D2.2 Architecture Requirements and Definition (v1)

WP2 System Requirements, Architecture Specification and Implementation

European Commission

Horizon 2020
European Union funding
for Research & Innovation

Document information

Project Identifier	ECSEL-2017-783221
Project Acronym	AFarCloud
Project Full Name	Aggregate Farming in the Cloud
Document Version	1.0
Planned Delivery Date	M12 (August 2019)
Actual Delivery Date	M12 (August 2019)
Deliverable Name	D2.2 Architecture Requirements and Definition
Work Package	WP2 System Requirements, Architecture Specification and Implementation
Task	T2.2 Architecture Requirements and Definition
Document Type	Report
Dissemination level	Public
Abstract	This document contains the functional and non-functional architectural requirements of the AFarCloud platform and the design of the architecture.

Document History

Version	***Date***	***Contributing partner***	***Contribution***
0.1	19th October 2018	TECN, ROVIMATICA, UPM, HUA, ITAV, HI-IBERIA	Contribution to scenario requirements, data requirements and messages
0.2	25th October 2018	TECN, DAC, TST, TTC, UPM, AVL, AVL-CD, AMS, UWB, AIT	Contribution to scenario requirements and data requirements

0.3	19th November 2018	UWB, ROVIMATICA, UNIPR, INTRA	Update of requirements (UNIPR) Cloud services requirements (INTRA) Update of value ranges in the 2.3 Livestock data table and addition of butyric and lactic acid concentrations (UWB) High-performance image processing platform (ROVIMATICA);
0.4	30th November 2018	MDH, BOSIT, HUA, TTC, TECN	Update of requirements
0.5	23th January 2019	AVL-CD, ROTEC, UPM, HUA, PDMFC, BEV, APP4M, ACREO	Update of livestock data with more specific information. (UPM) Update of requirements: - vehicle, security and development requirements (AVL-CD) - requirements and guidelines (ROTEC) - security and distributed intelligence (HUA) Requirements updates, WSNs, UAVs (PDMFC, BEV, APP4M) Sensors (ACREO)
0.6	28th January 2019	TECN, UPM, HUA, AVL-CD, ROTEC, ROVI, ESTE, INTRA, AIT, CENTRIA, UNIVAQ, EXODUS, NXP, RISE ACREO, SPACEMETRIC, AMS, IMAG	General architecture Update in data requirements from data provided by sensors Update of middleware diagram (TECN)
0.7	26th February 2019	AMS, HUA, TECN, MDH	Update to 7.5 (AMS) Update of 3.5. Ground Vehicle data (HUA, TECN) Mission Management Tool description (MDH)
0.8	1st March 2019	TECN, HUA	Update of 3.5 Ground Vehicle data, AUV data (HUA) Update of middleware architecture (TECN)

0.9	4th March 2019	TECN	Update of the interfaces of the Image Processing Platform (TECN) Update of the interfaces of the FMS and the Middleware (TECN)
0.10	29th March 2019	BOSIT, TECN	Added draft descriptions to cloud repositories and access to third-party data (BOSIT). Update of the list and description of the middleware components, update of the middleware architecture and components diagram, new data flow diagrams added (TECN).
0.11	4th April 2019	IMAG, UPM, TECN, MDH	Update of Annex1: LoRa WAN features, MQTT and MQTT-SN descriptions. (IMAG) Description of the Mission Processing & Reporter and the Mission Manager (UPM). Update of the description of the general description of the architecture components (TECN), update of the MMT description (MDH)
0.12	9th April 2019	TECN, CUNI	Update of the description the architecture components (TECN), update of MMT description (CUNI)
0.13	17th April 2019	TECN	Update of the description of the architecture components, update of the data flow between components
0.14	24rd April 2019	TECN	Semantic middleware figure update. New data flow diagram for REST compatible devices. Description of DDS Manager.
0.15	30th April 2019	TST, TECN, ITAV	MQTT Broker/Clients description (general topic, QoS settings, fw updates) (TST), update of DDS Manager and Environment Reporter. New cloud resources monitoring requirements (TECN). Env. Reporter specification (ITAV)

0.16	14th May 2019	ITAV, TECN, BEV, AMS	Description of Knowledge Extractor and Data Pre-Processor (ITAV). Distinction between Cloud Data Storage and Edge Data Storage. Definition of different architecture options for UAVs. Description of legacy systems databases. Proposal for specific MQTT topics (TECN), Proposal for fw updates over MQTT (AMS, TST). Integration of multi-hop WSNs (ROTEC). Update of development requirements (BEV, AVL), cloud resources monitoring requirements (TECN, CUNI)
0.17	15th May 2019	TECN, DAC, HUA, BEV, PDMFC	Updates in the System Configuration module in the FMS. Updates in MQTT Broker/Clients. Cloud Resources Monitoring. Summary of the data model in Annex 2 (TECN). Streaming Engine (DAC). Updates in REST services and Distributed Intelligence Requirements (HUA, TECN). Updates in UAV (BEV, PDMFC). Data Access Manager and Data Query (HIB)
0.18	24th May 2019	ITAV, TECN, BEV, AIT, UNIPR, TST, PDMFC, UPM	Update in Environment Reporter (ITAV), update in Data Flow Diagrams (TECN). Revision and update of requirements to new table structure: vehicle req (BEV, TECN), security and development req (AIT), communication req (UNIPR), sensor req (TST), cloud services req (PDMFC), general req (UPM), Image Data Manager (SM)
0.19	27th May 2019	TECN, TST, MDH, AIT, ROVI	Update of general req (TECN), HMI req. (MDH), vehicle req. (ROVI), MQTT Broker/Clients (TST, TECN), Device Manager (TECN), cyber-security management (AIT)
0.20	10th June 2019	ROVI, TECN, BEV, PDMFC	Diagrams updated for: architecture, middleware and data flow for devices sending data to the platform (TECN), Computer Vision Platform module renamed to Image Processing Platform and contents updated (ROVI, TECN), update of the Image Data

			Manager (TECN), update of Data Pre-processor and Data Fusion (BEV, PDMFC).
0.21	19thJune 2019	TECN, ITAV, ROVI, MTECH, SINTEF, UPM, VTT, RISE ACREO	Update of the middleware diagram (TECN), update of ER, DPP, DF and KE (ITAV, TECN), REST Services and Data Models for IPP (ROVI), description of CDX usage (MTECH), update of device registry (TECN, UPM), update of livestock data (UPM), update of soil data with measured grass values (VTT), internal review of scenarios requirements (SINTEF, RISE ACREO).
0.22	25th June 2019	TECN, INTRA, MDH	Update of REST services description (TECN), update of the System Configuration in the FMS (INTRA, MDH), update of the interfaces of the AFarCloud platform (TECN).
0.23	02nd July 2019	ITAV, MDH	Update of the Environment Reporter, the Data Pre-Processor, the Data Fusion and the Knowledge Extractor (ITAV), update of the Farm Management System section (MDH).
0.24	5th July 2019	TECN, APPS4	First version for internal review
0.25	30th July 2019	MDH, BEV, RISE	Updates based on comments from internal reviewers; updates on sensor requirements
1.0	1st August 2019	TECN	Version 1

Document Contributors

Partner name	***Partner member***	***e-Mail***
TECNALIA	Sonia Bilbao	sonia.bilbao@tecnalia.com
TECNALIA	Belén Martínez	belen.martinez@tecnalia.com
TECNALIA	Leire Orue-Echevarria	Leire.Orue-Echevarria@tecnalia.com
TECNALIA	Fernando Jorge Hernandez	fernando.jorge@tecnalia.com
ROVIMATICA	Miguel A. Aragón	miguel.aragon@rovimatica.com

ROVIMATICA	Patricio Alemany	patricio.alemany@rovimatica.com
UPM	Jesús Rodríguez	jesus.rodriguezm@upm.es
HUA	Vassilis Dalakas	vdalakas@hua.gr
HUA	Anargyros Tsadimas	tsadimas@hua.gr
HUA	Dimosthenis Anagnostopoulos	dimosthe@hua.gr
HUA	Mara Nikolaidou	mara@hua.gr
HUA	George Dimitrakopoulos	gdimitra@hua.gr
ITAV	Joaquim Bastos	jbastos@av.it.pt
IPB	Luís Miguel Pinheiro da Luz	luisluz@ipbeja.pt
HI-IBERIA	Inmaculada Luengo	iluengo@hi-iberia.es
HI-IBERIA	Gaia Rubio	grubio@hi-iberia.es
DAC	Bartosz Jachimczyk	bartosz.jachimczyk@dac.digital
TST	Arturo Medela	amedela@tst-sistemas.es
TTC	Martijn Rooker	martijn.rooker@tttech.com
AVL	Patrik Maier	Patrik.Maier@avl.com
AVL	Bernhard Frohner	Bernhard.Frohner@avl.com
AVL	Zhendong Ma	Zhendong.Ma@avl.com
AVL-CD	Daniel Puckmayr	daniel.puckmayr@avl.com
AVL-CD	Wolfgang Hollerweger	Wolfgang.Hollerweger@avl.com
AVL-CD	Bernhard Frohner	Bernhard.Frohner@avl.com
AMS	Johannes Loinig	Johannes.Loinig@ams.com
AMS	Ernst Haselsteiner	Ernst.Haselsteiner@ams.com
AMS	Isabella Wagner	Isabella.Wagner@ams.com
UWB	Roman Čečil	rcecil@kky.zcu.cz
UWB	Michal Kepka	mkepka@kgm.zcu.cz
UWB	Tomas Mildorf	mildorf@kgm.zcu.cz
UWB	Martin Čech	mcech@kky.zcu.cz
AIT	Erwin Kristen	Erwin.Kristen@ait.ac.at
AIT	Christoph Schmittner	Christoph.Schmittner@ait.ac.at
AIT	Arndt Bonitz	Arndt.Bonitz@ait.ac.at
INTRA	Theofanis Orphanoudakis	Theofanis.Orphanoudakis@intrasoft-intl.com
INTRA	Dimitrios Skias	Dimitrios.SKias@intrasoft-intl.com
UNIPR	Antonio Cilfone	antonio.cilfone@unipr.it

UNIPR	Gaia Codeluppi	gaia.codeluppi@unipr.it
UNIPR	Luca Davoli	luca.davoli@unipr.it
UNIPR	Gianluigi Ferrari	gianluigi.ferrari@unipr.it
MDH	Afshin Ameri	afshin.ameri@mdh.se
MDH	Baran Curuklu	baran.curuklu@mdh.se
MDH	Branko Miloradovic	branko.miloradovic@mdh.se
BOSIT	Iván Gómez	Ivan.gomez@bosonit.com
ROTEC	Leonardo Napoletani	leonardo.napoletani@rotechnology.it
PDMFC	Francisco Damião	Francisco.damiao@pdmfc.com
PDMFC	Nuno Ramalho	nuno.ramalho@pdmfc.com
BEV	Dário Pedro	dario.pedro@beyond-vision.pt
BEV	João Carvalho	joao.m.carvalho@beyond-vision.pt
BEV	Fábio Azevedo	fabio.azevedo@beyond-vision.pt
BEV	Ricardo Sacoto	ricardo.martins@beyond-vision.pt
RISE ACREO	Cristina Rusu	cristina.rusu@ri.se
RISE ACREO	Åke Sivertun	ake.sivertun@ri.se
ESTE	Carlo Ferraresi	ferraresi@estetechnology.com
CENTRIA	Mikko Himanka	Mikko.Himanka@centria.fi
UNIVAQ	Marco Santic	marco.santic@univaq.it
EXODUS	Anna Palaiologk	a.palaiologk@exodussa.com
NXP	Axel Nackaerts	axel.nackaerts@nxp.com
SPACEMETRIC	Daniel Åkerman	da@spacemetric.com
IMAG	Alex Jonsson	alexj@imagimob.com
CUNI	Tomas Bures	bures@d3s.mff.cuni.cz
CUNI	Petr Hnetynka	hnetynka@d3s.mff.cuni.cz
MTECH	Johanna Häggman	johanna.haggman@mtech.fi
SINTEF	Mariann Merz	Mariann.Merz@sintef.no
SINTEF	Gorm Johansen	Gorm.Johansen@sintef.no
VTT	Miranto Akseli	Akseli.Miranto@vtt.fi
APPS4	Diogo Silva	diogo.silva@apps4mobility.com
UPM	Jesús Rodríguez	jesus.rodriguezm@upm.es
UPM	Jose-Fernán Martínez	jf.martinez@upm.es

UPM	Gregorio Rubio	gregorio.rubio@upm.es
UPM	Pedro Castillejo	pedro.castillejo@upm.es
UPM	Victoria Beltrán	mv.beltran@upm.es

Internal Reviewers

Partner name	***Partner member***	***e-Mail***
MDH	Baran Curuklu	baran.curuklu@mdh.se
BEV	Dário Pedro	dario.pedro@beyond-vision.pt
RISE ACREO	Cristina Rusu	cristina.rusu@ri.se

Table of Contents

Table of Figures

Tables

Definitions and Acronyms

Acronym	***Definition***	***Remark***
ADF	Acid Detergent Fiber	
AMQP	Advanced Message Queuing Protocol	
API	Application Programming Interface	
CAN	Controller Area Network	
CAPEX	Capital Expenditure	
CEC	Cation-Exchange Capacity	
CIR	Color InfraRed	
CoAP	Constrained Application Protocol	
CRUD	Create, Read, Update and Delete	
CoTS	Commercial off The Shelf	
CWSI	Crop Water Stress Index	

Acronym	***Definition***	***Remark***
DoW	Description of Work	
DB	DataBase	
DDS	Data Distribution Service	
DSS	Decision Support System	
eCO2	equivalent CO2	
eTVOC	equivalent Total Volatile Organic Compounds	
FMS	Farm Management System	
FPCM	Fat-Protein Corrected Milk	
FR	Foundational Requirements	
GA	Genetic Algorithm	
GDPR	General Data Protection Regulation	
GIS	Geographical Information System	
GNSS	Global Navigation Satellite System	
GPS	Global Positioning System	
GPU	Graphics Processing Unit	
GUI	Graphical User Interface	
GV	Ground Vehicle	
HLP	High-Level Planning	
HMI	Human Machine Interface	
HTTP	HyperText Transfer Protocol	
ID	IDentifier	
IDL	Interface Description Language	
IDM	Image Data Manager	
IMU	Inertial Measurement Unit	
IoT	Internet of Things	
IPP	Image Processing Platform	
IT	Information Technology	
KE	Knowledge Extractor	
LLP	Low-Level Planning	

Acronym	***Definition***	***Remark***
LPWAN	Low Power Wide Area Networks	
MLP	Mid-Level Planning	
MQTT	Message Queuing Telemetry Transport	
MQTT-SN	Message Queuing Telemetry Transport for Sensor Networks	
MMT	Mission Management Tool	
MW	Middleware	
M2M	Machine to Machine	
NB-IoT	NarrowBand IoT	
NDF	Neutral Detergent Fiber	
NDRE	Normalized Difference Red Edge index	
NDVI	Normalized Difference Vegetation Index	
NDWI	Normalized Difference Water Index	
NGSI-LD	NGSI with Linked Data	
NP-hard	Non-deterministic Polynomial-time hardness	
N-S-E-W	North-South-East-West	
NTP	Network Time Protocol	
OGC	Open Geospatial Consortium	
OPEX	OPerational EXpenditure	
OT	Operation Technology	
QoS	Quality of Service	
REST	REpresentational State Transfer	
RGB	Red Green Blue	
SIFT	Scale Invariant Feature Transform	
SPOF	Single Point Of Failure	
SOM	Soil Organic Matter	
SfM	Structure for Motion	
SIL	Safety Integrity Level	
SL	Security Level	
SLO	Service-Level Objective	

Acronym	***Definition***	***Remark***
SPARQL	SPARQL Protocol and RDF Query Language	
SQL	Structured Query Language	
SSL	Secure Sockets Layer	
TBD	To Be Described	
TLS	Transport Layer Security	
TVOC	Total Volatile Organic Compounds	
UAV	Unmanned Aerial Vehicle	Also mentioned as vehicle(s) dependent on the context
UDP	User Datagram Protocol	
(U)GV	(Unmanned) Ground Vehicle	Also mentioned as vehicle(s) dependent on the context
USB	Universal Serial Bus	
UTF	Unicode Transformation Format	
VQT	Value, Quality, Time (data format)	
WMS	Web Map Service	
WSN	Wireless Sensor Network	
XML	EXtensible Mark-up Language	
6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks	

1. Introduction

This document contains the functional and non-functional architectural requirements of the AFarCloud platform, as well as the design of the architecture.

In order to define the architectural requirements for the AFarCloud platform, this document has taken as input the methodology described in deliverable D6.1 (more specifically, the system viewpoint and the goals and subgoals identified in the AJA tables) and the end-user requirements collected through questionnaires in Tasks 2.1 and 7.1.

Based on these requirements, the design of the AFarCloud platform architecture has been carried out. This platform consists of three main functional components: (i) the Farm Management System, (ii) the Semantic Middleware and (iii) the Deployed Hardware. Besides, the AFarCloud platform will interconnect with other data sources like third-party data and legacy systems databases.

The Farm Management System offers: a Mission Management Tool (MMT), a Decision Support System (DSS), a system configurator and applications for the user to manage and monitor the whole system; plan cooperative missions involving Unmanned Aerial Vehicles (UAV) and ground vehicles ranging from fully autonomous UGVs to legacy systems; configure the above-mentioned systems including their key hardware components (mission relevant sensors and other component important for performing a mission); and make decisions pre-, during-, and post-mission.

The Semantic Middleware offers, among others, components for: data storage and retrieval from the Cloud; managing and cataloguing images; registration of IoT devices, animals and vehicles in the farm; data flow management inside the platform; managing, controlling and acquiring data from IoT devices and missions involving ground and aerial vehicles; data processing and knowledge extraction.

The Deployed Hardware layer involves the functionalities related to unmanned aerial vehicles, semi-autonomous ground vehicles, actuators, sensors and other IoT devices.

1.1. Structure of the document

The document is organised as follows. Chapter 2 provides the list of architectural requirements grouped by categories: general, vehicle, user interface, communication, distributed intelligence, sensors, devices and WSN, safety and security, cloud services, cloud resources monitoring and development tool requirements.

Chapter 3 gathers the kind of information that will be used and exchanged inside AFarCloud platform regarding geographic regions, environment data, crops and soil data, livestock and milk quality information, ground and aerial vehicles.

In Chapter 4, the approach followed in the project for sharing of resources among farms is explained. Besides, the three main functional components of the platform (i.e. Farm Management System, Semantic Middleware and Deployed Hardware) are described in detail and their expected functionalities are presented.

Chapter 5 is dedicated to the Farm Management System and its three main components: the Mission Management Tool that provides services for (i) defining, (ii) planning, (iii) monitoring, (iv) controlling, (v) analysing, and finally (vi) saving mission-related data (incl. sensor data, status of all connected hardware such as sensors, actuators, robots/vehicles where applicable) in steps (i) – (v) of a mission; the Decision Support System, which provides expert recommendations using algorithms that extract conclusions from data; the System Configuration that handles the configuration of system hardware (vehicles, sensors, actuators, etc.).

Chapter 6 describes the Semantic Middleware and all its internal components: cloud data storage, cloud resources monitoring, data interoperability, data access manager and data query, device registry, streaming engine, device and mission managers, mission and alarm processing & reporters, environment reporter, data pre-processor, data fusion and knowledge extraction, image processing and data manager, ISOBUS gateway, DDS manager, MQTT broker and clients and REST services.

Finally, Chapter 7 contains some data flow diagrams depicting a) how missions are sent to UAVs and (semi-)autonomous UGVs, b) how data flows from the vehicles and from the IoT devices to the middleware and to the FMS, c) how to configure the sampling rate on a MQTT device.

2. Scenarios Requirements Matrix

The purpose of this section is to provide the list of architectural requirements of a flexible and secure AFarCloud platform in terms of data collection, data communication/processing, and device control. To this end and following the strategy for demonstrations detailed in deliverable D7.1, the work has taken as input the list of demonstrator functionalities in D7.1 (added for readability as Annex 3. Scenario Functionalities (D7.1)), the list of user requirements defined in D2.1 (added as Annex 4. User Requirements (D2.1)) and the Scientific, Technical and Business Objectives described in the DoW.

Some general guidelines have been followed when defining the requirements:

- Each requirement should be **TRACEABLE** in order to allow future mapping between system and integration
- Design each requirement so to keep its **ATOMICITY** (that is, at the lowest possible detail level so that it should not be possible to split it in two or more components)
- Every requirement should be, at some point, **TESTED** so statements must be expressed as testable arguments.
- Try to keep every requirement as **CONSISTENT** and **UNAMBIGUOUS** as possible.
- Relevant information within a statement should always be clear to make the requirement **COMPLETE**.

In the requirements matrix the use of the word “**shall**” or ”**will**” denotes requirements that must be met. Use of the word “**should**” denotes requirements that are desirable. Use bold for "**shall**"/”**will**” and "**should**".

The requirements are grouped according to the following definitions:

- GEN – General requirements. Requirements that are not specific to any of the following categories (see below).
- VEH – Vehicle requirements. Specific requirements for UAVs, UGVs, and legacy systems.
- HMI – User interface requirements.
- COM – Communication requirements.
- INT- Distributed intelligence, cooperation algorithms, etc.
- SEN – Sensor and WSN requirements
- SEC – Safety and security requirements

- CLOUD – Cloud services requirements. AFarCloud services should be deployed on a multi-cloud environment that ensures scalability, performance and accessibility, alerting when a non-functional requirement is violated. AFarCloud platform should provide the five essential characteristics for cloud computing as defined by NIST[1]: (i) On-Demand Self-Service, (ii) Broad Network Access, (iii) Resource Pooling, (iv) Rapid Elasticity and (v) Measured Service.
- CLDMON – Cloud resources monitoring requirements. These are requirements for the monitoring component which, as described in the DoW, will monitor the availability and performance of the cloud resources with the main aim of triggering events. This component will verify if the non-functional requirements of the Cloud Services Provider (CSPs) and the SLOs are being fulfilled.
- DEV - Development Tool requirements. These are the set of requirements requested by partners which plan to use components from other partners.

For each requirement, the following information is provided:

- **Req. No.:** Requirement numbers shall start with the group followed by a unique number. Derived requirements (if any) have an additional number. For example: GEN-1-1 is the first derived requirement to requirement GEN-1. Requirements numbers may be changed in final version of documents. Letters may be used in early document versions in order to present requirements in a logical order e.g.GEN-1, GEN-1a, GEN-2, GEN-2a, GEN-2b, GEN-3 etc.
- **Req. Description:** definition of the requirement
- **Source:** input from where this architectural requirement was defined. This includes: functionalities from the demonstrators in D7.1, user requirements identified in D2.1 and objectives described in DoW.
- **Priority:** relevance of this requirement for end-users and for achieving the objectives in the DoW or the goals and functionalities of the demonstrators
- **Deadline:** expected year when a first release of this requirement should be ready in order not to put into risk the objectives of the project. Improvements can be done in future releases.
- Responsible WP: main WPs that are responsible for providing the requirement.
- Comment: additional information that can be useful for the comprehension of the requirement.

[1] The NIST Definition of Cloud Computing: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

2.1. General requirements

General requirements (GEN)					
Req no.	**Req. Description**	**Source** (see Annex 3. Scenario Functionalities (D7.1) & Annex 4. User Requirements (D2.1))	**Priority**	**Deadline**	**Responsible WP**
GEN-1	The system **shall** provide a framework (methods & collection of technologies) to facilitate the use of autonomous (ground and aerial) vehicles and sensors for precision farming	F1, F5, F28 UR1, UR2, UR5, UR7, UR8, UR12, UR18. DoW	High	Y1 (sensors), Y2 (vehicles)	WP2, WP3, WP4, WP5, WP6
GEN-2	Historical information about a mission **should** be available in the system for further usage: mission assessment, planning of future mission and input to the DSS for additional knowledge extraction.	DoW	Medium	Y1	WP2
GEN-3	A mission **shall** consist of a set of tasks to be performed by a group of ground or/and aerial vehicles with the purpose of achieving certain goals. When a non-autonomous vehicle, or job, is required, a human is assumed to operate the intended system.	F28 DoW	High	Y2	WP3, WP2

General requirements (GEN)					
Req no.	**Req. Description**	**Source** (see Annex 3. Scenario Functionalities (D7.1) & Annex 4. User Requirements (D2.1))	**Priority**	**Deadline**	**Responsible WP**
GEN-4	The system **shall** use a common data format for the data exchanged between software components	DoW	High	Y1	WP2
GEN-5	The system **shall** be designed in compliance with standards selected according to system domain i.e., web standards, telecommunication standards, user interface standards, WCAG 2.1, etc.	DoW	High	Y3	WP2, WP3, WP4, WP5, WP6
GEN-6	All relevant entities, vehicles and nodes in the system **shall** use Epoch date/time format	DoW	High	Y1	WP4, WP5, WP6
GEN-7	The system **shall** provide mechanisms to exploit data from at least one common existing farm management systems already in use in one of the demonstration sites (preferably at a holistic demonstration site i.e., AS09-AS11).	UR7	High	Y3	WP2
GEN-8	The system **shall** provide farmers with means to access information coming from other farms and external sources (i.e., meteorological data)	F1 UR7, UR12 DoW	High	Y2	WP4, WP2

General requirements (GEN)					
Req no.	**Req. Description**	**Source** (see Annex 3. Scenario Functionalities (D7.1) & Annex 4. User Requirements (D2.1))	**Priority**	**Deadline**	**Responsible WP**
GEN-9	Ground and airborne vehicles **shall** be used in at least one of the holistic scenarios (AS09-AS11)	DoW	High	Y2	WP7, WP6
GEN-10	The system **should** allow the registration of new sensors, actuators and vehicles in AFarCloud before the start of a mission.	DoW	Medium	Y2	WP2
GEN-11	The system **should** support several distinct user roles	DoW	Medium	Y2	WP3, WP2

2.2. Vehicle requirements

Vehicle requirements (VEH)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
VEH-1	The system **shall** support ISObus (ISO 11783) compatible ground vehicles (this may not apply to all UGVs, since they may be primarily robotic systems).	DoW	High	Y1	WP6	
VEH-2	UAVs **shall** transmit at least the following information to the middleware: position and orientation, speed, inner status (fuel/battery left) and task status.	DoW	High	Y1	WP6	The information will be sent via DDS
VEH-3	GVs **should** transmit at least the following information to the middleware: position, speed and inner status (fuel/battery left).	DoW	Medium	Y2	WP6	Tractor interface shall be the ISO-XML file.
VEH-4	UAVs **should** detect, react to, and report internal faults and error states, at least safety critical.	DoW	Medium	Y2	WP6	Actions might have to be determined from the vehicle’s status

Vehicle requirements (VEH)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
VEH-5	GVs **should** detect, react to and report internal faults and error states, at least safety critical.	DoW	Medium	Y2	WP6	Actions might have to be determined from the vehicle's status. Detection + Reaction is done directly by vehicle. SAE J1939-73 defines Diagnostic messages sent on CAN bus containing DTCs (Diagnostic Trouble Codes - basically a number up to ~550000), the meaning of the DTCs is vehicle specific. So, the AFarCloud Platform will need to know at least those DTCs which shall be used for 'actions'.
VEH-6	Data regarding UAV and satellite monitoring **should** be collected and stored in AFarCloud data repositories.	DoW	Medium	Y1 (UAV), Y2 (satellite monitoring)	WP4, WP5, WP6	

Vehicle requirements (VEH)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
VEH-7	GVs **shall** have machine interfaces (e.g., displays) showing information from the AFarCloud system to the operator	DoW	High	Y2	WP3	A tractor will have a display (or tablet) mounted inside the cabin, which is providing information to the farmer. Difference in interface between 1) tablets (android, iOS) with mobile data (mobile MMT): direct connection to cloud; 2) tractor display interfaces usually CANbus + USB stick with ISO-XML; framework is e.g., embedded linux/ Codesys
VEH-8	Tractors **shall** have a power supply (12V) for connecting the machine interface	DoW	High	Y2	WP6	
VEH-9	UAVs **should** have a safe state for each operational situation whenever possible. See SEC-18.	DoW	Medium	Y1	WP6	E.g., a ROS node may be created to detect critical operational situation
VEH-10	GVs **shall** have a safe state for each operational situation.	DoW	High	Y1	WP6	

Vehicle requirements (VEH)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
VEH-11	UAVs **shall** switch to safe state whenever possible if a flight safety critical incident is detected.	DoW	High	Y2	WP6	E.g., a ROS node may be created to receive alerts and switch to safe mode.
VEH-12	GVs **shall** switch to safe state if at least one contributing device of a mission reports a safety critical incident.	DoW	High	Y2	WP6	Example: Tractor + implement is on the field, performing the tasks of a mission. All of the sudden, one positioning sensor on the field fails or gives implausible values. In this case the tractor shall switch to safe state --> I.e., stop movement, stop implement operation.

