPS available)	Yes	Yes, but no strict requirements on info placement
#UC6	AR headset (Magic Leap, or HoloLens), possibly also smartphone	1	Yes (indoor: AR device tracking, UWB, Computer Vision)	Yes	No strict requirements on info placement. A plant map will provide global situational awareness
#UC7	Smartphone	1	No	QR scanning code	No strict requirements on info placement.
#UC10	Smartphone	1	No (but it would be good to also match the AR device with the EV charging station if being close)	Yes	No strict requirements on info placement.

The use cases and involved partners have no specific impositions / restrictions on the brands / models to be used for the smartphones, so the ones meeting the elicited requirements to the greatest extent will be selected. Regarding AR headsets, preliminary tests are being conducted with Magic Leap [34] and Microsoft HoloLens [35] devices. Both seem to be appropriate decisions, but the one achieving best performance will be used in the pilots.

Relevant requirements for selecting the development AR framework

Table 15: Relevant requirements for selecting the development AR framework.

UC	Any Restriction / Special Feature Desired?	Preference (e.g., for interoperability with other developments)
#UC4	<ul style="list-style-type: none"> - Need to take and send pictures of the SynField node - Need to display the information from the node, ideally next/close to it 	- Slight preference on web-based technologies, but priority on features being provided in an accurate and robust manner.
#UC6	<ul style="list-style-type: none"> - Need to analyze pictures / video of the captured environment - Need to track the captured environment, possibly with the help of visual markers, and to obtain the device position / orientation. - Need to have image tracking capabilities - Need to load 2D/3D models of the factory floor, with overlaid info about the detected sensors - Need to be compliant with Magic Leap / HoloLens headsets 	- No restrictions. No preferences. Priority is to provide an accurate and robust support for the required features.
#UC7	<ul style="list-style-type: none"> - Need to scan QR codes - Need to retrieve / load 3D CAD models of the target cabinet being scanned 	
#UC10	<ul style="list-style-type: none"> - Need to scan QR codes - Need to provide pictures of the captured EV charging station 	

As there are neither preferences nor restrictions on the selection of the AR development framework, the use of the Vuforia engine for Unity [36] has been selected as a common framework to support all use cases requiring AR features. On the one hand, and most importantly, it will allow providing efficient support for all required features from all use cases. On the other hand, it will allow sharing development efforts between the involved partners and re-usability for any interested stakeholder or third-party agent applying to Open Calls.

3.4.2 Interfaces

As discussed in D4.2 and earlier in this deliverable, the IoT AR module need to interact with other modules of the IoT-NGIN platform to provide their targeted features. The next figure provides high-level sequence diagrams for different sensor data update and presentation cases that are meant to encompass all potential situations (or at least similar cases) required by the AR use cases:

- (i) Periodic or even-driven update of the data from/about sensors, thanks to Machine Learning analysis (WP3) and/or by status updates from the sensors themselves

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

- (ii) Discovery of sensors / elements via non-visual methods and associated presentation of their data via an AR interface
- (iii) (User-transparent or User-Driven) Discovery of sensors / elements via visual methods and associated presentation of their data via an AR interface
- (iv) (User-transparent or User-Driven) Discovery of sensors / elements within pre-defined distance and associated presentation of their data via an AR interface
- (v) Sensor actuation via an AR interface and associated presentation of the updated data via the same AR interface.

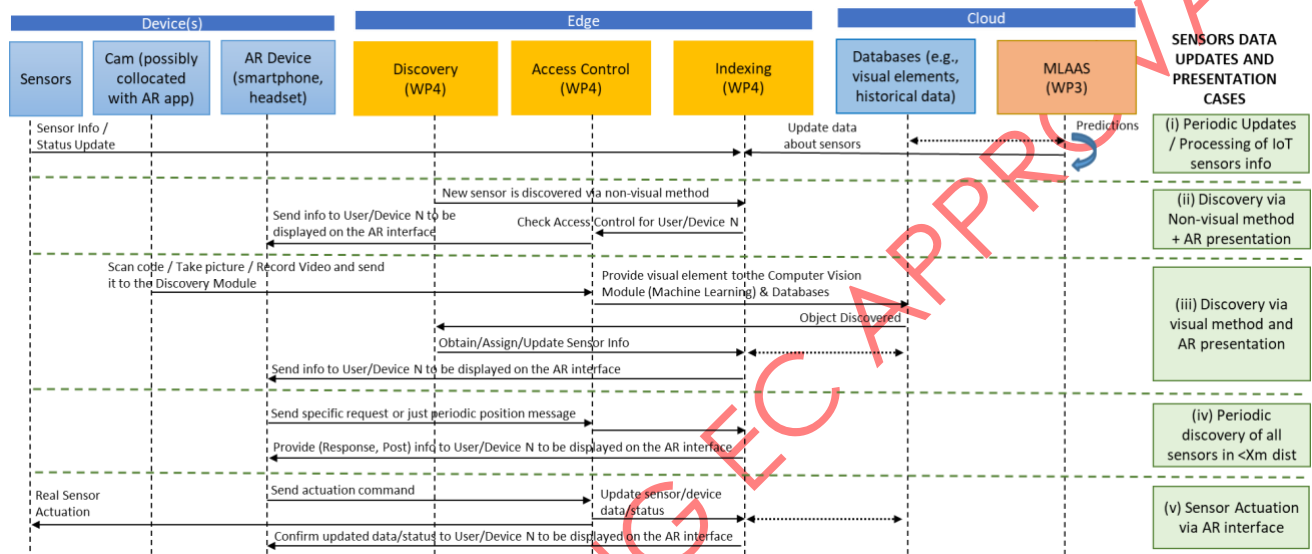


Figure 22. High-level sequence diagrams for AR presentation and actuation cases.

Table 24 in Annex 1 provides (tentative) details about the required interfaces with other IoT-NGIN components, along with the associated protocols, messages and fields / info to be exchanged.

4 Implementation for the IoT-NGIN Living Labs

Based on the technical specifications and design of the Aml tools provided in Section 3, in this section the use of these tools in the scope of the LL UCs is analyzed. Use case specific implementations are provided, where relevant, as in the case of the device discovery counterparts, including analysis of ML models developed, as for the computer vision related tasks employed for image recognition. Instantiations of the device indexing and access control components are also described in detail for use case scenarios, providing technical insights for instantiation in similar scenarios in other application domains. Moreover, use case specific preparatory activities are analyzed for the AR/MR module, anticipating its development in the foreseen application areas.

4.1 IoT Device Discovery

The IoT Device Discovery is being implemented in four use cases of three different Living Labs of the project. Each of the use cases uses one or several discovery methods as it is indicated in Table 16. The table also shows if the tracking of the objects is needed in the use case.

Table 16: Discovery methods implemented in each use case

Living Lab	Use Case	Discovery method	Tracking
Smart Agriculture	UC#4 Crop diseases prediction, smart irrigation and precision aerial spraying	Computer Vision Code Scanning	No
	UC#5 Sensor aided crop harvesting	Computer Vision Code Scanning	No
Industry 4.0	UC#6 Human-centred safety in a self-aware indoor factory environment	Ultra Wide Band Visual Light Positioning Computer Vision	Yes
Smart Energy	UC#10 Driver-friendly dispatchable EV charging	Computer Vision Code Scanning	No

The status of the implementation of the IoT Device Discovery in each living lab is described in the following sections. It can be more or less advanced depending on the living lab since the final implementation is due by the end of the WP4. These final results will be reported in deliverable D4.4 Moreover, it has to be noted that code scanning does not require special implementation or adaptation per use case, so it is not further analyzed in the following subsections.

4.1.1 Smart Agriculture

General Info / Requirements / Installation

The main objective here is to develop an AI-based object recognition model able to detect the Synelaxis SynField IoT devices in photos taken by mobile devices in the field. This will be the first step towards indexing of the SynField devices on the overall network and eventually the Smart Agriculture LL.

The methodology and framework followed to build the model is described below. Using python's library 'tensorflow', a training job by using the pretrained model 'SSD ResNet50 V1 FPN 640x640' [37]. The architecture of the model is shown in Figure 23.

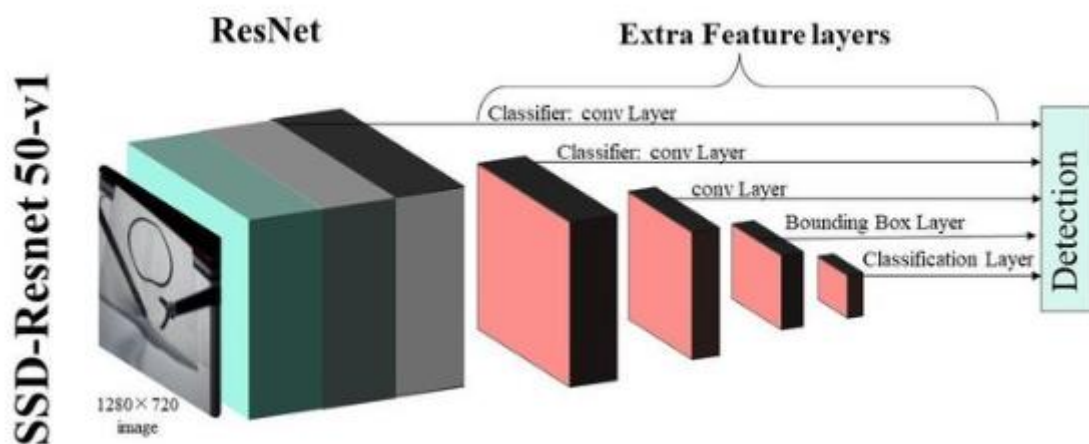


Figure 23: Model Architecture [38].

The output of the model is the detection of the required object [Synelaxis SynField device]. The whole process of setting up the model, the required APIs, how the input photos were labelled and how we carried out the training process is detailed below.

The model has been built in python and requires the following technologies:

- Python 3.9+
- Anaconda

Installation Setup

The initial setup was split into 6 sections:

1. Workspace & Training Files Organisation
2. Preparation/annotation of image dataset
3. TF* records generation
4. Training Pipeline configuration
5. Training and monitoring the model
6. Export resulting model object

Workspace structure

The structure of our workspace is defined by the directory tree in Figure 24.

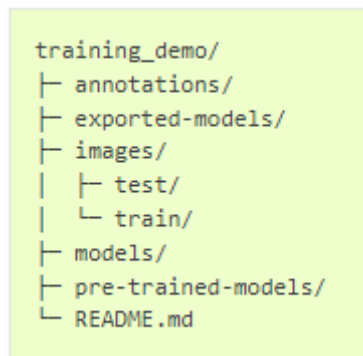


Figure 24: Directory Tree.

Each of the folders/file shown in the above tree is explained in the following.

- **annotations:** This folder will be used to store all *.csv files and the respective TensorFlow *.record files, which contain the list of annotations for our dataset images.
- **exported-models:** This folder will be used to store exported versions of our trained model(s).
- **images:** This folder contains a copy of all the images in our dataset, as well as the respective *.xml files produced for each one, once labelling is used to annotate objects.
- **images/train:** This folder contains a copy of all images, and the respective *.xml files, which will be used to train our model.
- **images/test:** This folder contains a copy of all images, and the respective *.xml files, which will be used to test our model.
- **models:** This folder will contain a sub-folder for each of training job. Each subfolder will contain the training pipeline configuration file *.config, as well as all files generated during the training and evaluation of our model.
- **pre-trained-models:** This folder will contain the downloaded pre-trained models, which shall be used as a starting checkpoint for our training jobs.
- **README.md:** This is an optional file which provides some general information regarding the training conditions of our model. It is not used by TensorFlow in any way, but it generally helps when you have a few training folders and/or you are revisiting a trained model after some time.

Model Description

The model has been built in order to detect the object of interest (SynField device). The model has been trained by a series of photos as well as two videos capturing the object from different angles. The input of the model is a photo of the area where a SynField device resides. Figure 25 shows a few samples of our training set.



Figure 25: Samples of the training set.

The available dataset includes 116 photos and two videos which were processed to give 5 frames per second. The two videos have a total duration of 63 seconds. Cutting them in 5 frames per second, we ended up with a total of 332 additional photos.

The dataset was split into three sets. A training set, a testing set and a validation set. The ratio for creating the above data sets was set at 80:10:10.

Dataset Preparation/TF Records Creation

The first step has been to prepare the dataset, annotating the photos via bounding boxes indicating the SynField devices. In order to perform the labelling process, the Labelling [39] tool has been used. Figure 26 shows the interface of the application.



Figure 26: Labelling Interface.

We built an XML version of the photographs by labelling them. The photos were divided into training and test sets. A label map was required by Tensorflow, which translates each of the utilized labels to an integer value. Both the training and detection methods make advantage of this. The generated file was named 'label_map.pbtxt' and has the following format:

```

item {
  id: 1
  name: 'The_Box'
}

```

Figure 27: The label of the object.

After the photos were annotated and defined the datasets into training and testing subsets, we converted those annotations into TFRecord format.

Model Selection

Instead of obtaining features from each RoI, which is cropped and rescaled every time in R-CNN, Fast R-CNN analyzes the network and extracts features for the entire picture once. It then employs RoI pooling, a variant of SPPNet's pyramid pooling, to generate a feature vector of the appropriate length. The categorization and localisation of this feature vector follows. Because the calculations for overlapping regions are pooled, this approach is more successful than R-CNN.

The actual architecture of the fast R-CNN includes four building blocks:

1. Region proposal network
2. Feature extraction using CNN
3. RoI pooling layer
4. Classification & Localization

Figure 28 below shows the above steps.

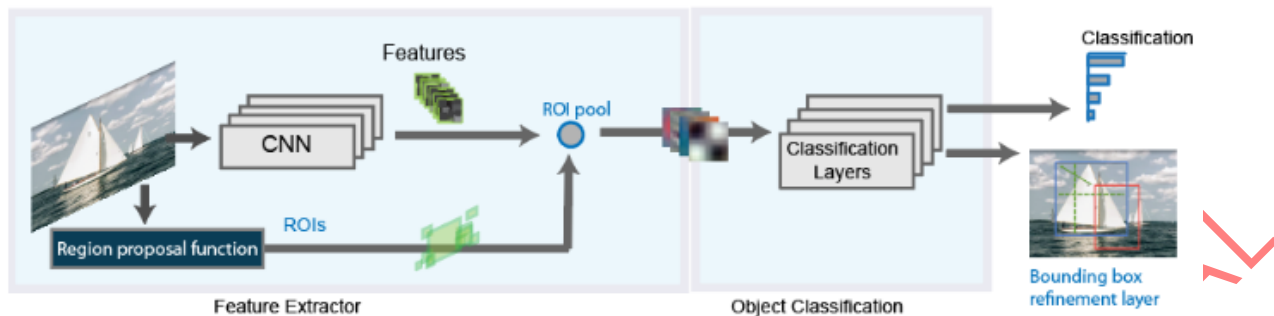


Figure 28: The architecture of Fast R-CNN

R-Region CNN's proposal network and Feature extraction modules are quite similar. Instead of putting each cropped and rescaled RoI through a feature extractor like VGG-16 to build a convolutional feature map, the entire input picture is sent through one. In the RoI pooling layer, the features (i.e. convolutional feature map) are coupled with a region proposal network that employs an SS method to generate a fixed-length feature vector. The classification and localization modules receive each of these feature vectors. The classification module uses a softmax probability to classify $K+1$ (1 for background) item types. For each of the K object classes, the localization module generates four real-valued integers [40].