2.3. HMI requirements

HMI requirements (HMI)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
HMI-1	The operator **shall** be given a warning if an abnormal situation occurs. Comment: Could be an alarm and/or a visual indication.	UR3	High	Y2	WP3, WP6	
HMI-2	The operator **shall** see ground and airborne vehicles locations in a map	F28 UR28	High	Y1	WP3, WP6, WP2	
HMI-3	Manual mode **shall** overrule automatic mode	UR3	High	Y1	WP3, WP6	
HMI-4	The data gathered by the system **shall** be displayed to the operator in a user-friendly way.	UR1	High	Y2	WP3, WP2	
HMI-5	Cow geofencing: the operator **should** receive a warning when a cow gets close to previously established borders	F8, F23, F24 UR19, UR24	Low	Y2	WP3, WP2, WP4	

HMI requirements (HMI)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
HMI-6	The operator **shall** receive a warning if a cow shows unusual behaviour that could be indicative of sickness, such as staying in one place for long periods of time or not drinking water.	F8-F13, F23, F24 UR19, UR24	High	Y3	WP4, WP2, WP3	Details of sickness-related behaviours shall be consulted with experts at a later date.
HMI-7	The operator **should** receive an alert if a cow is being excluded by the rest.	F8, F13, F23, F24	Low	Y3	WP4, WP2, WP3	The excluded cow should be individually identified for the operator.
HMI-8	Data required by the users and higher-level inferred information **shall** be available and represented in a GUI, in a georeferenced way, eventually layered, when applicable	UR1	High	Y2	WP3, WP2, WP4	
HMI-9	The farm operator **shall** be able to consult information, notifications and alerts about provided milk quality (FPCM), estimated CO2 production and estimated water consumption.	F26	High	Y2	WP3, WP4	

HMI requirements (HMI)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
HMI-10	A mission management tool **shall** allow farmers to generate plans for the support of the coordinated actions to be carried out by (semi)-autonomous vehicles (ground and aerial).	F28 UR6	High	Y1	WP3, WP6	
HMI-11	A decision support system **shall** support farmers providing pre-mission data analysis, real-time data analysis during missions and post-mission data analysis.	F2, F4, F7, F16, F22 UR1	High	Y2	WP3 (support: WP2, WP4)	The DSS must be domain specific, and a set of problems must be chosen for this purpose.
HMI-12	The system **should** define user groups that can access different HMI solutions (or technologies).	UR1	Medium	Y3	WP3	E.g., an HMI solution for mission management will be different than those for AUV control; or solutions that show more or less detailed information (see Chapter 5).
HMI-13	The system **shall** provide a user interface to configure which data is sent from a vehicle to the cloud	F28	High	Y2	WP3, WP6	

HMI requirements (HMI)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
HMI-14	Sensor data **shall** be visualized in form of dynamic charts	UR4, UR10	High	Y2	WP3, WP2, WP4	
HMI-15	User **should** be allowed to define alerts based on observed values or their timeline	F1-F16, F18, F20-F26, F28	Medium	Y3	WP3	
HMI-16	The system **should** be able to generate reports based on the collected data (for example milk productivity, milk quality, animal health status, etc.)	F26 UR22, UR27	Medium	Y3	WP3	
HMI-17	The user interface **shall** be compatible with the system architecture of the vehicle (e.g., CAN, Ethernet, etc)	F19, F28 UR3-UR5	High	Y3	WP3, WP6	

2.4. Communication requirements

Communication requirements (COM)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
COM-1	There **shall** be (at least) a local network to support systems' communications in all demonstrators.	To support all functionalities	High	Y1	WP2, WP7	
COM-2	Communication between different vehicles (i.e., UAVs, UGVs, and where applicable, legacy systems) **should** be supported depending on the scenario requirements (i.e., if in the scenario there are both UAVs and smart tractors, then UAV and GV connectivity may be considered).	F28	Medium	Y3	WP6, WP2	
COM-3	Communication between Wireless Sensor Networks (WSN) nodes **should** be possible through multi-hop communications.	To support all functionalities	Medium	Y1 (at least in 1 scenario), Y2 (other scenarios)	WP2	

Communication requirements (COM)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
COM-4	The in-vehicle gateway in the tractor **should** provide communication between the cloud and the onboard ISOBUS and CAN bus.	F17, F19, F28 UR2, UR7	Medium	Y2	WP6, WP2	
COM-5	The in-vehicle gateway in the tractor **should** be able to collect data from vehicle on-board systems (e.g., diagnostic, usage, operational scenarios) and forward them over-the-air to the cloud.	F17, F19, F28 UR5	Low	Y2	WP6, WP2	
COM-6	The system **should** be able to send software update from the cloud to a tractor through an in-vehicle gateway for over-the-air (OTA) firmware or software update.	F17, F19, F28	Low	Y3	WP6, WP2	
COM-7	Communication between battery powered WSN nodes **shall** be realized by energy efficient technologies (with low battery consumption).	DoW	High	Y1	WP2	

Communication requirements (COM)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
COM-8	Communication link between ruminal probes and concentrator/relay **should** be based on technology that is not hindered by body tissues.	F8-13, F25	Medium	Y1	WP5	
COM-9	Software updates **should** consider vehicle status (e.g., if an update can be applied without influencing ongoing vehicle operation).	F17, F19, F28 UR3	Medium	Y3	WP6/WP2	Especially in a domain where the availability of vehicles and machinery is so critical, it is important to ensure that software updates are only applied if the timing suits the planning of farming tasks and the vehicle is able to ensure correct operation after an update.
COM-10	Software updates of UAV **should** not be attempted while the UAV is in use.	F17, F19, F28 UR3	Medium	Y3	WP6/WP2	

Communication requirements (COM)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
COM-11	The vehicle **should** be able to test if a software update was successfully applied and revert back to an operational status if the update was not successful	F17, F19, F28	Low	Y3	WP6, WP2	
COM-12	The AFarCloud communication architecture **shall** support various transmission ranges (depending on the communication technology and use case): long (beyond 1 km), medium (between 1 km and 100 m), and short (below 100 m)	DoW UR8	High	Y1	WP2	
COM-13	The AFarCloud communication architecture **shall** support various data rates (depending on the communication technology and use case): very high (higher than 50 Mb/s), high (between 1 Mb/s and 50 Mb/s), medium (between 10 kb/s and 1 Mb/s), low (between 250 b/s and 10 kb/s), very low (below 250 b/s).	DoW	High	Y1	WP2	

Communication requirements (COM)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
COM-14	Information **shall** flow among different types of networks (e.g., from a local WiFi network to a long-range LoRa network), i.e., interoperability between heterogeneous networks (i.e., with “translation” between different communication protocols) will be guaranteed.	DoW UR8	High	Y2	WP2	
COM-15	The AFarCloud communication architecture **shall** provide different levels of data reliability (e.g., TCP or UDP transmissions)	DoW	High	Y1	WP2	
COM-16	The AFarCloud communication architecture **shall** provide different levels of real-time communication/timing constraints (e.g., hard/soft real-time or non-real-time)	DoW	High	Y1	WP2	

Communication requirements (COM)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
COM-17	The AFarCloud communication architecture **shall** provide different levels of communication energy efficiency, depending on the considered use case and specific node lifetime objective	DoW	High	Y2	WP2	
COM-18	The AFarCloud communication architecture **shall aim at** using a common (single) network layer protocol across the various demonstrators. Tentatively IP (v4 o v6) for very high, high, medium and low bit rates and LoRaWAN for very low bit rates.	DoW	High	Y1	WP2	Non-proprietary protocols allow addressing in the same way computers and WSN nodes (OpenThread).
COM-19	The AFarCloud communication architecture **should** allow the creation of remote disconnected networks.	DoW	Medium	Y2	WP7	When the involved distance is too high the existence of remote disconnected networks should be possible.
COM-20	Low delay future communications (like 5G) **should** be kept in mind, at least between the UAVs and the local network servers	DoW	Medium	Y2	WP6, WP2	For some mission planning and collision avoidance algorithms, it's preferable to distribute the code in order not to drain the

Communication requirements (COM)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
						UAV battery. But to keep the algorithm validity the request, processing and response must be done under ~100ms.
COM-21	Remote data collection **shall** be supported by the deployment of one or more WSN.	F1	High	Y1	WP2, WP5	The number of nodes will depend on the specific demonstrator (e.g., a large area may be covered by hundreds of nodes organized in different WSNs)
COM-22	Remote data collection **shall** be realized by guaranteeing communication among UAVs/GVs and the cloud through proper communication protocols (5G for very high transmission rates or LoRa for very low transmission rates).	F17, F19, F28	High	Y3	WP6, WP2	

Communication requirements (COM)

Req no.	Req. Description	Source	Priority	Deadline	Responsible WP	Comment
COM-23	Remote animal monitoring **shall** be implemented by the communication from body-worn sensor nodes (e.g., collars on cows), possibly equipped with GPS for geo-localization, and the cloud through proper communication protocols (e.g., LoRa for long-distance communications, while pasturing, or WiFi for short-distance communications, with cows close to the barn).	F17, F19, F28	High	Y2	WP5, WP2	Collected data examples: GPS animal position, behaviour, etc.

2.5. Distributed intelligence requirements

Distributed intelligence requirements (INT)					
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**
INT-1	The system **shall** improve the profitability of crops production in the farm.	F15, F19, F20 UR10, DoW	High	Y3	WP2, WP3, WP4, WP5, WP6
INT-1-1	The system **shall** determine the status of the plant through the monitoring of the soil and climate conditions.	F1, UR12, UR18, UR31	High	Y3	WP3, WP4, WP5
INT-1-2	The system **shall** be able to determine when to water the crops.	F15, F16, UR1, UR31	High	Y3	WP3, WP4, WP5
INT-1-3	The system **shall** be able to predict diseases on crops.	F14, UR10	High	Y3	WP3, WP4, WP5
INT-1-4	The system **shall** be able to detect presence of weeds.	UR16	High	Y2	WP4, WP5
INT-1-5	The system **should** be able to monitor macro and micronutrient levels.	UR29	Medium	Y3	WP3, WP4, WP5
INT-1-6	The system **shall** be able to determine the best time to harvest within a reasonable timeframe.	F4, UR17	High	Y3	WP3, WP4, WP5
INT-1-7	The system **shall** help to reduce the usage of pesticides in the crops by 30%	F5 UR18	High	Y3	WP3, WP4, WP5

Distributed intelligence requirements (INT)					
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**
INT-2	The system **shall** improve the profitability of livestock and milk production in the farm.	F10, F13 UR19, UR23, UR25. DoW	High	Y3	WP2, WP3, WP4, WP5, WP6
INT-2-1	The system **shall** help improve animals' welfare w.r.t. standards measures.	F10, F11, F12, F13, F23, F24, F25, F27 UR19, UR21, UR22, UR23, UR24, UR25, UR28.	High	Y2	WP2, WP3, WP4, WP5. Welfare improvement will be specified by the experts in the project.
INT-2-2	The system **shall** monitor the location of cows.	F8, F23 UR24	High	Y3	WP4, WP5
INT-2-3	The system **shall** monitor the activity of cows (steps, and other movements).	F13 UR19	High	Y3	WP4, WP5
INT-2-4	The system **shall** monitor eating and rumination periods, and **shall** estimate daily intakes, grazing habits and watering habits.	F25, F27 UR21, UR22	High	Y3	WP4, WP5
INT-2-5	The system **shall** allow calving detection of animals	F13 UR25	High	Y3	WP4, WP5

Distributed intelligence requirements (INT)					
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**
INT-2-6	The system **shall** allow in heat detection of animals	F13 UR20	High	Y3	WP4, WP5
INT-2-7	The system **shall** estimate reproduction rates of animals	F13 UR23	High	Y3	WP4, WP5
INT-3	In WSN, the system **should** notify the coordinator node when another node leaves the system.	DoW	Medium	Y3	WP5
INT-4	In WSN, the system **shall** provide node heartbeat mechanism	DoW	High	Y2	WP5
INT-5	The system **shall** be designed for efficient distributed mining of large-scale amounts of data. Data exchange should be small or compressed.	DoW	High	Y2	WP2
INT-6	The system **shall** be designed to guarantee interoperability between communicating entities in the platform (e.g., using a well-defined API)	DoW	High	Y2	WP2
INT-7	The data from heterogeneous sensors, third-party systems, and external data warehouses **shall** be pre-processed, aggregated and fused by appropriate algorithms, running at sensor nodes (dew computing), at border gateways/platforms (edge computing), and in the cloud (cloud computing).	DoW	High	Y3	WP4, WP5

Distributed intelligence requirements (INT)					
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**
INT-8	Processing of large amounts of data **shall** be possible to enable descriptive analytics (e.g., to monitor raw milk quality, CO2 production or water consumption), and predictive analytics (visualize and forecast milk production and other parameters e.g., raw milk quality, CO2 production or water consumption).	F2, F4, F16, UR1	High	Y3	WP2
INT-9	Knowledge extraction algorithms **shall** use pre-processed data, and/or raw data, stored in AFarCloud repositories	DoW	High	Y2	WP2, WP4
INT-10	Knowledge extraction algorithms **shall** provide output to AFarCloud repositories, i.e., DBs and/or ontology	DoW	High	Y2	WP2, WP4
INT-11	DSS and MMT decision making algorithms **shall** query AFarCloud repositories to exploit pre-processed data and/or extracted information in their decision processes	DoW	High	Y1	WP2, WP3, WP4
INT-12	The system **shall** anonymize sensitive data before sharing externally.	DoW	High	Y3	WP3
INT-13	The system **should** be able to determine when a remote disconnected network needs its data to be collected (by UAV or (U)GV)	DoW	Low	Y3	WP5, WP6

Distributed intelligence requirements (INT)					
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**
INT-14	Cooperative data maps used for navigation and awareness **should** be shared and used by the multiple UAVs	DoW	Medium	Y3	WP6

2.6. Sensor and WSN requirements

Sensor requirements (SEN)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
SEN-1	The quality control software based on computer vision in vineyards **should** require the weather parameters (air temperature and vapour pressure deficit).	F1, UR12	Medium	Y2	WP4, WP5	The weather parameters are needed to calculate water stress.
SEN-2	The system **should** monitor the environmental conditions in the vineyard through a Wireless Sensor Network (WSN). The parameters to be measured are: air temperature, humidity, leaf wetness, rainfall, soil temperature, soil moisture, solar radiation, atmospheric pressure, wind speed and wind direction.	F15, UR31	Medium	Y2	WP4, WP5	

Sensor requirements (SEN)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
SEN-3	The system **shall** gather soil information from the WSN including soil moisture, temperature, electrical conductivity and various depth.	F14, UR31	High	Y1	WP4, WP5	
SEN-4	The system **should** provide a specification about the registration of computing nodes in the distributed system	DoW	Medium	Y2	WP4	
SEN-5	The system **should** allow to pre-process on the sensor level	DoW	Medium	Y3	WP4	
SEN-6	The system **should** monitor the conditions of an individual cow in the herd.	F9, F10, F11, F12, F13, F24, UR21, UR23	Medium	Y2	WP4, WP5	
SEN-7	The system **should** backup/cache measured data if connection is temporarily lost.	DoW	Medium	Y3	WP4	
SEN-8	The system **shall** monitor the environmental conditions in both the stable and the surroundings	F1, DoW (stable), UR12	High	Y2	WP4, WP5	
SEN-9	The system **shall** exchange sensor data by standardized interfaces and protocols	DoW	High	Y1	WP5, WP4, WP2	

Sensor requirements (SEN)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
SEN-10	The system **shall** provide sensor data in format suitable for integration with systems for herd management	UR7	High	Y1	WP5, WP4, WP2	
SEN-11	The system **should** register metadata for sensors (accuracy, range, calibration, location, maintenance, validation…)	DoW	Medium	Y2	WP5, WP4	

2.7. Safety and security requirements

Safety and security requirements (SEC)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
SEC-1	The system architecture **shall** have a security by design principles approach	UR3 DoW	High	Y2	WP2, WP6	
SEC-2	Novel components for GV **shall** be developed along ISO25119.	DoW	High	Y3	WP6, WP2	

Safety and security requirements (SEC)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
SEC-3	The in-vehicle gateway for the tractor **shall** provide mechanisms for authentication to prevent identity spoofing.	DoW	High	Y2	WP6, WP2	
SEC-4	The in-vehicle gateway for the tractor **shall** provide mechanisms for data integrity to prevent tampering of over-the-air data transmission to and from the cloud.	DoW	High	Y2	WP6, WP2	
SEC-5	The in-vehicle gateway for the tractor **shall** provide data confidentiality and sender/receiver authentication for data in transit over the Internet (from the cloud and to the cloud).	DoW	High	Y2	WP6, WP2	
SEC-6	End-to-end data consistency **shall** be ensured.	DoW	For SL(0): Low Others: High	Y2	WP6, WP7	Depends on the defined security level (SL) Similar to SEC 7 and SEC 21

Safety and security requirements (SEC)

Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
SEC-7	A security risk assessment **shall** be performed for security relevant system components, to define the necessary security.	DoW	High	Y1	WP6 / WP7	A security risk assessment procedure will determine the necessary security level.
SEC-8	The communication between the different actors **shall** be secure by adequate security measures to prevent data spying and data manipulation.	DoW	For SL(0): Low Others: High	Y2	WP6 / WP7	Depends on the defined security level (SL) Similar to SEC 5 and SEC 21
SEC-9	Only authenticated users **shall** have access to the user interfaces and the system components.	DoW	For SL(0): Low Others: High	Y2	WP6 / WP7	Depends on the defined security level (SL)
SEC-10	The control system **shall** provide the capability to identify and authenticate all human users.	DoW	For SL(0): Low Others: High	Y2	WP6 / WP7	Depends on the defined security level (SL)

Safety and security requirements (SEC)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
SEC-11	The assigned privileges of an authenticated user (human, software process or device) **shall** be enforced to perform the requested action on the control system and monitor the use of these privileges.	DoW	For SL(0): Low Others: High	Y2	WP6 / WP7	Depends on the defined security level (SL)
SEC-12	The integrity of the control system **should** be ensured to prevent unauthorized manipulation.	DoW	For SL(0): Low Others: High	Y2	WP6 / WP7	Depends on the defined security level (SL)
SEC-13	The confidentiality of information on communication channels and in data repositories **should** be ensured to prevent unauthorized disclosure.	DoW	For SL(0): Low Others: High	Y2	WP6 / WP7	Depends on the defined security level (SL)
SEC-14	In case of detected security violations, the proper authority **should** be notified, the event **should** be reported and corrective actions **should** be taken timely manually.	DoW	For SL(0): Low Others: Medium	Y2	WP6 / WP7	Depends on the defined security level (SL)

Safety and security requirements (SEC)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
SEC-15	The availability of the control system against the degradation or denial of essential services **should** be ensured. System loads by resource management **should** be prevented.	DoW	For SL(0): Low Others: Medium	Y2	WP6 / WP7	Depends on the defined security level (SL)
SEC-16	Software updates **should** be performed periodically when provided from the control system supplier	DoW	For SL(0): Low Others: High	Y2	WP6 / WP7	Depends on the defined security level (SL)
SEC-17	The system **should** include mechanisms to create roles and groups of users to give certain permissions to users who belong to a group or have a certain role. This way, we can quickly change the permissions of many users at once. In addition, we can have a much more specific user differentiation.	DoW	High	Y2	WP6 / WP7	Depends on the defined security level (SL)

Safety and security requirements (SEC)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
SEC-18	Safe states and the shutdown paths for all autonomous vehicles **shall** be defined for all states of a mission.	F28	High	Y2	WP6 / WP7	shutdown path = way to reach the safe state. Safety relevant
SEC-19	In case of collision of a vehicle with other vehicles or obstacles, the emergency shutdown path of all vehicles in the mission **shall** be triggered	F28	High	Y2	WP6 / WP7	i.e., triggered by the airbag sensor of the colliding device Safety relevant
SEC-20	An emergency stop button **shall** be mounted at the main operator’s viewpoint as well as in the cabin of manned vehicles. This emergency stop button **shall** trigger the safe state of all devices in the mission with as less software functions as possible.	F28	High	Y2	WP6 / WP7	Safety relevant
SEC-21	Security mechanism towards Man-In-the-Middle and/or replay attacks **should** be implemented.	DoW	Medium	Y2	WP6 / WP7	Depends on the defined security level (SL)

Safety and security requirements (SEC)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
SEC-22	Security mechanisms for end-to-end communications **shall** be implemented, if possible.	DoW	For SL(0): Low Others: High	Y2	WP6 / WP7	At the Sensor Node level, it is possible that some of them don't have the capability/processing power for standard security such as HTTPS. Depends on the defined security level (SL) Similar to SEC 5 and SEC 21

2.8. Cloud services requirements

Cloud services requirements (CLOUD)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
CLOUD-1	AFarCloud users **shall** be able to provision resources without any	DoW	High	Y1	WP7	Self-service with ease of use - The service management functionality should tie into

Cloud services requirements (CLOUD)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
	interaction with the service provider's staff.					the broader offering repository such that defined services can be quickly and easily deployed and managed by the end user.
CLOUD-2	AFarCloud users **shall** be able to access resources over the Internet using ubiquitous clients (e.g., a web browser) from a range of client devices (e.g., smartphones, tablets, laptops).	DoW	High	Y1	WP7	Ubiquitous access to cloud computing means that any person with the right credentials, from any device through any network connection and from any country should be able to access the platform.
CLOUD-3	AFarCloud platform **shall** support resource pooling, multi-tenancy and dynamic assignment of computing resources based on customer demand	DoW	High	Y2	WP7	Multi-tenancy ensures that every customer is on the same version of the software. As a result, no customer is left behind when the software is updated to include new features and innovations.
CLOUD-4	AFarCloud platform **shall** support rapid elasticity so that resources can be quickly provisioned and released, sometimes automatically, based on demand	DoW	High	Y1	WP7	The cloud platform should ensure that applications have the exact system capabilities they need at all times. The cloud hosting provider should allow clients to expand or minimize services as needed,

Cloud services requirements (CLOUD)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
						resulting in an efficient and convenient scaling process.
CLOUD-5	AFarCloud platform **should** support metering of services compatible with usage and charging models	DoW	Medium	Y2	WP7	Service Management - To productize the functionality of cloud computing, it is important that administrators have a simple tool for defining and metering service offerings. A service offering is a quantified set of services and applications that end users can consume through the provider — whether the cloud is private or public. Service offerings should include resource guarantees, metering rules, resource management and billing cycles.
CLOUD-6	Updated Heterogeneous Systems Support - Cloud management **shall** leverage the latest hardware, virtualization and software solutions with high-availability and resilience.	DoW	High	Y1	WP7	To be fully reliable and available, the cloud needs to be able to continue to operate while data remains intact in the virtual data centre regardless if a failure occurs in one or more components. Additionally, since most cloud architectures deal with shared

Cloud services requirements (CLOUD)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
						resource pools across multiple groups both internal and external, security and multi-tenancy must be integrated into every aspect of an operational architecture and process. Services need to be able to provide access to only authorized users and in this shared resource pool model the users need to be able to trust that their data and applications are secure.

2.9. Cloud resources monitoring requirements

Cloud Resources Monitoring requirements (CLDMON)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
CLDMON-1	The AFarCloud cloud resource monitoring component **shall** monitor the availability of the contracted cloud resources. This availability **shall** be measured in principle as uptime and shall be compared with the one offered by the cloud service provider (CSP) in its Cloud Service Level Agreement (CSLA). In the event the threshold is passed, an alarm **shall** be triggered, and the user informed.	DoW	High	Y1 (first version)	WP4	Monitoring of the availability of a cloud service offering
CLDMON-2	The AFarCloud cloud resource monitoring component **shall** monitor the performance of the contracted cloud resources. This performance metric **shall** be compared against a metric inserted by the user. In the event the threshold is passed, an alarm **shall** be triggered, and the user informed.	DoW	High	Y2	WP4	Monitoring of the performance of a cloud service offering
CLDMON-3	The AFarCloud cloud resource monitoring component **shall** monitor the response time of the contracted cloud resources, initially taking also into consideration the	DoW	High	Y2	WP4	Monitoring of the response time of a cloud service offering

Cloud Resources Monitoring requirements (CLDMON)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
	latency of the network. The response time **shall** be compared against a value inserted by the user. In the event the threshold is passed, an alarm **shall** be triggered, and the user informed.					
CLDMON-4	The AFarCloud cloud resource monitoring component **shall** monitor the use of resources in the Virtual Machine (VM).	DoW	High	Y2	WP4	Monitoring of the use of resources Among others, it is expected to measure the following aspects: memory use, disk use, CPU use (%). Other values to measure the use of resources will be evaluated in the course of the project. A formula is expected to be generated for this purpose.

Cloud Resources Monitoring requirements (CLDMON)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
CLDMON-5	The AFarCloud cloud resource monitoring component **shall** monitor the workload of the jobs and the timeliness of the job execution. This will in particular provide answers on statistical distribution of job execution time to allow detecting outliers in the job execution and providing means for adaptation of how jobs are scheduled.	DoW	High	Y2	WP4	Monitoring of the workload of the jobs deployed in the cloud
CLDMON-6	All violations **shall** be logged, and the log **shall** be obtainable by the users. The log **shall** hold the following parameters and values: • CSP Id/info • Violated parameters • Value of violated parameters • Time and date of parameters The log **should** be read only, hashed and signed by the AFarCloud cloud resource monitoring component.	DoW	High	Y2	WP4	Log of violations
CLDMON-7	The component Cloud Resource Monitoring **should** include a User Interface where the following data can be included by the user:	DoW	Medium/High	Y1 (availability) / Y2	WP4	User Interface of Cloud Resource Monitoring

Cloud Resources Monitoring requirements (CLDMON)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
	• Endpoint of the cloud service contracted (e.g., IP) • Availability as per the SLA • Maximum response time (in ms) • Maximum Performance (TBD) Furthermore, the UI **shall** also provide a dashboard where the monitored values can be easily seen by the operator of the application.					

2.10. Development Tool requirements

Development Tool requirements (DEV)						
Req no.	**Req. Description**	**Source**	**Priority**	**Deadline**	**Responsible WP**	**Comment**
DEV-1	The development of dependable autonomous systems **shall** be assisted by a design flow management tool.	DoW (TO5)	High	Y2	WP6	A design flow management tool ensures a definable development process from the requirements gathering process over to the different design phases until to the final validation process.
DEV-2	The implementations **shall** be validated by proper validation methods.	DoW	High	Y2	WP6, WP7	A validation verifies the correct implementation and ensure a safe and secure operation. MoMuT tool is being developed for the model-based test case generation.
DEV-3	The collection of safety evidences **shall** be assisted by AVLs FSM-Tool.	DoW	High	Y2 (M22)	WP6	A Functional Safety Management Tool should help the consortium responsible for managing safety activities to collect evidences and coordinate safety tasks.
DEV-4	A set of simulated tests **should** be run prior to any software upgrade.	DoW	Medium	Y1	WP7	A complete set of tests should be defined and developed, in order to ensure the proper

						function of all the modules whenever any change is made in the system. This set needs to be adapted over the time to consider and test the new features.
DEV-5	The LoRa network **should** be implemented and validated for multi-robot scenarios.	DoW	Medium	Y1	WP7	Using multiple UAVs simultaneously requires a stable and reliable communication between them. Using DDS to communicate requires the proper setup of a network.
DEV-6	The multispectral images and passive sensors dataset **should** contain data from fields others than the demonstrators.	F20/F6/F3	Medium	Y3	WP6	The quality of the fields' condition evaluation benefits from the existence of more data. Having data (properly labelled) from fields around the world, not only helps in the evaluation of a test field, but also prevents the possible development of local-specific algorithms (due to some possible characteristic that is common to the small set of the demonstrators but is not common to a wider set).
DEV-7	The communication times of passive sensors **should** be properly evaluated.	DoW	Medium	Y2	WP3, WP6	The aerial vehicles read the passive sensors data by descending near to them, and then generate the signals to communicate. Knowing the times needed for this process is

						crucial to the mission planning, as the flight time and manoeuvres needed are greatly dependent on the available energy/battery.
DEV-8	The design flow management tool **shall** support the security risk analysis process – according IEC62443	DoW	High	Y1	WP6, WP7	The CSFlow tool supports the security risk analysis process for a selected real demonstrator part (Austrian Use Case) and the AFarCloud architecture in general.
DEV-9	The design flow management tool **shall** support security level implementation process for the defined security zones and conduits - according IEC62443	DoW	High	Y2	WP6, WP7	The CSFlow tool supports the security level implementation process for the defined security zones and conduits of a selected real demonstrator (Austrian Use Case) and the AFarCloud architecture in general.
DEV-10	The design flow management tool **shall** support the selection of proper security counter measurement (requirements) and the assignment of the final archived security level	DoW	High	Y3	WP6, WP7	The CSFlow tool supports the selection of proper security counter measurement (requirements) and the assignment of the archived security level of a selected real demonstrator (Austrian Use Case) and the AFarCloud architecture in general.

3. Data Requirements

This chapter gathers the kind of information that will be used and exchanged inside AFarCloud. The purpose is to list the data that needs to be included in the AFarCloud information model to be defined in Task T2.4.

3.1. Regions

Regions will be defined by the position of the centroid (latitude and longitude) of the area and the width and length of the area. This applies to:

- **Weeds region**: area where weeds are found. Multiple weeds regions could be sent simultaneously to the MW.
- **Dead plants region**: area where dead plants are found. Multiple dead plant regions could be sent simultaneously to the MW.
- **Water stress region**: area where water stress is detected. Multiple water stress regions could be sent simultaneously to the MW.
- **Generic/non-specific area**: area that does not have a specific category

Data	Units	Data type	Range of values	Granularity/ Resolution	Invalid/ default value	Description/comments
Area: Latitude	decimal degree	Double	From -90 to +90	0.00001 decimal degree	Invalid: -1000	Latitude of the centroid of the area
Area: Longitude	decimal degree	Double	From -180 to + 180	0.00001 decimal degree	Invalid: -1000	Longitude of the centroid of the area

Data	Units	Data type	Range of values	Granularity/ Resolution	Invalid/ default value	Description/comments
Area: Width	Meter	Float	From 0 to 500	0.01 meter	Invalid: -1	Width of the region expressed in meters.
Area: Length	Meter	Float	From 0 to 500	0.01 meter	Invalid: -1	Length of the region expressed in meters.

3.2. Environment data

Data about the environment normally provided by observations from the sensors or exchanged with other systems.

Data	Units	Data type	Range of values	Granularity/ Resolution	Invalid/ default value	Description/comments
Air relative humidity	%	float	0-100%	0,1%		Data provided by sensors.
Air quality (eCO2 and eTVOC)		float				Data provided by sensors.
Air temperature	ºC	float	-40 to 80	0,1ºC		Data provided by sensors.
Air ambient temperature	ºC	float	-40 to 100	0,01ºC		Data provided by sensors.

Data	Units	Data type	Range of values	Granularity/ Resolution	Invalid/ default value	Description/comments
Atmospheric Pressure	hPa	float	300 – 1100	0.01		Data provided by sensors.
CO2 equivalents	ppm	integer	250 - 32768	1		
Light Sensors	lux	Integer				Data provided by sensors.
Rainfall	Millimetres (mm)	float				Data provided by weather service.
Wind speed	meters/second (m/s)	Float				Data provided by weather service.
Wind direction	N, NE, E, SE, S, SW, W, NW	Integer				

3.3. Crops and soil data

Data related to crops.

Data	Units	Data type	Range of values	Granularity/ Resolution	Invalid/ default value	Description/comments
Diameter plant (Dendrometer)	cm	float	0-5		Invalid: -1	Data provided by a dendrometer

Data	Units	Data type	Range of values	Granularity/ Resolution	Invalid/ default value	Description/comments
Equivalent Total Volatile Organic Compounds (eTVOC)	ppb	integer	0 - 32768	1	Invalid: -10	Data provided by sensors
Leaf wetness		float	1-15			
Soil moisture	%	Integer	~250 - 820	1	Invalid: -1	Data provided by sensors
Soil volumetric water content	%	float	Dielectric constant 10-80	0.05	Invalid: -1	Data provided by sensors
Soil total ions conductivity	S/m	float	Conductivity 0-120 mS/m;	0.1	Invalid: -1	Data provided by sensor
Soil temperature	ºC	float	-55 - 125	0.0625 - 0.5	Invalid: -100	Data provided by sensor
Water used to irrigate	Litres (l)	Float			Invalid: -1	AS-04 is interested in measuring the amount of water needed to obtain a ton of grapes. Rainfall should also be considered.

Data	**Units**	**Data type**	**Range of values**	**Granularity/ Resolution**	**Invalid/ default value**	**Description/comments**
Dry matter	g/kg	Float	0 - 1000		Invalid: 0-1	Amount of dry matter (DM) in sample (from sensor)
D-value	g/kg	Float	0 - 1000		Invalid: 0-1	Dry matter digestibility / DM (from sensor)
Crude protein	g/kg	Float	0 - 1000		Invalid: 0-1	Amount of Protein in sample (from sensor)
NDF-fiber	g/kg	Float	0 - 1000		Invalid: -10	Amount of Non-Digestible Fiber in sample (from sensor)
Grass height	cm	Float	0 - 100		Invalid: -10	Measured grass height (entered manually)
Yield sample weight	g/m^2	Float	0 - 10000		Invalid: -10	Weight of a yield sample (entered manually)
Fresh yield	kg/ha	Float	0 - 100000		Invalid: -10	Fresh yield per hectare (calculated)
Dry yield	kg/ha	Float	0 - 100000		Invalid: -10	Dry yield calculated from DM and fresh yield (calculated)

3.4. Livestock and milk quality data

Data related to cows, their feeding and the quality of the milk.

Data	Units	Data type	Range of values	Granularity/ Resolution	Invalid/ default value	Description/comments
Acetic acid concentration - rumen	mmol/l	float	20 - 120	0,1	Value, Quality, Time (VQT) format	Data provided by ruminal probe. Usual value: 45 - 100
Activity	steps/day	integer	0 – 50.000	1	VQT format	Data provided by sensor/actuator. Pedometer activity data – steps. Usual value: 180 per hour
Animal body temperature	ºC	float	20 - 50	0,1	VQT format	Data provided by sensor
Butyric acid concentration - rumen	mmol/l	float	5 – 30	0.1	VQT format	Data provided by ruminal probe. Usual values: 8 - 20
Dissolved ammonia concentration - rumen	mmol/l	float	2 – 30	0.1	VQT format	Data provided by ruminal probe. Usual value: 5 - 20
Dissolved methane concentration - rumen	mmol/l	float		0.1	VQT format	Data provided by ruminal probe.
Lactic acid concentration - rumen	mmol/l	float	0 - 100	0,1	VQT format	Data provided by ruminal probe. Usual value: 0 – 3.3

Data	Units	Data type	Range of values	Granularity/ Resolution	Invalid/ default value	Description/comments
Milk - somatic cells	cells/ml	integer	0 - 5 000 000	1	VQT format	Data provided by the Herd management system.
Milk conductivity	mS/cm	float	5 - 30	0,1	VQT format	Data provided by the herd management system. Usual value: 4 - 15
Milk temperature	ºC	float	0 - 40	0,1	VQT format	Data provided by sensor or milking machine/robot.
Milk yield	L	float	0 - 150	0,1	VQT format	Data provided by the herd management system. Usual value: 0 - 80
Position: Latitude	decimal degree	float	From -90 to +90	0.00001 decimal degree	Invalid: -1000	Animal position. GPS/GNSS data
Position: Longitude	decimal degree	float	From -180 to + 180	0.00001 decimal degree	Invalid: -1000	Animal position. GPS/GNSS data
Propionic acid concentration - rumen	mmol/l	float	5 - 60	0,1	VQT format	Data provided by ruminal probe. Usual value: 10 - 40

Data	Units	Data type	Range of values	Granularity/ Resolution	Invalid/ default value	Description/comments
Ruminal ORP (oxido-reduction potential)	mV	float	-1000 - 0	1	VQT format	Data provided by ruminal probe. Usual value: -300 to -100
Ruminal pH	pH	float	4 - 9	0,01	VQT format	Data provided by ruminal probe. Usual value: 5 - 7.5
Ruminal temperature	°C	float	20 - 50	0,1	VQT format	Data provided by ruminal probe. Usual value: 32 - 42
Cattle weight when fattening starts	decimal degree (Kg)	Float	0.0 to 1000.0 Kg Kg = kilograms	0.5 Kg	0.0 Kg	Cow-related data
Fertility rate per year	Calf (Number of calves born per year) One calf per year and cow is the optimal rate	Integer	1-3 (calves)	1 calf	0 calves	Cow-related data
Calving	Dimensionless	Boolean	True/false			The cow is giving birth

Data	Units	Data type	Range of values	Granularity/ Resolution	Invalid/ default value	Description/comments
Behaviour habits (in heat animals)	In heat time happens in periodic intervals (the fewer days, the better). About 17 days of the heat cycle, the heat lasts from a few hours to a day. Days	Integer	0 to 30	1	0	The cow is in a state where she is ready to mate with a male animal
Watering	Measured as an average of litres of water drunk by 20 calves per year	Integer	to 10000 litres	1 litre	0 litres	

Data	**Units**	**Data type**	**Range of values**	**Granularity/ Resolution**	**Invalid/ default value**	**Description/comments**
Rumination	seconds (timespan, e.g., 3 hours grazing, 2 hours ruminating, etc.) Integer number	Integer	0-36000 s	1 s	0 s	The cow is ruminating
Dying off	Dimesionless	Boolean	True/false			The cow is in critically bad health
Vaccination t	€ per head of livestock	Float	0.0 to 10000.0 € € = Euro	0.5 €	0.0 €	Vaccines received by the cow
Number of kg of feed/animal year	decimal degree (Kg)	Float	0.0 to 1000.0 Kg Kg = kilograms	0.5 Kg	0.0 Kg	
Feed costs per year	decimal degree	Float	0.0 to 10000.0 € € = Euro	0.5 €	0.0 €	
Note/remark on animal		string	-	-		Personnel comments on animal health status, treatment, …

3.5. Ground Vehicle data

Data related to ground vehicles.

Data	**Units**	**Data type**	**Range of values**	**Granularity/ Resolution**	**Invalid/ default value**	**Description/comments**
Hitch Draft Sensor	kn	Float	-60kn to + 60kn	0.01kn		
Hitch Position Sensor	Percentage (%)	Float	0 to 100%	0,10%		
Inertial sensor		Double				Inertial measurement units with: acceleration (m/s/s), angular rate (rad/s), magnetic field (Gauss), orientation(rad/quaternions/rotation matrix), pressure (mPa)
Inner status Signals		Integer	From 0 to x	1	Default: 0	Gears, Diagnosis information, etc.
Linear Position	mm	Float	0.5 – 2 mm	0.1 mm		To measure the linear movement.
Position: Latitude	Degrees	Integer	From -90º to +90º	1º	Invalid: -1000º	Ground position. NMEA Standards - GNSS Data

Data	**Units**	**Data type**	**Range of values**	**Granularity/ Resolution**	**Invalid/ default value**	**Description/comments**
Position: Longitude	Degrees	Integer	From -180º to + 180º	1º	Invalid: -1000º	Ground position. NMEA Standards - GNSS Data
Speed of the vehicle	kph	Float	From 0 to 40kph	1/256 kph	Invalid: -1	Velocity
Speed sensors	rpm	Float	from 0 to 8031rpm	1/8 rpm	Invalid: -1	Engine Speed, Transmission speeds, Powertake Off
Total fuel used	l	Float		0.5	Invalid: -1	
Fuel level	Percentage	Float	From 0.0% to 100.0%	0.4%	Invalid: -1	
Number of km travelled per ground vehicle						
Number of driving hours per ground vehicle	h			3 minutes		This information is needed to check if the farmer reduces costs in the use of vehicles (i.e., looking for cows). To be calculated by checking the timestamp related to the moment a mission starts and ends.

Data	Units	Data type	Range of values	Granularity/ Resolution	Invalid/ default value	Description/comments
Amount of fertilizers/pesticides applied per ground vehicle	l	Float				This parameter should be calculated per mission. Only for ISOBUS tractors and ISOBUS implements.
Identification of the tractor		String				

3.6. UAV data

Data	Units	Data type	Range of values	Granularity/ Resolution	Invalid/ default value	Description/comments
UAV Identifier		Integer				
Position: Latitude	Decimal	Float	From -90 to +90	0.00001 decimal degree	Invalid: -1000	Vehicle position. GPS/GNSS data
Position: Longitude	Decimal	Float	From -180 to + 180	0.00001 decimal degree	Invalid: -1000	Vehicle position. GPS/GNSS data

Data	Units	Data type	Range of values	Granularity/ Resolution	Invalid/ default value	Description/comments
Position: Altitude	Decimal	Float		0.0001 m	Invalid Min.Float	Absolute, 0 at sea level
Speed of the vehicle	Decimal	Float	From 0 to 20 m/s	0.1 m/s	Default: 0	
Orientation: Yaw	Decimal	Float	[0, 180]	0.1	Invalid: -1	
Orientation: Pitch	Decimal	Float	[0, 45]	0.1	Invalid: -1	
Orientation: Roll	Decimal	Float	[0, 45]	0.1	Invalid: -1	
Battery: capacity	Decimal	Float			Invalid: -1	Capacity in Ah (last full capacity)
Battery: percentage	Decimal	Float	[0, 1]		Invalid: -1	Charge percentage on 0 to 1 range

4. General Architecture

4.1. Architecture approach for sharing of resources among farms

There are different alternatives when defining the type of data access for farms. Depending on whether there will be data sharing or not between farming entities, different types of architecture designs can be specified:

a. Isolation: each scenario is isolated and there is neither sharing of information nor services among farms. Each scenario would deploy its own AFarCloud architecture, with its own database in an isolated and dedicated repository in the Cloud. Data is owned by the farm and being in the Cloud guaranties persistence of resources and data accessibility from anywhere.
b. Federation: In this case, some scenarios share information or services among them. Each scenario would deploy its own AFarCloud architecture, with its own database in an isolated and dedicated repository in the Cloud. However, in this case, some farms are federated, which implies some resource or data sharing. Replication services would be implemented to add the information to be shared to all the databases that are federated.
c. Centralization: all the scenarios share the same AFarCloud database, which is deployed in the Cloud. Access to the information should be filtered by the farms.

In AFarCloud, we propose a combination of these data sharing scenarios as illustrated in Figure 1.

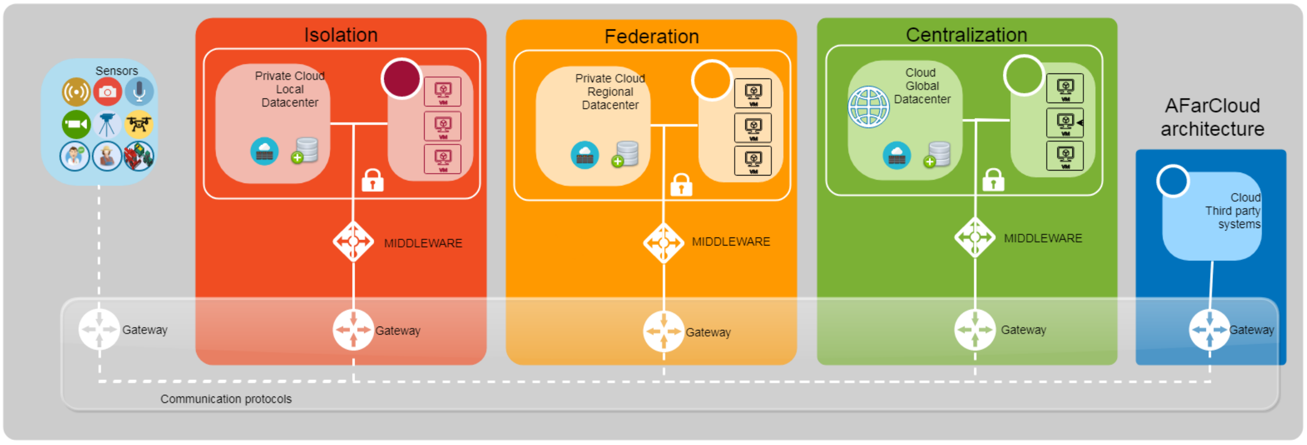

Figure 1: AFarCloud data sharing perspective

Data gathered from the Farm Management System, will be pre-processed and stored in a private cloud database in the farm level (*isolation*). The data processing will take place locally using centralised algorithms.

It will be possible to forward to *federation* some processing results that produce critical data (such as key performance indicators or events e.g., diseases). Each participant will be able to select among different federations (e.g., all farms in the same region or in the country or all farms with vineyards, etc). AFarCloud should be able to propose the most relevant/appropriate federations.

Additionally, it may be interesting that some of the data maintained in the federation level could be sent or processed in a central level (*centralisation*). The results obtained from central processing would benefit all the participants.

This architecture requires a specific amount of processing power and storage capacity that will be obtained with the help of cloud services.

4.2. Functional and components architecture

The AFarCloud platform will consist of three main functional components:

- The Farm Management System
- The Semantic Middleware
- Deployed Hardware

Besides the above-mentioned functional components, the AFarCloud platform will interconnect with other data sources like third-party data and legacy systems databases. Figure 2 depicts in more detail the functionalities and protocols covered in AFarCloud while Figure 3 represents the share of responsibilities among WPs for the AFarCloud architecture.

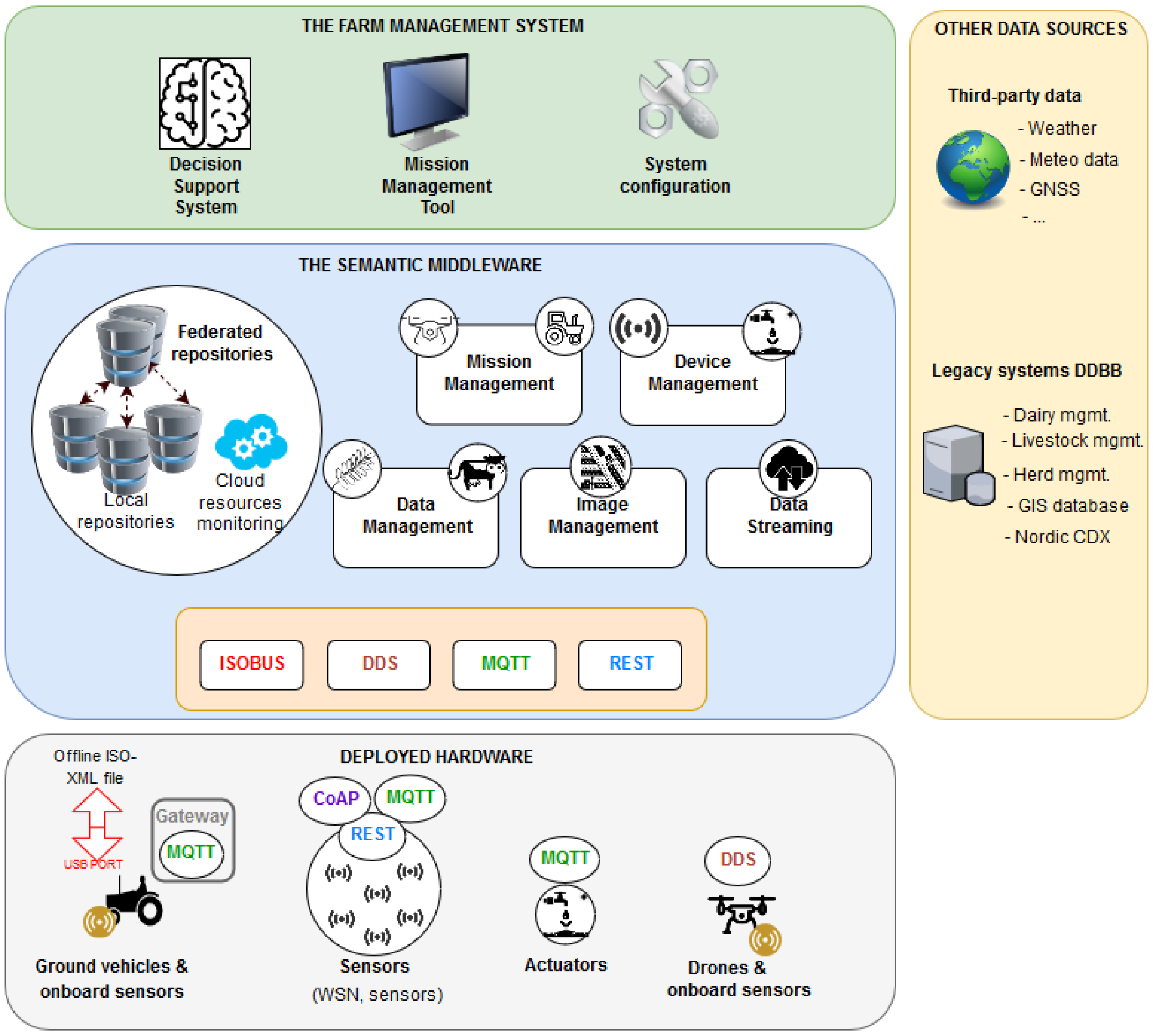

Figure 2: AFarCloud architecture

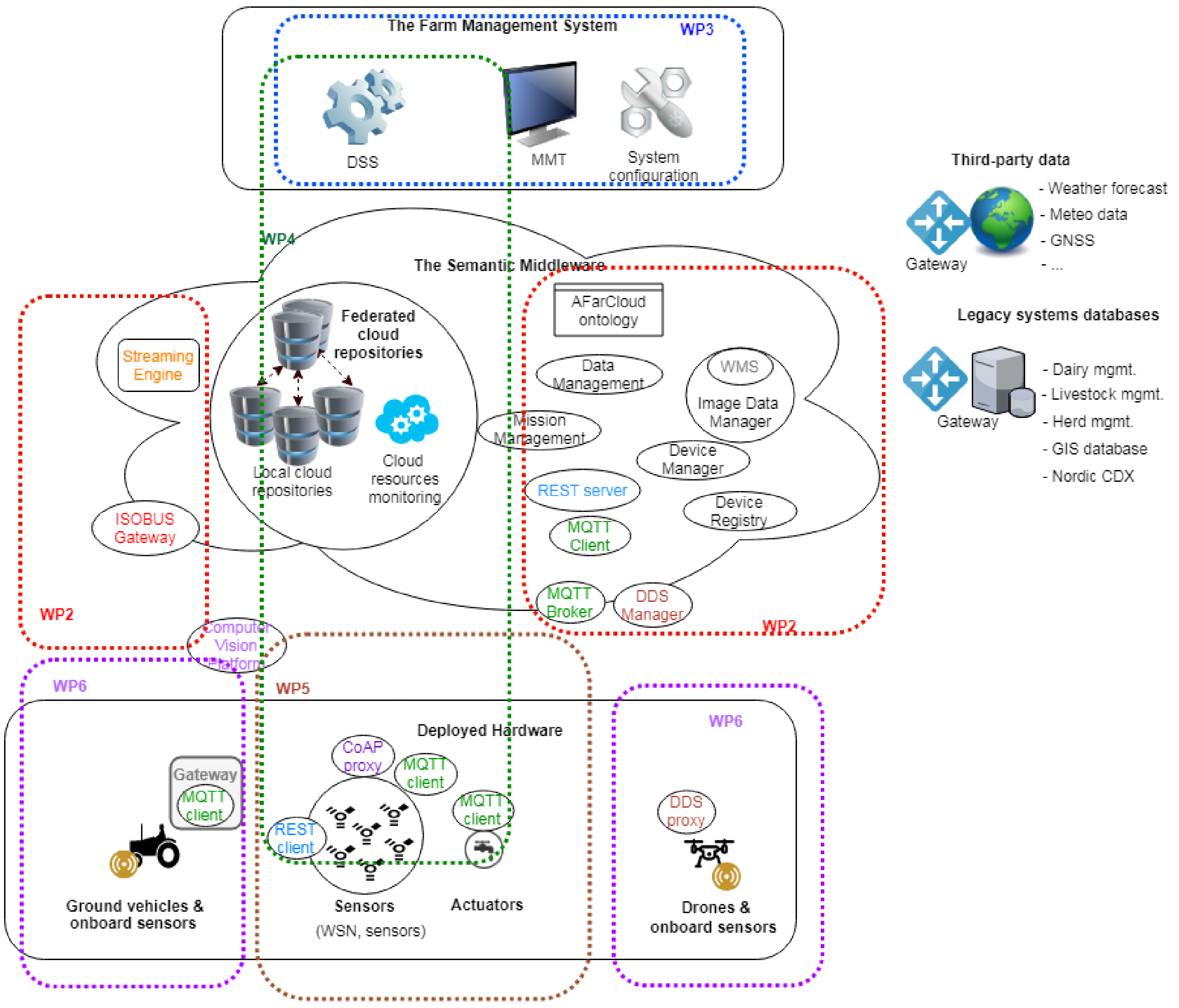

Figure 3: How WPs are reflected in the AFarCloud architecture

An initial description of each of these functional components is outlined below.

4.2.1. The Farm Management System

__Description:__

The Farm Management System will offer a Mission Management Tool (MMT), a Decision Support System (DSS), a system configurator and applications for the user to manage and monitor the whole system, plan cooperative missions involving unmanned aerial vehicles and ground vehicles ranging from fully autonomous UGVs to legacy systems; configure the above-mentioned systems including their key hardware components (mission relevant sensors and other component important for performing a mission); and make decisions pre-, during-, and post-mission.

__Functionalities:__

The main functionalities identified until now (M12) to be provided by this component are described below:

- MMT (incl. graphical user interface):
    - Maps with locations of ground/aerial vehicles and sensors (if possible)
    - Alarms
    - Mission management of ground/aerial vehicles:
        - Mission planning and control
        - Mission progress monitoring
        - Ground/aerial vehicle status monitoring
- Decision support system pre-, post-mission, and for real-time data analysis during the mission:
    - Crop monitoring, including vineyard related processes
    - Animal health monitoring
    - Milk quality monitoring
- System configuration:
    - Configuration of the monitoring functionalities based on scenario selection;
    - Provide mechanisms to support the configuration of the system including UAVs, GVs and agricultural machinery prior to a mission in order to make sure that they are prepared for the mission and that the user can collect and analyse status reports;
    - Notification of available firmware updates for sensors, actuators and GVs;
    - Configuration of the sampling rate of sensors;
    - Sending commands to actuators;
    - Alarms configuration: definition of the alarms to be provided by the MMT to the operator;
    - (Graphical) interface for registering ground/aerial vehicles, actuators and sensors in the farm. Through this interface, the operator will provide the input information that will be used by the Device Registry of the Semantic Middleware.
- Cyber security management
- Reasoning based on existing data: e.g., to detect if an animal exits a dedicated area; to detect collision risks for vehicles, etc.

4.2.2. The Semantic Middleware:

Description:

A middleware is a software layer used to hide the underlying complexity of hardware, in distributed systems, so that application layers can access to that hardware in a unified way. The AFarCloud middleware will use semantic models, specified by an ontology to abstract the heterogeneity of the underlying hardware, and to ensure that all information is stored according to a common information model that guaranties interoperability. The semantic middleware will act as a communication centralizer, disseminating messages between the Farm Management System and the hardware deployed.

The Semantic Middleware will be in charge of unifying data coming from, or directed to, different types of cyber-physical systems deployed:

- WSNs, sensors and actuators;
- Semi-autonomous ground vehicles: these vehicles will be able to autonomously perform tasks defined by the Farm Management System. **It is important to highlight that, in AFarCloud, automation for ground vehicles is related to task execution (i.e., activation of an implement when a tractor reaches a given location), and not to vehicle navigation.**
- Autonomous aerial vehicles: these vehicles will be able to autonomously carry out missions defined by the Farm Management System. Fully autonomous navigation will be part of the mission for aerial vehicles.

Considering the heterogeneous nature of the elements deployed in the hardware layer, the semantic middleware will offer different types of interfaces:

- *DDS interface for fully autonomous vehicles able to implement autonomous navigation (i.e., UAVs)*: The Data Distribution Service (DDS) for Real-Time Systems protocol is a standard defined by OMG[2]. DDS provides a communication environment based on a publish/subscribe architecture which is very suitable for networks with moving nodes. The main DDS features are listed below:
    - Reliable, scalable and real-time data exchanges using a publish/subscribe pattern;
    - Automatic discovery of connected entities;

[2] https://www.omgwiki.org/dds/

- Automatic Quality of Service management for control over every aspect of data distribution, such as data durability (i.e., data persistency), resource usage, and reliability (i.e., guarantee the delivery of messages);
- Management of the data persistency: time and space decoupled reception and delivery of messages:
  - Space decoupling: IPs of the DDS nodes do not need to be static.
  - Time decoupling: if desired, nodes can receive multiple messages addressed to them, sent even before they are connected to the network. This is especially suitable for handling of unreliable environments with communication channels of poor quality and/or high latency, or to manage alerts (i.e., an alarm is sent to a UAV to avoid collision but at that moment, the UAV loses connectivity. As soon as the connectivity is established, the UAV will receive the alarm)
  - Unlimited buffer: DDS nodes can serialize unlimited buffers of samples. Multiple samples per topic are allowed. Even nodes that join late to a DDS partition will be able to receive all samples previously addressed to them.

In AFarCloud, all communications that are performed throughout both ends of the system (the Farm Management System and Deployed Hardware) related to the command and control of fully autonomous vehicles, will go through the DDS interface of the middleware, as DDS guaranties data persistency (no data is lost) and a real-time delivery of the data (essential for autonomous vehicles able to perform missions). The combination of both features, together with the deployment of data analytics techniques, would even allow the re-planning of ongoing missions if necessary. Thus, this DDS interface will be used by all aerial vehicles able to carry out autonomous AFarCloud missions defined by the Farm Management System and to send the data gathered during missions back to the middleware. All data exchanges will be compliant with the agreed data model.

DDS compliant vehicles will be able to directly exchange DDS messages between themselves. These messages would be defined by the project, in case it is needed: e.g., two UAVs could share an anti-collision plan (safety related plans).

- *ISO-XML/MQTT interface for compatible semi-autonomous ground vehicles able to implement tasks autonomously:* Due to their lack of auto-steer abilities, semi-autonomous ground vehicles will not be able to perform standard missions that involve autonomous navigation (i.e., go to a waypoint) defined by the planning algorithms in the Farm Management System.

However, they will be able to perform special missions defined by the Farm Management System, involving the execution of tasks autonomously, supervised by vehicle operators. To carry out these tasks, they will rely on their implements. The communication between the ISOBUS compatible implements of a semi-autonomous ground vehicle and the Farm Management System is standardized and simplified through the use of ISO-XML files. The ISO-XML format (described on the ISO 11783-Part 12 standard) is the only standardized way defined by ISOBUS for exchanging information between ISOBUS compatible entities.

The Semantic middleware will offer an interface able to generate an ISO-XML file to define each of these special missions. ISO-XML files will be manually loaded to semi-autonomous ground vehicles. Besides, this interface will also offer means to convert manually loaded XML files of log data collected from the CAN-bus of ground vehicles during a mission, in order to be uploaded into the middleware.

The Semantic Middleware will also offer a MQTT link that will be used by semi-autonomous ground vehicles to send data of their onboard sensors, gathered during the execution of missions.

MQTT (Message Queuing Telemetry Transport) is a publish/subscribe communication protocol standardized by ISO (ISO / IEC PRF 20922). It is light weight, open, simple, and designed so as to be easy to implement. These characteristics make it ideal for use in many situations, including constrained environments such as for communication in Machine to Machine (M2M) and Internet of Things (IoT) contexts where a small code footprint is required, network bandwidth is at a premium, and low energy consumption is highly desirable. Due to its characteristics and features it is considered as an optimal protocol to employ in a context such as the one posed by AFarCloud.

- *MQTT or REST interface for compatible WSNs, sensors, actuators:* The Semantic Middleware will offer two different interfaces (MQTT and REST) to all devices responsible for collecting measurements (WSN, standalone sensors, etc.) or for implementing actions (actuators). By exposing these two interfaces, the Semantic Middleware guarantees compatibility not only with the MQTT and REST protocols, but also with the CoAP protocol, since it is based on the REST model.

Most of the AFarCloud middleware is hosted by the cloud infrastructure that will be deployed in the project, to take advantage of the features provided by cloud resources. Cloud computing is based on the use of remote servers hosted on the Internet to manage infrastructure and data, which provides many benefits such as flexibility and scalability in infrastructure design, cost reduction or guaranteed reliability. Reliability is reached through cloud monitoring, which uses automated tools to manage the cloud infrastructure and services. Besides, a cloud deployment can be used no matter the type of architecture that will be adopted finally (isolated, federated, centralized). The AfarCloud cloud infrastructure will define interfaces based on a set of REST services.

Other components of the middleware need to be deployed at the Edge and/or in the facilities of the farm. These components are the following:

- The Image Processing Platform (IPP): due to the large size of the images needed by this component and taken by UAVs, the loading of these images is carried out offline (i.e., through a memory stick) for Y1, to minimize errors in the transmission of files and speed up the process of loading. From Y1 on, we will study the viability of doing some pre-processing onboard the UAVs in some of the cases.
- The DDS Manager: as this module is responsible for processing the real time communications with UAVs, it is deployed as close as possible to the place where data is generated, to minimize latencies.
- The Data Pre-Processor and the Data Fusion: processing data close to the source reduces latency as data does not have to traverse over the network to the Cloud for processing. By only sending important data over the network, the edge computing reduces both the data traversing the network, and the processing time.

Functionalities:

The main functionalities identified up to now (M12) is to be provided by the Semantic Middleware are described below:

- Data Storage:
    - A set of repositories (semantic, SQL and NoSQL) to store data in the cloud. In addition, as in AFarCloud not all the knowledge extraction will be done at the cloud level and there will also be intelligence at sensor level (edge level) when needed, the platform will provide an Edge Data Storage to give support in local processing done by the Data Pre-Processor and the Data Fusion. The Edge Data Storage is out of the scope of D2.2.
    - As explained in Section 4.1, a three-level cloud repository is proposed:

        - Local cloud repository (Y1): it's the private cloud repository of an individual farm.

        - Federated cloud repository (from Y1 on): it's a common cloud repository shared by all individual farms that are part of a specific federation of farms. Each participant should be able to select among different federations: e.g., all farms in the same region. The data to be shared in this common federated cloud repository must be agreed during the project. For the moment, several ideas have been proposed:
            - Sharing of high-level information inferred in farms: e.g., animal behaviour (why do cows give more milk in a farm?), gas emissions (why do cows emit less methane in a farm?), etc. Data too dependent on location and weather conditions is considered not relevant to be shared.
            - Sharing of resources like vehicles or human power between neighbour farms: e.g., calendar of occupation of resources.
        - Centralized repository (from Y1 on): a common repository shared by all scenarios. The data to be shared in this common centralized cloud repository must be agreed during the project.
    - Semantic cloud repository: a SPARQL server that provides access to the AFarCloud ontology
    - Historical data cloud repository: a SQL Database
    - Big data cloud repository: a NoSQL Database
    - Sensitive data will be encrypted
- Cloud Resources Monitoring:
    - Monitoring of the liveliness of all the middleware components deployed in the Cloud
    - WSN monitoring
    - Other sensors monitoring
- Data Interoperability (by means of a common AFarCloud Data Model):
    - Provides a Common Data Model whose purpose is to define a common view for the real environment that can be achieved by heterogeneous cyber-physical systems (e.g., vehicles, sensors, actuators, MMT, etc.) and humans, to enable the integration and cooperation of them.
    - The data model will be composed of three parts:

        - The *semantic data model* will represent the concepts managed by the AFarCloud project. These concepts and their associated properties will be depicted in form of graphs and formalized in the AFarCloud ontology.
        - The relational data model will represent the model of the SQL tables of the AFarCloud database;
        - The non-relational data model will represent the model of the NoSQL database in AFarCloud.
- Data Management:
    - <u>Data Access Manager</u>: CRUD operations to the cloud repositories (the AFarCloud ontology, the SQL and non-SQL databases). This component will allow, for example, to manage connections with the databases, or to execute SQL, NoSQL and SPARQL queries.
    - <u>Data Query</u>: to process any query (semantic, relational or non-relational) to store or retrieve data. It is the access point that other components must call to ask for or store data, without having to worry about where the data are stored or the syntax of the query. For example, the Data Query could offer a method to get the location of a cow (e.g., getCowPosition). As the Data Query knows where this information is stored, it will create the specific SPARQL, SQL or NoSQL query and will send it to the Data Access manager to be processed. This provides the user an abstraction level on the data storage infrastructure. This component will also be able to aggregate information stored in different cloud repositories.
- <u>Streaming Engine</u>:
    - Management of the delivery of real time data streams read from the AFarCloud cloud repositories to the DSS in the Farm Management System.
    - The component will be implemented in Apache Kafka.
- Device Management:
    - <u>Device Registry</u>:
        - Management of the registry of aerial/ground vehicles, actuators, sensors, animals, etc. in the farm. This process is performed every time a new element joins the AFarCloud platform.
    - <u>Device Manager</u>:
        - Management of specific operations to be implemented by standalone devices or groups of devices:
            - Notification of software updates (WSNs, sensors, actuators, gateways, etc.)