Model Training

The model used has not been developed from scratch; instead, we used one of Tensorflow's pre-trained models, *SSD-Resnet 50-v1*. This has been saved in the pre-trained-models folder created.

Some changes at the configuration file *pipeline.config* have been performed. Changes include:

- setting the number of the labelling classes (line 3). For our case it is 3
- setting the batch size (line 131): We put 8. This varies based on your system
- `fine_tune_checkpoint` (line 161):
 - `pre-trained models/ SSD-Resnet 50-v1/checkpoint/ckpt-0`
- `fine_tune_checkpoint_type` (167): Set to 'detection'
- `use_bfloat16`: false (line 168). Set this to false if you are not training on a TPU
- `label_map_path` (lines 172, 182): "annotations/label_map.pbtxt"
- `input_path` (line 174): "annotations/train.record"
- `input_path` (line 186): "annotations/test.record"

Once the training started, we have monitored its progress through Tensorboard. The model was successfully trained and evaluated giving an overall performance of ~94%. The output is the photo including the prediction of the model as shown in Figure 32.

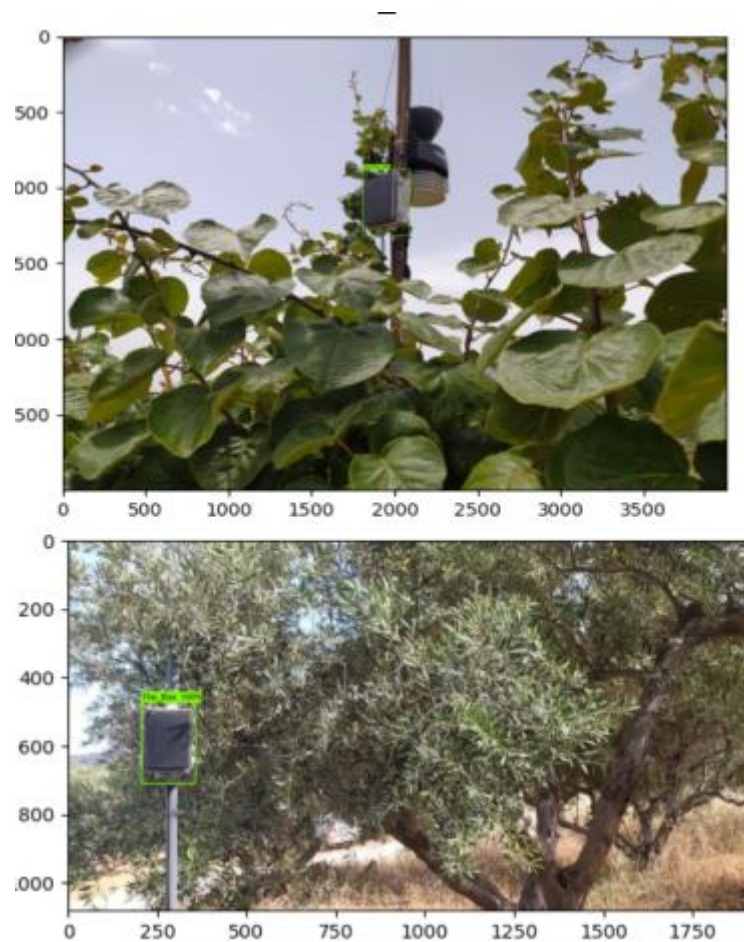


Figure 29: Model results samples.

Prediction times (0.88 seconds average), loss classification, loss localization, loss regularization and total loss for the algorithm are shown in Figure 30 and Figure 31 below.

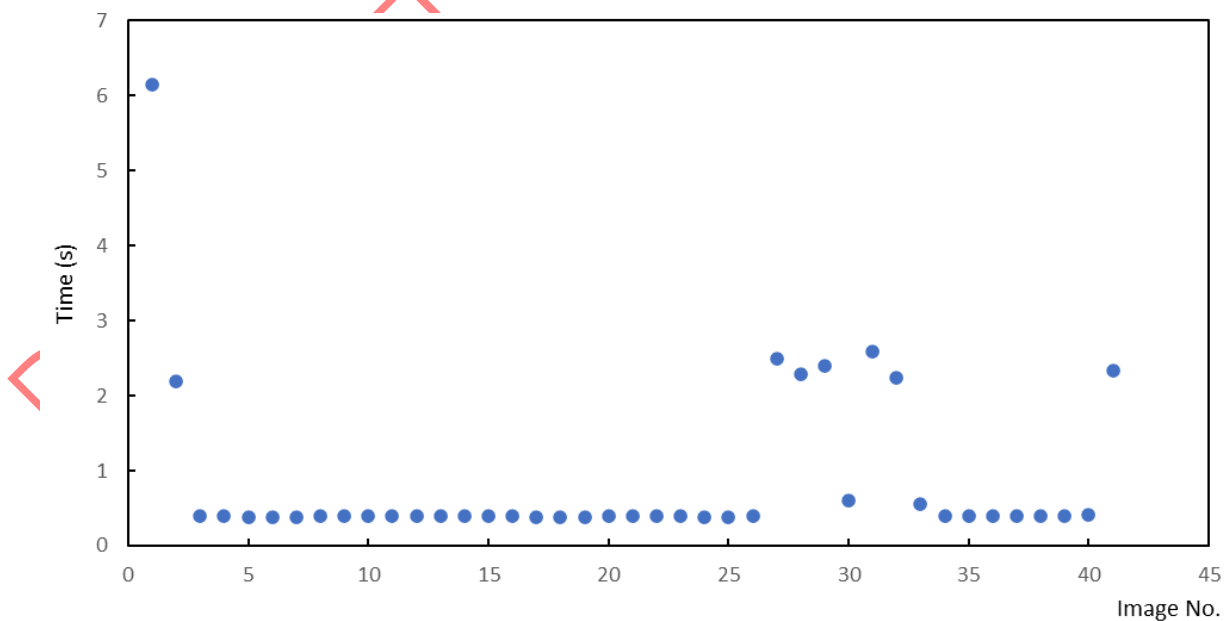


Figure 30: Prediction times.

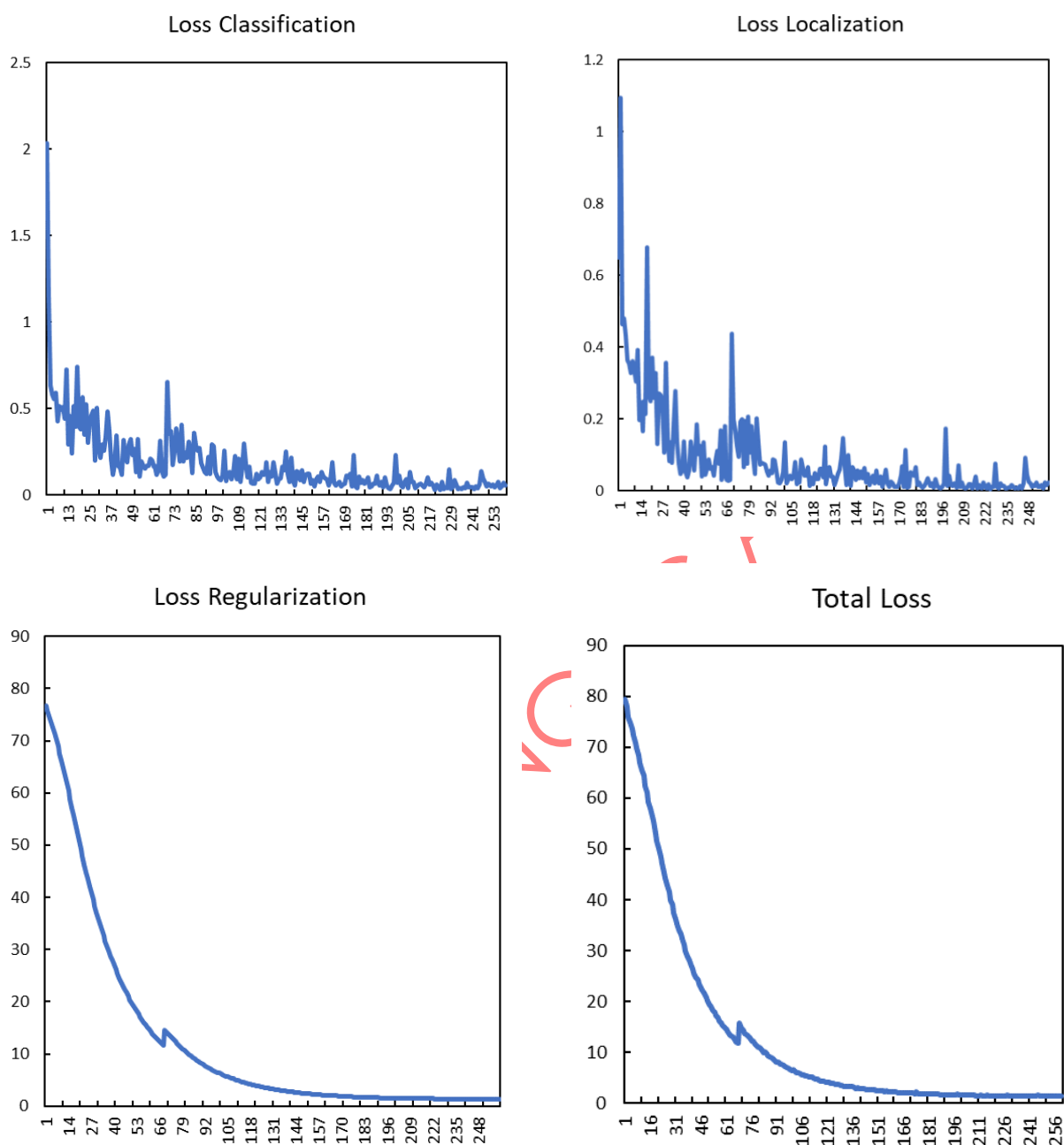


Figure 31: Loss characteristics.

4.1.2 Industry 4.0

As already explained in D4.2, the following IoT Device Discovery methods will be applied in the Industry 4.0 LL:

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

- VLP-based positioning. To be tested in specific areas for AGV devices in order to validate the feasibility of the solution in terms of positioning accuracy and tracking capability compared to other methods.
- UWB-based positioning solution for precise positioning of AGV devices.
- Computer-vision solutions to provide wide-scope positioning capabilities and the possibility to identify and track different kind of objects/people in the industry area.

For all the cases, it will be necessary to deploy specific hardware devices in the scenario. Also, an edge server will be necessary with an instantiation of the following modules or services linked to the IoT Device Discovery features:

- UWB solution Edge service. Used for the configuration and setup of the system. Does not require any specific configuration/instantiation/implementation.
- Computer Vision service.

Some test of the UWB and Computer vision methods of the IoT Device Discovery module have already been performed in the Industry 4.0 Living Lab. The preliminary results that were obtained are illustrated in Figure 32.

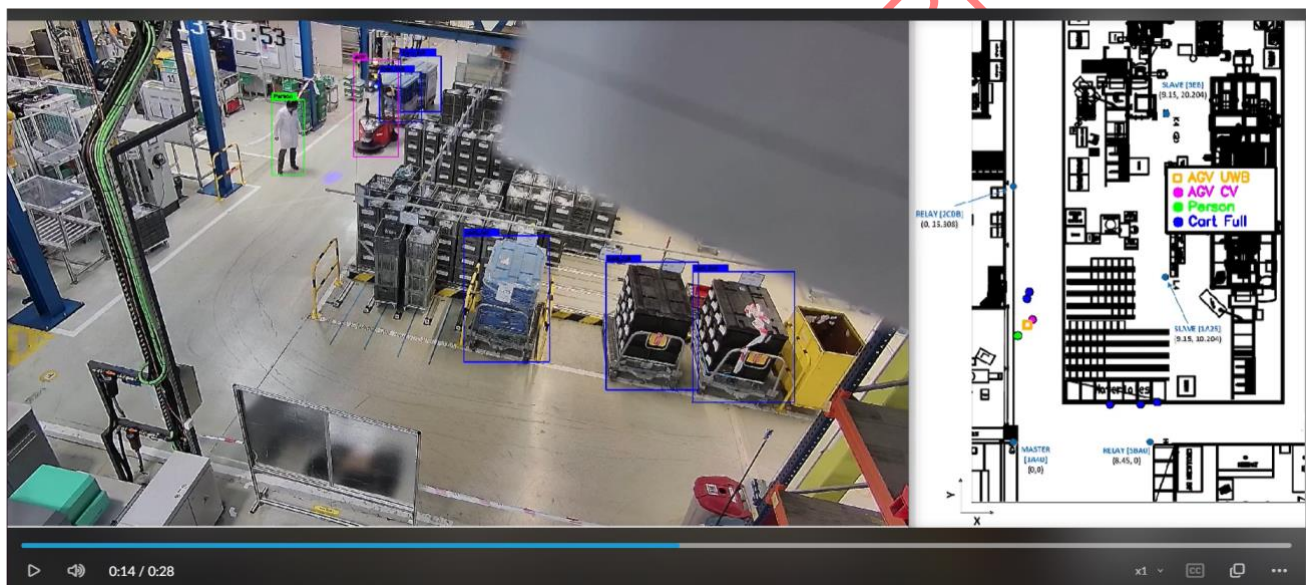


Figure 32: IoT Device discovery preliminary results in the Industry 4.0 Living Lab.

The figure is a snapshot of a video that shows the detection and positioning of some objects by Computer Vision and Ultra Wide Band. The image on the left side is a video frame of the factory with the distortions of the camera corrected. On this image, it is also possible to see some color boxes surrounding the objects detected by Computer Vision. The blue boxes mark the objects identified as "Cart Full", the green boxes are the objects of type "Person", and the purple boxes are the "AGV". The image on the right side shows a map of the factory with the positions of the detected objects. The objects detected by Computer Vision are marked with circles following the same color code as in the left image. It also shows the position of the AGV detected by Ultra Wide Band, which is marked with an orange square. It can be seen in the image that the positioning of the CV (purple circle) and the UWB (orange square) are almost coincident on the map. Finally, it is also shown on the map the position of the 5 UWB anchors deployed in the factory (blue circles).

4.1.3 Smart Energy

In the Smart Energy Grid Active Monitoring/Control Living Lab, UC#10 “Driver-friendly dispatchable EV charging” is considered for WP4 demonstration activities. In the UC#10, electric mobility plays the role of a flexibility provider for the stabilization of the electricity grid heavily penetrated by distributed renewable energy plants. In this scenario, the charging station is the IoT device to be recognized and to be enabled for the interaction using AR technology; the electric vehicle deployed in the Italian living lab, equipped with a camera, are being used to collect images of the charging stations at different times of the day and in different time conditions, so as to be able to guarantee learning by artificial intelligence system and obtain a reliable discovery and recognition mechanism.



Figure 33: Smart Energy charging station screenshots taken from the videos made by EV cameras.

4.2 IoT Device Indexing

The IoT Device Indexing (IDI) is an essential part of the IoT-NGIN project. It could be described as a middleware between the end users or applications that wish to communicate with a device to obtain its data. Instead of coming in direct contact with it, it is more efficient and secure to query that device's Digital Twin.

The efficiency stems from the fact that it becomes significantly easier to both keep track of the latest updates and have a stored record of all devices' modifications. The security is guaranteed in the physical as well as the digital layer: the user/application does not come in direct contact with the device; therefore, it cannot easily act maliciously and the Digital Twin, which represents the device, is protected with the use of the *IoT Device Access Control* component.