            - Actuators commands
            - Requests on devices (WSNs, sensors, gateways): e.g., change on sampling rate
        - Management of the data coming from standalone devices or groups of devices.
        - Device Manager manages data flows with connected devices (e.g., sensors, actuators, gateways, WSNs, etc.) while Mission Manager manages data flows with connected vehicles (i.e., GVs, UAVs).

- Mission Management:
    - <u>Mission Manager</u>:
        - Delivery of the mission to the semi-autonomous ground vehicles and UAVs.
        - Delivery of events to the semi-autonomous ground vehicles, UAVs or other (farming) devices/systems. We will consider as events all high-priority actions or data (i.e., command to abort a mission) sent by the middleware that should be considered by the devices deployed in the Hardware. Events will be generated by the Farm Management System as a result of an analysis of data.
        - Management of the data coming from vehicles (result of the mission, alarms, data from sensors and actuators on board, etc)
    - <u>Mission Processing & Reporter</u>: mission status report management. Validation of the mission status report to ensure that this information is clean, and its format is correct
    - <u>Alarm Processing & Reporter</u>: management of alarms from aerial/ground vehicles, sensors and actuators. Validation of alarms to ensure that this information is clean, and its format is correct. We consider as alarms messages sent from a device, sensor or vehicle to the middleware to inform about abnormal behaviour at equipment or functional levels.
    - <u>Environment Reporter</u>: deals with raw sensor data from aerial/ground vehicles and sensors, checking basic aspects (e.g., validating ranges), before forwarding them to the Data Pre-Processor, and/or before storing them in historical DB, both during missions (online), and pre- or post- mission (offline). The sensor reports sent by this component can be used by the Farm management System to be aware of any sensor failure(s) and react upon it if necessary.
    - <u>Data Pre-Processor</u>: performs online (raw) sensor data cleaning and filtering, to avoid missing values, impossible combinations and outliers.
    - <u>Data Fusion</u>: aggregates data already pre-processed by the Data Pre-Processor, from multiple sources, providing a more consistent, accurate and useful information. This data could be associating vehicles proprioceptive data to exteroceptive data acquired

by embarked environmental sensors. Data Fusion should be done right after data pre-Processing.

- Knowledge Extractor: exploits environment data (raw, pre-processed, fused or including metadata) related to livestock and/crops, to extract knowledge. The latter should be stored at the cloud repositories to be eventually used by MMT/DSS in the Farm Management System if needed. The outcomes from the Knowledge Extractor are validated by the Environment Reporter.

- Image Management:
  - Image Processing Platform (IPP):
    - The IPP will be a ground station deployed at the Edge of the architecture (in the farm facilities) that will process the images provided by sensors embarked in the UAVs. The images of a flight will be stored in a memory stick or SD card, and downloaded to the IPP.
    - By means of some deep learning algorithms, the IPP will "stitch" all images in a unique image. This image is known as a georeferenced/orthorectified mosaic.
    - In the IPP, this image will be processed by a set of image processing algorithms. As a result of this process, the platform will return a series of data. Three different customizations of the IPP will be provided, in order to adjust to each of the demonstrator needs: (i) livestock location and tracking; (b) vigour, water stress and weeds & dead plants detection; and (c) estimation of the main cropping indexes.
      - Generally, these data, and not the images due to their large size, will be stored in the cloud repository of the MW.
      - Only in cases where a long-term image storage is needed, the IPP will send specific images to the Image Data Manager in the MW for future reference.
  - Image Data Manager:
    - *Image processor*: the image processor will provide image data for the specified image or image set, the ground location, map projection and the image format selected. This component will offer an OGC WMS interface, so the Farm Management System will be able to retrieve images in the case the users need them. The image processor will also be used to improve the image geometries.
    - *Image catalogue*: the image catalogue will keep track of the image data location (images could be stored in a file system or in Amazon S3 compliant systems), the acquisition time, the geometric extent on the ground, the geometric models,

etc. A typical query to the catalogue could be: e.g., "*give me all Sentinel-2 images from July this year, covering this point*".

- Cyber security management:
  - Management of the cyber-security technologies to be implemented in the Semantic Middleware.
- MQTT Broker and MQTT Clients (Publisher & Subscriber) in the Middleware:
  - Management of communication with the MQTT devices of the hardware deployed. The QoS settings of the MQTT communications must be configured correctly according to the AFarCloud requirements.
- REST Server and REST services for data acquisition and actuation on devices:
  - Management of communication with devices of the hardware deployed offering a REST interface.
- ISOBUS Gateway:
  - Translation of missions for ground vehicles in the AFarCloud format, to an ISO-XML file to be manually loaded by a GV (i.e., through a USB port).
  - Translation of the log information of the execution of a mission by a GV, to the AFarCloud data format.
- DDS Manager:
  - Management of the DDS communications with the DDS compatible devices of the hardware deployed. The QoS settings of the DDS Manager should be correctly configured according to the AFarCloud requirements.

4.2.3. Deployed Hardware

This functional component will provide means to deploy and integrate the services and data provided by the ground/aerial vehicles, sensors and actuators into the AFarCloud platform. Deployed Hardware can be divided into four different categories:

1. Unmanned aerial vehicles (UAVs). The degree of autonomy will vary. Some UAVs will be fully autonomous and will be controlled by the MMT through the planning algorithm, whereas a few UAVs are deployed solely for specific sensors such as, hyperspectral imaging.
2. Semi-autonomous ground vehicles. There are two sub-categories (i) UGVs and (ii) tractors incl. legacy systems. In the former sub-category there are two types of vehicles, domain specific farming UGVs and general use UGVs. In the latter sub-category traditional farming vehicles are found.
3. Actuators

4. Sensors

The main functionalities identified up to now (M12) to be provided by this component are described below:

4.2.3.1. Unmanned Aerial vehicles (UAVs)

Aerial vehicles will perform different actions in AFarCloud: (a) collect data gathered by their onboard sensors during a mission execution and send it to the Middleware; (b) carry out special missions to collect data from short-range sensors that lack of an internet connection and forward it to the Middleware; (c) carry out other missions defined by the Farm Management System.

In AFarCloud, all communications between the Farm Management System and UAVs will use a DDS link.

A. Functionalities and features of UAVs:

- Vehicle interface management: translation of the actions in an AFarCloud mission to the specific language of the UAV. Translation of the data collected by the UAV (e.g., sensor data, UAV status, UAV alarms, etc.) to the data format defined in AFarCloud (D2.6)
    - Receiving missions and commands from the MW: The UAV's DDS Proxy will be subscribing to the topic *mission*. The DDS Proxy interprets the desired command and translates the information into a UAV specific command or message, accordingly. From this step, the message is treated as a normal command inside the UAV's framework. Besides, whenever a command/mission is received, the UAV will generate an acknowledgement message that is reported to the MW by means of the *mission_report* topic.
    - Sending UAV status to the MW. During all its operation, the UAV will be sending, periodically, messages concerning its state. The message sending to the MW is done through the IDL topics *pose* (for vehicle location and orientation) and *battery* (to inform the energy status of the overall system);
    - Sending sensor measurements to the MW. Sensor measurements will be divided into two sets: images and other measurements. The images taken by the onboard camera will be reported to the MW using the topic *image*. The data obtained by other onboard sensors will be reported in the topic *observation*.
- Alarms management: UAV error states detection and reporting. The UAV reports errors to the MW both by the topic *alarm* and *mission_report*. In the *alarm* topic errors and alarms related to the UAV's hardware integrity and/or availability will be reported. In the *mission_report,* are

reported errors with respect to the success (and failure) in the execution of a desired command/mission.

- Management of the cyber-security of communications.
- Management of reliability of communications. In order to ensure that the UAV is communicating, there are several alternatives. One possibility is to set a timeout in both the UAV and DDS Manager. From the DDS Manager side, this timeout can be set regarding the periodic messages i.e., the pose and battery topics. To use this alternative from the UAV side, it would be necessary to implement an acknowledge message whenever a new pose is received by the DDS manager (using, for example, the event topic). Another possibility is to use the built-in discovery service of DDS to detect the presence of publishers and subscribers (similar to a heartbeat message option).
- Software updates. The UAV functionalities might be affected with software updates. For this reason, the system will not manage automatic software updates. UAVs should only allow this kind of actions by a qualified person (to ensure the proper function of all the modules, a set of tests must be run and validated prior to any further flight).
- User interface (Mobile MMT). It will allow the user to visualize relevant data from both UAV fleets and single UAV information.

B. Interfaces of UAVs:

UAVs should implement a DDS Proxy in order to interface with the Semantic Middleware. The DDS Proxy could be implemented in two ways depending on the type of the UAV. We have two types of UAVs in AFarCloud: a) UAVs with open or accessible on-board software; and b) UAVs with proprietary software (closed systems). Making changes in closed systems is usually complicated, or occasionally, forbidden by the manufacturer. In addition, we should bear in mind that UAVs need accident insurance to fly. The introduction of modifications in a product can lead not only to the loss of the guarantee offered by the manufacturer, but also to the loss of the insurance policy. For this reason, we have designed two alternatives to integrate an UAV in AFarCloud.

B.1. Interface with the Semantic Middleware for open vehicles: in this case it will be possible to install the DDS Proxy onboard the UAV.

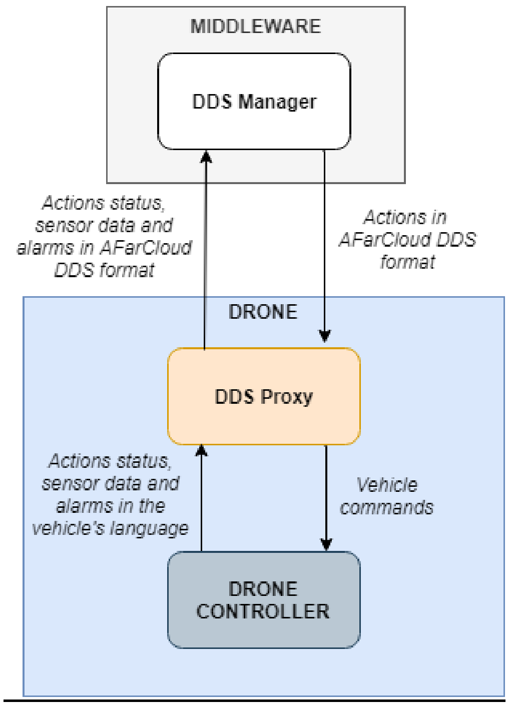
MIDDLEWARE
DDS Manager
Actions status, sensor data and alarms in AFarCloud DDS format
Actions in AFarCloud DDS format
DRONE
DDS Proxy
Actions status, sensor data and alarms in the vehicle's language
Vehicle commands
DRONE CONTROLLER

Figure 4: Interface with the Semantic Middleware for open vehicles

B.2. Interface with the Semantic Middleware for proprietary vehicles: in this case, the recommendation will be to install the DDS Proxy of each vehicle in the Ground Control Station (or mobile application) that manages the UAV.

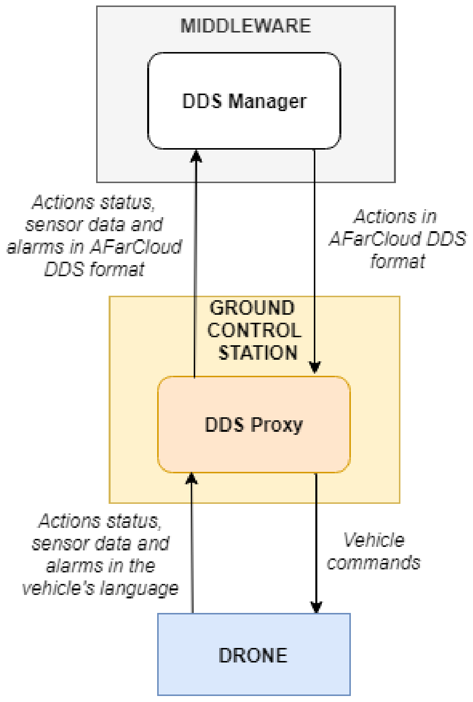
MIDDLEWARE
DDS Manager
Actions status, sensor data and alarms in AFarCloud DDS format
Actions in AFarCloud DDS format
GROUND CONTROL STATION
DDS Proxy
Actions status, sensor data and alarms in the vehicle's language
Vehicle commands
DRONE

Figure 5: Interface with the Semantic Middleware for proprietary vehicles

The goal of the DDS Proxy in both of the proposed architectures is the following:

- DDS Proxy: it provides a DDS compatible communication channel to the UAV, offering a bridge between the messages that the UAV is expecting (they could be based on ROS or any standard solution used by the UAV) and DDS messages. On the one hand, the DDS Proxy translates the DDS messages with generic commands issued by the DDS Manager of the Middleware into vehicle specific commands. On the other hand, it collects data, events and faults from the vehicle and translates these messages into AFarCloud DDS messages. DDS messages will be based on the data format for UAVs defined in D2.6.
  Apart from being the link with the Middleware, the DDS Proxy will also provide a DDS link with the DDS Proxy of other UAVs (if a direct UAV to UAV communication is needed).

4.2.3.2. Semi-autonomous ground vehicles (GVs)

In AFarCloud, we consider under this category ground vehicles that are not equipped with an autosteer system, (i.e., unable to perform autonomous navigation “go to waypoint” commands), but capable of executing actions autonomously.

A. Functionalities and features of GVs:

- Vehicle interface: load ISO-XML files containing a mission for a GV on the CAN bus of the GV. Unload log data from the GV’s CAN bus about the result of a mission execution. Translation of the data collected by the GV (e.g., CAN data, GV status, etc.) to the data format defined in AFarCloud (D2.6)
- Software updates management: allow operators/administrators to deploy secure firmware updates to GVs.
- Alarms management: GV error states detection.
- On-board user interface: a display showing information and alerts from the AFarCloud system to the operator.
- Management of the cyber-security of communications.

B. Interfaces of GVs:

- ISOBUS interface: communication at vehicle level between the GV and its implements.
- MQTT interface: to send data from sensors onboard the GV to the Middleware.
- USB interface: to load data (ISO-XML file obtained through the ISOBUS Gateway) into the GV’s CAN bus manually. To unload mission log data from the GV’s CAN bus manually.

4.2.3.3. Actuators

A. Functionalities and features of actuators:

- Actuator interface: translation of AFarCloud actions to the specific language of the actuator. Translation of the data collected by the actuator (e.g., actuator status, actuator alarms, etc.) to the data format defined in AFarCloud (D2.6)
- Software updates management: allow operators/administrators to deploy secure firmware updates to actuators.
- Alarms management: actuator error states detection.
- Management of the cyber-security of communications

B. Interfaces of actuators:

- MQTT interface: communication with the MQTT Broker of the MW.
- CoAP interface: communication with the REST services of the MW.

C. Available actuators:

The tractor and its implements can be considered as a group of actuators such as auxiliary valves, spraying nozzle, seeding rate etc. Their utilization as actuators is strongly dependant on the specific set of tractors and implements that will be available on the demonstrator sites.

The main actuators to be developed within the project are shown in Table 1.

Table 1. Overview of actuators within the project

Actuator(s)	Information	Functionality
AIR NTP actuator - sanitising device for air	Air treatment system for the sanitisation of microbiological contamination and chemical substances of indoor air	Moulds and microbiological contamination and filtering action for improvement of indoor air quality
WATER NTP actuator - treatment device for water	Water treatment system for production of treated water for irrigation of green-house or bounded area	Bio-stimulation of crop growth by using treated water and sanitization of contaminated surfaces
Electronic control unit (ECU) as gateway	ECU gateway for monitoring the data on CAN and ISOBUS network	It gives the possibility to receive commands or data from the “cloud” or from a WSN

4.2.3.4. Sensors

In AFarCloud, we consider under this category groups of dedicated sensors (e.g., WSNs, collars) and stand-alone sensors.

A. Functionalities and features of sensors:

- Sensor interface: translation of AFarCloud actions (e.g., change sampling rate) to the specific language of the sensor. Translation of the data collected by the sensor (e.g., observations, alarms, etc.) to the data format defined in AFarCloud (D2.6).
- Software updates management: allow operators/administrators to deploy secure firmware updates to sensors.
- Alarms management: sensor error states detection.
- Management of the cyber-security of communications

B. Interfaces of sensors:

- REST interface: communication with the REST services of the MW.
- MQTT interface: communication with the MQTT Broker of the MW.
- CoAP interface: communication with the REST services of the MW.

C. Available Sensors

A summary of the available sensors within the project is shown in Table 2.

Table 2. Overview of sensors used within the project

Sensors type
Multispectral sensor, thermal camera (crops, soil information) – FIXED / Handheld Multispectral sensor, thermal camera (crops, soil information) – MOBILE / UAV/ tractor
Soil (electrical detection - humidity, temp, conductivity) – FIXED Soil (spectral detection, humidity, temp, conductivity) – MOBILE
Environmental: T, Humidity, CO2, CH4, TVOC, light intensity, dendrometer, leaf wetness, etc.
Vehicle data and its implements (e.g., GPS position, current speed/ torque/ fuel consumption)

Aggregated ruminal probe (temperature, pH, ORP, etc.)
Inertial sensors - Smart collar (cow)

The proposed sensors have different degree of development, some are commercial, some are already developed by the partners, and some will be further developed during the project's duration. Depending on the utilized platform, there are sensors that will be mount on UAVs, sensors that will be connected to UGVs, attached to livestock as collars, placed under ground as soil sensors, and in green houses (other miscellaneous solutions will also be possible).

4.2.4. Other Data Sources

4.2.4.1. Third-party data

Third-party data will provide information for understanding the environment surrounding, the involved ground/aerial vehicles and sensors. For the case of simple information (i.e., numerical data like a temperature), the access to these data could be also done from the semantic middleware if needed. For the case of complex information like maps, a Geographical Information System (GIS) should be used for managing digital maps and available geographical data. The information of interest for the project could be, among others:

- Weather forecast
- Meteorological data
- Classified digital Satellite data/Air images
- Soil maps/Soil strata
- Digital terrain model and Aspect (N-S-E-W)
- Surrounding vegetation

4.2.4.2. Legacy systems databases

The AFarCloud platform will have access to the data and provided by several legacy systems databases:

- Nordic Cattle Data eXchange (CDX):
  - Nordic CDX is a REST API that provides data of milking stations and robots taken from the national cattle information systems of Finland, Sweden, Denmark and Norway.

    - Within the AFarCloud project, the access to cattle database via Nordic CDX API is allowed for Finnish implementation. Further, the access to the farm data is possible only for the farms that have allowed the access to their data. The access is granted during the existence of the AFarCloud project.
    - The possible integration in the AFarCloud cloud repositories of data of interest for the project will be studied.
- GIS database:
    - This GIS system provides access to a GEO database with topography (2M grid) soil maps, vegetation maps, hydrography and models to estimate soil wetness (in situ probes to give ground data through measurements of soil parameters, nutrients, etc.)
    - By using these data, farmers will be able to evaluate the spectral response from crops and suggest proper counter measures in case of sign of bad crop.
- Herd management systems (AfiFarm and FarmSoft):
    - These two systems offer information about data retrieved from herds of cows located in Czech Republic.
    - Access to these data will be provided and the possible integration in the AFarCloud cloud repositories of data of interest for the project will be studied.
- Sensowave’s livestock management system:
    - Offers information about the data taken by Sensowave’s ear tags and collars for cows. These devices provide data about animal location and health condition (heat, calving, temperature, etc.)
    - These data will be integrated in the AFarCloud cloud repositories.

4.3. Cyber-security management

4.3.1. Motivation

Cyber-security is a very important issue in modern agriculture, according to system security and data confidentiality. Today’s agriculture production plants are equipped with an uncountable number of interconnected computers and modern electronics equipment. These components define the system which must be considered in a cyber security risk assessment.
To perform a cyber security assessment, the location of the components of the architecture must first be considered. Figure 2 displays the associated components of the AFarCloud architecture, which shows:

(a) a set of components located outside on the field or in the stable:

   - ➔ Deployed Hardware

(b) a set of components located inhouse:
   - ➔ Hardware infrastructure for deployment (e.g., WiFi routers, computers, etc.)
   - ➔ Edge components of the Semantic Middleware
   - ➔ Farm Management System

(c) a set of components provided by third party companies. These components are out of the area of influence for the asset owner.
   - ➔ Third party data
   - ➔ Legacy systems databases

(d) a set of components deployed in the Cloud:
   - ➔ The Semantic Middleware, including the AFarCloud repositories (for the time being, the Cloud repositories are being provided by the project partners).

On three of the mentioned component areas (a, b and d) the asset owner has fully control over the security measures to be installed or used to harden the system against cyber security attacks. For the last area (c) the asset owner can only perform a very accurate provider selection and must trust that the defined security specifications are fulfilled. Security monitoring is the only activity the asset owner can perform to increase the confidence.
The following three security aspects define the motivation of cyber-security attacks (with examples for explanation). These aspects are in the focus of the cyber security management in AFarCloud:

<u>Espionage</u>

- Unauthorised Data access
- Data leakage
- Loss of know-how (IP) and production data
- Phishing
- Trojans
- IP Theft
- Spyware

<u>Destruction and Exaction</u>

- Causing physical damage to farming equipment
- Deterioration of product quality
- Ransomware
- Data manipulation
- Data destruction

<u>Sabotage and Mis-usage</u>

- Loss of availability of the farming equipment
- Loss of production

- Deterioration of product quality
- Botnets
- Distributed denial of service (DDoS) attack
- Man-in-the-middle attack

To limit the possible cyber-attack risks in AFarCloud an exemplary cyber-security management process must be installed to identify the system vulnerabilities with the possible attack vectors, and to propose suitable countermeasures.

4.3.2. Security improvement

A first step to improve security is to reduce the number of data interchange ports to a minimal necessary limit, using certified secure components and operating the internet activities with care.
A second step is to perform a cyber security threat analysis with a risk assessment to identify for the entire system the actual and the necessary security level.
Table 4 lists general security requirements, which define measures to improve the overall system security.

Tool support

For a large system the security management process, according to the concept of a dedicated security standard is a very complex task. This is due to the fact that a lot of steps and analysis, documentation and requirements preparation must be performed very accurately. These steps are mandatory for each system area, each system conduit and components, depending on the requested security level. In the course of the project, a security process management tool (CSFlow) - provided by AIT - will be used. This tool will be enhanced for security in agriculture. CSFlow guides the system design experts through all necessary design phases to perform all specified security steps, according to a dedicated security standard. For more details about security standard (Section 4.3.3).
A process management tool, as like the CSFlow™ tool represents a big support to manage the prescribed design steps and the implementation of the security requirements in a precise and traceable way.
The development of CSFlow and the detailed description of the cyber security management contribution will be performed in WP6 and in WP7 in a real security application for a demonstrator use case.
The cyber security process tasks must be coordinated with the asset owner, who must agree and support the mandatory steps to harden the system according to the state of the art.

It is very important that the system to be assessed, is described completely with a sufficient level of detail and correctly defined system borders. Defining the system in a not adequate way implies the hazard that important system security issues can be overseen, whereas a too large system definition may lead to unnecessary security risk analysis efforts which usually do not help to improve the security of the system.

The cyber security management process consists of the four main cycles:

- Security assessment and analysis
- Security measure installation
- Security guidance and
- Security verification

4.3.3. Safety / Security standard landscape

The overview of well-known safety and security standards shows that they come from the industrial and mobility (vehicles, train, avionic) domain. The focus of agriculture standards is in safety regulations, to prevent illnesses and injuries from agriculture work by using pesticides, save use of heavy machines and animal-friendly treatment of animals (livestock). Agriculture security standards mainly handle the area of food and nutrition security while for the agriculture IT / OT security no dedicated standards are defined at the moment. But in this case, today's well-established industrial control security standards are a perfect and good suitable source for applications.

The overview lists some cyber security standards from the IT / OT domain:

ISO/IEC 15408 establishes the general concepts and principles of IT security evaluation.

ISO/IEC 27000 provides the overview of Information Security Management Systems

ISO/IEC 27001 formally specifies a management system for information security

ISO/IEC 27002 describes guidelines for organizational information security and information security management practices.

ISO/IEC 27005 brings guidelines for information security risk management.

ISO/IEC 62443 Industrial Communication Networks - Network and System Security series

For the cyber security management process in AFarCloud, the **ISO/IEC 62443** security standard will be the base for all ongoing system assessments and cyber-security measure implementations.

The core goal in **ISO/IEC 62443** standard is to define Foundational Requirements (FRs) and Security Levels (SLs).

The SL defines the necessary protection level again potential attacks, such as shown in Table 3:

Table 3. Cyber security level definitions

SL 0	No specific requirements or security protection necessary
SL 1	Protection against casual or coincidental violation
SL 2	Protection against intentional violation using simple means with low resources, generic skills, and low motivation
SL 3	Protection against intentional violation using sophisticated means with moderate resources, IACS specific skills, and moderate motivation
SL 4	Protection against intentional violation using sophisticated means with extended resources, IT specific skills, and high motivation

The FR's defines the necessary measures to fulfil the requirements for the SL , in seven subgroups, Table 4.

Table 4. Foundational requirements overview

FR1 – IAC	Identification and Authentication Control ○ Password and user authentication
FR2 – UC	Use Control ○ Mapping of roles in the management process ○ System usage policy
FR3 – SI	System Integrity ○ Session handling, cryptography and monitoring to detect changes
FR4 – DC	Data Confidentiality ○ Encryption ○ End to end data encryption
FR5 – RDF	Restricted Data Flow ○ Less connection ○ Network segmentation
FR6 – TRE	Timely Response to Events ○ Event / Action Logging ○ Monitoring ○ Anomaly / Inconstancy detection
FR7 – RA	Resource Availability ○ System backup ○ System recovery

4.3.4. MQTT security

MQTT is a good opportunity to bring security in the IoT based control system, where secure data exchange and an end-to-end isolation between the data source and data sink is a main security topic for the system design concept.

Data exchange with MQTT can be protected by several well-applied security measures on the different network layers.

Network Layer: E.g. using of a secure network or VPN.

Transport Layer: Data encryption according the SSL (*Secure Sockets Layer*), or the new labelled TSL (Transport Layer Security) protocol specification.

Application Layer: Using data encryption and device authentication. MQTT support the authentication of devices with a client identifier and username/password credentials.

The applied security measures depend on the available computing resources in the MQTT clients, because the big plus of MQTT is the lightweight software footprint, which allows MQTT for small microcontroller applications, too.

4.4. Wireless Sensor Networks

The wireless sensor networks (WSN) is one of the key building blocks for fulfilling the AFarCloud objectives by allowing the extraction of the data that is collected by the sensors and making it possible that the data can be forwarded to the AFarCloud services.

An open field, a closed barn or a limited area have different requirements for the technology type of wireless that, in the sensor networks, can be used. While in closed/limited areas, the sensor networks have to deal with distances around tens of meters for wireless communication, ranges of hundreds of meters to kilometres are possibly required on open fields such as corn fields or olive fields, for example.

Wireless communications range depends on the type of technology used and can be divided in short range radios and long-range radios (see Figure 6).

With short range radios, to cover an area greater than a single radio can, wireless solutions can be built based on mesh networks and associated protocols to solve the radio signal range issue. For increased coverage, where it is not feasible to create a short-range radio mesh network, long range radios such as the ones built for LPWANs (Low Power Wide Area Networks) allow a solution for covering such extended areas.

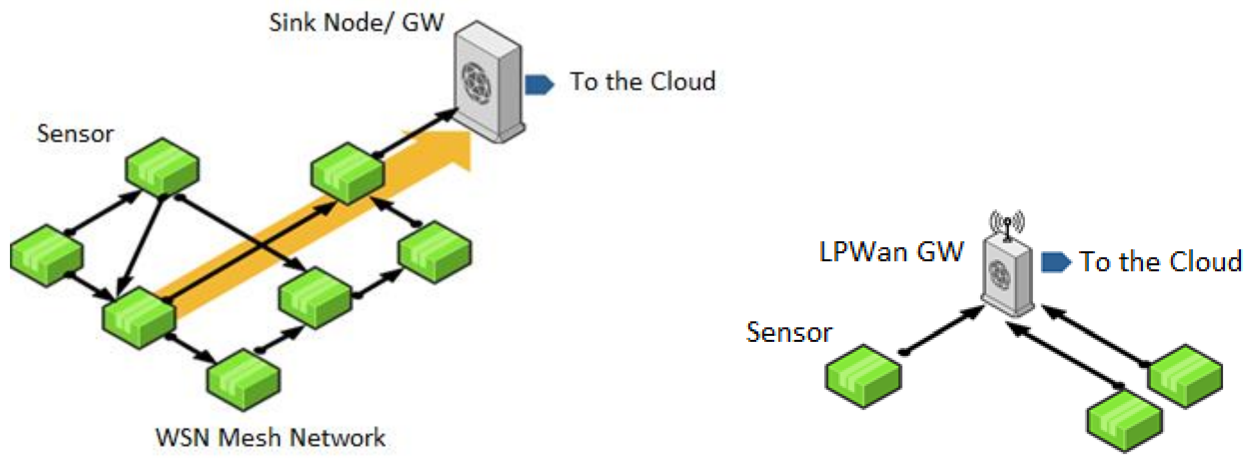

Figure 6: Short range radio mesh network (on the left), LPWAN long range radio (on the right)

Still, LPWANs increased range come with lower data rate throughput and higher latency as a cost of the increased range, thus is not ideal for continuous monitoring. For such purposes, the lower range devices offer higher throughput and lower latency but also have higher energy consumption.

As such important factors like the necessity for real time monitoring, data throughput, latency, range, energy consumption and quality of service, among other parameters, are the key aspects to consider on implementing a wireless sensor network for the AFarCloud project.

Other important factors for the sensor network, not related with the technical aspect of how data is transmitted, are the CAPEX (Capital Expenditure) and OPEX (Operational expenditure) of such networks. Direct costs such as the radio and gateway hardware and operational costs such mobile operator costs, battery maintenance need to be considered for a WSN solution.

4.4.1. AFarCloud Sensor networks

Annex 1. WSN Technologies contains a detailed description on WSN Technologies, an explanation about their pros and cons, and their possible usage for AFarCloud.

Specifically, for the scope of AFarCloud wireless sensor networks and based on Annex 1 description of available wireless supporting protocols, the following protocols are recommended:

Short-range networks:

6LoWPAN over 802.15.4 radios → Allows the creation of wireless sensor networks with constrained devices that can integrate seamlessly with IP networks. 6LoWPAN is an open protocol, has both the hardware for nodes and gateways available, and the protocol stack is available for several types of architectures (ARM, Atmel, Linux). Integration with higher level data transport protocols such as

MQTT, MQTT-SN and CoAP can also be done at the sensor level, without the need of a specialized protocol gateway.

WIFI → The standard for low range high speed networks, ideal for devices that are not power constrained.

Bluetooth → Bluetooth allows connectivity at the personal level and also supports 6LoWAN as a communication protocol. It is ideal as the communication protocol for supporting body sensors in body area networks, for example devices to monitor cattle body temperature and motion.

Bluetooth can also be used for presence detection, for example an animal wearing a collar that just advertises its Bluetooth ID, if non-mobile gateway on an area detects that advertisement, it can infer the animal presence in that area.

Long-range networks:

Without incurring in operator costs and radio coverage, the only viable solution for a LPWAN network is the implementation of a LoRa based network. Still, the initial CAPEX can be high due to the necessity of adding multiple LoRa Gateways for high availability purposes.

The CAPEX costs for a LoRa network can be levelled by the use of the LoRaWan standard that implements a network layer with a set of services that can be private or public over the LoRa radio protocol. LoRaWan can build networks that can be shared among farms and other users of the LoRaWan network and so use gateways that were deployed by other entities. An example of such open network is the TheThingsNetwork. In this case, users can use the available LoRaWan gateways or deploy their own, so they also take part of the network and in return, any gateway can relay sensor node information to end application/servers, in this case the AFarCloud servers.

4.4.2. Store and Forward networks

These networks are remotely located and are out of range of any connectivity to the AFarCloud servers either by LPWAN or other means of wireless connectivity. It can also be assumed that power on these networks is a scarce resource, which means that nodes are battery powered and may have solar power or other energy harvesting methods to keep batteries topped up. To save power most nodes are in sleep state, only waking up to gather the required sensing data.

The nodes sensing data can be stored temporarily in each node and collected, in a future time, by a fly by UAV or approaching vehicle. In this case the collecting vehicle must approach and be in range of each sensor to allow the data download and triggering the data download process. This process depends on the state of the node, if dormant or not and can consume a lot of the node energy.

Another approach is to let the network behave as a standard network with nodes and a gateway, where the gateway has a higher capacity power source backed up with solar or wind generator, thus it is always awake and ready to receive data. The gateway also has capacity for storage of all received data from all nodes, which means that the collecting vehicles only needs to visit the collecting gateways.

For such networks, LPWAN LoRa based network with a LoRa gateway and associated hardware for high speed data link to upload data to the collecting vehicle is an ideal solution. LoRa gateways can receive data from nodes that are kilometres away, and gateway hardware and software can be modified to support the store and forward network topology.

The software for this specialized gateway needs to be modified from the standard LoRa gateway software, so it packs the data into data units with an associated GUID (Global Unique Identifier), that is uploaded to the collecting vehicle. In a future pass for gathering more data, the collecting vehicle informs the gateway of the AFarCloud data units that were previously collected and successfully transferred, and so the gateway can delete the associated data unit safely.

5. The Farm Management System

5.1. The Mission Management Tool

5.1.1. Description

The goal of the Mission Management Tool (MMT) is to provide the operators with a central user interface, and a set of services accessed through this interface. These services aim at: (i) defining, (ii) planning, (iii) monitoring, (iv) controlling, (v) analysing, and finally (vi) saving mission-related data (incl. sensor data, status of all connected hardware such as sensors, actuators, robots/vehicles where applicable) in (i) – (v) of a mission. The stages of monitoring and control are performed during the mission and also include sensors as part of any system, and also systems that are not subject to programming. This means that all values relevant for the mission will be accessible to the operator. Some of these services are provided by various software solutions within the MMT itself whereas others are part of the FMS (i.e., the DSS and the System Configuration solutions), and other components that the MMT accesses through the MW. Thus, the MMT acts as a command and control centre for planning and supervising the missions performed by the vehicles i.e., UAVs, UGVs, and if applicable legacy systems.

It is assumed that the MMT is found in an office environment. Whereas the MMT will provide full functionality, the mobile MMT, accessed through an approx. +10-inch tablet, has the purpose of providing selected set of services when mobility near the mission site is required. In this context three different configuration of the mobile MMT are assumed:

1. Operator view: the generic mobile solution.
2. Tractor driver view: dedicated solution for placed in a tractor cockpit, and to be used together with other monitors.
3. UAV pilot view: dedicated solution for UAV pilots to be used together with the GUI of the base station.

Services for these three views are related to monitoring of the mission, and analysis of the data including access to the DSS. This means that planning, and control of a mission will not be possible to do through the mobile MMT (except solutions for termination of a mission due to safety and security risks). A detailed list of the services/capabilities of these both MMT and operator mobile MMT, including the differences, are as follows:

1. Providing the operator with maps from different sources containing different (and/or equivalent) type of information of an area. The area here can be any location, although it refers to the location of a mission;
2. Providing the operator with list of assets of the vehicles such as, their status and properties including their equipment such as sensors and actuators. This service is provided together with the System Configuration solutions;
3. Providing the operator with relevant information obtained from Decision Support System (DSS). This input can be shown as an overlay of the map or using appropriate modality;
4. Providing the operator with weather data and other external relevant information;
5. Allowing the operator to define “forbidden” zones which should be avoided by the vehicles (N/A to mobile MMT);
6. Allowing the operator to define mission goals that will be used by the planners for solving the problem (N/A to mobile MMT);
7. Communication with the planners to plan the actions of the vehicles in order to solve the mission (N/A to mobile MMT), sending the plan to the vehicles through the Middleware (N/A to mobile MMT);
8. If required, allowing the operator to modify the planned mission (N/A to mobile MMT);
9. Communication with the MW to receive status updates from vehicles;
10. Communicates with other Ground Control Stations (GCS) to inform them of the flight plan and mission objectives (N/A to mobile MMT). This is done through the Middleware;
11. During a mission, providing the operator with messages and alerts received from the vehicles;
12. Allowing the operator to abort or re-plan a mission and execute it (the mobile MMT will have a sub-set of functionalities).

Figure 7 shows an overview of the MMT and its connections with the Robotics Agents through the Middleware. In the figure one robotic agent with its three main components are shown. Such an agent is a (semi-)autonomous robot i.e., UAVs, UGVs, and if applicable, legacy systems, as well as non-moving equipment (which can be subject to hierarchical planning).

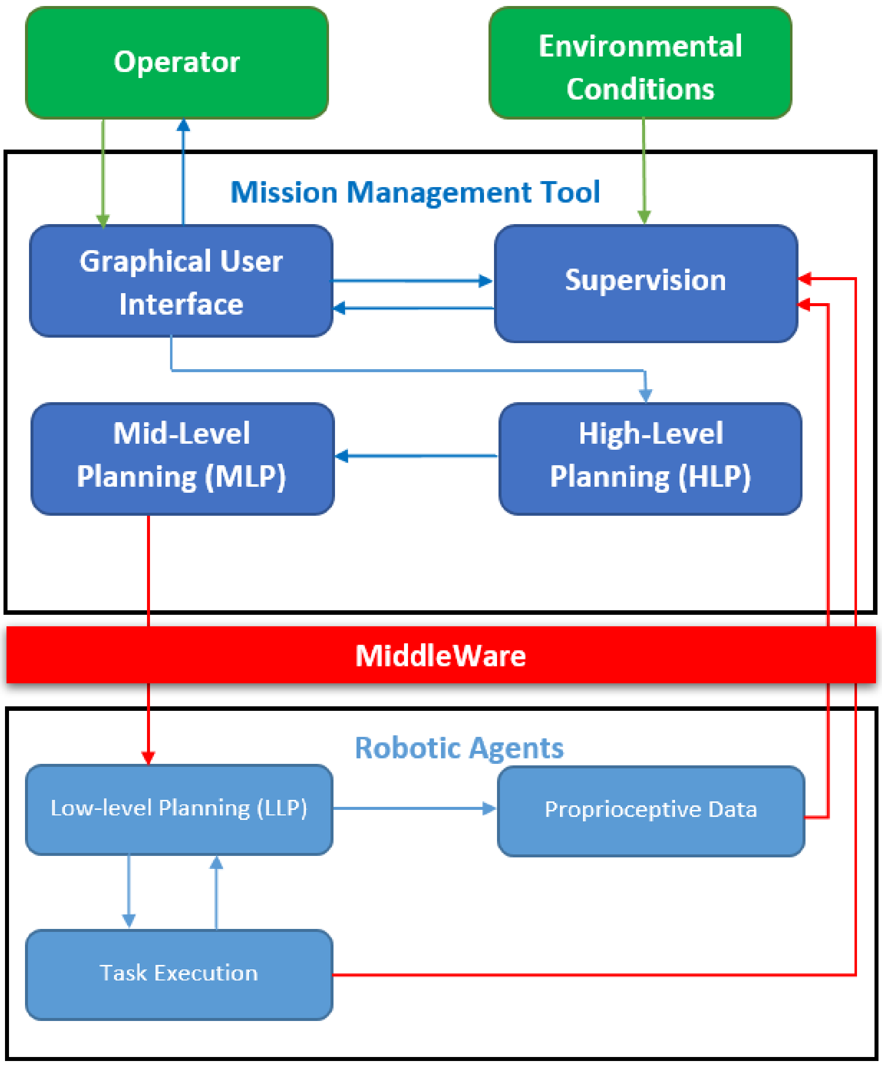

Figure 7: Overview of the MMT and its connections with the Robotics Agents through the MW.

5.1.1.1. The Graphical User Interface

As mentioned above the MMT provides the operators with a Graphical User Interface (GUI) for services 1-12 above. Since different categories of users will use the MMT, it is important to adapt the GUI to meet these requirements. Working with the plans of the missions will require a moderate to large screen size (15 inch or preferably above), whereas monitoring a mission, or visualization of the data on/near the field will be done with a tablet.

The MMT's GUI will provide the operator with a geographic map showing the mission area and different objects such as vehicles and sensors. Through this GUI, during a mission, it will be possible to visualize measured data, trajectories of the vehicles, as well as other relevant data such as UAVs battery time/consumption or other critical parameters).

The maps will be accessed through OGC Web Map Service (WMS) providers, including AFarCloud's Image Data Manager (IDM). This allows for integration of different maps from different sources, e.g., satellite maps, precipitation, weather maps, etc. Different maps can be presented as different layers providing additional information that the operator might need. MMT map presentations should be expandable using a plugin-based design. This will allow for easy integration of other map services including weather maps, satellite maps, etc.

An important feature of the MMT will be the advanced customizable visualizations that will provide insights into the current and historical data, and thus will allow building high-level data awareness. These services i.e., Visualization tools for High-Level Awareness Framework (HLAF), will complement the MMT and the DSS to create a complete solution of tools for creating more accurate and useful mission plans. The HLAF services will go from simple view of historical values in a line-chart (with ability to arbitrarily zoom in and out) to comparing data from different sources (and potentially a different scale) and data from different time periods. Furthermore, this will allow also observing trends in the data. Connected with design of data processing pipelines for particular demonstrators, it may also allow for what-if analysis, that based on historical data, shows the potential impact of particular value settings. This will be especially important in order to fine-tune parameters of algorithms used for autonomous decisions. These features are meant to be used on medium to large screens (15 inch or preferably above). As the real-time data insights require relatively significant computation power, the data will be processed in the cloud and the visualizations will be served via web-based services.

5.1.1.2. Mission Planning and Planning Algorithms

Planning a mission requires access to different set of resources. This includes maps, vehicles and their capabilities. In addition, a set domain specific tasks must be defined. Obviously, the vehicles must have sensors, actuators, and other SW/HW (all related to vehicle's capabilities) so that at least one vehicle is able to perform a specific task, in case this task becomes part of a mission plan. The operator will be able to choose a set of desirable tasks and the MMT will communicate with mission planners to generate a list of possible plans. The operator will be presented with an option to choose one of the given plans and save it for a future use or execute the plan immediately. When the plan is selected for execution, it is being forwarded to the MW's Mission Manager Component. In addition, the communication between MMT and MW is used to retrieve mission progress data and present it to the operator.

5.1.1.2.1. Hierarchical planning

Planning missions for multi-agent systems is a complex problem as there are several factors that affect planning. Thus, there are many possible ways to plan a mission since there is a need to use

algorithms that can optimize a plan based on a given criteria. Due to the complexity of the problem and many unknown factors, the strategy here is to have several planners implemented, with vast range of capabilities, based on different paradigms to "compete" for the most appropriate plan for the given problem/mission and circumstances. The most basic separation between planner alternatives is to optimal and sub-optimal planners. Optimal planners produce optimal plans, however they are usually very slow (due to the problem complexity being NP-hard at least). Sub-optimal planners are more suitable for situations where the necessity for a feasible plan outweighs the need for optimality, e.g., UAV autonomy is not very long (~ 15 mins), so if there is a need to re-plan, the planning process needs to be fast, i.e., the UAVs cannot wait hours for a new plan to be produced.
In the addition to aforementioned categories, planning process can be divided into levels, or a hierarchy, as follows:

- High-Level Planning (HLP)
- Mid-Level Planning (MLP)
- Low-Level Planning (LLP)

HLP represents the most abstract level of planning. It consists in providing the schedule and breakdown of tasks that need to be performed by the multi-agent system in order to accomplish the mission. The output of a planning process is a plan, i.e., a list of ordered tasks allocated to vehicles. At this level, tasks are assumed to be atomic and can require different equipment for its completion. Task duration and energy consumption are roughly estimated. Transits between tasks are roughly estimated as well, usually using Euclidian heuristics. Obstacles and environment influence are ignored at this stage of the planning process. The set of high-level tasks that HL planners receive is as follows (note that solely UAVs are assumed, a more generic solution with GVs and other resources will be designed after M13):

- **ACTIVE_TRACK:** the UAV will track a moving subject using the vision system and without a GPS tracker on the subject. The subject to track is defined by a rectangle on the live video view.
- **FOLLOW_TARGET:** the UAV will follow GPS coordinates continually sent to the UAV maintaining separation and a constant altitude.
- **HOTPOINT:** the UAV will repeatedly fly circles of a constant radius around a specified point called a Hot Point.
- **INSPECT:** go to a given location and take photos or record video.
- **PANORAMA:** go to a given location and take photos while rotating the camera 180 or 360 degrees.
- **SURVEY:** Boustrophedon coverage of an area is the default method for surveying a 2D space (alternative methods to boustrophedon are plausible, including dynamic/adaptive methods).
- **TRANSIT:** go towards a given location.

The planner used for HLP is based on a simple Genetic Algorithm (GA) and has been adapted to the specific problem of multi-agent mission planning.

MLP is the intermediate level of the hierarchical planner. It exploits the high-level plan to compute waypoint sequences based on an optimization approach according to some predetermined criteria, which UAVs shall follow in order to achieve the desired goal. The output is provided to the LLP module as a set of waypoints (a waypoint sequence for each device). MLP takes as input HL tasks described in the paragraph above and breaks them down. Each task is split into the corresponding set of commands (to be defined in detail for each task). Depending on the constraints and the objective function, different optimization problems can be formulated in order to compute the MLP output. For example, a cost function can include the time execution, the energy consumption and the number of available quadcopters, while, a maximum amount of time, energy and number of quadcopters available for the task execution can be considered as constraints.

MLP takes HLP output as an input and “refines” the plan by calculating trajectories and breaking down tasks into commands. MLP can be seen in most cases as path planning and task planning process. Path planning includes calculating the path between tasks, however, in this context considered, various information that were intentionally ignored at HLP level (obstacles and other information from the environment) are considered at MLP level. Task planning is the process of planning the execution of the task. At this level, tasks are not seen as atomic entities. For example, if a UAV should survey an area taking photos, task planning deals with solving the way that UAV will cover the surveyed area and how the equipment should be activated.

For each of the aforementioned high-level tasks the MLP will compute a list of waypoints and specific commands/signals that will be also provided to the LLP in order to correctly execute the mission. Particularly:

- **ACTIVE_TRACK:** the MLP will compute the waypoints to reach the intendent position where the moving subject is located. Then, the waypoints are fed into the LLP along with the ACTIVE_TRACK command to signal that when reaching the target location (last waypoint within the list) the UAV shall detect and track a moving subject by only using the onboard vision system and algorithms.
- **FOLLOW_TARGET:** the MLP will continuously get new coordinates position from the HLP and will re-plan the waypoints accordingly (a maximum/minimum rate which depend on several aspects that will be discussed later). Then, each plan will be dynamically fed to the LLP.
- **HOTPOINT:** comments the MLP will compute the waypoints to get to a location where it can start flying circles around the target location. Then, the resulting plan will be fed into the LLP.

- **INSPECT:** the MLP will compute the waypoints to get to the given location. Then, the waypoints are fed into the LLP along with the INSPECT command to signal that once the UAV has reached the last waypoint it shall take photos or record videos.
- **PANORAMA:** the MLP will compute the waypoints to get to the given location. Then, the waypoints are fed into the LLP along with the PANORAMA command to signal that once the UAV has reached the last waypoint it shall take photos while rotating the camera 180 or 360 degrees.
- **SURVEY:** the MLP will compute the waypoints to achieve the Boustrophedon coverage of an area which boundaries have been obtained by the HLP through corresponding GPS coordinates. Then, the resulting plan, i.e., the waypoints to get to the boundary of the given area along with the waypoints related to its coverage, will be fed into the LLP.
- **TRANSIT:** the MLP will compute the waypoints to get to the given location and will feed them into the LLP.

LLP deals with the motion planning, i.e., how to follow the path (actuator control) that has been generated by the MLP, or other low-level instances within task planning like turning sensors on and off. LLP is out of the scope of T3.2 and usually is addressed within vehicle specific module.

5.1.1.3. Supervision of Missions and Systems

An important part of the MMT is supervision of missions and vehicles. Therefore, it is essential that MMT provides the operator with relevant warnings when an alarm happens e.g., a vehicle/sensor malfunctions, a vehicle has very low battery level for the remaining part of the mission, or two vehicles are too close to each other (or a building). These alarms will be set through the “System Configuration” component of Farm Management System and accessed by the MMT.

During a mission, the operator will be able to view the mission progress by observing the locations of the vehicles and their current state or see the camera live feed from the vehicles, when possible. They will also be able to see deviations from proposed plan and request a re-plan if necessary. In order to allow the operator to review the missions later, mission plan and mission progress will be saved though the MMT for future reuse and analysis.

5.1.2. Software Interfaces

MMT will provide interfaces for development of tools and external components that need to be integrated with MMT and its GUI (see Figure 8). These interfaces allow the AFarCloud partners to develop plugins for MMT that can easily be added to it and provide new functionalities and user interfaces. In cases where plugin interface cannot be used, Apache Thrift will preferably be employed.

5.1.2.1. Interfaces for Farm Management System

The DDS, the System Configuration are other parts of the Farm Management System that require to be accessible through MMT's GUI will be developed as plugins for MMT to allow for extensibility and code separation. This means that they will need to be developed with the same technologies as MMT or contain a proxy developed in MMT's platform that communicates with the tools.

5.1.2.2. Interfaces for visualization tools for high-level data awareness

In order to support the operator during the process of planning, performing and evaluating a mission (the main MMT functionalities), services for data analysis in general (with/without the DSS) will be provided also, in order to grant high-level data awareness (HLAF). These services will be integrated either with the MMT or the DSS depending on the purpose. The objective of these services is to bring high-level data awareness and insights (developed in T6.3). This framework will be hosted in the cloud and will offer a web-based interface for development and configuration of the visualizations and for development and configuration of data processing pipelines. This will allow to use the tool remotely while the heavy (offline and real-time) data processing will happen in the cloud. The tool will be integrated into MMT via web-components (and potentially an IFRAME). This integration will further require sharing of authentication/authorization tokens from MMT.

5.1.2.3. Map Providers

Interfacing with different map providers will follow the plugin design as well. Each map provider will have a plugin developed for the MMT that acts as a proxy to communicate with different map providers using WMS protocols and presents the final image on MMT's GUI.

5.1.2.4. Other Data Sources

Any other data source can also be accessible and be presented in the GUI to the operator following the same plugin design. Communication with a data source can be based for example on REST protocols and be presented to a proxy plugin, which in turn presents the results on the MMT's GUI.

5.1.2.5. Middleware

Communication with different parts of Middleware will be based on Apache Thrift. This includes the Device Manager, the Mission Manager, the Data Query and the Device Registry.

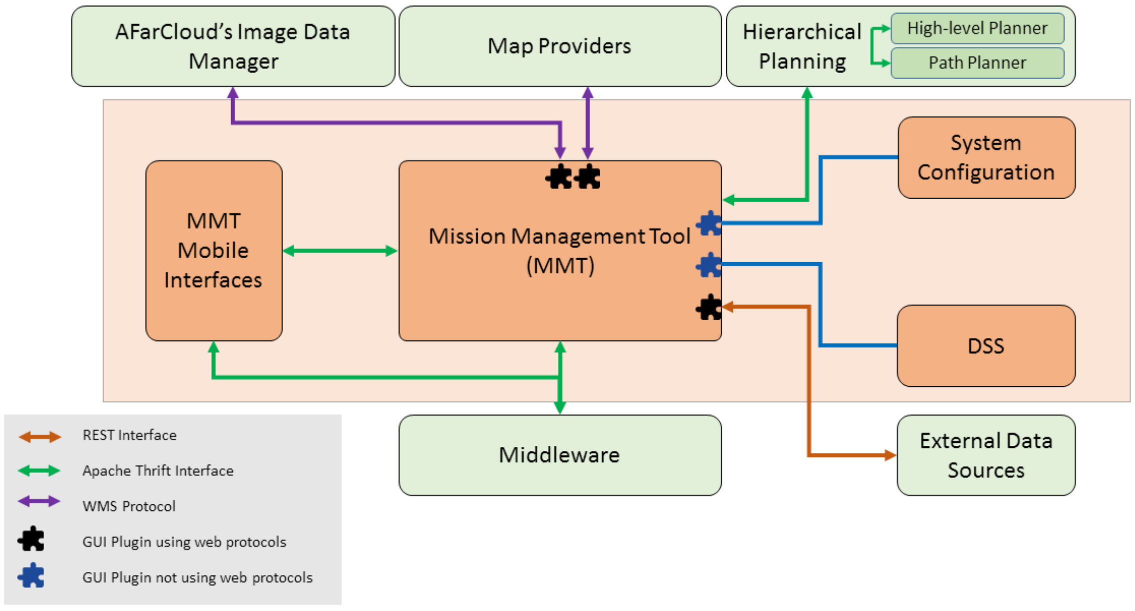

Figure 8: Interfaces of the Mission Management Tool

5.2. Decision Support System

5.2.1. Description

The goal of the DSS is to provide expert recommendations using algorithms that extract conclusions from data. DSS complies with criteria of scalability and adaptability, as users' requirements are different in each scenario (see Figure 9).

Algorithms are the core of the DSS as they provide outputs. Algorithms can be classified in two classes, according to the outputs they provide:

- **Calculation of complex metrics for crop and animal welfare from raw data.** Note that these complex metrics are useful only if they are solving the right problem and in a way that is understood by the users. For example, the metrics for calculating "percentage of water stress in a crop": (i) soil humidity in several points, (ii) solar radiation, (iii) amount of watering, (iv) raining, (v) type of soil, etc.
- **Recommendation algorithms.** This stage represents the next step after calculating the output as above. The recommendation algorithms' goal is to integrate metrics (computed by algorithms) and suggest different alternatives, or solutions to the user, in order to help him/her reach an objective (defined in users' requirements). An example recommendation could be:

"when the crops are watered consider the following: (i) levels of water stress forecasting next days, (ii) low levels of disease risk, (iii) expected amount of watering".

There shall be algorithms to monitor data or metrics to alert users of low or high levels that can jeopardize defined objectives.

Data for algorithms will be in the AFarCloud repository. The DSS will not store any data, although algorithms could use local repositories for their calculations.

5.2.2. Interfaces

A. Interface for Farmer and Farm Cloud (DSS)

This interface is for configuration purposes. It is used for starting/stopping the execution. The farm operators can perform the following actions:

1. List the installed algorithms;
2. Start an algorithm. The user must define a name for the algorithm and a configuration. The configuration will be a global configuration, i.e., a high-level configuration: sensor position, model to apply, etc. Thus, it does not refer to internal parameters of the algorithm itself. The DSS registers the name and sends the Entry Point a "start command". Internally, the Entry Point knows that the algorithm is started and returns a unique identifier. The algorithms must have the ability to be instantiated several times (each time with a unique identifier or algorithm_id), since the same algorithm is used in several farms with different configurations and data.
3. Stop an algorithm (stop receiving alerts and recommendations). The DSS will connect to the Entry Point and it will stop the algorithm (algorithm_id).
4. The user can ask the DSS for the list of running algorithms and their id.
5. Others (e.g., configure alerts).

The algorithms will be able to communicate with the DSS to inform about their status, and in case there are errors during the execution.

B. Interface for Farm Cloud (DSS) and algorithms

This interface connects the DSS with the algorithms in partner's premises. Thus, all partners that develop algorithms for DSS need to implement an API to listen commands or send their status. This interface will process internally to stop, start and configure the algorithms according to the farm's operator specifications.

The DSS cannot manage the internals of the algorithms that partners develop, therefore the DSS needs a way to control them with basic commands. The proposed basic commands are:

- Start a recommendation algorithm, sending the name of the algorithm and the configuration file (if needed). Returning an internal id.
- Stop a recommendation algorithm (by means of its internal id).
- Send the status of the algorithm to the DSS (by request or periodically).

Therefore, all partners will need an interface to communicate their algorithms with the DSS. For that purpose, it is proposed an API service that will be addressed by the DSS whenever a command needs to be sent. For sending the status of the algorithms, another API interface will be implemented in the DSS.

C. Interface Algorithms and the Middleware

All the algorithms will need to access to the database to collect data in order to generate their outputs, to store them in the database and to send alerts via the publish/subscribe mechanism. These interfaces are not implemented by the DSS.

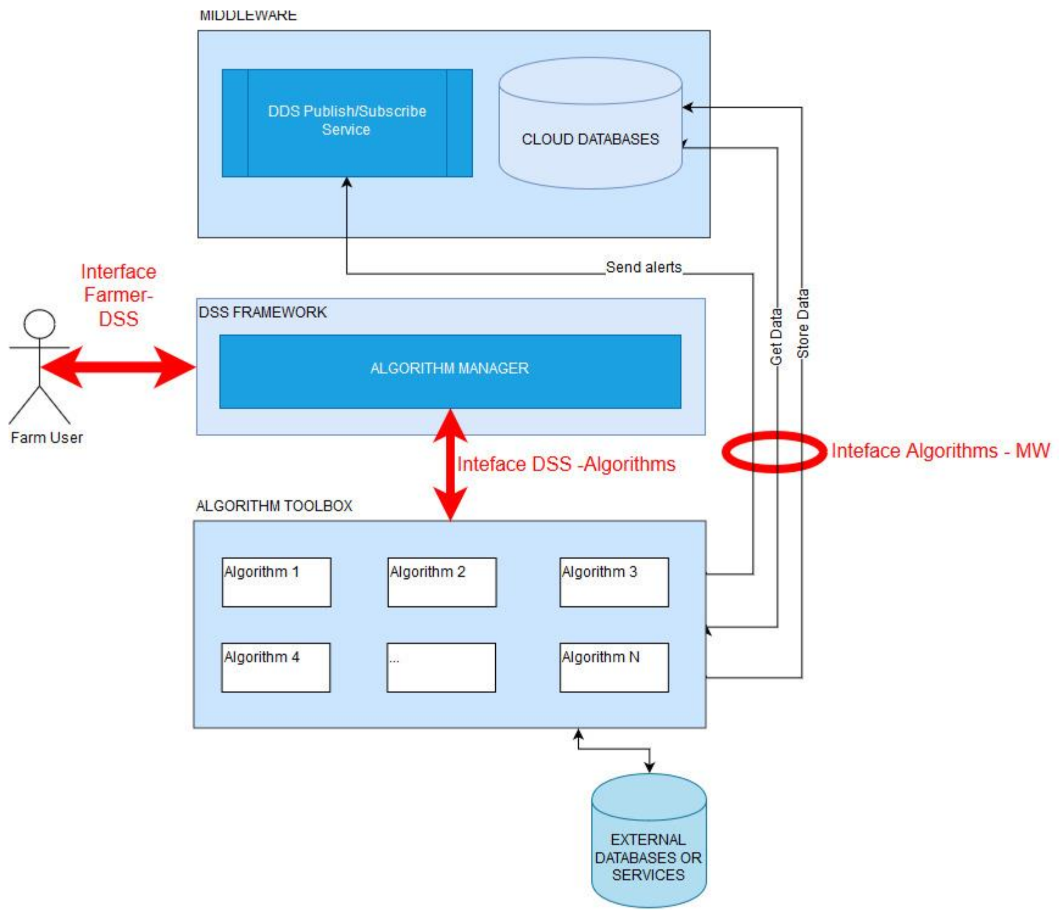

Figure 9: The AFarCloud DSS architecture and interfaces

5.2.3. Components

The basic components of the DSS are as follows:

– **Algorithm Manager:** This component is in charge of managing the algorithms that will be deployed as part of the AFarCloud platform in the farm. The user can interact with it to define and configure the algorithms available in the farm and to stop or start an algorithm. The Algorithm Manager will store internally which algorithms are available in each farm and the status of them (running or stopped) and the URL and port where the algorithm is accessible through APIs.

– **Algorithm Toolbox:** With this approach the algorithm could run either in the partner's premises or in the cloud. The algorithms will have access to the AFarCloud repositories to collect the data needed to operate. The results will be saved in the AFarCloud database (for visualization purposes or for further analysis), and optionally, they can generate alerts via a publish/subscribe based mechanism (to be implemented in the Middleware). This mechanism will publish relevant alerts, warnings or errors to interested subscribers (users or other components).

5.3. System Configuration

This module handles the configuration and settings of system hardware (vehicles, farming machinery, sensors, actuators, etc.). To this end the following high-level architecture of the platform configuration module is foreseen as shown in Figure 10:

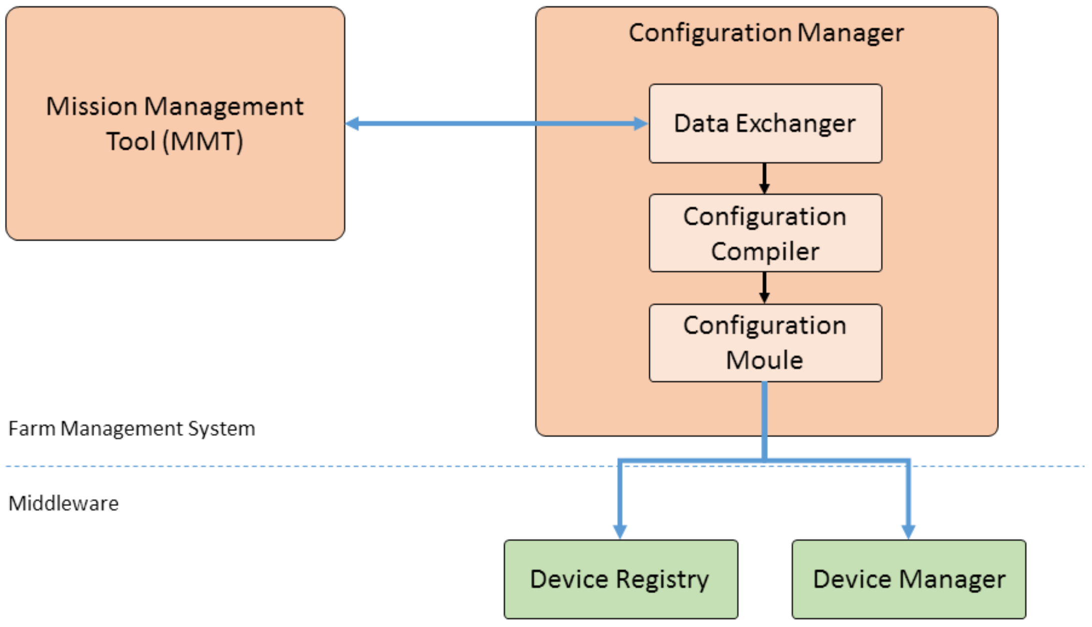

Figure 10: Interfaces of the System Configuration

- The <u>Data Exchanger</u> module will implement the interface to the MMT and respective User Interfaces to support the following:
    - Configuration of sensors and actuators (e.g., sampling rates and other operational parameters);
    - Alarms configuration;
    - Notification of available firmware updates for sensors, actuators and GVs;
- The <u>Configuration Compiler</u> module will make any necessary format adaptations in order to translate configuration information received from the MMT/UI to be transmitted towards the underlying AFarCloud platform infrastructure.
- The <u>Configuration Module</u> will implement the interfaces to the underlying AFarCloud platform infrastructure via the Middleware Device Registry and Device Manager in order to transmit configuration information.