The IDI component can be easily set up and integrated for any type of device or scenario. In this subsection, the instantiation for a use-case scenario under the scope of the Smart Agriculture Living Lab UC4, is presented in detail. However, the use of the component in other use cases is similar, so further analysis is not included for other use cases. The scenario involves

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

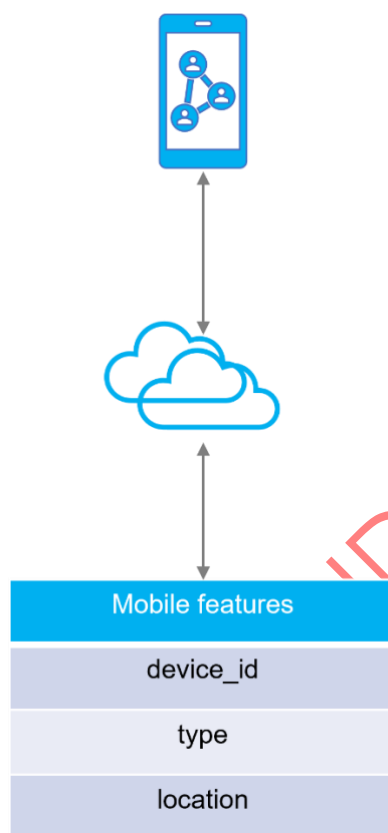
two devices: one of type *Mobile* and one of type *Synfield* (equipment used in Smart Agriculture).

First, for each device type a *Service* in IoT Agent must be created in order to have the devices authenticate themselves when sending measurements.

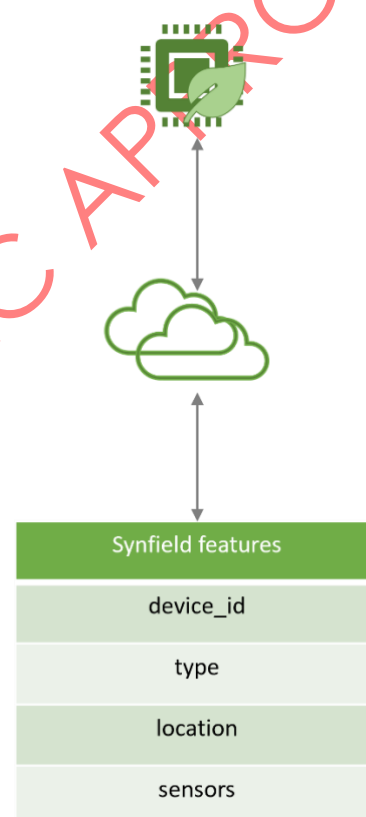
Next, the IoT Agent needs to register the devices to the Context Broker. Registering effectively means issuing the command to create the digital representation of the device. A device in its Digital Twin format can be described by a set of features.

A mobile Digital Twin in the specific use case has the following features (Figure 34-a):

1. A unique id
2. The device type and
3. Location coordinates



(a) For the Mobile device



(b) For the SynField device

Figure 34: Mobile device and Digital Twin relation.

Similarly, a Synfield Digital Twin can be described with the following features (Figure 34-b):

1. A unique id (*device_id* field in the figure)
2. The device type (*type* field in the figure)
3. Location coordinates (*location* field in the figure)
4. The measurements from its sensors (*sensors* field in the figure)

Now the devices can send their measurements through the IoT Agent, which will relay the update to the Context Broker in order to update the Digital Twin's context. This update

triggers the subscription mechanism of the Broker and a notification with the updated context is sent to the Historic Data Registry. The Registry appropriately stores the notification. Every time a new update occurs, a new record will be created.

An external system (user/application) can either query the API of the Context Broker or the Data Registry, depending on their needs.

An external component (*Inactivity Checker*) works in parallel to keep up-to-date the 'new' and 'active' statuses of the devices.

The above use case is graphically represented in Figure 35 for the mobile device, while the process is similar for other types of devices.

An important feature of IDI is its ability to support multi-tenancy. The FIWARE components use two headers in their HTTP requests to indicate the database the Digital Twin should be stored in. Only by applying those headers can an external system yield the requested information. The Historic Data Registry considers these headers as to mirror the separation occurring in the Context Broker and support uniformity across the IDI module.

In the use case, the Mobile and Synfield devices use the headers described in Table 17, while the flow of operations to integrate them is described in Figure 35.

Table 17: Multitenancy for use case #4.

	Fiware-Service	Fiware-ServicePath
Mobile Device	mobile	/
Synfield Device	synfield	/

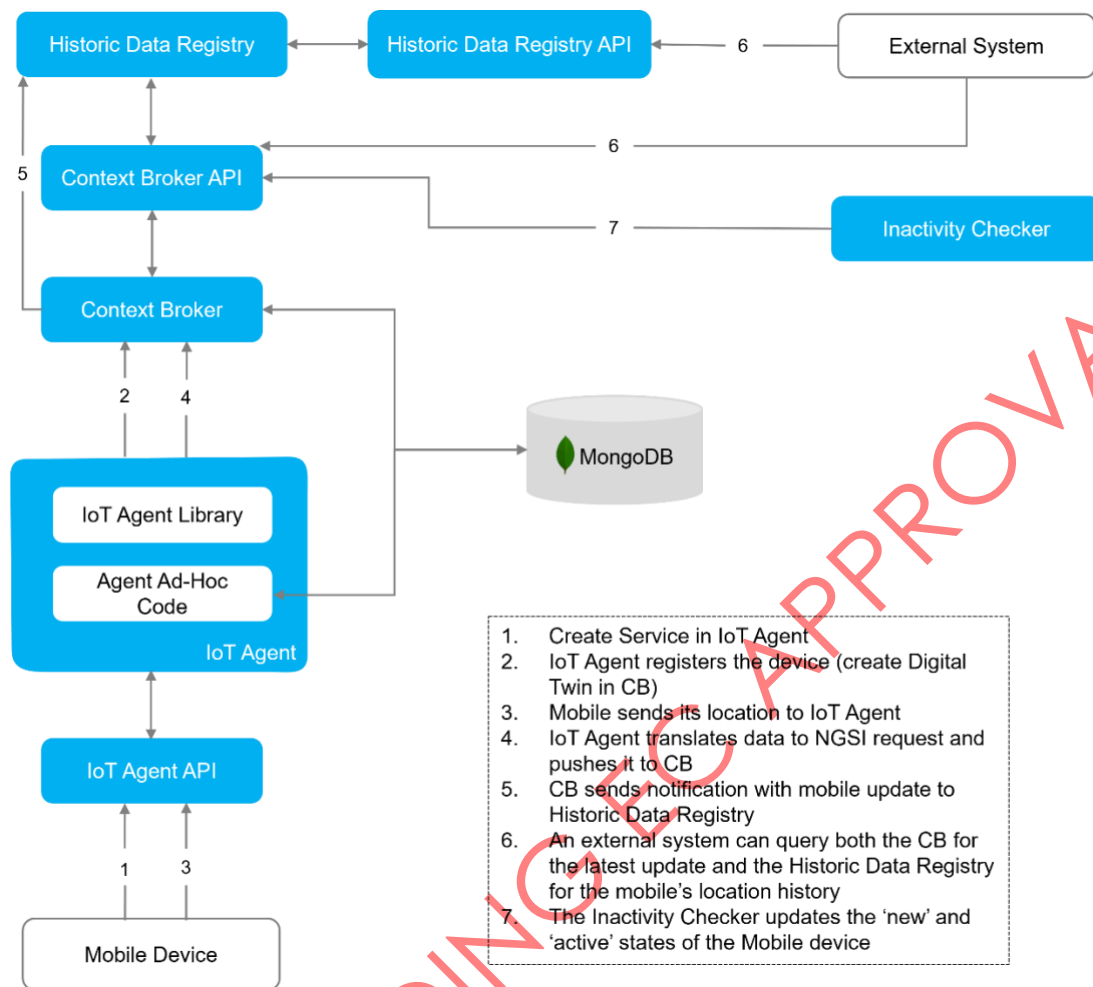


Figure 35: IDI use case for Mobile Device.

4.3 IoT Devices access control

Access Control lies at the heart of WP4. Its role of authorizing, authenticating and protecting services means it can be easily integrated with the other components of WP4 or it can even be used as a standalone tool.

As several access control mechanisms can be quite generic to cover diverse use cases, like the ones covered by the OpenID Connect Authentication plugin, no special implementation is needed for those across use cases. However, additional access control rules can be use case specific, like in the proximity plugin, which in that case requires relevant configuration. In IoT-NGIN, proximity constraints are foreseen for the Smart Agriculture LL UC4, as well as for the Smart Energy LL UC10. In this subsection, the instantiation of IDAC services under the scope of these Use Cases is presented. The sequence of operations related to OpenID Connect and proximity-based authorization is depicted in Figure 36.

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

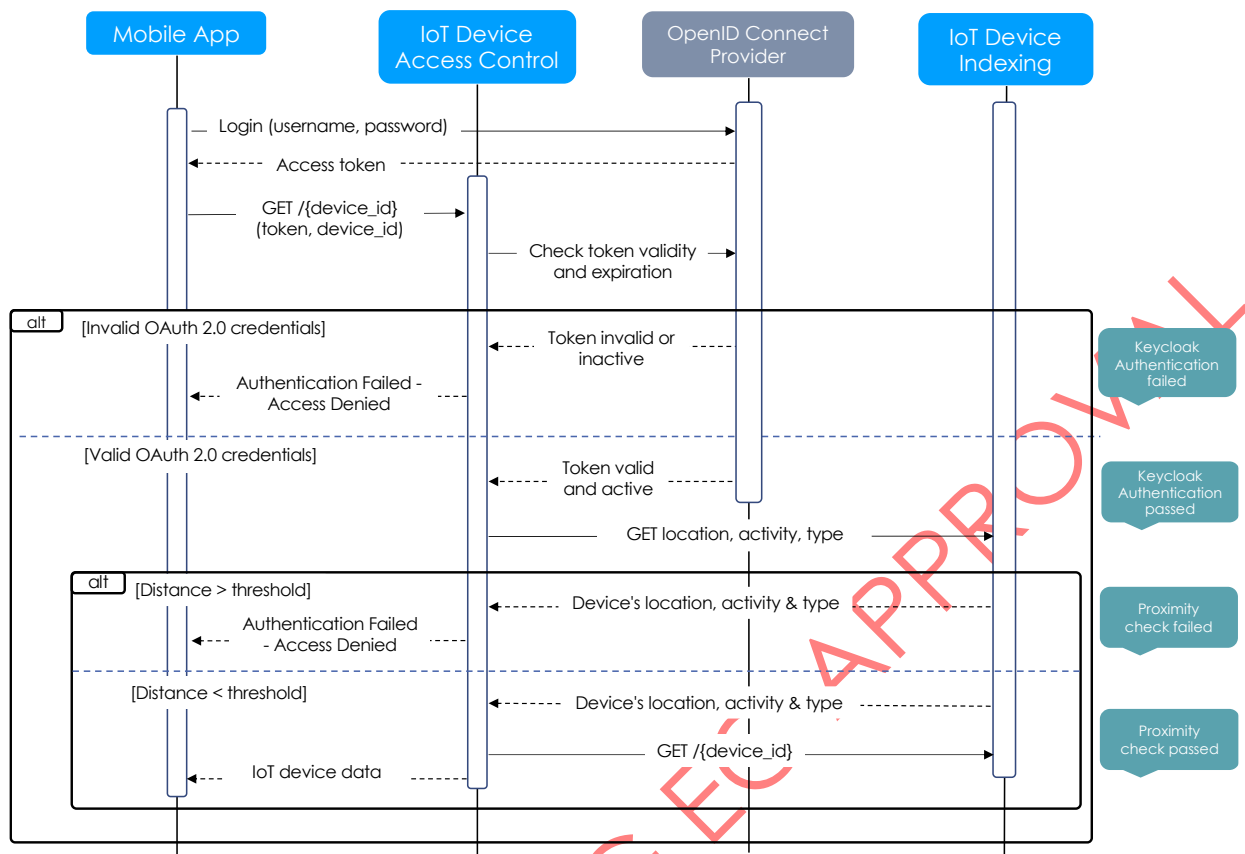


Figure 36: Sequence Diagram for Access Control in accessing IoT data, applicable in Smart Agriculture LL UC4 and Smart Energy LL UC10.

For both UC4 and UC10, we consider an IoT device registered in the IDI component and a user trying to gain access to this device through a smart phone application. The IDI API is being protected by the IDAC module. The API has been configured as a Service in Kong, by adding the API's endpoint. In order to expose this service to the users, a route has been added. The route specifies a path to the Kong's Proxy URL where requests to the service are going to be handled, and it also specifies the methods and the protocols to be accepted. A two-layer protection has been configured for the service through custom plugins, namely the Keycloak and Proximity plugins [See section 3.3]. The Keycloak plugin has been configured to work with a Keycloak client responsible for the user authentication, while the Proximity plugin has been configured with the device indexing endpoint needed for the inner workings of the plugin, a threshold for the maximum distance that a user needs to have from a device in order to be authorized and the IoT device type allowed for the users to have access to.

Also, another service has been added that allows the users to update device information. This service is connecting to a middleware that correctly formats the information before updating them on the Device Indexing module. This service is protected by the Keycloak plugin.

The user, by logging in with the app gains an access token by the same Keycloak client configured in the Keycloak plugin. To proceed, the request is sent by the app along with headers containing the access token and the id of the user's device. Kong will receive the request in the appropriate route. First, the request passes through the Keycloak plugin where

the validity of the access token is checked. The potential response cases for the outcome can be seen in Table 18.

Table 18: Response codes under different scenarios in the OpenID Connect Authentication (Keycloak) plugin.

Case	Status
Token not sent with the request	400
Token is expired (for public clients)	400
Token cannot be inspected (for private clients)	403
Token is not active (for private clients)	401
Token is not valid (for public clients)	401
Token is valid and active	200

If the token was indeed valid and active and the plugin proceeded with the '200' status code, then the request passes through the Proximity plugin which checks whether the user's current distance from the IoT device is lower than the configured threshold. The potential response codes for this plugin can be seen in Table 19.

Table 19: Response codes under different scenarios in the *proximity* plugin.

Case	Status
Missing any required header	400
Trying to access a path that is not allowed for users	403
Requested device type is out of scope	403
User's device was inactive	401
Distance between devices is greater than the threshold	401
Distance is lower than the threshold	200

If the distance was lower than the threshold and the plugin proceeded with the '200' status code, then the user is authorized to access the resources requested. The request reaches the Device Indexing component, the information is retrieved, and the response is sent to the user through Kong.

An example of attempting to access IoT device data for UC4 is presented in the figures below. Specifically, in Figure 37, on the black terminal the logs of the OpenID Connect plugin, while on the blue terminal the logs of the proximity plugin are depicted. For test purposes of UC4 the proximity threshold is set to 1 meter. The user made two attempts to access the Smart Agriculture device data, one being further away than the configured threshold and the other being close enough. In Figure 38 the data retrieved on the Mobile App from the second attempt are presented.

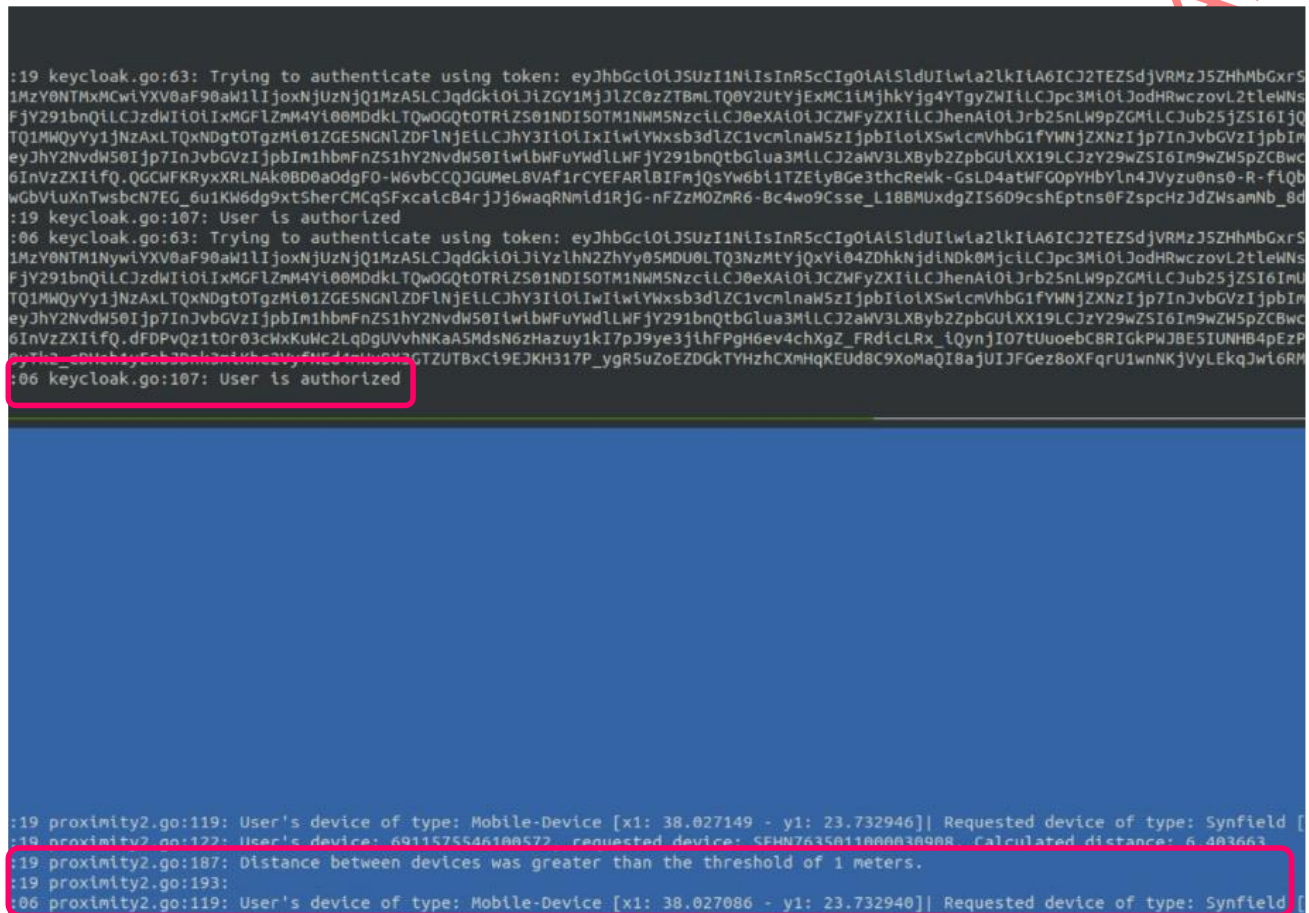


Figure 37: IoT Devices Access Control Plugins' Logs (for the OpenID Connect Authentication plugin in the black terminal, while for the Proximity plugin in the blue terminal).

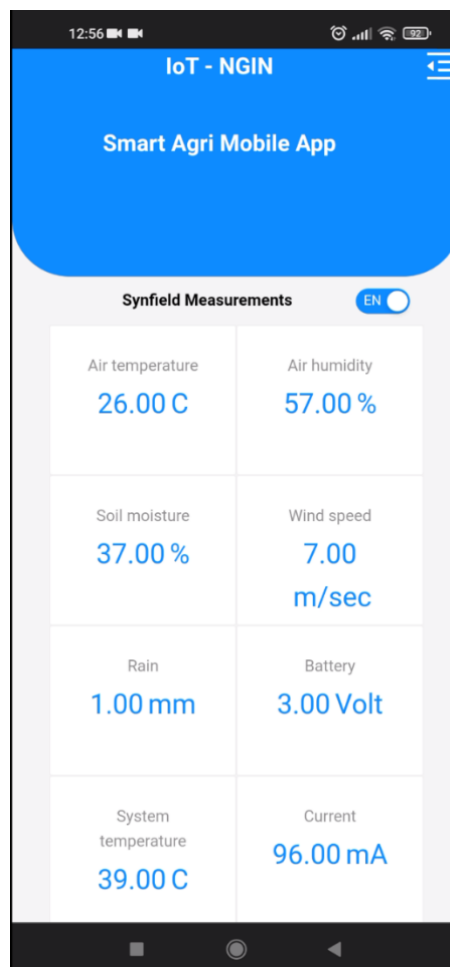


Figure 38: Retrieved Smart Agriculture Device data after successfully passing access control for both the Proximity and the OpenID Connect Authentication plugins.

4.4 AR/MR module

The AR/MR modules for IoT interaction (presentation and actuation) heavily rely on other modules for: IoT discovery; Access Control; Device Indexing; and other Machine Learning solutions and databases to allow more accurate discovery and obtaining more comprehensive information about the discovered element(s). At the time of writing, functional versions of all these modules are available, but their release occurred a couple of months ago. Accordingly, just isolated and specific implementation and testing activities have been conducted for the AR/MR modules for IoT interaction, but no integration with other deployed modules of the IoT-NGIN platform yet.

Next, the main implementation and testing activities that have been performed are summarized.

Selection of AR frameworks: after an initial comparative analysis, some trials and demos were implemented and tested for Unity AR and Vuforia frameworks.

Special attention was given to analyze and test:

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

- their capabilities and performance for device tracking (position, orientation)
- their capabilities and performance for image tracking, even including integrated ad-hoc modules for such a purpose
- their capabilities to send / receive data from remote application, even for real-time conferencing sessions and 5G-powered connections.
- their capabilities to interactively load / hide overlaid textual information, and graphical 2D / elements, attached to markers or other visual elements

This allowed getting familiar with these frameworks, and suggesting the selection of Vuforia engine for Unity for implementing the use cases

Tests with AR devices: testing apps and demos with smartphones (Samsung Galaxy), tablets (Galaxy Tab) and AR headsets (Magic Leap, and Microsoft HoloLens 2)

Tests with overlaid info to videos: some tests have been also conducted to verify the capabilities for:

- displaying associated visual information attached detected (classes of) elements (e.g., persons, AGVs and carts for #UC6)
- displaying 2D/3D models of a map / space (e.g., factory plant in #UC6), together with all detected sensor in it and information about them

Figure 39 shows an example for the above two features. Although they are implemented as an overlay to a 2D video right now, their application to AR interfaces will be smooth as the associated modules and APIs can be applied therein without major efforts.

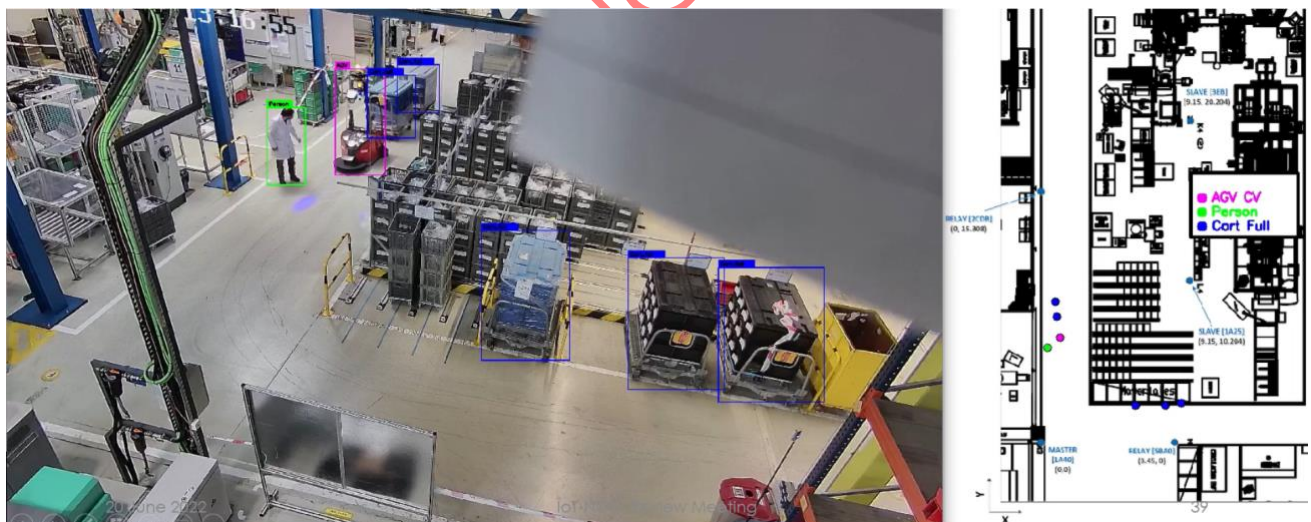


Figure 39: Presentation of overlaid and contextual information to be applied for AR interfaces.

The implementation, integration and validation activities will be completed and fully documented in D4.4, as planned in the Grant Agreement.

Selection of the backend framework

In reference to the chosen framework and based on the sequence diagrams of previous sections a macro blocks architecture in Figure 40 for the AR/MR component has been used with the main aim to investigate the best solution for the backend part.

Initial test has been realized for the AR/MR component with the main purpose to create an API environment for connecting with the external database and the MLaaS. At this purpose *NODE.js* has been selected as web server. A test with Unity is documented below.

Tests with Unity client object and Nodejs webserver

We have managed to receive and send data between the *Nodejs* server and the Unity client.

We equip our *Nodejs* server with a json data and make an object to send it when a Unity client asks for it.

We created a *dummyObject* in Unity and add c# script to enable connection among *Unity* and *Nodejs* web server.

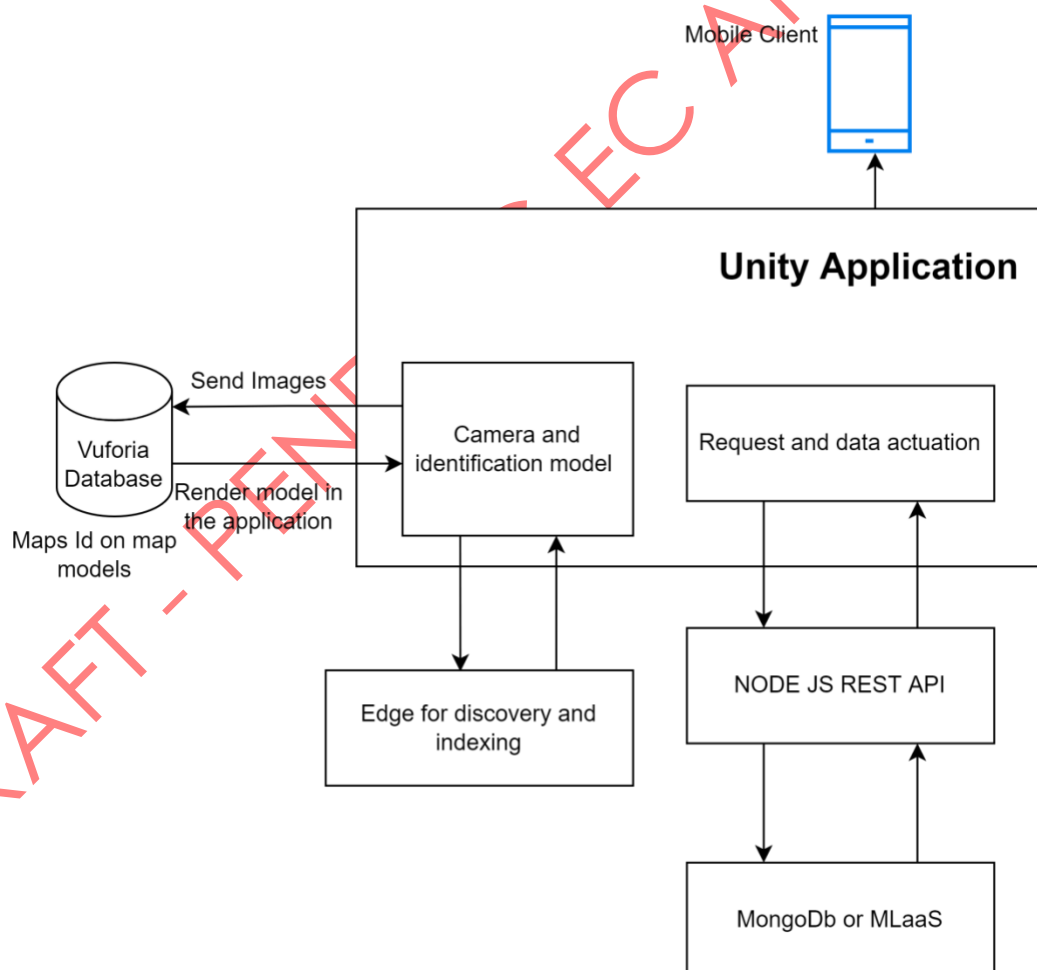


Figure 40: Macro blocks architecture for the AR/MR component.

5 Installation and user guidelines

The implementations of the IoT-NGIN Aml tools are available at the “Enhancing IoT Tactile and Contextual Sensing_Actuating” group of IoT-NGIN GitLab group [41]. In this section, installation guidance is provided for these tools, followed by user guidelines, describing basic functionality of each tool for the candidate.

5.1 IoT Device Discovery

5.1.1 Installation guidelines

Computer vision based solution

The installation guidelines for the Computer Vision solution of the IoT Device Discovery will be included in the deliverable D4.4.

UWB-based location solution

The following instructions are provided as a guideline for the Industry 4.0 UC#6 use case where the UWB location solution will be deployed.

The system requires specific hardware devices (UWB anchors and tags). The devices have been prepared with the binary files that allow an easy deployment (without the need of additional reprogramming steps). The system allows the configuration of the main parameters without the need of modifying the source code. However, the binary files with the firmware release will be also provided in case that a reinstallation is required.

For the setup and installation of the IoT Device Discovery module based on UWB, the following guidelines must be followed:

- Install the WiFi AP configured for the project.
- Place at least four anchor devices (one of them will be labelled as master) in the area that wants to be covered assuring that the anchors will have a good Line of Sight (LoS) with the devices that will be tracked.
- Build a reference coordinates system for positioning the anchors. The accuracy of these positions is very relevant, as it will directly affect the performance of the location solution.
- Switch on the anchor devices. They can be powered by batteries or can be connected to a power socket through the USB interface.
- Once powered, each tag/anchor automatically generates its own WiFi SSID. Join that SSID with a computer and visit <http://192.168.4.1>. Set there the SSID of the WiFi AP configured for the project. From now on, all tags and anchors will have access to the Internet through that AP.
- Deploy the edge service (virtual machine) for configuring the UWB system in the same private network where the anchor devices are connected.
- Introduce the position of the anchors in the edge service (using the provided API).
- Set the main configuration parameters of the UWB positioning system by using the provided API (otherwise, default values will be applied).

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

- Deploy the Device Indexing Module in the edge server (in the same private network where the anchor devices are connected).
- Attach the UWB device to the equipment/vehicle/person that will be tracked and switch it on.
- The system will begin tracking the device and periodically uploading the locations to the Device Indexing Module.
- If there occur any problems restart the network remotely. If problems persist, reset manually the master anchor device and the UWB tag.