The platform configuration module will not be developed for the first-year demonstration and updated specifications of this module will be provided in the final version of this deliverable.

6. The Semantic Middleware

Figure 11 depicts the components and interfaces of the semantic middleware. Most of the elements of the Semantic Middleware are deployed in the Cloud, except for four components (i.e., the Image Processing Platform, the DDS Manager, the Data Pre-Processor and the Data Fusion), which are deployed at the Edge. Edge components must be implemented on the site, in the farm facilities.

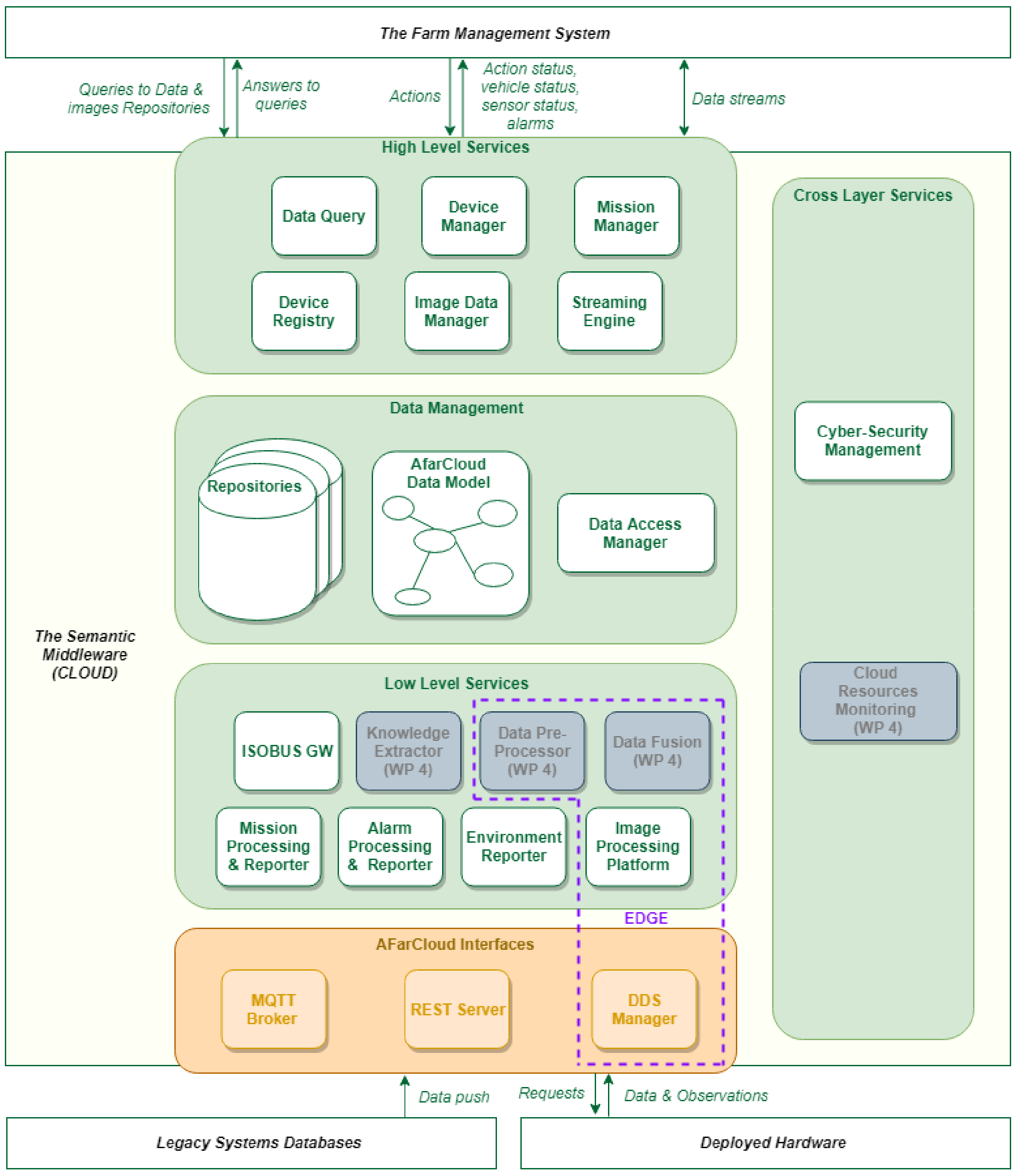

Figure 11: Components of the semantic Middleware

The Semantic Middleware provides the following interfaces to the rest of the elements of the AFarCloud architecture:

- Interface with the Farm Management System:
  - Apache Thrift interface: query cloud repositories, sending missions, collecting mission results, sensor status and alarms.
  - Web Map Service interface: retrieving images from cloud repositories.
  - Apache Kafka interface: delivering real time data streams to the DSS.
  - REST interfaces: collecting data from other data sources (e.g., third-party data).
  - USB interface: load/unload of ISO-XML files.
- Interface with Deployed Hardware:
  - DDS interface: sending actions to UAVs, collection of action results, sensor status and alarms from UAVs. Communications with DDS compatible devices (i.e., UAVs) are managed in real-time.
  - ISOBUS interface: sending actions to ISOBUS compatible vehicles (i.e., GVs). Collecting the results of actions carried out by GVs.
  - MQTT interface: managing communications with MQTT devices (e.g., sensors onboard GVs, standalone sensors, groups of dedicated sensors, actuators). Communications with MQTT devices are managed in real-time.
  - REST interface: managing communications with devices offering a REST interface.
  - USB interface: load of hyperspectral/multispectral images taken by UAVs in the Image Processing Platform.
- Interface with legacy systems databases:
  - REST interfaces: collecting data from other data sources.

The following sections of this chapter describe each of the components of the Semantic Middleware.

6.1. Cloud Data Storage

6.1.1. Description

The data managed in AFarCloud will be stored in 3 types of repositories: semantic, relational and NoSQL. Depending on the needs of the information to be stored, the appropriate repository will be chosen.

The major benefits of semantic repositories, in comparison to relational databases, are: the data schema can be changed without affecting the data instances; implicit knowledge can be automatically inferred without having to explicitly store it based on either semantic rules or the logic of ontological languages; seamless integration of distributed data sets and linked data models. This provides greater flexibility and scalability to AFarCloud data model and the possibility of inferring events based on rules like the detection of possible collisions between vehicles based on their Euclidean distance. For this reason, the semantic repositories will be used to store the last updated information or “photo” of the farm.

Relational databases will be used to store historical information about the farm and the missions. This kind of repositories allow multiple users to access the database simultaneously and offer built-in locking and transactions management functionality which ensures security and reliability and prevents collisions.

Finally, NoSQL repositories (like MongoDB, HBase or InfluxDB) will be used for storing observations from IoT devices and sensors. It is expected that this volume of data can be large and grow in the future and measurements coming from different sensors are not related between them so there is not the need of structured data.

The data model for all these repositories will be described in deliverable D2.6.

6.2. Cloud Resources Monitoring

6.2.1. Description

The aim of this component is to ensure that the cloud resources where AFarCloud is deployed on are behaving in accordance to the expected non-functional requirements (NFR). For the time being the non-functional requirements that will be covered by this component are availability, workload job processing performance, overall performance, and response time (see section 2.9 Cloud resources monitoring requirements).

More specifically, the objective of this component is to monitor the cloud resources, namely virtual machines, database as a service and storage as a service. In addition, compare their actual values with the Service Level Objectives (SLOs) identified by the developer in order to alert said developer when a violation has occurred. It also has the aim of analysing the workload that a data processing job is taking in order to be able to optimize the execution of such processing jobs or to select additional cloud resources for a better performance or improved response time.

6.2.2. Components

The cloud resource monitoring component is envisioned to have the following components (for more detail, please see D4.3):

- NFRMonitoring manager: this component is the core of the cloud resource monitoring AFarCloud component. Once the developer manifests the need to start monitoring a cloud resource, it configures and starts the different agents needed for the NFRMonitoring metering to function properly as well as the monitoring registry.
- NFRMonitoring metering: This component collects the data from the different cloud services where the AFarCloud components are deployed in. The needed data are inserted through the UI component or through a REST API.
- WorkloadMonitoring: this component will evaluate how the processing job is behaving in terms of workload in the contracted cloud resource.
- NFRMonitoring registry: This sub-component is in charge of storing the data collected from the metering sub-component in a time-series database.
- SLO Assessment: responsible of the aggregation, if needed, of the raw monitored metrics whose values need to be assessed with respect to the predetermined SLOs, and comparison of the theoretical values vs. the real values.
- ViolationManager: Once the SLO Assessment component detects a violation of the SLO, this subcomponent registers the violation in the service registry data base and alerts the developer e.g., via an email that a violation has occurred.
- UI: This is the user interface where the developer will insert the data related to the IP of the cloud service contracted, the thresholds of the values to be monitored, and so on. This information may also be sent to the cloud resource monitoring through a REST API. This component will also include the dashboard where the actual values of the monitored metrics will be shown at real time, as well as a summary of the violations occurred by cloud service offering.
- CloudServiceRegistry: this is the database where the service catalogue is stored, and where the occurred violations are stored. While the violations could be queried from the NFRMonitoring registry, initially it is envisioned to store the violations on a different database, namely this one, for performance issues on the queries.
- UserManagement: this is a generic module to manage the users that can access to the cloud resource monitoring module.

6.3. Data interoperability (AFarCloud Data Model)

Data Interoperability in AFarCloud will be managed by using a common data model for information storage and common data formats for information exchange. More information will be available in D2.6.

6.4. Data Access Manager

6.4.1. Description

This component provides interfaces able to insert/retrieve/update information in the AFarCloud ontology and in the relational and non-relational databases. Queries to the ontology will be implemented through a SPARQL Endpoint. Apache Jena Fuseki will be used as SPARQL server. Queries to the relational database will be implemented by means of SQL statements. For non-relational databases the query language will depend on the NoSQL database that is chosen, e.g., MongoDB, HBase, InfluxDB, etc.

6.4.1. Interfaces

The Data Access Manager (DAM) defines three interfaces: OntoManager (in Table 5), RDBManager (in Table 6) and NRDBManager (Table 7). The main goal of these interfaces is to manage the connection to the semantic repository using the Jena library and to the relational and non-relational databases, respectively. These interfaces are used by the Data Query for CRUD operations to the AFarCloud repositories.

6.4.1.1. OntoManager

Table 5. OntoManager interface

(*public*) void **addModel** (File rdf) (*public*) void **addModel** (Model m)	Adds the content in the RDF file or the model to the default graph of the dataset if it does not exist.
(*public*) void **replaceModel** (File rdf) (*public*) void **replaceModel** (Model m)	Create/replace the default model of the dataset with the content in the RDF file or with the model m.
(public) void **replaceInfModel**(Model m)	Put (replace) the inferred model of a Dataset
(public) **deleteModel** (dsServiceURI)	Deletes the default model of the dataset.
(public) **getModel** (dsServiceURI)	Returns the default model of the dataset.

(*public*) List<Map<String, Object>> **queryModel**(String squery);	This method provides the endpoint to query the dataset.
(*public*) void **updateModel** (String squery)	This method provides the endpoint to update (INSERT, DELETE) the dataset.
(*public*) void **updateOntologyFile**()	This method saves the content in the dataset to an RDF file.
(public) String[] **getRegisteredReasoners()**	Returns the list of registered reasoners configured in the properties file.

6.4.1.2. RDBManager

Table 6. RDBManager interface

(*public*) **queryDatabase**(query)	This method provides the endpoint to query the database.
(*public*) **openDBConnection**()	This method establishes a database connection
(*public*) **closeDBConnection**()	Closes the connection to the DB.
(*public*) **insertDatabase**(query)	This method provides the endpoint to insert data to the database

6.4.1.3. NRDBManager

Table 7. NRDBManager interface

(*public*) **queryDatabase**(query)	This method provides the endpoint to query the database.
(*public*) **openDBConnection**()	This method establishes a database connection
(*public*) **closeDBConnection**()	Closes the connection to the DB.
(*public*) **insertDatabase**(query)	This method provides the endpoint to insert data to the database

6.4.2. Components Diagram

The Data Access Manager component relates to the rest of the components in the architecture, as shown in Figure 12:

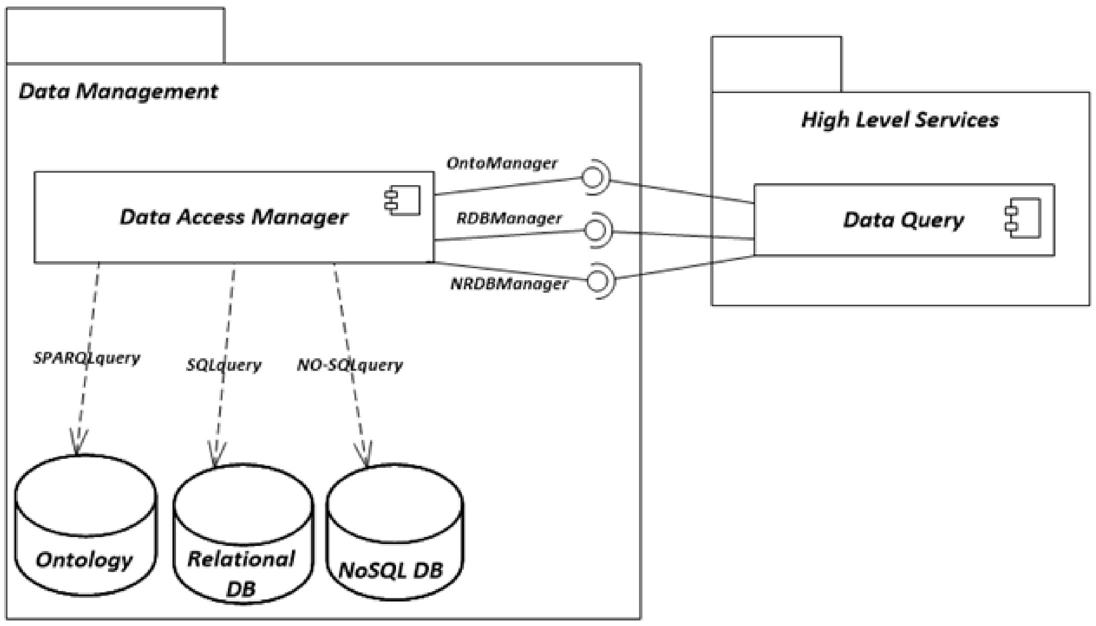

Figure 12: Data Access Manager Components Diagram

6.5. Data Query

This component processes any query made to manipulate data in the AFarCloud repositories. The queries are the mechanism used by the rest of the AFarCloud components to consult or update any information from the AFarCloud repositories. This module will provide predefined methods for certain queries (e.g., getCapabilities(vehicle), getTasks(Mission, Vehicle), etc.) and will translate them to the database specific syntax that will be forwarded to the Data Access Manager. This way, components do not need to be aware of where and how the data are stored.

6.5.1. Interfaces

The Data Query (DQ) defines three interfaces: SemanticQueryFeeder that is used by other modules to send queries to the ontology, SQLQueryFeeder and NoSQLQueryFeeder for relational and NoSQL databases, respectively. These interfaces will provide methods with predefined queries for the most common functionalities and also generic methods (ontoQuery and dbQuery) for more complex queries.

6.5.2. Components Diagram

Figure 13 shows the relation of the Data Query with the rest of the middleware components.

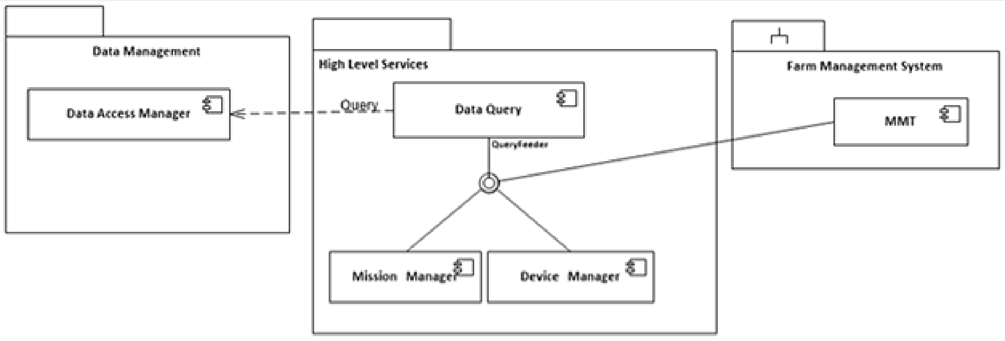

Figure 13: Data Query components diagram

6.6. Device Registry

6.6.1. Description

The Device Registry registers the animals, IoT devices and vehicles in a farm (i.e., sensors, collars, tractors, UAVs, etc.) and their static features in the AFarCloud data repositories. In the case of sensors, this static information includes the type of information observed by the sensor, the range of possible values, the units of measure, the location (if static), the identifier, etc. For UAVs, it includes the equipment onboard, the type of commands that the vehicle can understand, etc.
For sensors, the data model to be used will be based on the one provided by FIWARE, a framework for open source developments.

6.6.2. Components diagram

The Device Registry calls the Data Query to store and update the information related to the IoT devices and vehicles in the farm. This component offers the registration functionality as a REST service that can be invoked by the MMT. This service hides the complexity of the registration process from the farmer.

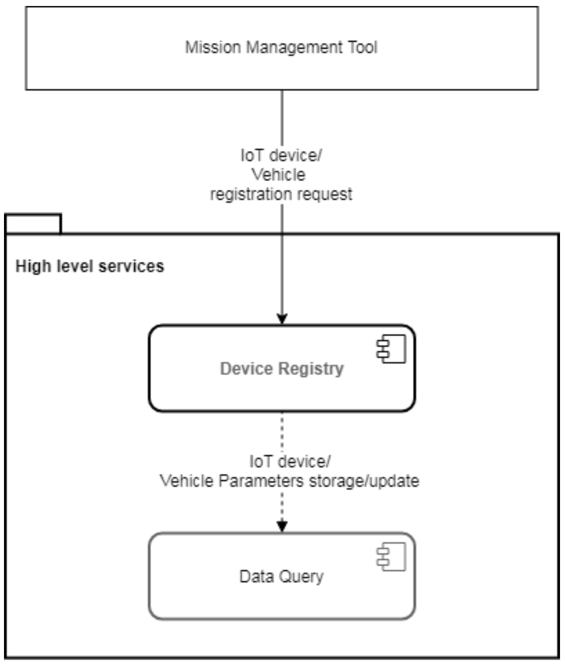
Mission Management Tool
IoT device/ Vehicle registration request
High level services
Device Registry
IoT device/ Vehicle Parameters storage/update
Data Query

Figure 14: Device Registry component diagram

6.7. Streaming Engine

6.7.1. Description

The Streaming Engine (SE) publishes and subscribes to streams of records in real-time. The main features of this component are the following:

- High performance and scalability in data streams management;
- Reliably getting data between distributed systems, applications and data sources;
- Handling Big Data operations and enabling integration with a large number of heterogeneous data sources and advanced database solutions.

The SE provides real-time streaming data pipelines that reliably exchange data between the AFarCloud repositories and the Decision Support System. During the project we will analyse the possibility to directly integrate other components of the AFarCloud platform (e.g., MQTT broker, REST server, DDS Manager, etc.) with the SE by corresponding connectors. The SE will implement a publish/subscribe interface, by means of a well-known technology (e.g., Apache Kafka).

6.7.2. Components diagram

The following figure presents the general architecture of the SE, and a proposal of its interfaces:

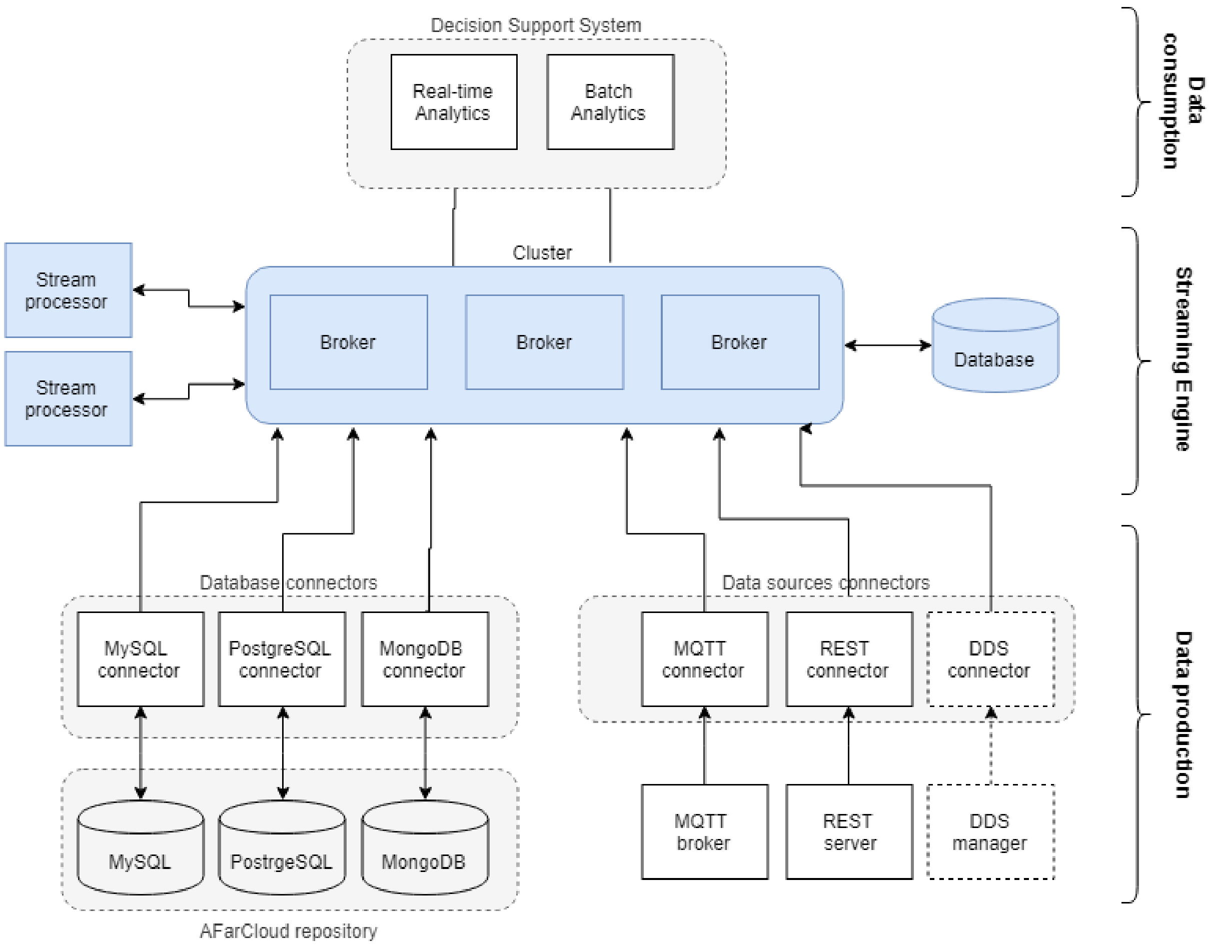

Figure 15: Components of the Streaming Engine (SE)

The SE may be integrated with two groups of data sources:

- AFarCloud cloud repositories
- Measurement devices

For each type of data source, a corresponding connector which unifies data format needs to be applied. Data streams are transmitted from connectors into the Cluster, which distributes messages under a publish/subscribe paradigm into the Streaming Processors, the Database and the Decision Support System (e.g., Real-time Analytics and Batch Analytics engines).

The main Streaming Engine components are described below:

- **Cluster –** the SE runs as a cluster on one or more servers. The cluster stores streams of records in categories called topics. Each record consists of a key, a value, and a timestamp. Brokers (a part of the cluster) are the primary storage and messaging components of SE.
- **Database connectors -** there could be various source connectors (e.g., MySQL, SQL, PostgreSQL, MongoDB, etc.) that could obtain a snapshot of the existing data in a database

and then monitor and record all subsequent row-level changes to that data. All of the events for each table are recorded in a separate topic, where they can be easily consumed by applications and services.

- **Data source connectors -** integration with existing MQTT brokers and REST server. This component connects to a MQTT broker, REST server, etc. and publishes data to the specified topics.
- **Stream processor -** is a node in the processor topology that represents a real-time single processing step. It takes continual streams of data from input topics, performs some processing on this input, and produces continual streams of data to output topics. This component enables i.e., aggregates of streams or joins streams together. For example, this component might be used to detect and generate alerts.
- **Database (persistency) -** the SE enables also integration with a dedicated data warehouse to store streaming records in a fault-tolerant durable way. The streaming platform relies heavily on the filesystem for storing and caching messages.

6.8. Device Manager

6.8.1. Description

This component is responsible for the management of all operations related to standalone devices (sensors, actuators, gateways, etc.) and groups of devices (e.g., WSNs) connected to the AFarCloud platform. The management of data flows with vehicles (e.g., GVs, UAVs) will be carried out by the Mission Manager. By device operations we understand the following actions on this kind of devices:

a) Setting the sampling rate of sensors;
b) Sending notifications on firmware updates to sensors and actuators;
c) Sending commands to actuators;
d) Receiving alarms generated by sensors and actuators. The Device Manager will forward these alarms to the MMT;
e) Receiving device data (e.g., measures);
f) Storing all data sent by sensors and actuators (e.g., alarms, device data, etc.) in the AFarCloud repositories.

Operations a), b) and c) will be triggered by the system operator through the MMT.

6.8.2. Components diagram

The Device Manager manages all requests on devices done by the system operator through the MMT. For MQTT devices, requests to change the sampling rate of sensors, actuator commands and notification of firmware updates will be published through the MQTT Broker, that will publish messages on specific topics addressed to the involved MQTT devices.

Alarm Processing & Reporter and the Environment Reporter transfer data coming from devices (i.e., alarms, measures and other device data) to the Device Manager. The Device Manager sends this data to the Data Query to be stored in the AFarCloud repositories, and forwards alarms to the MMT in the FMS.

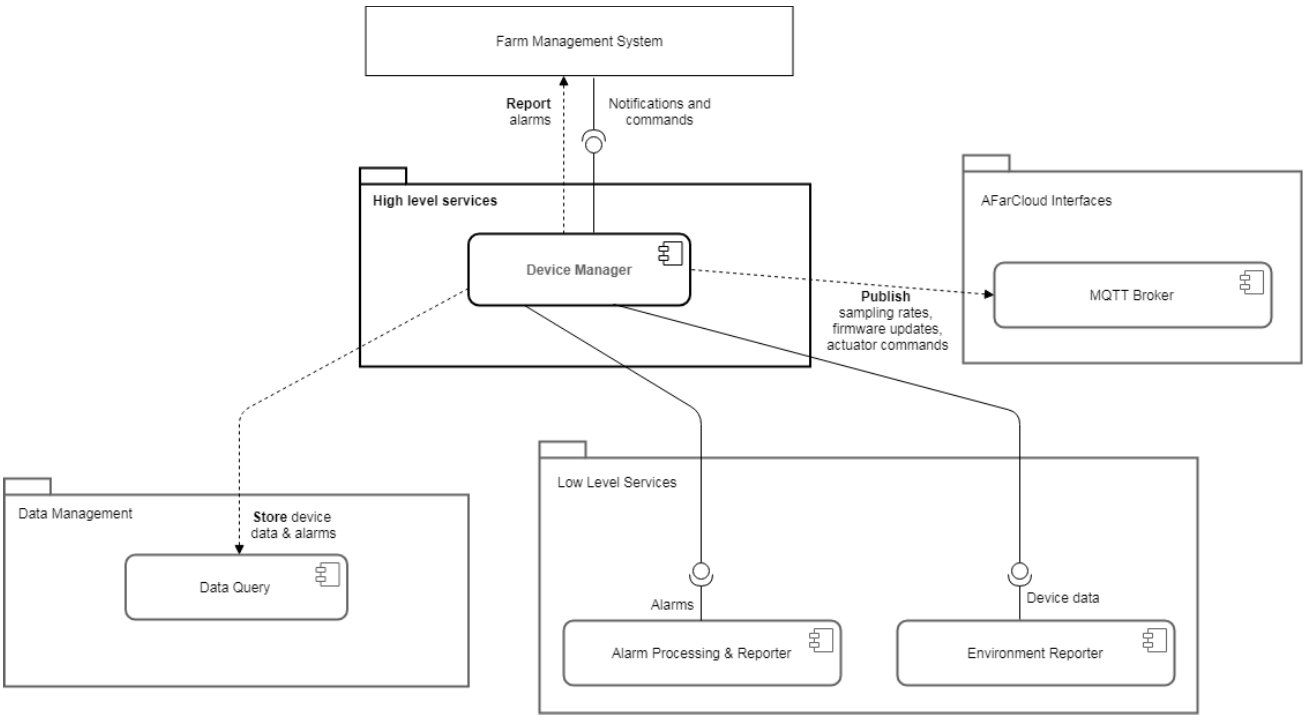

Figure 16: Device Manager components diagram

6.9. Mission Manager

6.9.1. Description

This component manages all operations and data flows in which vehicles (i.e., GVs, UAVs) are involved. The main duties of this component are related to the delivery of the mission plan defined by the Farm Managing System, as well as receiving the information from the vehicles that participate in the mission according to its progress.

Also, this component is responsible for sending events to elements in the Hardware Layer. We will consider as events all relevant data (i.e., command to abort a mission) sent by the middleware that should be considered by vehicles or other devices. Events will be generated by the Farm Management System as a result of an analysis of data.

6.9.2. Components diagram

As it was mentioned before, the Mission Manager component interacts with the other software components of the semantic middleware architecture in four ways:

a) Receives vehicle information about the mission progress (Mission Processing & Reporter), alarms (Alarm Processing & Reporter) and data from sensors onboard the vehicles (Environment Reporter);
b) Sends the mission plan to UAVs (DDS Manager) or ground vehicles (ISOBUS Gateway);
c) Stores mission plans and data received from devices, in the AFarCloud cloud repositories (Data Query);
d) Sends events to UAVs, ground vehicles, or other devices.