For the installation of the firmware binaries, the following steps can be followed:

- DW1001 firmware. It can be installed using a JTAG device.
- ESP32 firmware. It can be installed through the USB interface. The ESP-IDF environment is required.

VLP-based location solution

The following instructions are provided as a guideline for the Industry 4.0 UC#6 use case where the VLP location solution will be deployed.

The system requires specific hardware devices (VLP LEDs and VLP device). The VLP equipment (LEDs) will be prepared with the binary/library files that allow an easy deployment (without the need of additional reprogramming steps). The binary files with the firmware release will be also provided in case that a reinstallation is required.

The deployment of the final VLP-based solution is still work in progress. Herein, some initial instructions for its installation are provided; the final guidelines will be provided in the D4.3 deliverable.

- Install the WiFi AP configured for the project.
- Install the VLP LEDs in the ceiling following the deployment indications. The placement of the LEDs requires an accurate measurement of its position (x,y,z) in a reference coordinates system.
- The VLP LEDs must be permanently connected to a power supply. Once powered they will begin transmitting data without any additional configuration.
- Deploy the Device Indexing Module in the edge server (in the same private network where the VLP device is connected).
- Attach the VLP device to the equipment/vehicle that will be tracked and switch it on.
- Connect to the VLP device through the WiFi network using SSH (user and password will be provided together with the deployment indications).
- Introduce the position of the VLP LEDs in the infrastructure.csv configuration file [LEDid,x,y,z]
- The VLP device will be prepared to automatically track its position once it is in the area of the VLP lights and to send the location updates to the IoT Device Indexing Module.

5.2 IoT Device indexing

5.2.1 Installation guidelines

The following technologies have been used to successfully complete the installation of the IDI:

- Docker (version 20.10.17 has been used in this guide)
- Kubernetes (GitVersion:"v1.22.4" has been used)
- Helm (v3.7.0 has been used)

The FIWARE components of IDI can be installed via a Helm chart that includes the following components:

- Orion Context Broker
- IoT Agent
- MongoDB
- Mosquitto Broker (MQTT)
- RabbitMQ (AMQP)

The repository can be cloned with the following command:

```
$ git clone https://gitlab.com/h2020-iot-nginx/enhancing_iot_tactile_contextual_sensing_actuating/device-indexing/device-indexing-module.git --depth 1 --branch v1.0.0
```

In the 'values.yaml' file there are a few variables that should be configured before deploying. These are:

- In the 'orion' values, the 'ingress.hosts.host' and 'ingress.tls.hosts'. Set to your Orion's domain name.
- In the 'iotagent-json' values, the 'iota.provideUrl' should be set to the URL the IoT Agent is exposed to for requests to its API

Values that have to do with credentials are also configurable and the rest can be configured, depending on the requirements of the installation.

In order to install the Helm chart, use the following command:

```
$ cd device-indexing-module
$ helm install -f values.yaml .
```

In case the 'values.yaml' file has been updated, the changes can be applied as follows:

```
$ helm upgrade --reuse-values orion -f values.yaml .
```

The Historic Data Registry can be installed via a Kubernetes Manifest.

The repository can be cloned with the following command:

```
$ git clone https://gitlab.com/h2020-iot-nginx/enhancing_iot_tactile_contextual_sensing_actuating/device-indexing/historic-data-registry.git --depth 1 --branch v1.0.0
```

The Manifest is configurable and can be created and reapplied, if changes have occurred, as follows:

```
$ cd historic-data-registry
$ kubectl apply -f config/k8s/combo.yml
```

To affirm the state of the deployments:

```
$ kubectl get pods
```

The Inactivity Checker that was mentioned in previous sections is available as a configurable Kubernetes Manifest.

The repository can be cloned as follows:

```
$ git clone https://gitlab.com/h2020-iot-nginx/enhancing_iot_tactile_contextual_sensing_actuating/device-indexing/inactivity-checker.git --depth 1 --branch v1.0.0
```

The Manifest can be deployed with the following commands:

```
$ cd inactivity-checker
$ kubectl apply -f config/k8s/cronjob.yml
```

5.2.2 User guidelines

All the IDI components expose several interfaces. In order to ease the interaction with the available interfaces, a Python Client has been implemented.

The repository can be cloned with the following command:

```
$ git clone https://gitlab.com/h2020-iot-nginx/enhancing_iot_tactile_contextual_sensing_actuating/device-indexing/device-indexing-client.git --depth 1 --branch v1.0.0
```

In order for the client to work correctly, a '.env' file should be set up with the variables listed in Table 20.

Table 20: Environmental variables for the IDI client configuration.

Environmental variable	Description	Example value
ORION_URL	The URL Orion is available at	http://localhost:1026
IOT_AGENT_URL	The URL IoT Agent is available at	http://localhost:4041
ENTITIES_LIMIT	Used to set the limit for Orion's pagination mechanism	10
TIMEZONE	Timezone of device	Europe/Greece
PROTOCOL	Protocol of device	IoTA-JSON
TRANSPORT	Transport protocol of device	HTTP
IOT_AGENT_NORTH_BOUND	The URL IoT Agent uses to receive messages from physical devices, through HTTP	http://localhost:7896
DEVICE_REGISTRY	The URL the Historic Data Registry is available at	http://localhost

This client can be used to connect the Historic Data Registry with the Orion Context Broker, via the *Create Subscription interface*, in order to receive a copy of the devices' changes.

In *'drone/subscribe.py'* one can find the functions to properly form a subscription request and send it. The *'subscribe()'* method should be configured according to the needs of each use case.

The *'main.py'* file contains a full example on how to setup the instance for the Orion and IoT Agent Clients and how to call the functions that implement the different interfaces.

The functions that implement the interfaces for the Historic Data Registry can be found in *'drone/db.py'* file.

5.3 IoT Devices access control

5.3.1 Installation guidelines

Before installing the project, the following are prerequisites:

- Docker installed (version v20.10.12 has been used in this installation guide)
- Kubernetes installed and configured (client version v1.24.1 has been used)
- Helm installed (v3.7.0 has been used)

Then, users must clone the repository of the module:

```
$ git clone https://gitlab.com/h2020-iot-
ngin/enhancing_iot_tactile_contextual_sensing_actuating/access-control.git --depth 1 --
branch v1.0.0
```

Before installation, users can change the username and password for the PostgreSQL database that backs up Kong. To do that, they must go to manifests/kong.yaml and change `env.pg_user` and `postgresql.postgresqlUsername` to a new username and `env.pg_password` and `postgresql.postgresqlPassword` to a new password. They must **NOT** make any other modifications as it can cause unexpected results.

To install the module run:

```
$ ./install.sh
```

For convenience, Figure 41 depicts the command line screenshot presenting the installation steps for the IoT Device Access Control module.



```

Access Control
*****
This is the Access Control module part of WP4 of the
IoT-NGIN project.
Installation will begin shortly..

!Please make sure you have installed on your machine the prerequisites before installation!
-> Checking if Kong's chart already exists..

NAME          CHART VERSION  APP VERSION  DESCRIPTION
kong/kong      2.7.0         2.7          The Cloud-Native Ingress and API-management

Kong chart already exists. Skipping...

- Installing Kong...

Release "access-control-kong" does not exist. Installing it now.
NAME: access-control-kong
LAST DEPLOYED: Thu May 19 13:11:05 2022
NAMESPACE: iot-ngin-access-control
STATUS: deployed
REVISION: 1
NOTES:
To connect to Kong, please execute the following commands:

HOST=$(kubectl get svc --namespace iot-ngin-access-control access-control-kong-kong-proxy -o jsonpath='{.status.loadBalancer.ingress[0].ip}')
PORT=$(kubectl get svc --namespace iot-ngin-access-control access-control-kong-kong-proxy -o jsonpath='{.spec.ports[0].port}')
export PROXY_IP=${HOST}:${PORT}
curl $PROXY_IP

Once installed, please follow along the getting started guide to start using
Kong: https://docs.konghq.com/kubernetes-ingress-controller/latest/guides/getting-started/

✓ Done!

- Installing Konga GUI...

Release "access-control-konga" does not exist. Installing it now.
NAME: access-control-konga
LAST DEPLOYED: Thu May 19 13:11:08 2022
NAMESPACE: iot-ngin-access-control
STATUS: deployed
REVISION: 1
NOTES:
1. Get the application URL by running these commands:
  export NODE_PORT=$(kubectl get --namespace iot-ngin-access-control -o jsonpath='{.spec.ports[0].nodePort}' services iot-ngin-konga)
  export NODE_IP=$(kubectl get nodes --namespace iot-ngin-access-control -o jsonpath='{.items[0].status.addresses[0].address}')
  echo http://$NODE_IP:$NODE_PORT

✓ Done!

Installation complete! Ready to use the Access Control module!

```

Figure 41: Installation complete for the IoT Device Access Control module.

5.3.2 User guidelines

After installation a new namespace will be created on the user's cluster, namely `iot-ngin-access-control`. The Konga GUI is a NodePort and users can visit it and setup an account. First run:

```
kubectl get svc -n iot-ngin-access-control
```

And then from the results check PORT from `iot-ngin-konga`. This will be the port to visit Konga. Visiting Konga, users will be able to register an administration account and sign in the login page shown in Figure 42.

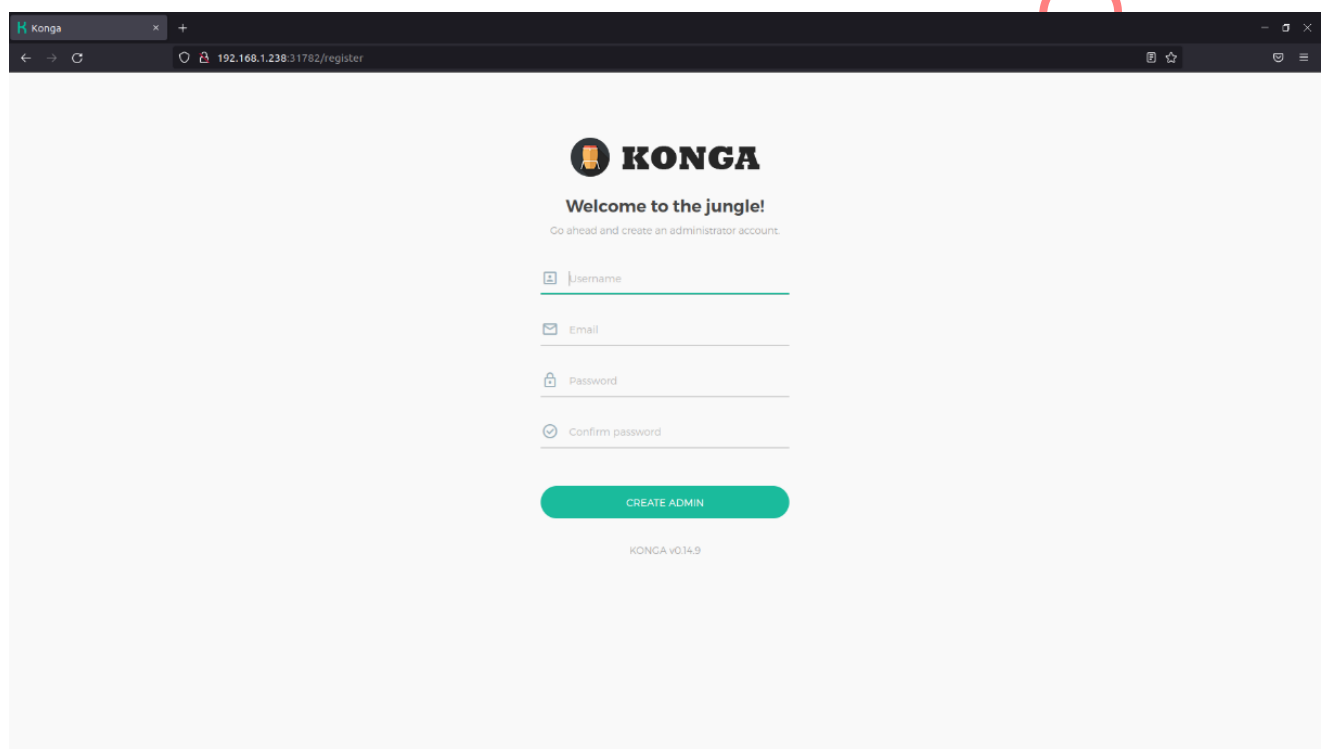


Figure 42: Login page for the IoT Device Access Control dashboard.

Then, users will need to connect Konga to the Kong Admin Service in order to be able to manage Kong, as shown in Figure 43.

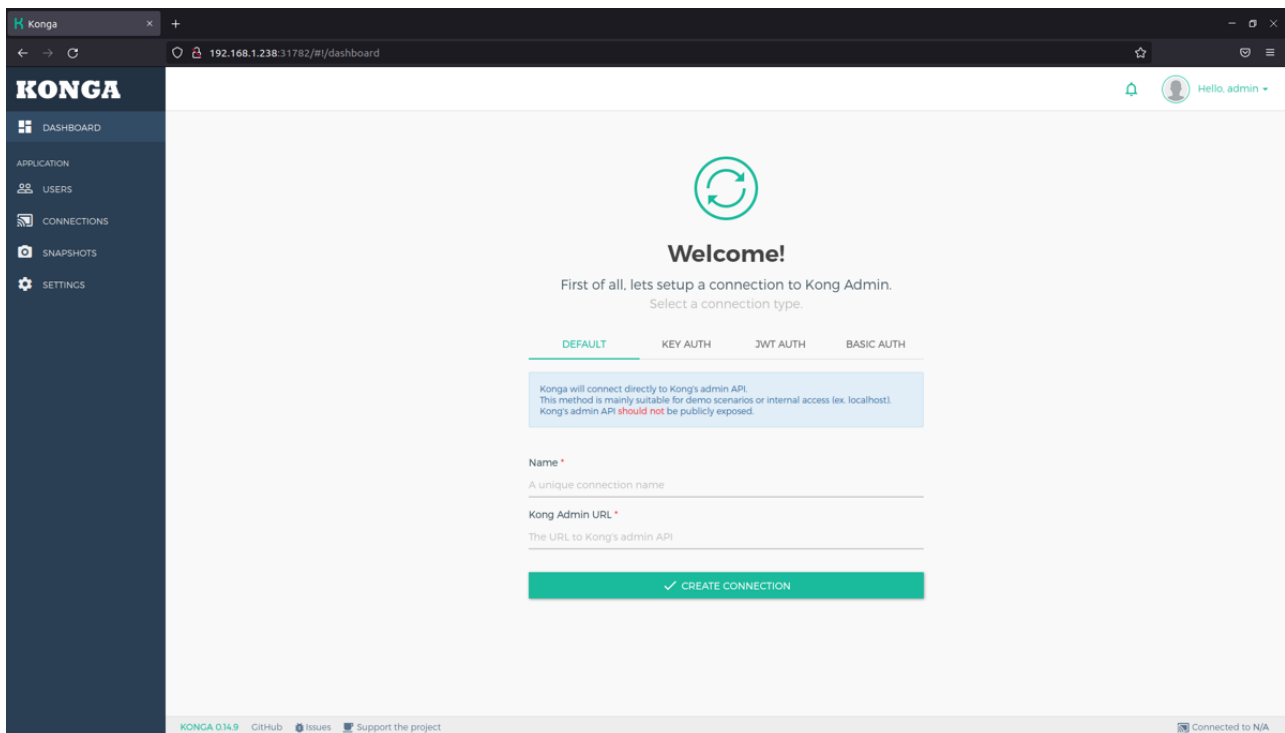


Figure 43: Welcome screen of the IoT Device Access Control dashboard to connect with its Admin service.

Then, users will be ready to add new services from the Services tab (Figure 44 and Figure 45).

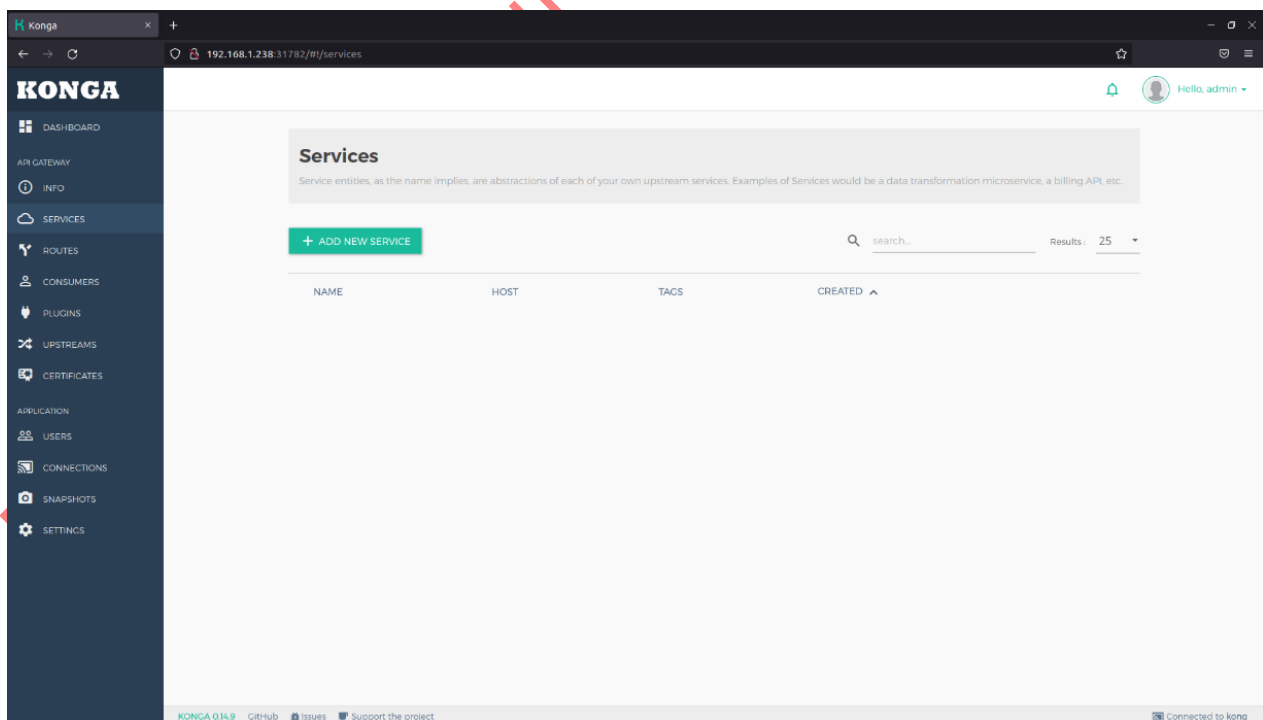


Figure 44: Services tab in the IoT Device Access Control dashboard.

The screenshot shows the Konga dashboard with a 'CREATE SERVICE' modal open. The modal contains the following fields:

- Name** (optional): The service name.
- Description** (optional): An optional service description.
- Tags** (optional): Optionally add tags to the service.
- Uri** (shorthand attribute): Shorthand attribute to set **protocol**, **host**, **port**, and **path** at once. This attribute is write-only (the Admin API never "returns" the uri).
- Protocol** (semi-optional): The protocol used to communicate with the upstream. It can be one of **http** or **https**.
- Host** (semi-optional): The host of the upstream server.
- Port** (semi-optional): The upstream server port. Defaults to **80**.
- Path** (optional): The path to be used in requests to the upstream server. Empty by default.
- Retries** (optional): The number of retries to execute upon failure to proxy. The default is **5**.
- Connect timeout** (optional): The timeout in milliseconds for establishing a connection to the upstream.

Figure 45: Service creation in the IoT Device Access Control dashboard.

To be able to use the expose the Service, a route must be added through the route tab on the created service details (Figure 46 and Figure 47).

The screenshot shows the Konga dashboard with the 'Routes' tab selected for the 'device-indexing' service. The 'Routes' section includes a search bar and a table with the following columns:

Name / ID	Hosts	Paths	Protocols	Methods	Regex priority	Created
no data found...						

Figure 46: Accessing Routes for the device-indexing service.

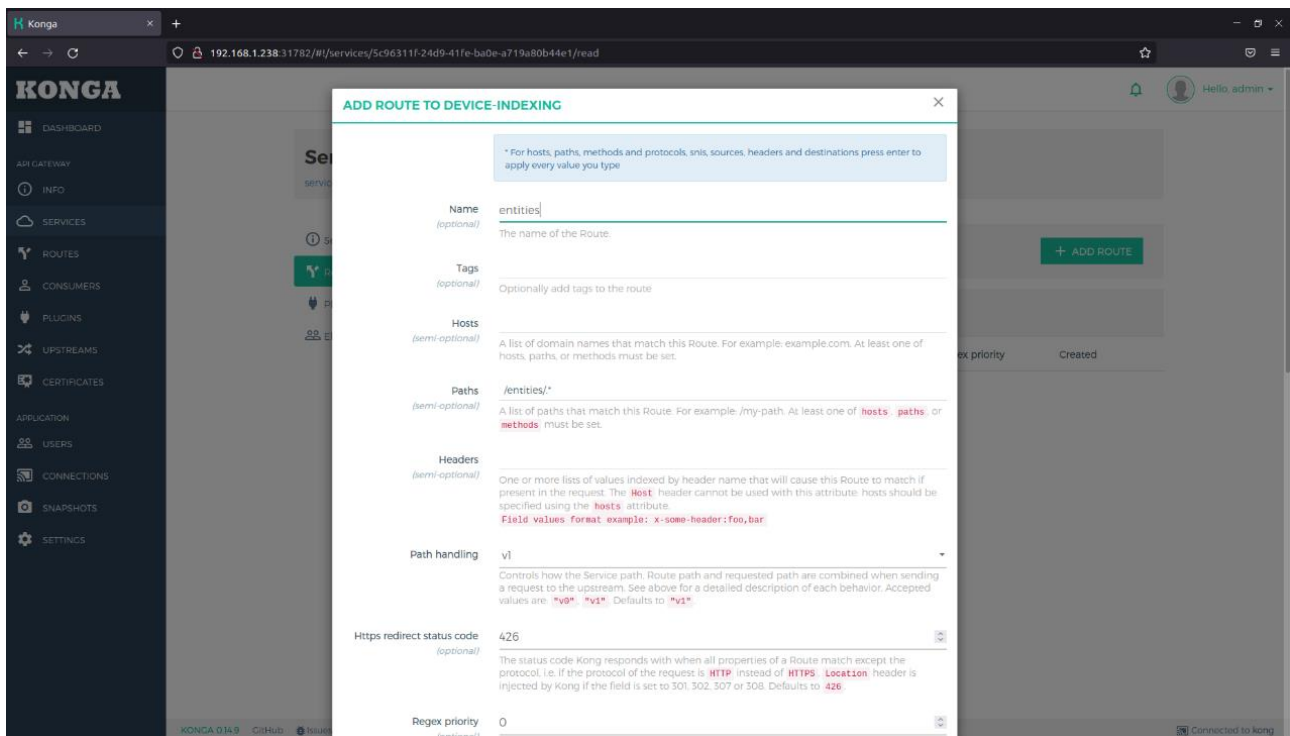


Figure 47: Route creation in the IoT Device Access Control dashboard.

Plugins can be added either globally or on the service level or on the route level through the relevant plugin tabs (Figure 48).

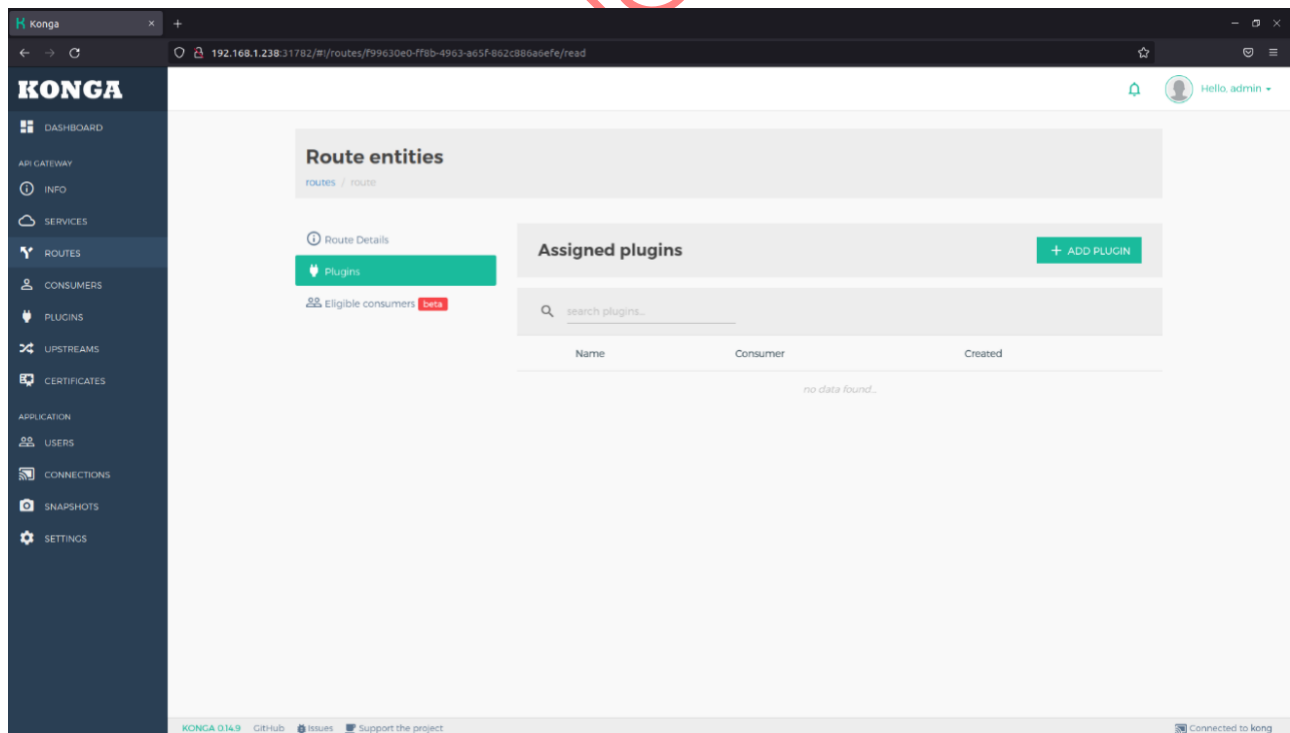


Figure 48: Adding plugins for Route entities.

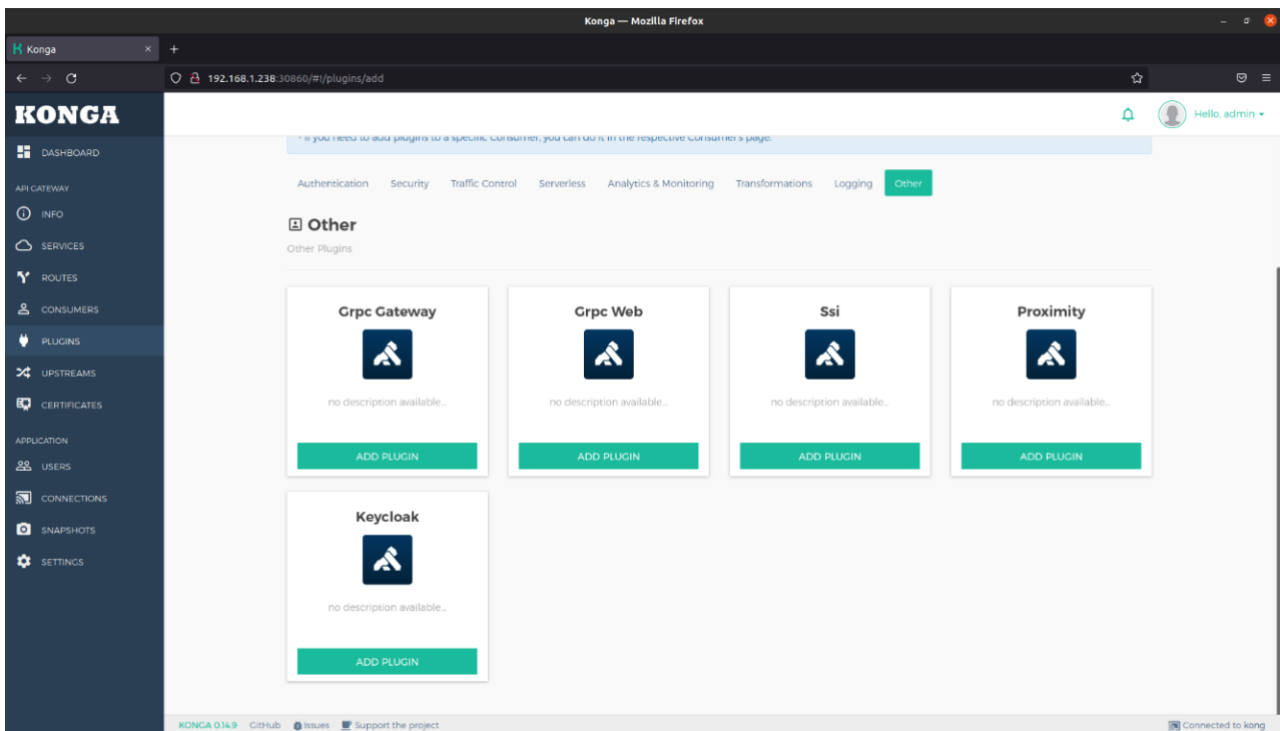


Figure 49: Adding custom plugins.

The custom plugins can be added and used by pressing the “ADD PLUGIN” button and going to the “Other” tab (Figure 49).

For further guidance please read the relevant guide (GUIDE.pdf) on Gitlab [42].

Please note that in order to use the Keycloak and SSI plugins, users need to have deployed their own instances of these components and make the appropriate configurations when setting up the plugins.

To remove the component, run (Figure 50):

```
$ ./uninstall.sh
```


```
*****  
Removing the Access control module...  
  
- Removing Konga...  
release "access-control-konga" uninstalled  
✓ Done!  
  
- Removing Kong...  
release "access-control-kong" uninstalled  
✓ Done!  
  
- Deleting Namespace...  
namespace "iot-ngin-access-control" deleted  
✓ Done!  
  
Access Control module uninstalled.   
To install again run: ./install.sh
```

Figure 50: Uninstalling the Access Control module.

5.4 AR/MR module

As indicated in Section 4, although some preliminary proof-of-concept solutions for the AR/MR module are available at the time of writing, and some tests have been conducted, they mostly refer to isolated implementations and validations for specific features and sub-component of such a module, but are neither full-fledged implementation nor integrated with other components / modules of the IoT-NGIN platform yet. Development and testing activities will be intensified in the next period toward the final deliverable D4.4. At that point, installation and usage guidelines will be shared and detailed, as for other IoT-NGIN modules in this deliverable.

6 Conclusions

The present report has provided technical updates related to the design and implementation of the IoT-NGIN tools towards supporting ambient intelligence in the context of tactile internet. The tools refer to device discovery -conceived as both device recognition and localization/positioning- device indexing, access control and AR/MR interaction.