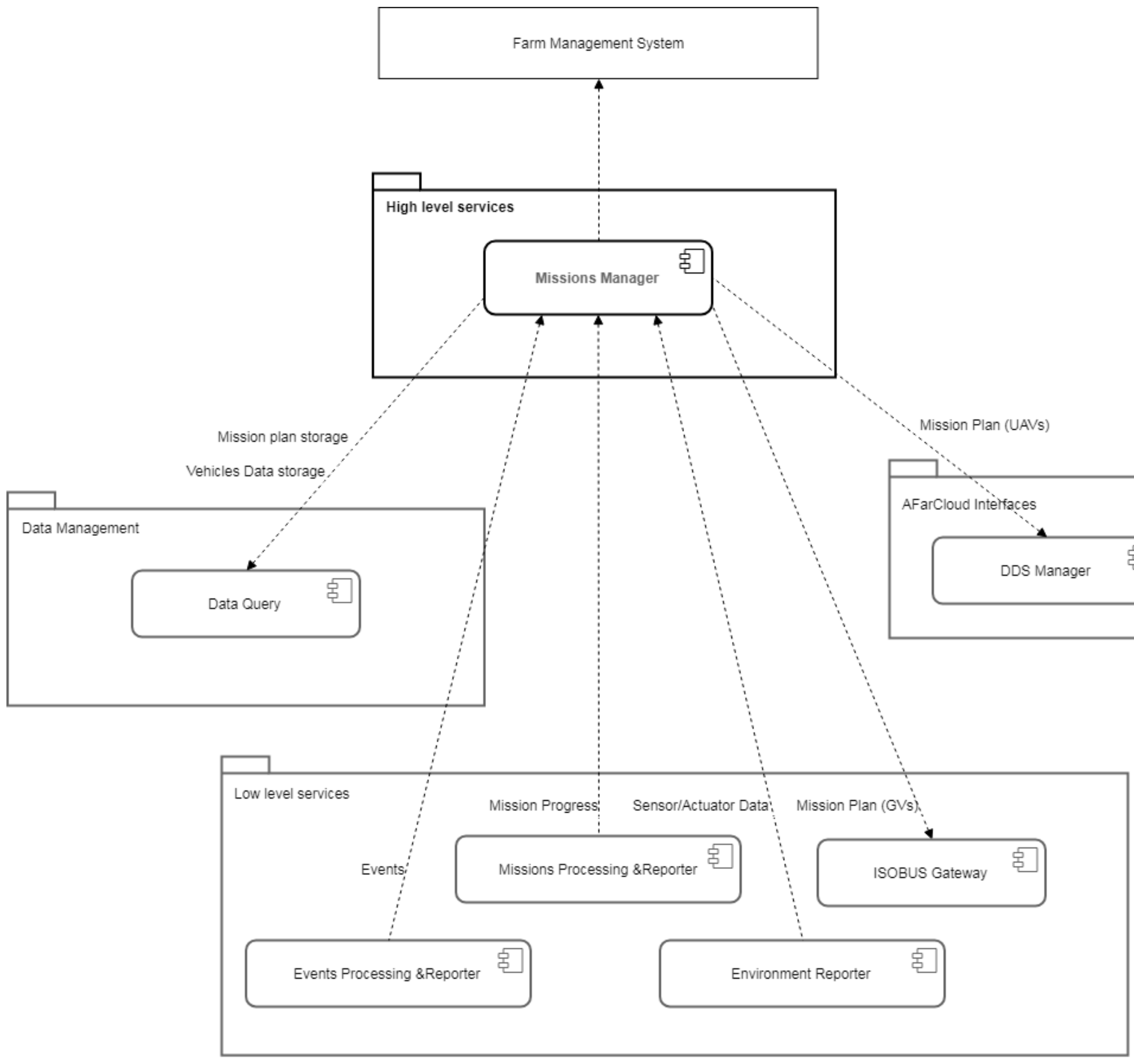

Figure 17: Mission Manager components diagram

6.10. Mission Processing & Reporter

6.10.1. Description

This component is responsible for two tasks in the semantic middleware architecture: firstly, it reports the status of the mission to the High-Level Services of the middleware (specifically, to the Mission Manager). Secondly, it processes the information to ensure that it is received with the quality and cleanness expected from the data (e.g., correct data formats)

6.10.2. Components diagram

The Mission Processing and Reporter component interacts with the other software components of the semantic middleware architecture in two ways as well: first, it receives the information, according to the data format defined for the AFarCloud project, either from the DDS Manager that is ruling data transfers between the distributed DDS parts, or from the ISOBUS interface with ground vehicles. Secondly, and once the information received has been processed so that it is internally manageable, it will be transferred to the Mission Manager component via the *sendMissionStatus* method invocation, which will transfer it back to the Farm Management System, where the mission progress can be displayed.

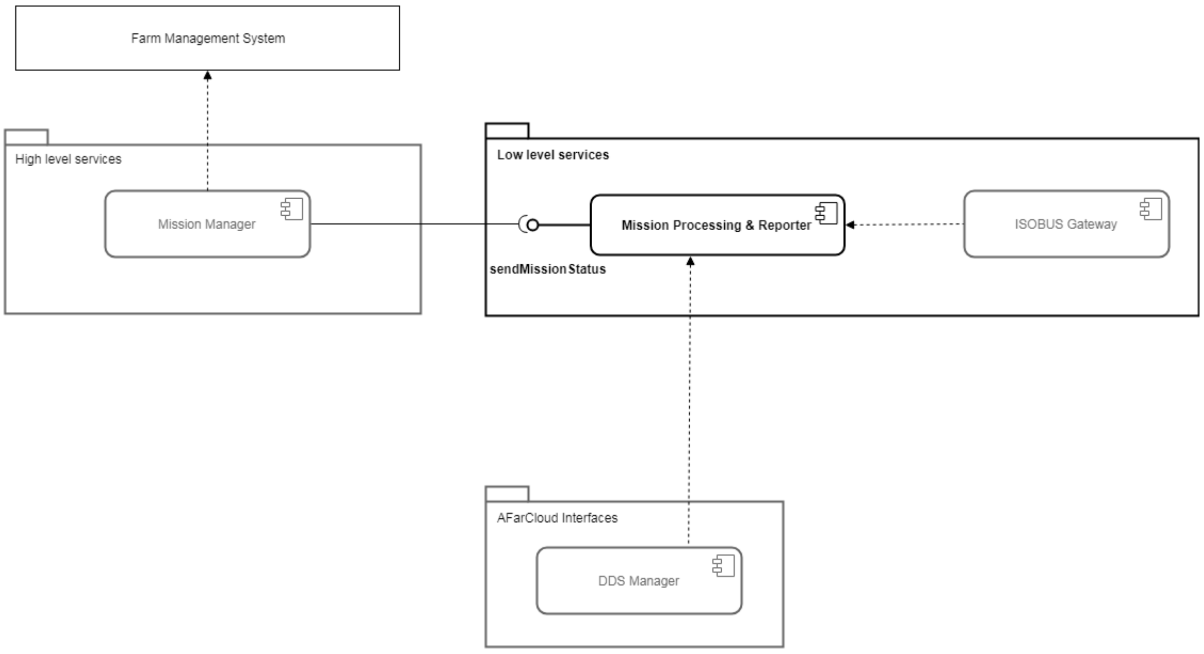

Figure 18: Mission Processing & Reporter components diagram

6.11. Alarm Processing & Reporter

6.11.1. Description

We will consider as alarms messages sent from a device, sensor or vehicle to the middleware to inform about abnormal behaviour at equipment or functional levels. The alarm at equipment level indicates that the equipment is not functioning correctly e.g., the IMU in a UAV. Functional level represents a higher level of abstraction than equipment level. An alarm at functional level means that an entity will not be able to execute an action that requires a certain functionality like localisation or navigation, probably due to an alarm at equipment level.

6.12. Environment Reporter

6.12.1. Description

The Environment Reporter (ER) deals with raw data provided by sensors in the AFarCloud (AFC) hardware layer (e.g., standalone devices, devices on semi-autonomous ground vehicles and aerial vehicles) arriving through the MQTT, REST or DDS interfaces. The core function of the ER should be to report and validate any data or information related to environment, approving it to be stored in AFC repositories (in the cloud). Such data can be coming directly from the sensors, but can also be pre-processed data, fused data or extracted information (on environment, crops and livestock) coming from the components responsible for such operations, respectively, which have been previously fed with raw data, by the ER, or which requested stored data, through queries. The ER checks basic aspects of the received data, such as, validating it according to the expected value ranges considering the respective sensor specifications. This occurs before forwarding such data to the Data Pre-Processor (DPP) and before data or information is sent for storage at AFC repositories, both during missions (online), and pre- or post- mission (offline). The ER can also verify if the received raw data needs pre-processing involving data fusion, e.g. in case of missing georeference metadata associated to sensors measurements, respectively, and forward that indication to the DPP together with the validated raw data to be pre-processed appropriately.

Since the majority of raw data needs pre-processing and data fusion, e.g. to filter duplicates and/or include georeference metadata, only when the ER receives back the data already pre-processed, either coming directly from the DPP, or from the Data Fusion (DF) component if data fusion has been performed, it can then, after new validation, send such pre-processed data to AFC cloud for storage. This will occur via the Mission Manager or the Device Manager components, or eventually directly via the Data Query component, to which the ER will be implicitly confirming the proper acquisition of the relevant respective data, or it can instead eventually report any failure explicitly.

Summarizing, the ER validates, relays and reports sensor data. The latter reports can be used by the MMT in the Farm Management System to be aware of any sensor failure(s) and react upon it if necessary. Moreover, the ER validates, and relays extracted environment-based information or knowledge associated to livestock and crops to be exploited in the cloud, namely by MMT and DSS.

6.12.2. Components diagram

The Environment Reporter (ER) component interacts with other components, receiving and providing data, as follows. First, it receives data according to the defined AFarCloud format, via DDS Manager in case such sensor data comes from aerial vehicles, or via MQTT or REST in case the data comes from ground vehicles' onboard sensors or deployed sensors in the farm. Secondly, raw data, should be kept in a local DB located at the edge level, close to where effective data pre-processing should take place, i.e., at Data Pre-Processor (DPP) and Data Fusion (DF) components.

The DPP sends the outcome of pre-processing data either to the DF component if data fusion is necessary, or directly to the ER otherwise. In other cases, uncorrelated to DPP operation, the DF and the Knowledge Extractor (KE) components send fused data or extracted information/knowledge, respectively, to the ER for it to validate and send to the AFC repositories for storage. On the other hand, the ER can send data to two distinct components, which are the Mission Manager and the Device Manager. In order for that to take place, ER's *postSensorReport* and *postDeviceReport* methods are used, respectively. On both situations, after receiving such data, those components should forward it to the Data Query, which will store it in the appropriate AFarCloud repositories. Figure 19 presents the components diagram of the ER.

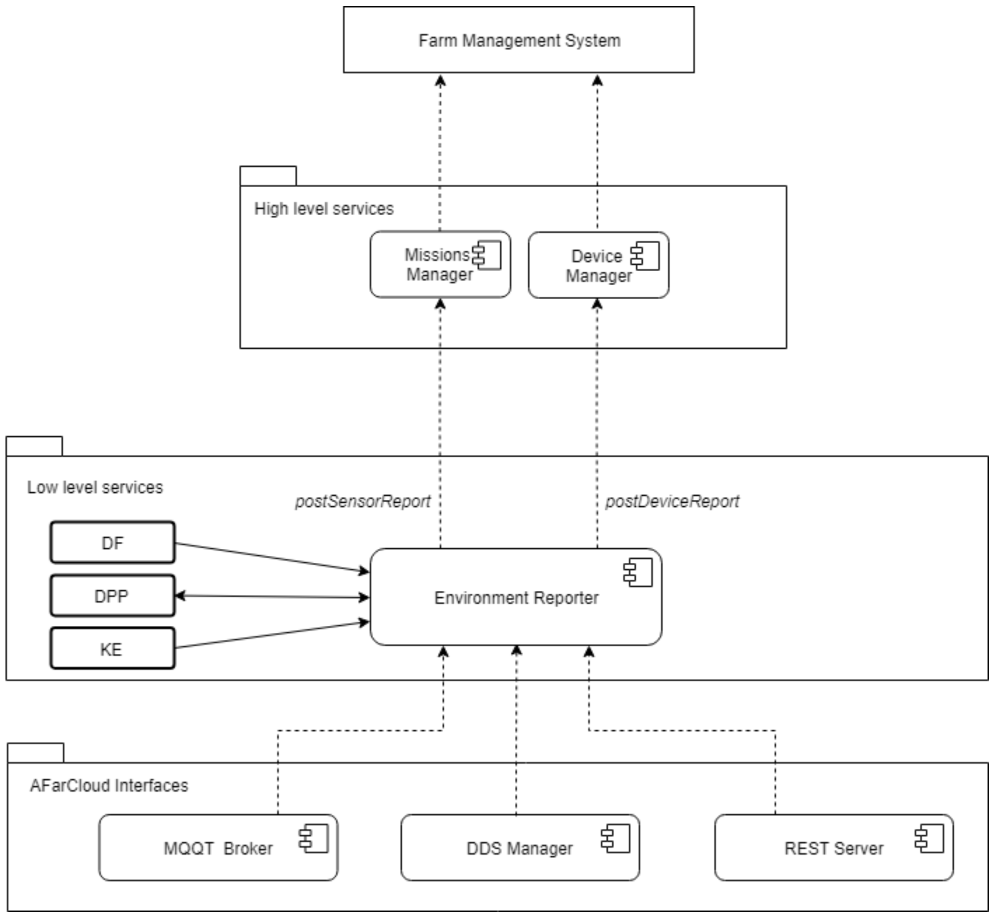

Figure 19: Environment Reporter components diagram

6.13. Data Pre-Processor

6.13.1. Description

The Data Pre-Processor (DPP) component performs (raw) sensor data cleaning and filtering, to avoid missing values, impossible combinations and outliers. All this processing is done online (during missions), with some latency. Pre-processing could be done at the edge level, where raw data is acquired from sensors and/or retrieved if previously stored, to diminish input lag of the values acquired by the sensors. Pre-processed environment data related to livestock and crops, depending on the type of sensor can eventually already include metadata. These data should be stored preferably at an Edge Data Storage for further processing, however can also be forwarded to the cloud repositories/DB to be eventually exploited by the MMT/DSS in the Farm Management System if needed, standalone or together with information/knowledge extracted by the Knowledge Extractor. Also, if such data needs to be correlated with other relevant or associated data from other sensors, which is called Data Fusion, such process is be done within the Data Fusion server/component. By fusing data, it is possible to have a global vision of the data collected from the several sensors in a certain domain.

A more detailed description of the Data Pre-Processor component is available in deliverable D4.1 “Data fusion server”.

6.14. Data Fusion

6.14.1. Description

The Data Fusion (DF) component aggregates data from multiple sources in one single form, making data more readable, consistent and accurate. Such fusion could be associating vehicles proprioceptive data to exteroceptive data acquired by embarked environmental sensors, e.g., associating position (from vehicles own GPS chipset/sensor), as metadata, to the environment sensor readings, namely exploiting timestamps and sensor IDs, respectively. This fusion is done after data from multiple sensors is pre-processed by the Data Pre-Processor, for this reason Data Fusion is not done in real time. Such processing also considers the discrepancies in the timing/synchronization of the data streams that are being fused.
Data fusion is not necessary for all data, but only in some specific cases. In other cases, pre-processing data is sufficient for further processing and reasoning. It is necessary to determine which data will be fused and which will not.

A more detailed description of the Data Fusion component is available in deliverable D4.1 "Data fusion server".

6.15. Knowledge Extractor

6.15.1. Description

The Knowledge Extractor (KE) component exploits environment data (pre-processed, fused and/or including metadata, e.g., georeferenced data) related to livestock and crops, which is previously stored in the AFarCloud cloud repositories, for analysis and extraction of information or knowledge. The outcome of this process should be stored at the cloud repositories, to be eventually used by MMT and/or DSS if needed, standalone or together with data previously pre-processed and/or fused by the Data Pre-Processor and the Data Fusion components, respectively. The outcomes from the Knowledge Extractor should be validated by the Environment Reporter, which is the component in the Middleware that validates and relays such extracted information/knowledge to be stored in the AFarCloud repositories.

A more detailed description of the Knowledge Extractor is available in deliverable D4.4 "Livestock and crop quality assessment framework".

6.16. Image Processing Platform

6.16.1. Description and use cases

The main objective of this component is the integration of the image processing algorithms needed to extract data from the optic sensors, such as: multispectral cameras, hyperspectral sensors, visible cameras or thermal cameras.

Due to the special nature of this component in terms of hardware capabilities (i.e., depending on the data sources and the complexity of the algorithms to be performed, different processing capabilities may be required), in each local or holistic demonstrator, the partners involved will define a customized Image Processing Platform, according to the functionalities required and the hardware and software architecture planned for the demonstrator. Thus, the Image Processing Platform (IPP) is a component deployed at the Edge of the AFarCloud architecture.

The main features of each of the three possible customizations of the IPP, are presented below:

Livestock location and tracking:

The goal of this customization of the IPP is to use machine learning algorithms on aerial images to detect and count pieces of cattle. The results can be used to support farmers with daily inspection, or to help them find and locate lost/new-born animals in large grazing areas. To enable such functionality, the IPP uses a convolutional neural network and deep learning.

Aerial images are captured by RGB cameras mounted on UAVs, stored in a memory stick or SD card, and downloaded to the IPP. As a result of the processing, the IPP returns the detected animals. Users (farmers) can choose to keep the results in text or in image format. The results and/or referenced images are sent to the AFarCloud repositories (as text) and/or to the Image Data Manager (as image) for long term storage.

In the following figure, it is described how the data flows from the devices deployed in the farm, to the AFarCloud Cloud infrastructure:

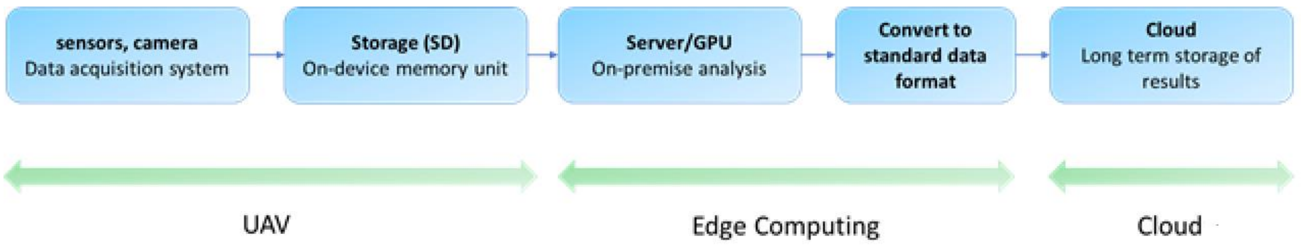

Figure 20: Data flow of the IPP for livestock location and tracking

Vigour, water stress and weeds & dead plants detection:

The objective of this customization of the IPP is to analyse the quality of vineyard cropping through georeferenced mosaics provided by UAVs, which are obtained by the pre-processing of a georeferenced images set of the field.

As a source of the analysis, georeferenced images taken by a UAV flight with a multispectral camera and a thermal camera embarked are used. Besides, GPS and IMU data are also collected. This data is stored in a memory stick or SD card, and downloaded to the IPP. Once this information is uploaded to the IPP, a *multiband georeferenced aerial image* of a pilot area in the vineyard is obtained. The computer vision algorithms of the IPP detect and match hundreds of overlapping images, accurately estimate internal and external camera parameters, create point cloud representations of the 3D surface and finally combine everything into a unique georeferenced multiband *orthomosaic* image.

One of the most notable sets of algorithms for this purpose is *Structure for Motion* (SfM). SfM typically utilizes scale invariant feature transform (SIFT) to locate important features in an image, known as keypoints. Keypoints are then matched in each image based on the minimization of Euclidian distance. These keypoints are then tracked from image to image, enabling the accurate estimation of both camera orientation, as well as the keypoint location.

The result of the process carried out by the IPP under this customization, is the following:

- Vigour and water stress:

  Plant vigour is based on the calculation of the normalized difference vegetation index (NDVI), applied on the specific spectral band data taken, by measuring the difference between near-infrared (which vegetation strongly reflects) and red light (which vegetation absorbs). NDVI vary in ranges between -1 and +1.

  Water stress is based on the calculation of the crop water stress index (CWSI) by measuring the difference between the canopy and air temperature. CWSI vary in ranges between 0 and +1.

- Dead plants and weeds detection:

  These parameters are evaluated with the combination of NDVI index and the training of a convolutional neural network. The goal of NDVI in this task is the vegetation detection, which means, the discrimination between pixels that represent green vegetation and the other pixels. This process enables to simplify and speed up the subsequent plant detection and classification subtasks, providing a mask image. Pixels that belong to vegetation and no vegetation need to be classified between crop and weeds and dead plants.

Generally, the resulting data and not the images due to their large size, will be stored in the cloud repository of the MW.

A data flow diagram for this customization of the IPP is shown below:

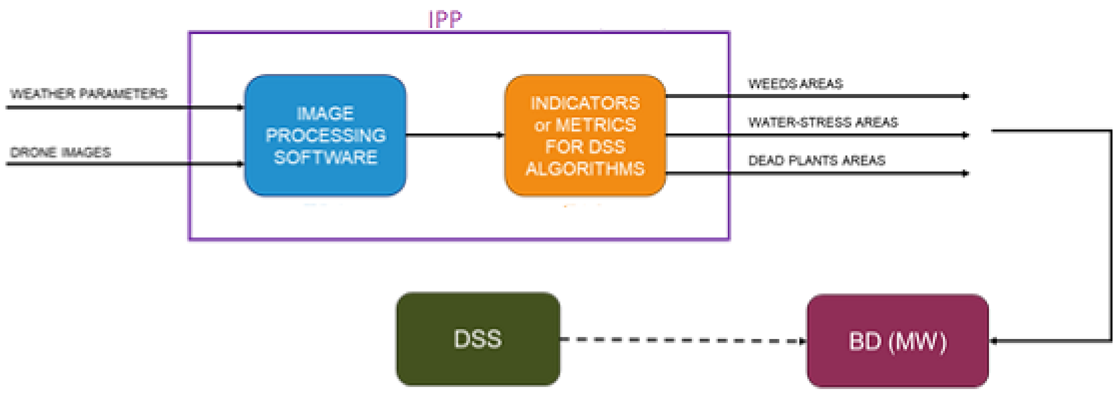

Figure 21: Data flow of the IPP for vigour, water stress, weeds & dead plants detection

Estimation of the main cropping indexes:

This customization of the IPP uses as data sources the images taken by a high-resolution camera and a multispectral camera embarked in a UAV. The goal for the first year of the project is to store these images in a memory stick or SD card, and downloaded to the IPP. Through these images, the IPP estimates the main cropping indexes.

The drawback of this process is the amount of time it takes. We are aware that some farmers that were already using computer vision techniques to obtain spectral vegetation index maps, have stopped doing so because it is a time-consuming process even when running on a high-performance machine (i.e. taking the SD card, copying all files to a computer, running the multispectral mosaic algorithm, and saving the resulting maps may take around 2 hours).

From the first year of the project onwards, we will focus on trying to make the UAV completely autonomous and being able to generate the maps in nearly real-time (e.g. 20 seconds delay maximum). For UAVs with enough capabilities, we will run a grid map algorithm (which is similar to a multispectral mosaic algorithm) in real time, on the Session Border Controller of the UAV. Once the process is finished, the UAV will send the results to the cloud repository once it has landed. For UAVs lacking these high-performance capabilities, we might need to send the information to the cloud and run this algorithm there. This can be seen as distributed computing, as it is described in Figure 22.

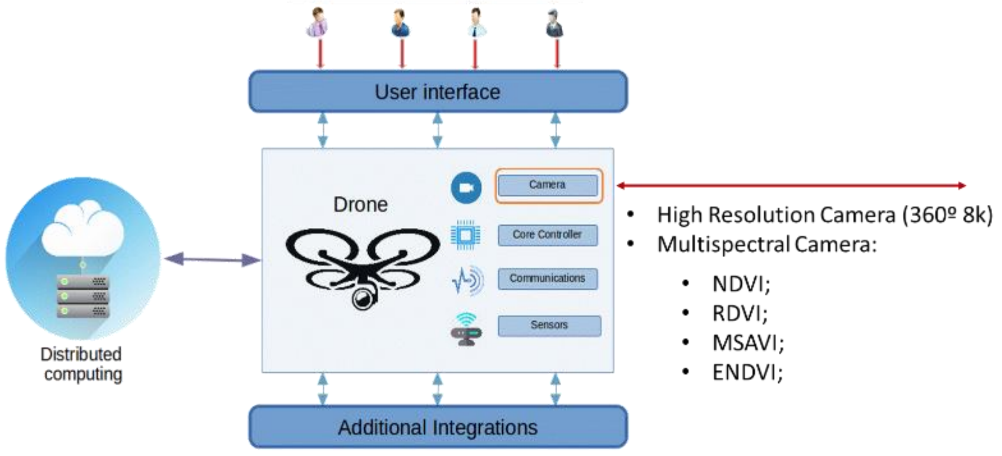

Figure 22: Data flow of the IPP for the estimation of the main cropping indexes through distributed computing

For the estimation of the main cropping indexes, the IPP implementation is based in OpenCV. OpenCV has four distinct motion models, illustrated next:

- Translation → Two scene images shifted by (x,y) (2 parameters);
- Euclidean → Two scene images shifted by (x,y) and/or rotated by an angle (3 parameters);

- Affine → Two scene images transformed by rotation, translation, scale, and shear (4 parameters);
- Homography → Two scene images transformed according not only to 2D effects (as the previous ones) but also with some 3D transformations (8 parameters).

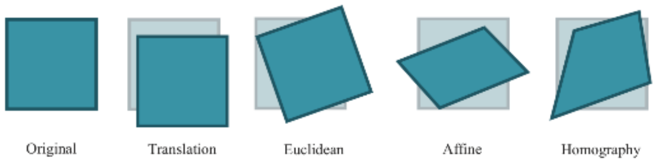

Figure 23: Homography procedure

To be applied to our agriculture case study, from the detected aligned images' keypoints in common - using Fast Library for Approximate Nearest Neighbours (FLANN matcher) - a perspective transformation matrix (homography) is computed between both scenes. Then, the image to be aligned is cropped from the overlapped detected corners and stretched to the same size as the red filtered. The framework's alignment node is the main responsible for converting and aligning the captured images. After the multispectral imagery correction, it is possible to proceed with the lenses' alignment. The image alignments are performed according to the previously mentioned homography method.

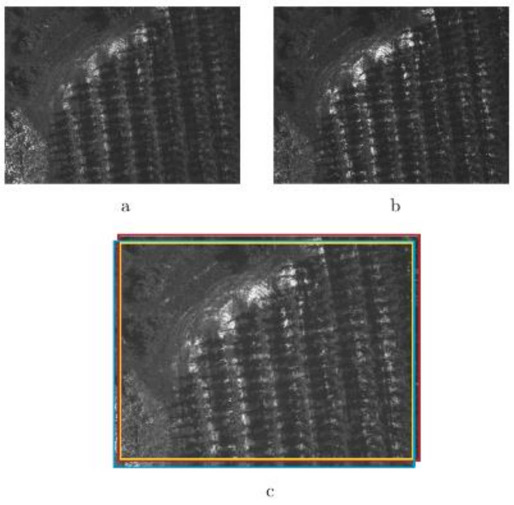

Figure 24: Misalignments between scene images from different lenses. The blue square is surrounding the blue band image (a), whereas the red square is the contour of the red band image (b). The yellow square represents the cropped images after transformations and illustrates the scene area in common (c).

According to the index fusion node, the alignment node's received message needs to be decomposed to obtain the turned-on indices, the images required to calculate those requested indices and the respective GPS and IMU sensor's information.

Hence, the index fusion node is essentially composed by the following two steps:

1. Spectral indices bands' fusion (NDVI, MSAVI, RDVI, ENDVI)
2. Mapping

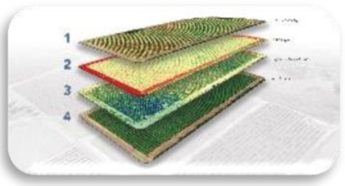

Figure 25: Image layers

Nowadays, grid maps are extensively used in mobile robotic mapping, thus there will be a specific C++ Grid Map library, as a ROS interface, capable of storing distinct types of terrain features, like elevation, variance and colour. Grid maps can also have an unlimited number of layers and it is possible to convert them into other ROS message types, such as PointCloud2, OccupancyGrid and GridCells. For viewing purposes, it is also fully integrated with the ROS' 3D visualization tool, known as RViz.

This Grid map will be constructed by the multispectral camera images and by GPS and IMU data. With this information it is possible to create a dynamic map without using stitching. After the dynamic mapping is created and with the index fusion calculated, it is possible to evaluate the crop healthy (this evaluation was done by NDVI index)

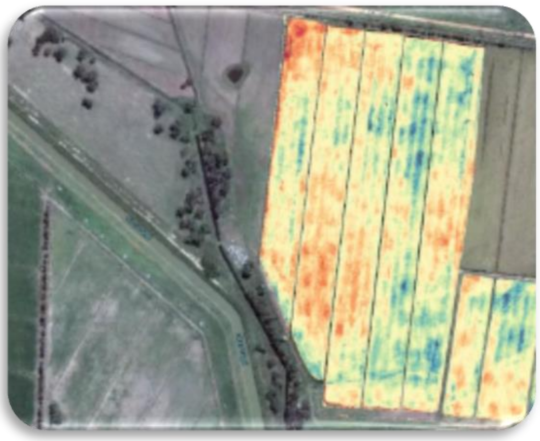

Figure 26: NDVI representation in a crop

6.16.2. Interfaces

The interaction with the other functional components will be:

- Aerial vehicles: Image sensors embarked in UAVs will provide georeferenced mosaics/images to the high-performance IPP. The load of images from UAVs to the IPP will be done through an external storage device (e.g., USB disk, SSD card, etc.)
- Third Party Data: Weather data will be used by the IPP to estimate the water stress in the crop.
- Cloud Data Storage: Data (not images) extracted by the IPP will be stored in the cloud repository in the Middleware.
- Image Data Manager: Images generated by the IPP will be stored in the Image Data Manager of the Middleware.

6.17. Image Data Manager

6.17.1. Description

The Image Data Manager handles, stores and catalogues georeferenced images, image data and other types of raster data. It maintains a record of various data associated with each image, such as the image location data, the acquisition time, the geometric extent on the ground, the geometric models, etc. It can be used to retrieve image data as service (WMS) or image files, where the images can be ortho-corrected, reformatted, trimmed and resampled. The Image Data Manager is based on Keystone Enterprise (COTS).

Following an example of usage is provided: after image data is analysed by the Image Processing Platform and a resulting raster map is created (e.g., with pest affected vegetation), this raster map is uploaded to the Image Data Manager and a WMS layer is created. The WMS server is serving the WMS layer to the MMT, which shows the map to the end user. The retrieved images, or a selection from them, are transferred to the Image Data Manager to be stored and catalogued so they can be used later.

6.17.2. Interfaces

- Image Processing Platform:
  - Receives images that need to be stored and catalogued;
  - Receives result map data in raster-form to be shown in MMT;
  - Sends images to Image Processing Platform for time-series and longitudinal analysis.
- MMT:
  - Shows image data through the WMS service.

6.18. MQTT Broker and MQTT Clients in the Middleware

6.18.1. Description

MQTT is a Client Server publish/subscribe messaging transport protocol. Since MQTT decouples the publisher from the subscriber, client connections are always handled by an MQTT broker. It is important to note that the MQTT broker uses the topic of a message to decide which client receives which message. In MQTT, the term "topic" refers to an UTF-8 string that the broker uses to filter messages for each connected client. The topic consists of one or more topic levels. Each topic level is separated by a forward slash (topic level separator).