The updates on the technical design have been based on both updated requirements of the LL use cases claiming use of the components, as well as better perception of the needs across use cases as the use case descriptions and preparations/implementation become more mature. Moreover, the technical design incorporates the updates derived during the development and testing of the relevant tools. The current versions of these components are available as open source on the project's GitLab group. The present document, acting as technical report of these tools, provides insights on the implementation and usage of the tools in specific use cases, as well as on porting them in different use cases, where this is applicable. It also reports the integrated functionality of the Aml tools in these use cases.

In addition, installation and user guidance is provided for the available implementations, encouraging the adoption, experimentation and extensions by third parties.

The future work includes the finalization of the components' implementation and their integration, as well as their component-level validation, in the context of the planned use cases. This work is expected to be delivered in D4.4 "Enhanced IoT Tactile & Contextual Sensing/Actuating (Final Version)", due by the second quarter of 2023.

DRAFT - PENDING EC APPROVAL

7 References

- [1] ITU, "The Tactile Internet - ITU-T Technology Watch Report," 2014. [Online]. Available: https://www.itu.int/dms_pub/itu-t/oth/23/01/T23010000230001PDFE.pdf.
- [2] Synelix, "SynField," Synelix, 2022. [Online]. Available: <https://www.synfield.gr/>.
- [3] RedShift, "What Is a Digital Twin? How Intelligent Data Models Can Shape the Built World," 2021. [Online]. Available: <https://redshift.autodesk.com/articles/what-is-a-digital-twin>.
- [4] A. Yilmaz, O. Javed and M. Shah, "Object tracking: A survey.," *Acm computing surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.
- [5] J. Berclaz, F. Fleuret and P. Fua, "Robust people tracking with global trajectory optimization," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, New York, 2006.
- [6] G. D. Evangelidis and E. Z. Psarakis, "Parametric image alignment using enhanced correlation coefficient maximization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 10, p. 1858–1865, 2008.
- [7] I. Huerta, M. B. Holte and T. B. Moeslund, "Chromatic shadow detection and tracking for moving foreground segmentation," *Image and Vision Computing*, pp. 42-53, 2015.
- [8] C. Kim, F. Li, A. Ciptadi and J. M. Rehg, "Multiple hypothesis tracking revisited," *Proceedings of the IEEE international conference on computer vision*, pp. 4696-4704, 2015.
- [9] G. Brasó and L. Leal-Taixé, "Learning a neural solver for multiple object tracking," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6247-6257, 2020.
- [10] Z. Wang, Z. Liang, L. Yixuan and S. Wang, "Towards real-time multi-object tracking," in *European Conference on Computer Vision*, Cham, 2020.
- [11] L. Bertinetto, J. Valmadre, J. Henriques, A. Vedaldi and P. Torr, "Fully-Convolutional Siamese Networks for Object Tracking," in *European conference on computer vision*, Cham, 2016.
- [12] P. Voigtlaender, J. Luiten, P. H. Torr and B. & Leibe, "Siam r-cnn: Visual tracking by re-detection," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6578-6588, 2020.
- [13] B. Shuai, A. Berneshawi, X. Li, D. Modolo and J. Tighe, "Siammot: Siamese multi-object tracking," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12372-12382, 2021.

- [14] S. Ren, K. He, R. Girshick and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, p. 28, 2015.
- [15] A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, "Simple online and realtime tracking," *IEEE international conference on image processing (ICIP)*, pp. 3464-3468, 2016.
- [16] N. Wojke, A. Bewley and D. Paulus, "Simple online and realtime tracking with a deep association metric," *IEEE international conference on image processing*, pp. 645-3649, 2017.
- [17] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan and et al, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Chan, 2014.
- [18] C. Y. Wang, A. Bochkovskiy and H. Y. M. Liao, "Scaled-yolov4: Scaling cross stage partial network," *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pp. 13029-13038, 2021.
- [19] M. Tan, R. Pang and Q. V. Le, "Efficientdet: Scalable and efficient object detection," *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10781-10790, 2020.
- [20] J. Yan, C. Tiberius, G. Bellusci and G. Janssen, "Feasibility of Gauss-Newton method for indoor positioning," *IEEE/ION PLANS*, pp. 660-670), 2008.
- [21] IoT-NGIN, "D4.2 - Enhancing IoT Ambient Intelligence," H2020-957246 IoT-NGIN Deliverable Report, 2021.
- [22] FIWARE, "FIWARE-NGSI v2 Specification," 2018. [Online]. Available: <https://fiware.github.io/specifications/ngsiv2/stable/>.
- [23] Telefonica, "Orion Context Broker," GitHub, 2022. [Online]. Available: <https://github.com/telefonicaid/fiware-orion/blob/master/README.md>.
- [24] FIWARE, "IoT-Agent tutorials," 2021. [Online]. Available: <https://github.com/FIWARE/tutorials.IoT-Agent>.
- [25] FIWARE, "FIWARE IoT stack documentation - Multitenancy," [Online]. Available: <https://thinking-cities.readthedocs.io/en/release-v4.1/multitenancy/index.html>. [Accessed 2022].
- [26] Kong Inc., "Kong Gateway," 2022. [Online]. Available: <https://konghq.com/products/api-gateway-platform>.
- [27] P. Tselentis, "Konga," 2020. [Online]. Available: <https://pantisel.github.io/konga/>.
- [28] Google, "Go," [Online]. Available: <https://go.dev>. [Accessed 2022].
- [29] OpenID, "OpenID Connect," 2022. [Online]. Available: <https://openid.net/connect/>.

- [30] Keycloak, "https://www.keycloak.org," [Online]. Available: <https://www.keycloak.org>.
- [31] IoT-NGIN, "D5.3 - Enhancing IoT Data Privacy & Trust," H2020-957246 IoT-NGIN Deliverable Report, 2021.
- [32] Y. Sheffer, D. Hardt and M. Jones, "JSON Web Token Best Current Practices," Internet Engineering Task Force (IETF), 2020.
- [33] IoT-NGIN, "Privacy Preserving Self Sovereign Identities," 2022.
- [34] Magic Leap Inc., "Magic Leap 2," 2022. [Online]. Available: <https://www.magicleap.com/>.
- [35] Microsoft, "Microsoft HoloLens 2," 2022. [Online]. Available: <https://www.microsoft.com/en-us/hololens>.
- [36] PTC, "Vuforia Engine Developer Portal," PTC, 2022. [Online]. Available: <https://developer.vuforia.com>.
- [37] Tensorflow, "SSD ResNet50 V1 FPN 640x640'," 2020. [Online]. Available: http://download.tensorflow.org/models/object_detection/tf2/20200711/ssd_resnet50_v1_fpn_640x640_coco17_tpu-8.tar.gz.
- [38] F. Rashidi Fathabadi, J. L. Grantnep and I. Abdel-Qader, "Box-Trainer Assessment System with Real-Time Multi-Class Detection and Tracking of Laparoscopic Instruments, using CNN," *Acta Polytechnica Hungarica*, vol. 19, no. 2, pp. 7-27, 2022.
- [39] Heartexlabs, "Labellmg," GitHub, 2022. [Online]. Available: <https://github.com/tzutalin/labellmg/archive/master.zip>.
- [40] Saurabh Bagalkar, "A Deep Dive Into The Improved Fast R-CNN Approach," 2019. [Online]. Available: <https://medium.com/alegion/deep-learning-for-object-detection-and-localization-using-fast-r-cnn-85d52e3928a1>.
- [41] IoT-NGIN, "Enhancing IoT Tactile and Contextual Sensing_Actuating," Gitlab, 2022. [Online]. Available: https://gitlab.com/h2020-iot-ngin/enhancing_iot_tactile_contextual_sensing_actuating.
- [42] IoT-NGIN, "IoT Devices Access Control Guide," GitLab, 2022. [Online]. Available: https://gitlab.com/h2020-iot-ngin/enhancing_iot_tactile_contextual_sensing_actuating/access-control/-/blob/main/GUIDE.pdf.

Annex 1 IoT-NGIN Aml tools Interfaces

IoT Device Discovery

Table 21: IoT Device Discovery interfaces.

IoT Device Discovery module		
Description	The module will enable the IoT-NGIN architecture with the capabilities to support a more dynamic and advanced recognition (identification and localization) of available IoT devices in the scenario. These capabilities will either run in the edge or in own IoT devices.	
Provided Interfaces	VLP and UWB positioning Interface	
	Description	The interface provides positioning updates of the IoT registered devices. The location of the device can be performed either using Visual Light Positioning or Ultra-Wide Band technology, depending on the infrastructure deployed/available in the use case. Updates are transmitted to the IoT Device Indexing module using a REST API. The positioning calculation and the message update runs on the IoT device that is located (periodic updates).
	End-point URL	None. The Interface acts as a client (publishes information to the End-Point URL provided by the IoT Device Indexing module)
	Protocol used	HTTP
	Methods	POST (will publish the information)
	Message	The message includes the following parameters: IoT device identifier, timestamp, localization (x,y coordinates), location method (UWB, VLP), reliability (can be used as an indication of the quality of the measurement)
	Computer Vision Interface	
	Description	The interface provides positioning updates of objects discovered on video streams by Computer Vision. Updates are transmitted to the IoT Device Indexing module using a REST API. The detection, tracking and position calculations and the message update runs on an edge server provided with a GPU.
	End-point URL	None. The Interface acts as a client (publishes information to the End-Point URL provided by the IoT Device Indexing module)
	Protocol used	HTTP

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

	Methods	POST
	Message	The message includes the following parameters: IoT device ID, object ID, timestamp, localization (x,y coordinates)
	Code scanning – single image	
	Description	Through this interface, the user will be able to initiate a code scan operation through a mobile app or a web service.
	End-point URL	/codescanner/api/scan/
	Protocol used	HTTPS
	Methods	POST
	Message	<pre>{ "scan_device": {int:deviceId}, "format": "{string: code_format}", "image_url": "{string: image_url}", "image": {formData: image_data} }</pre>
	Response	<pre>{ "code_value": "string: {code_value}", "format": "string: {format}" }</pre>
	Code scanning – list of images	
	Description	Through this interface, the user will be able to initiate a code scan operation for a known code format through a mobile app or a web service.
	End-point URL	/codescanner/api/scan/list/
	Protocol used	HTTPS
	Methods	POST
	Message	<pre>{ "scan_device": {int:deviceId}, "images": [{ "format": "{string: code_format}", "image_url": "{string: image_url}", "image": {formData: image_data} }] }</pre>

		<pre> }, { "format": "{string: code_format}", "image_url": "{string: image_url}", "image": {formData: image_data} }] } </pre>
	Response	<pre> { "scanned_codes": [{ "code_value": "string: {code_value}", "format": "string: {format}" }] } </pre>
Required Interfaces	<p>IoT Device Indexing interface: Provides the REST API for communicating the positioning updates and logging scanning activity.</p> <p>IoT Device Access Control: Provides the REST API for access control interactions.</p> <p>User Application: Interaction through HTTP.</p>	

IoT Device Indexing

Table 22: IoT Device Indexing interfaces.

IoT Device Indexing																									
Description	<p>This module consists of 2 FIWARE technologies, the Orion context broker and the IoT Agent as well as the Historic Data Registry. The Orion context broker is responsible for storing the most recent version of a device's digital twin. The IoT Agent is to receive a device's measurements, translate the information into NGSI-compatible models and, then, forward them to the context broker.</p> <p>In the case of an external source (e.g. client application) applying changes to a device's data, the update message would follow a backward course to reach the device.</p> <p>The Historic Data Registry holds a record of all the modifications that have transpired for a device's data.</p>																								
Provided Interfaces	IoT Agent Interfaces <table> <tr> <th colspan="2">HTTP device data updates</th></tr> <tr> <td>Description</td><td>The interface allows the devices to send data updates, through HTTP</td></tr> <tr> <td>End-point URL</td><td>/iot/d?k={apikey}&i={device_id}</td></tr> <tr> <td>Protocol used</td><td>HTTP</td></tr> <tr> <td>Methods</td><td>POST</td></tr> <tr> <td>Message</td><td> <pre>{ "location": "4.645312789411754, - 2.829921052375175", "timestamp": "2022-05-19T13:42:21.425354Z", "sensors": [{ "sensor_id": 1, "sensor_name": "Θερμοκρασία Αέρα", "sensor_name_en": "Air temperature", "units": "C", "measurement": "20.60872742541275" }, { "sensor_id": 2, "sensor_name": "Υγρασία Αέρα", "sensor_name_en": "Air humidity", "units": "%", "measurement": "47.53497996921379" }] }</pre> </td></tr> <tr> <th colspan="2">MQTT device data updates</th></tr> <tr> <td>Description</td><td>The interface allows the devices to send data updates, through MQTT</td></tr> <tr> <td>End-point URL</td><td>/[{apikey}]/[{device_id}]/attrs</td></tr> <tr> <td>Protocol used</td><td>MQTT</td></tr> <tr> <td>Methods</td><td>POST</td></tr> <tr> <td>Message</td><td> <pre>{ "location": "4.645312789411754, - 2.829921052375175",</pre> </td></tr> </table>	HTTP device data updates		Description	The interface allows the devices to send data updates, through HTTP	End-point URL	/iot/d?k={apikey}&i={device_id}	Protocol used	HTTP	Methods	POST	Message	<pre>{ "location": "4.645312789411754, - 2.829921052375175", "timestamp": "2022-05-19T13:42:21.425354Z", "sensors": [{ "sensor_id": 1, "sensor_name": "Θερμοκρασία Αέρα", "sensor_name_en": "Air temperature", "units": "C", "measurement": "20.60872742541275" }, { "sensor_id": 2, "sensor_name": "Υγρασία Αέρα", "sensor_name_en": "Air humidity", "units": "%", "measurement": "47.53497996921379" }] }</pre>	MQTT device data updates		Description	The interface allows the devices to send data updates, through MQTT	End-point URL	/[{apikey}]/[{device_id}]/attrs	Protocol used	MQTT	Methods	POST	Message	<pre>{ "location": "4.645312789411754, - 2.829921052375175",</pre>
HTTP device data updates																									
Description	The interface allows the devices to send data updates, through HTTP																								
End-point URL	/iot/d?k={apikey}&i={device_id}																								
Protocol used	HTTP																								
Methods	POST																								
Message	<pre>{ "location": "4.645312789411754, - 2.829921052375175", "timestamp": "2022-05-19T13:42:21.425354Z", "sensors": [{ "sensor_id": 1, "sensor_name": "Θερμοκρασία Αέρα", "sensor_name_en": "Air temperature", "units": "C", "measurement": "20.60872742541275" }, { "sensor_id": 2, "sensor_name": "Υγρασία Αέρα", "sensor_name_en": "Air humidity", "units": "%", "measurement": "47.53497996921379" }] }</pre>																								
MQTT device data updates																									
Description	The interface allows the devices to send data updates, through MQTT																								
End-point URL	/[{apikey}]/[{device_id}]/attrs																								
Protocol used	MQTT																								
Methods	POST																								
Message	<pre>{ "location": "4.645312789411754, - 2.829921052375175",</pre>																								

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

	<pre> "timestamp": "2022-05-19T13:42:21.425354Z", "sensors": [{ "sensor_id": 1, "sensor_name": "Θερμότητα Αέρα", "sensor_name_en": "Air temperature", "units": "C", "measurement": "20.60872742541275" }, { "sensor_id": 2, "sensor_name": "Υγρασία Αέρα", "sensor_name_en": "Air humidity", "units": "%", "measurement": "47.53497996921379" }] </pre>
Get Services	
Description	The interface allows the user to get the available services created for a group of devices
End-point URL	/iot/services
Protocol used	HTTP
Methods	GET
Message	<pre> { "count": 1, "services": [{ "commands": [], "lazy": [], "attributes": [], "_id": "62823e0fb1d33b1856abb124", "resource": "/iot/d", "apikey": "4jggokgpepnvsb2uv4s40dv", "service": "synfield", "subservice": "/", "__v": 0, "static_attributes": [], "internal_attributes": [], "entity_type": "Synfield" }] } </pre>
Create Service	
Description	The interface allows for the creation of a service. A service is used by IoT Agent to identify the devices to connect, decide on the correct context broker in the case of a HA architecture.

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

End-point URL	/iot/services
Protocol used	HTTP
Methods	POST
Message	[<pre>{ "apikey": "4jggokgpepnvsb2uv4s40dv", "entity_type": "Synfield", "resource": "/iot/d" }</pre>]
Update Service	
Description	The interface allows for the specifications of a service to be updated
End-point URL	/iot/services?resource={resource}&apikey={apikey}
Protocol used	HTTP
Methods	PUT
Message	{ <pre>"entity_type": "Thing"</pre> }
Delete Service	
Description	The interface allows for the deletion of a service
End-point URL	/iot/services/?resource={resource}&apikey={apikey}
Protocol used	HTTP
Methods	DELETE
Message	"The request was successful and the entity/service/device was updated/deleted"
Get Devices	
Description	The interface is used to retrieve all the devices created through the IoT Agent.
End-point URL	/iot/devices
Protocol used	HTTP
Methods	GET
Message	{ <pre>"count": 1, "devices": [{ "device_id": "deviceS1", "service": "synfield", "service_path": "/", "entity_name": "urn:ngsi-ld:Device:S1", "entity_type": "Synfield", "transport": "HTTP", "attributes": [{ "object_id": "location", "name": "location", "type": "geo:point" }, { "object_id": "sensors", "name": "sensors", "type": "StructuredValue" }] }]</pre> }

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

	<pre> }], "lazy": [], "commands": [], "static_attributes": [], "protocol": "IoT-A-JSON", "explicitAttrs": false }] } </pre>
Create Device	
Description	The interface allows for the creation of a new device through IoT Agent. The Agent translates the information of the sent message to an NGSI-compatible data model, before forwarding it to the context broker.
End-point URL	/iot/devices
Protocol used	HTTP
Methods	POST
Message	<pre> [{ "device_id": "deviceS1", "entity_name": "urn:ngsi-Id:Device:S1", "entity_type": "Synfield", "protocol": "IoT-A-JSON", "transport": "HTTP", "attributes": [{ "name": "location", "type": "geo:point" }, { "name": "sensors", "type": "StructuredValue" }] }] </pre>
Update Device	
Description	The interface allows for the update of a device's characteristics defined during its creation
End-point URL	/iot/devices/{device_id}
Protocol used	HTTP
Methods	PUT
Message	<pre> { "entity_name": "urn:ngsi-Id:Thing:deviceS1" } </pre>
Delete Device	
Description	The interface allows for the deletion of a device
End-point URL	/iot/devices/{device_id}

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

Protocol used	HTTP
Methods	DELETE
Message	"The request was successful and the entity/service/device was updated/deleted"
Orion Context Broker interface	
Get Entities	
Description	The interface is used to retrieve all the available entities (digital twins) stored in the context broker's database
End-point URL	/v2/entities
Protocol used	HTTP
Methods	GET
Message	<pre>[{ "id": "urn:ngsi-Id:Device:S1", "type": "Synfield", "TimeInstant": { "type": "DateTime", "value": "2022-05-16T14:01:50.213Z" }, "location": { "type": "geo:point", "value": "2.5689596437815094, -0.4571677184094982" }, "sensors": { { "type": "StructuredValue", "value": [{ "sensor_id": 1, "sensor_name": "Θερμοκρασία Αέρα", "sensor_name_en": "Air temperature", "units": "C", "measurement": "20.628302761442384" }] } } }]</pre>
Get Entity	
Description	The interface allows for the retrieval of a single entity
End-point URL	/v2/entities/{entityId}
Protocol used	HTTP
Methods	GET
	<pre>{ "id": "urn:ngsi-Id:Device:S1", "type": "Synfield", "TimeInstant": { "type": "DateTime",</pre>

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

```

    "value": "2022-05-16T14:01:50.213Z"
  },
  "location":
  {
    "type": "geo:point",
    "value": "2.5689596437815094, -
0.4571677184094982"
  },
  "sensors":
  {
    "type": "StructuredValue",
    "value":
    [
      {
        "sensor_id": 1,
        "sensor_name": "Θερμοκρασία Αέρα",
        "sensor_name_en": "Air temperature",
        "units": "C",
        "measurement": "20.628302761442384"
      }
    ]
  }
}

```

Delete Entity

Description	The interface allows for the deletion of a single entity
End-point URL	/v2/entities/{entityId}
Protocol used	HTTP
Methods	DELETE
Message	"The request was successful and the entity/service/device was updated/deleted"

Get Subscriptions

Description	The interface allows for the retrieval of all available subscriptions
End-point URL	/v2/subscriptions
Protocol used	HTTP
Methods	GET

Message

```

[
  {
    "id": "620bc32f5c89d40d0d6a6e63",
    "description": "Dummy Subscription",
    "status": "active",
    "subject":
    {
      "entities":
      [
        {
          "idPattern": ".*",
          "typePattern": ".*"
        }
      ],
      "condition":
      {
        "attrs":
        [

```

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

	<pre> "ip", "location", "vulnerabilities"] }, "notification": { "timesSent": 3, "lastNotification": "2022-05-17T13:42:20.012Z", "attrs": ["location"], "attrsFormat": "keyValues", "http": { "url": "http://localhost:8080/notification" } } } }] </pre>
Create Subscription	
Description	The interface allows for the creation of a notification
End-point URL	/v2/subscriptions
Protocol used	HTTP
Methods	POST
Message	"The subscription was created successfully"
Update Entity	
Description	The interface allows for the update of one or more entities
End-point URL	/v2/op/update
Protocol used	HTTP
Methods	POST
Message	<pre> { "actionType": "append", "entities": [{ "type": "Synfield", "id": "urn:ngsi-ld:Device:S1", "location": { "value": "2.5689596437815094, - 0.4571677184094982" } }, { "type": "Thing", "id": "urn:ngsi-ld:Device:T1", "temperature": { "value": 22.9 } }] } </pre>

Historic Data Registry	
Get all device records	
Description	The interface is used to retrieve all the available records across all devices
End-point URL	/get_all_records
Protocol used	HTTP
Methods	GET
Message	<pre>[{ "id": "urn:ngsi-Id:Device:S1", "type": "Synfield", "device_data": [{ "location": "2.5689596437815094, - 0.4571677184094982", "sensors": [{ "sensor_id": 1, "sensor_name": "Θερμοκρασία Αέρα", "sensor_name_en": "Air temperature", "units": "C", "measurement": "20.628302761442384" }] }] }]</pre>
Get single device record	
Description	The interface is used to retrieve all the available records for a particular device
End-point URL	/get_all_records/{entityId}
Protocol used	HTTP
Methods	GET
Message	<pre>[{ "id": "urn:ngsi-Id:Device:S1", "type": "Synfield", "device_data": [{ "location": "2.5689596437815094, - 0.4571677184094982", "sensors": [{ "sensor_id": 1, "sensor_name": "Θερμοκρασία Αέρα", "sensor_name_en": "Air temperature", "units": "C", "measurement": "20.628302761442384" }] }] }]</pre>

]
Get single device last N records	
Description	The interface is used to retrieve the last N records for a particular device
End-point URL	/get_last_n_records/{entidyId}?last_n={last_n}
Protocol used	HTTP
Methods	GET
Message	<p>For last_n=1:</p> <pre>[{ "id": "urn:ngsi-Id:Device:S1", "type": "Synfield", "device_data": [{ "location": "2.68959644, -0.571677182", "sensors": [{ "sensor_id": 1, "sensor_name": "Θερμοκρασία Αέρα", "sensor_name_en": "Air temperature", "units": "C", "measurement": "12.76144" }] }] }]</pre>
Receive Context Broker notification	
Description	The interface is used to receive the notofication with the device update from the Context Broker
End-point URL	/notification
Protocol used	HTTP
Methods	POST
Message	<pre>{ "subscriptionId": "61faa9335c89d82d0d6a6e54", "data": [{ "id": "urn:ngsi-Id:Device:S1", "type": "Synfield", "location": "2.5689596437815094, - 0.4571677184094982", "sensors": [{ "sensor_id": 1, "sensor_name": "Θερμοκρασία Αέρα", "sensor_name_en": "Air temperature", "units": "C", "measurement": "20.628302761442384" }] }] }]</pre>

	}
Inactivity Checker	
Get Entities	
Description	The interface is used to retrieve all the available entities (digital twins) stored in the context broker's database
End-point URL	/v2/entities
Protocol used	HTTP
Methods	GET
Message	<pre>[{ "id": "urn:ngsi-Id:Device:S1", "type": "Synfield", "TimeInstant": { "type": "DateTime", "value": "2022-05-16T14:01:50.213Z" }, "location": { "type": "geo:point", "value": "2.5689596437815094, -0.4571677184094982" }, "sensors": { "type": "StructuredValue", "value": [{ "sensor_id": 1, "sensor_name": "Θερμοκρασία Αέρα", "sensor_name_en": "Air temperature", "units": "C", "measurement": "20.628302761442384" }] } }]</pre>
Update Entity	
Description	The interface allows for the update of one or more entities
End-point URL	/v2/op/update
Protocol used	HTTP
Methods	POST
Message	<pre>{ "actionType": "append", "entities": [{ "type": "Synfield", "id": "urn:ngsi-Id:Device:S1", "location": {</pre>

	<pre> "value": "2.5689596437815094, - 0.4571677184094982" } }, { "type": "Thing", "id": "urn:ngsi-Id:Device:T1", "temperature": { "value": 22.9 } } }] } </pre>
Required Interfaces	No interfaces are required for the operation of IDI.

IoT Device Access Control

Table 23: IoT Device Access Control interfaces.

Access Control										
Description	The module provides authorized access to the resources of the project. The users, using the appropriate authentication, are permitted to view, edit or delete resources of services that are hosted in different paths on the Kong's proxy URL.									
Provided Interfaces	Device Indexing Interface									
	<table> <tr> <td>Description</td><td>The interface provides access to the resources of the IoT Device Indexing module. A user is authorized to access information or delete a specific device using the <i>proximity</i> and the <i>OpenID Connect Authentication</i> plugins. For a more detailed view of the interface please visit: https://access-control-api.iotngin.lab.synelixis.com/ui/</td></tr> <tr> <td>End-point URL</td><td>http://{KONG_PROXY_URL}/entities/{DEVICE_ID}</td></tr> <tr> <td>Protocol used</td><td>HTTP</td></tr> <tr> <td>Methods</td><td>GET/DELETE</td></tr> <tr> <td>Message</td><td> Response: GET: <pre> { "id": "device_id", "type": "device_type", "Timestamp": { "type": "DateTime", </pre> </td></tr> </table>	Description	The interface provides access to the resources of the IoT Device Indexing module. A user is authorized to access information or delete a specific device using the <i>proximity</i> and the <i>OpenID Connect Authentication</i> plugins. For a more detailed view of the interface please visit: https://access-control-api.iotngin.lab.synelixis.com/ui/	End-point URL	http://{KONG_PROXY_URL}/entities/{DEVICE_ID}	Protocol used	HTTP	Methods	GET/DELETE	Message
Description	The interface provides access to the resources of the IoT Device Indexing module. A user is authorized to access information or delete a specific device using the <i>proximity</i> and the <i>OpenID Connect Authentication</i> plugins. For a more detailed view of the interface please visit: https://access-control-api.iotngin.lab.synelixis.com/ui/									
End-point URL	http://{KONG_PROXY_URL}/entities/{DEVICE_ID}									
Protocol used	HTTP									
Methods	GET/DELETE									
Message	Response: GET: <pre> { "id": "device_id", "type": "device_type", "Timestamp": { "type": "DateTime", </pre>									

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

	<pre> "value": "", "metadata": {} }, "active": { "type": "Boolean", "value": true, "metadata": {} }, "location": { "type": "geo:point", "value": "lat, lon", "metadata": { "TimeInstant": { "type": "DateTime", "value": "" } } }, "newDevice": { "type": "Boolean", "value": false, "metadata": {} }, "attributes_list": { "type": "StructuredValue", "value": [], "metadata": { "TimeInstant": { "type": "DateTime", "value": "" } } } } </pre>	
	Update Devices	
	Description	This interface provides an endpoint that enables users to update device information. It is protected by the <i>OpenID Connect Authentication</i> plugin.
	End-point URL	<code>http://{KONG_PROXY_URL}/iot/d?k={API_KEY}&i={DEVICE_ID}</code>
	Protocol used	HTTP
	Methods	POST

	Message	Request: body: {"attr_updated": "value"} Response: 200 OK
Required Interfaces	<p>IoT Device Indexing: Provides the REST API that holds the resources the users want to access and is implemented as a service in Kong. It is also needed by the Proximity custom plugin.</p> <p>In general, any interface that needs to be protected by the access control module.</p> <p>Keycloak Admin API: Used by the Keycloak custom plugin.</p> <p>Self-Sovereign-Identities Component: Used by the SSI custom plugin.</p>	

AR/MR module

Table 24: IoT AR module interfaces.

IoT AR Module		
Description	The module will enable the IoT-NGIN architecture with the capabilities to present the information from the discovered IoT sensors or elements, and to actuate them, via an AR interface.	
Provided Interfaces	Discovery	
	Description	Interface to be informed about new IoT sensors / elements discovered via any of the discovery methods (Computer Vision, VLP, UWB...), if the Access Control module (see Section 3.3) allows for that
	End-point URL	/iot/devices [(Complete) URI will depend on use the specific use case and deployment]
	Protocol used	HTTP(S), but MQTT also possible
	Methods	POST (to AR device)
	Message	See "New Device" Interface for the IoT Device Indexing Module (Section 3.2)
	Presentation	
	Description	Interface to be provided about period or requested updates for the information of specific IoT sensors registered in the Indexing Module, both the newly discovered or those previously discovered
	End-point URL	/iot/devices [(Complete) URI will depend on use the specific use case and deployment]

D4.3 - ENHANCING IOT TACTILE & CONTEXTUAL SENSING/ACTUATING

	Protocol used	HTTP(S), but MQTT also possible
	Methods	POST (to AR device), for period updates GET (from AR device) for requesting updates
	Message	See Messages for "Services", "New Device", "Entities", and "Get Entity" for the IoT Device Indexing Module (Section 3.2)
	Actuation	
	Description	Interfaces to update information, attributes or behaviour of registered IoT sensors, based on information previously retrieved from them via IoT Indexing modules.
	End-point URL	/iot/devices [(Complete) URI will depend on use the specific use case, deployment and what attributes / information need to be updated]
	Protocol used	HTTP(S), but MQTT also possible
	Methods	POST (from AR device) But potentially also PUT / DELETE, and GET to be confirmed with the actuation / modification
	Message	See Messages for "Data Updates", "Service Updates", "Device Update", and "Delete Device" for the IoT Device Indexing Module (Section 3.2)
Required Interfaces	In principle, no additional interfaces are required	