6.18.2. MQTT QoS configuration

The Quality of Service (QoS) is defined as the agreement between the sender and the receiver of a message. There are 3 possible QoS levels in MQTT to guarantee the delivery for a specific message:

- <u>QoS 0 – at most once</u>: only delivers the message to those online and subscribed to that topic at the time of data release (similarities to how UDP protocol works).
- <u>QoS 1 – at least once</u>: the message is delivered and there is evidence of that delivery but no assurance that the delivery is unique (potential duplicity of package delivered because the ACK has not been received / properly interpreted).
- <u>QoS 2 – exactly once</u>: is delivered and there is evidence of it.

The client that publishes the message to the broker defines the QoS level of the message when it sends the message to the broker. The broker transmits this message to subscribing clients using the QoS level that each subscribing client defines during the subscription process. If the subscribing client defines a lower QoS than the publishing client, the broker transmits the message with the lower quality of service.

QoS is a key feature of the MQTT protocol. QoS gives the client the power to choose a level of service that matches its network reliability and application logic. Because MQTT manages the re-transmission of messages and guarantees delivery (even when the underlying transport is not reliable), QoS makes communication in unreliable networks a lot easier.

Within the AFarCloud context, as long as it is considered that the duplication of messages is not critical, the ideal seems to be to publish with a quality of service level 1, having subscribers using QoS

1 as well. However, if there is an area of the deployment where such possible duplication becomes problematic, it would be necessary to resort to a QoS 2 level.

6.18.3. List of MQTT topics

Considering the data that is going to be exchanged in the AFarCloud scenarios, we propose to follow a topic structure based on a variation of the OMA NGSI-LD information model (see D2.6 for more details). An example of a topic for reporting observations from sensors following this model is: `project/scenario/service/type_of_device/deviceID/data`

where `type_of_device` represents a standalone sensor, a sensor mounted on a GV, an actuator or a gateway.

The topic structure proposed above is considered the ideal for the deployments within the AFarCloud demonstrations, although it is more complicated to subscribe to the specific data of interest, since it is necessary to know in detail where to subscribe.

However, to somehow ease this burden, when a client subscribes to a topic, it can subscribe to the exact topic or use wildcards to subscribe to multiple topics simultaneously. A wildcard can only be used to subscribe to topics, not to publish a message. There are two different kinds of wildcards: *_single-level (+)*, that replaces one topic level, and *_multi-level (#)*, that covers many topic levels.

It is worth noting that in a scenario where anyone can exploit all the data, keeping a single topic for all sensors could seem like a legitimate option. Nevertheless, it has implications of receiving too many data, perhaps even not knowing the format followed by a new device entering the system, and also may affect the overall functioning due to the processing capacity requirements that it would impose.

Another option, if there is a different cloud for each project scenario, would consist of reporting to structures based on the establishment of the so-called *mount points*.

Taking all these considerations into account, a list of the topics for the communication between MQTT devices and the MQTT Clients (publishers and subscribers) of the Middleware is provided below:

A. REPORTING A NEW MEASUREMENT SENT BY A DEVICE TO THE MIDDLEWARE

Proposed topic structure:

`afarcloud/scenario/service/sensor/sensorID/measure`

Proposed data model:

`ref: "#/definitions/Measurement"` (see Annex 2. Data Definitions)

B. REPORTING A LIST OF MEASUREMENTS SENT BY A DEVICE TO THE MIDDLEWARE

Proposed topic structure:

`afarcloud/`*`scenario/service/type_of_device/deviceID/`*`measureList`

Proposed data model:

`ref: "#/definitions/SensorDataList"` (see Annex 2. Data Definitions)

C. REPORTING ALARMS SENT BY A DEVICE TO THE MIDDLEWARE

Proposed topic structure:

`afarcloud/`*`scenario/service/type_of_device/deviceID/`*`alarm`

Proposed data model: TBD

D. SETTING SAMPLING RATES TO SENSORS

Proposed topic structure:

`afarcloud/`*`scenario/service/`*`sensor/`*`sensorID/`*`samplingperiod`

Proposed data model:

`ref: "#/definitions/SamplingPeriod"` (see Annex 2. Data Definitions)

E. SENDING COMMANDS TO ACTUATORS

Proposed topic structure:

`afarcloud/`*`scenario/service/`*`actuator/`*`actuatorID/`*`action`

Proposed data model: TBD

F. INFORMING ABOUT FIRMWARE UPDATES TO DEVICES

Proposed topic structure:

`afarcloud/`*`scenario/service/type_of_device/deviceID/`*`fwupdate`

The projected deployment will require for some of the equipment to receive certain firmware updates. To do so, MQTT could be used accordingly. Depending on the device to update, different mechanisms for firmware updates could be used:

- Standard MQTT management: MQTT subscribers of sensor nodes would subscribe at the MQTT Broker at start-up for firmware updates. A specific MQTT topic would represent a firmware update and could include the sensor type and vendor (e.g., /sensor/ams/ENS160). The MQTT message for a firmware update would include the software version and the location where to fetch the updated binary. This might be stored on a vendor specific platform. In case a firmware update for a sensor was triggered, it would fetch the software version from the sensor, compare it with the version on the sensor, and download the update if needed. The sensor node would fetch the binary from the location given by the MQTT message and would deploy it to the sensor via the available physical sensor interface (e.g., I2C, SPI). Software binaries might be provided in a vendor proprietary format. This might be even encrypted. Neither the MMT nor the MQTT broker/subscriber would need to interpret the content of such an update.
- Sensor ID replication: employing complementary tools that would allow us to replicate a sensor ID, furthermore, let that sensor subscribe to a certain topic in order to later on receive the available update. Once the transaction is completed, we could unsubscribe the real sensor again to avoid that undesired malicious activities. The equipment waiting to receive firmware updates would not be subscribed to a specific topic: thus, no one could maliciously subscribe to see what happens (performing some kind of phishing). When someone wanted to send something specific to that device, you could resort to subscribe it, using its unique ID, from a complementary tool, carry out the desired update and unsubscribe it later once the transaction is formalized (e.g., Mosquitto, a well-known open source MQTT Broker, allows it). There would be no conflict of repeated IDs as long as the real device and the simulated one are not connected at the same time. Depending on the broker used, the action before the arrival of two identical IDs would be one or the other: expel the one that was already there, or discard the one that arrives last. As hinted earlier, the Mosquitto MQTT broker implementation would expel the one that was currently subscribed, since it could be a spurious one that remained after a sudden disconnection or fall of the broker. In any case, this manual process could be considered as complementary to the other fully automated procedures. This option would rely on a “subscribe trigger” that is decoupled from the network connection and the subscription would be triggered via NFC (Near Field Communication) technology or a button on the sensor node.
- Firmware updates protection through certificates: by using SSL over MQTT, a chain of certificates could be setup. In this case, SSL certificates would be used to protect the confidentiality of the

firmware exchange on the network. In theory, only one certificate would be needed if we trust all the components of the project partners. However, this would mean that firmware updates from one partner are potentially seen by other partners as well. If partners require to protect their data from other partners, additional certificates could be installed. This would make sure that communication between the cloud and partner A is encrypted with a different key than communication with the cloud and partner B.

- Part-by-part firmware update: for devices that do not feature a huge memory, an approach based on downloading a full firmware update could mean the device is cannot handle it. In those cases, a part-by-part update would be welcome.

G. INFORMING ABOUT DSS ALERTS

Proposed topic structure:

afarcloud/*scenario/service/dss/dssID/*dssalert

The FMS will be able to inform the rest of the elements in the architecture about alerts generated by the DSS. The FMS will implement a MQTT Publisher that will publish alerts triggered by the DSS, so they can be available for interested subscribers.

6.19. REST Server and REST Services

Following a similar approach as in the case of MQTT we propose the following REST services for reporting observations from sensors, and for sending data from collar devices to the Environment Reporter (see Annex 2. Data Definitions for definitions of the data model).

SENSOR

POST **/sensor/measure** Store a new measurement
Parameters: name: body schema: ref: "#/definitions/Measurement"
Responses: 200: "Successful operation" 405: "Invalid input"

POST **/sensor/measureList** Store a list of measurements sent by a device e.g., a UAV or a gateway
Parameters: name: body schema: ref: "#/definitions/SensorDataList"
Responses: 200: "Successful operation" 405: "Invalid input"

COLLAR

POST **/collar/measure** Store a new measurement from a collar device
Parameters: name: body schema: ref: "#/definitions/CollarData"
Responses: 200: "Successful operation" 405: "Invalid input"

POST **/collar/measureList** Store a list of measurements from 1 to n collar devices
Parameters: name: body schema: ref: "#/definitions/CollarDataList"
Responses: 200: "Successful operation" 405: "Invalid input"

REGIONS

POST **/region/measure** Store a new region measurement from IPP
Parameters: name: body

schema: ref: "#/definitions/RegionData"
Responses: 200: "Successful operation" 405: "Invalid input"

POST **/region/measureList** Store a list of region measurements from 1 to n
Parameters: name: body schema: ref: "#/definitions/RegionList"
Responses: 200: "Successful operation" 405: "Invalid input"

6.20. ISOBUS Gateway

6.20.1. Description

The ISOBUS Gateway provides an interface for data exchanging between the AFarCloud middleware and the ISOBUS systems, by converting the planned field tasks into ISO11783-XML, the ISOBUS standard data format; the resulting file will be manually loaded by an ISOBUS-compliant semi-autonomous agricultural ground vehicle. In addition to this, the ISOBUS Gateway can also convert the XML data logged during a treatment in order to be uploaded into the middleware.

The ISOBUS gateway will basically act as:

- A Content Delivery Network (CDN), in order to deliver to each user every ISO11783-XML file that has been created by the architecture globally, with low latency, ensuring high performance when the file is downloaded by the user from the cloud. There will be an ISOBUS repository of ISO-XML files from where the user will download them. The CDN will provide the infrastructure to deliver these files;
- A Data Format Converter, from the mission sent by the FMS in an AFarCloud compatible data format (and forwarded by the Mission Manager to the corresponding ground vehicle) to the ISO11783-XML de-jure standard.

6.20.2. Functionalities

The relevant data flow is described in Section 7.3. The additional functions in this scenario that the FMS (through the MMT) will accomplish are:

1. Create a prescription map containing the task data, thus performing a data mining across these sources of data:
   I. GIS database;
   II. Data coming from any kind of sensors (e.g., on-the-field sensor networks, UAV images and sensors, on-board tractor sensor, etc.)
   III. Agronomist & DSS inputs to assess the mission's purpose, chemicals/water employed, etc.;
2. to feed the ISOBUS Gateway with a proper descriptor (hereafter mission file)

The exchanged data format (i.e., the mission file) between the Mission Manager and the ISOBUS Gateway should resemble a geospatial raster data (e.g., the prescription map or georeferenced treatment values for the selected task) containing the mission actions.

On the one hand, the functions of the ISOBUS Gateway when a user triggers the generation of an ISO11783-XML file to be downloaded are (see Figure 27):

1. to be invoked by the Mission Manager when the FMS sends a mission plan;
2. to get as input the mission plan and format it to ISO11783-XML file;
3. to export the ISO11783-XML file and store it in a specific ISOBUS repository, accessible by the MMT;
4. to deliver the file to the final user (i.e., the user, up to now (M12), will only download it and put it in a USB drive and upload it onto an ISOBUS tractor), via the CDN.

On the other hand, the end-user might upload an ISO11783-XML log file resulting of a treatment. This usually contains georeferenced, treatment-specific (depending on the implements employed) logs of the treatment itself, such as:

- the covered area;
- Punctual/average/total amount of mass (e.g., fertilizer, pesticide, seeds, etc.);
- Time elapsed;
- Linear length covered;

Thus, the ISOBUS Gateway shall be also able (see Figure 28):

1. to store the manually uploaded ISO11783-XML file in the ISOBUS repository;

2. to convert the XML log file back on the AFarCloud format to report on the result of GVs missions;
3. deliver it back to the Mission Processing & Reporter;

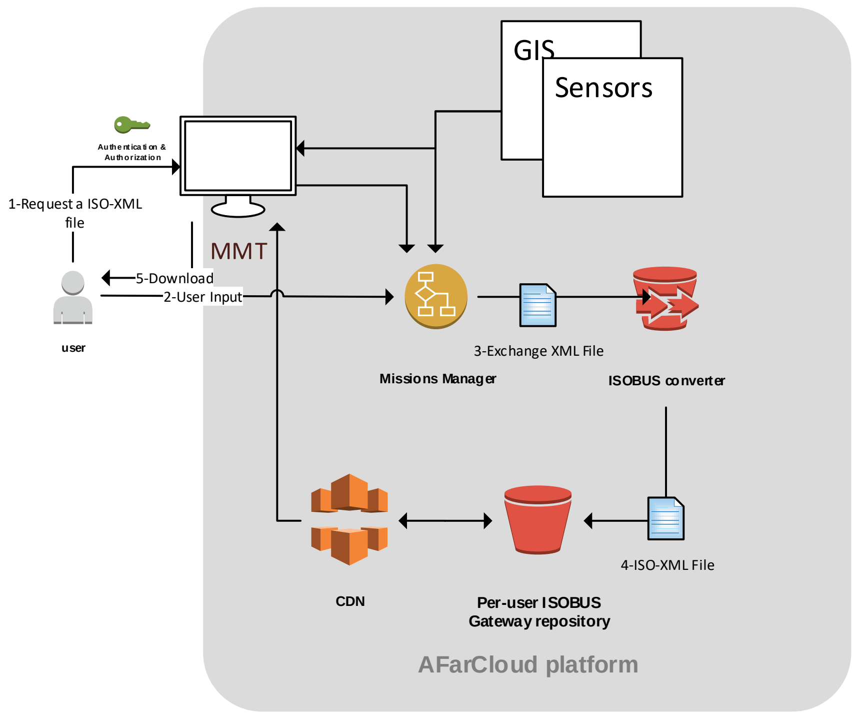

Figure 27: ISO11783-XML download

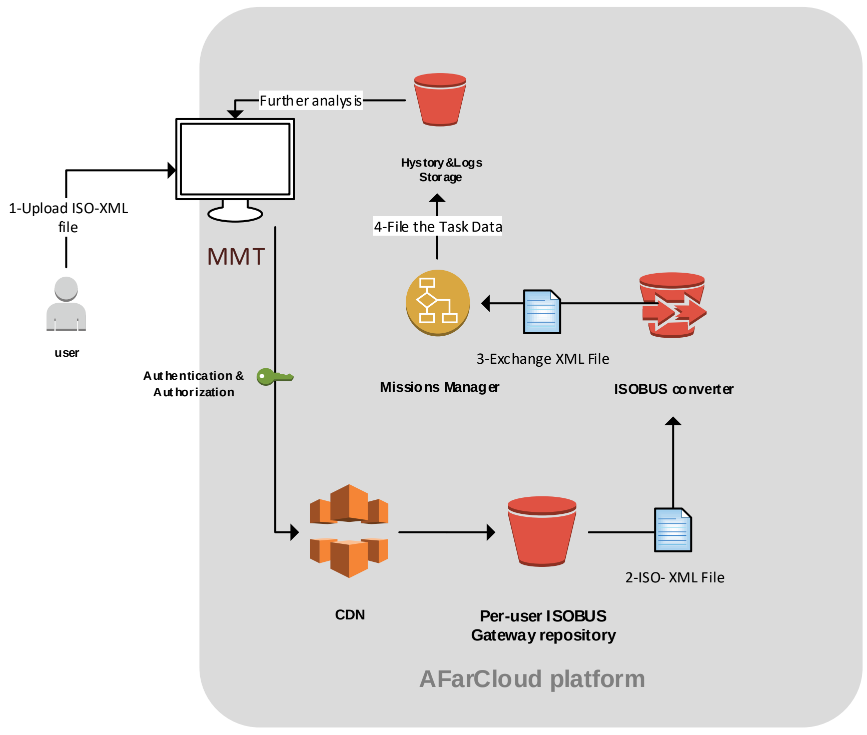

Figure 28: ISO11783-XML upload

6.20.3. ISO11783-XML

The ISO11783-XML files are formatted according to XML definitions version 1.0 with text coded in UTF-8 and they could reference optional binary-coded data files for grid cell definitions or logged process data. The validity of an ISO-XML file is defined by the schema ISO11783_TaskFile, that can be found at https://www.isobus.net/isobus/attachments/files/ISO11783_TaskFile_V2-1.xsd .

The task is the central XML element in that describes an ISOBUS task and it contains references to various other XML elements to express allocation of resources and specification of operations.

The XML file also allows defining the Treatment Zones that specify georeferenced values for the task. The treatment zones are obtained by the prescription map created (e.g., by the Farm Management System or by third-party data services) and they can be described as grid cell or as polygons as shown in Figure 29.

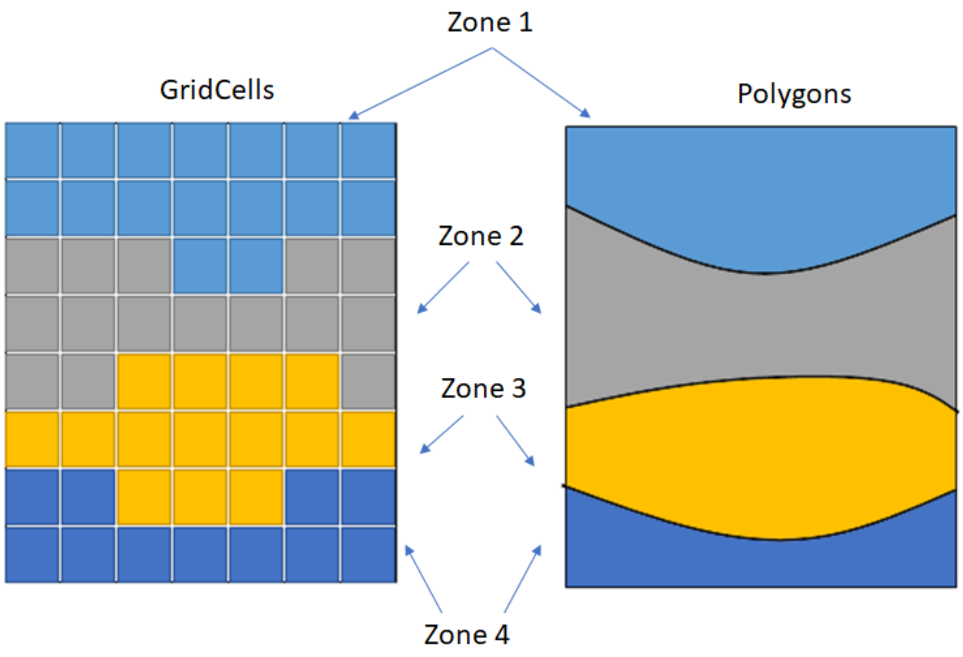

Figure 29: Treatment zone examples

6.21. DDS Manager

6.21.1. Description

The DDS Manager is based on the open source DDS libraries provided by ADLINK. This DDS Community Edition is a full-featured open source, genuinely free implementation of the Object Management Group (OMG) Data Distribution Service for Real-Time Systems standard.

The goal of the DDS Manager is to implement a DDS interface in the AFarCloud Middleware to allow the exchange of messages with DDS-compatible devices at the Hardware Level (i.e., UAVs). By means of this DDS interface, the Middleware will be able to (a) send actions to UAVs; (b) collect the status of actions; (c) collect data from sensors gathered by UAVs, and (d) collect alarms generated by UAVs. All data flows exchanged through this interface will follow the AFarCloud common data model defined in *D2.6 Semantic Middleware*, specifically the data format for missions with UAVs.

Conceptually, DDS manages a global data centric space, that we will call the AFarCloud DDS dataspace. DDS provides QoS-controlled data-sharing. A global data centric space can be divided into different isolated partitions. Under a data partition, applications communicate by publishing and subscribing to UTF-8 strings called Topics, identified by their topic name. A partition has to be explicitly joined in order to publish data in it or subscribe to the topics it contains. Subscriptions can specify time and content filters and get only a subset of the data being published on the Topic.

In AFarCloud, we will create a different DDS partition for each of the scenarios to be deployed. For a particular scenario, the DDS Manager will publish actions for UAVs in the scenario partition of the AFarCloud DDS dataspace and will subscribe to data published in this partition by the DDS Proxy of each of the UAVs. Different topics will be created to publish missions and events, and to subscribe to alarms, battery measurements, locations, observations and images as defined in D2.6.

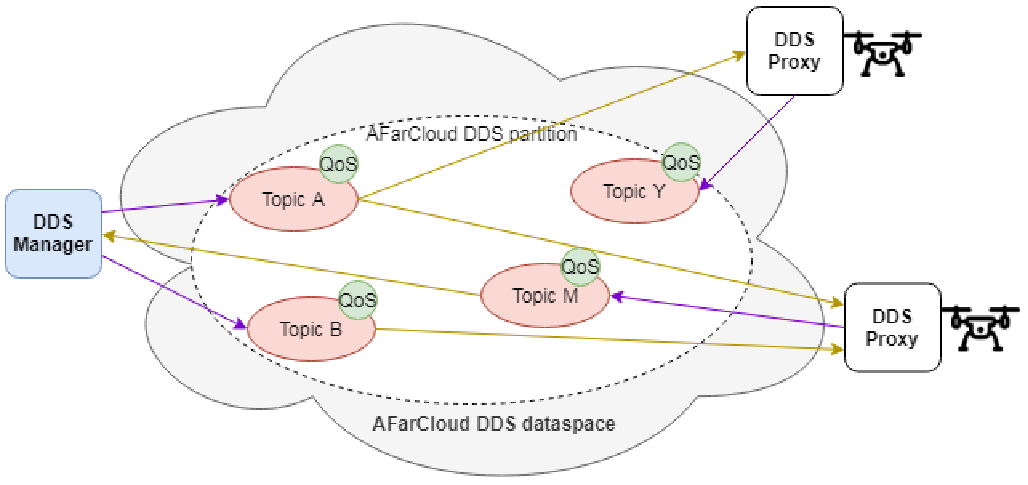

Figure 30: AFarCloud DDS dataspace

6.21.2. DDS Topic QoS configuration

DDS uses a ‘request *vs.* offer’ QoS-matching approach: a data reader matches a data writer if and only if the QoS it is requesting for the given topic does not exceed the QoS with which the data is produced. QoS settings can be linked to the topics defined by the DDS Manager. Different QoS settings can be applied to different topics. The QoS settings of the DDS Manager should be correctly configured according to the AFarCloud requirements. Depending on the durability and the reliability of the information that is published, we can group data into 3 categories: soft state, hard state and alarms:

- A soft state is a state that is periodically updated: e.g., the location of an UAV. In this case the suggested QoS parameters are:

Reliability → BestEffort

Durability → Volatile

History →KeepLast (n)

Deadline →updatePeriod

LatencyBudget →updatePeriod/3

DestinationOrder→SourceTimestamp

Liveliness →Fixed timeout

- A hard state is a state that is only sporadically updated and that often has temporal persistence requirements: e.g., a picture of an obstacle. In this case the suggested QoS parameters are:

Reliability → Reliable
Durability → Transient | Persistent
History →KeepLast (n)
DestinationOrder→SourceTimestamp
Liveliness →Fixed timeout

- Alarms are described as the occurrence of something noteworthy for our system, e.g., a collision alert, the battery below a given threshold, etc. In this case the suggested QoS parameters are:

Reliability → Reliable
Durability → any
History →KeepAll
DestinationOrder→SourceTimestamp
Liveliness →Fixed timeout

6.21.3. List of DDS topics

The management of the topics in DDS is different from that of MQTT in terms of memory usage and data porting. For scenarios in which it is desired to receive samples concurrently, it is necessary to carefully design the number of topics to be used, as the level of concurrency remains at the granularity of the subscriber. Taking into account these considerations and the different scenarios that will be carried out in AFarCloud, it is proposed to use a different DDS partition name for each of the scenarios to be deployed: e.g., partition name for scenario AS01: `scenario_AS01`

Besides, each of these partitions will manage the same set of AFarCloud topics. In order to define the list of DDS topics to be used in the AFarCloud scenarios, we have considered the data format for information transfer proposed in D2.6.

The end points for the communication between the Middleware and the UAVs will be the DDS Manager (in the Middleware) and the DDS Proxy of each of the UAVs. The DDS topics to be used by the DDS Manager in any of the AFarCloud DDS partitions are the following:

a) <u>Data sent from the DDS Manager to all DDS Proxies</u>:

- New missions:
  - Topic name for Publisher: “*mission*”
  - QoS settings for the topic: TBD
- New events:
  - Topic name for Publisher: “*event*”
  - QoS settings for the topic: TBD

b) <u>Data received by the DDS Manager from any DDS Proxy</u>:

- Report of a mission:
  - Topic name for Subscriber: “*mission_report*”
  - QoS settings for the topic: TBD
- Alarm:
  - Topic name for Subscriber: “*alarm*”
  - QoS settings for the topic: TBD
- Observation:
  - Topic name for Subscriber: “*observation*”
  - QoS settings for the topic: TBD
- Location and orientation:
  - Topic name for Subscriber: “*pose*”
  - QoS settings for the topic: TBD
- Battery level:
  - Topic name for Subscriber: “*battery*”
  - QoS settings for the topic: TBD
- Image:
  - Topic name for Subscriber: “*image*”
  - QoS settings for the topic: TBD

6.21.4. Components diagram

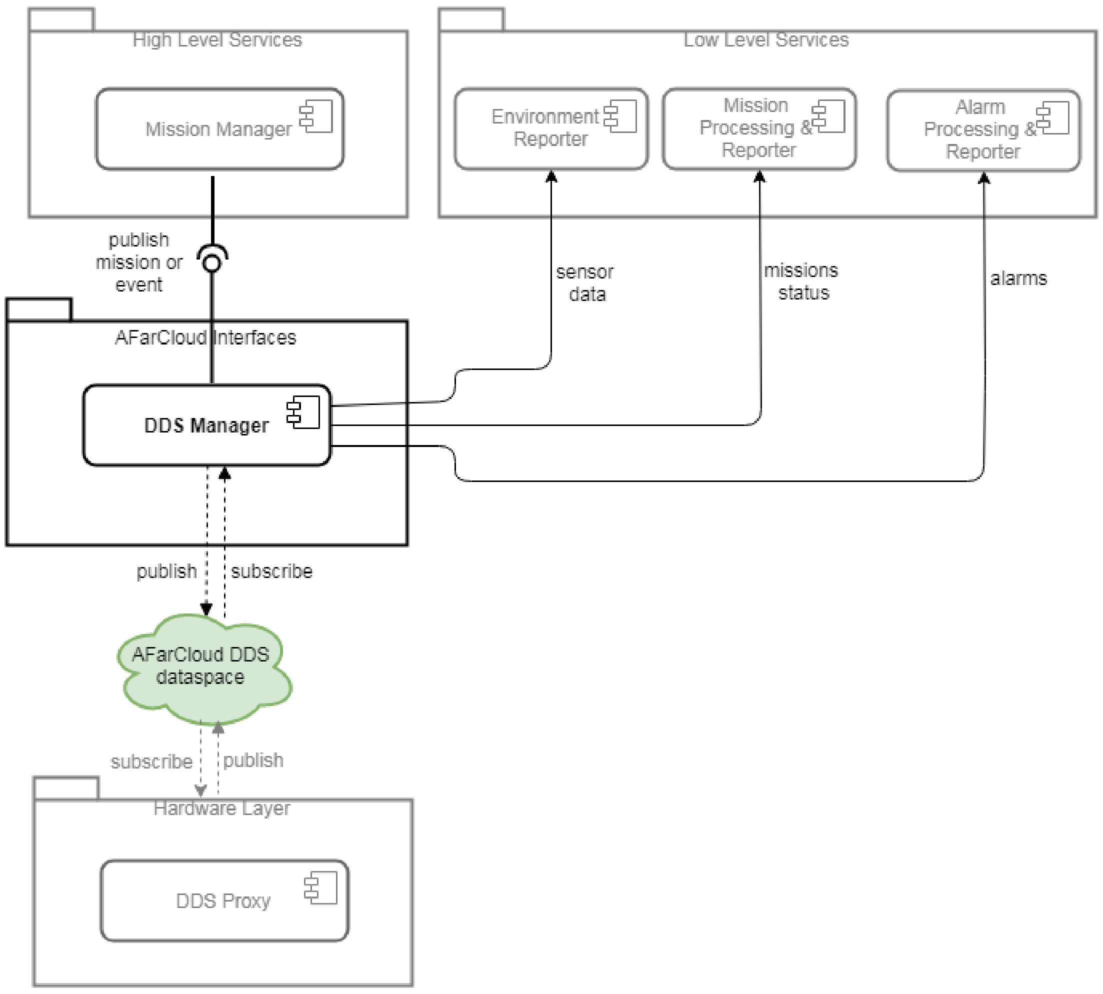

Figure 31: DDS Manager components diagram

6.21.5. Interfaces

Table 8. DDS Manager interfaces

(*public*) **publish** (topic_name, AFarCloudsmsg)	This method is used to publish data compliant with any of the AFarCloud's IDLs defined in D2.6, for the specified topic.

7. Data Flow diagrams

7.1. Send a mission to a UAV

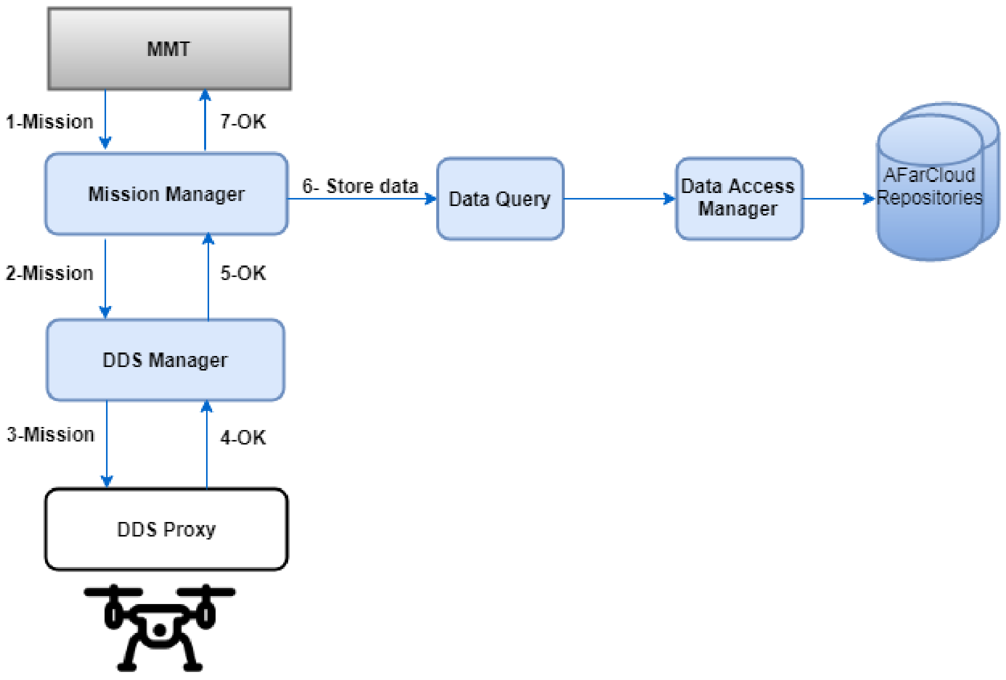

Figure 32: Send a mission to a UAV

7.2. A UAV sends data to the Farm Management System

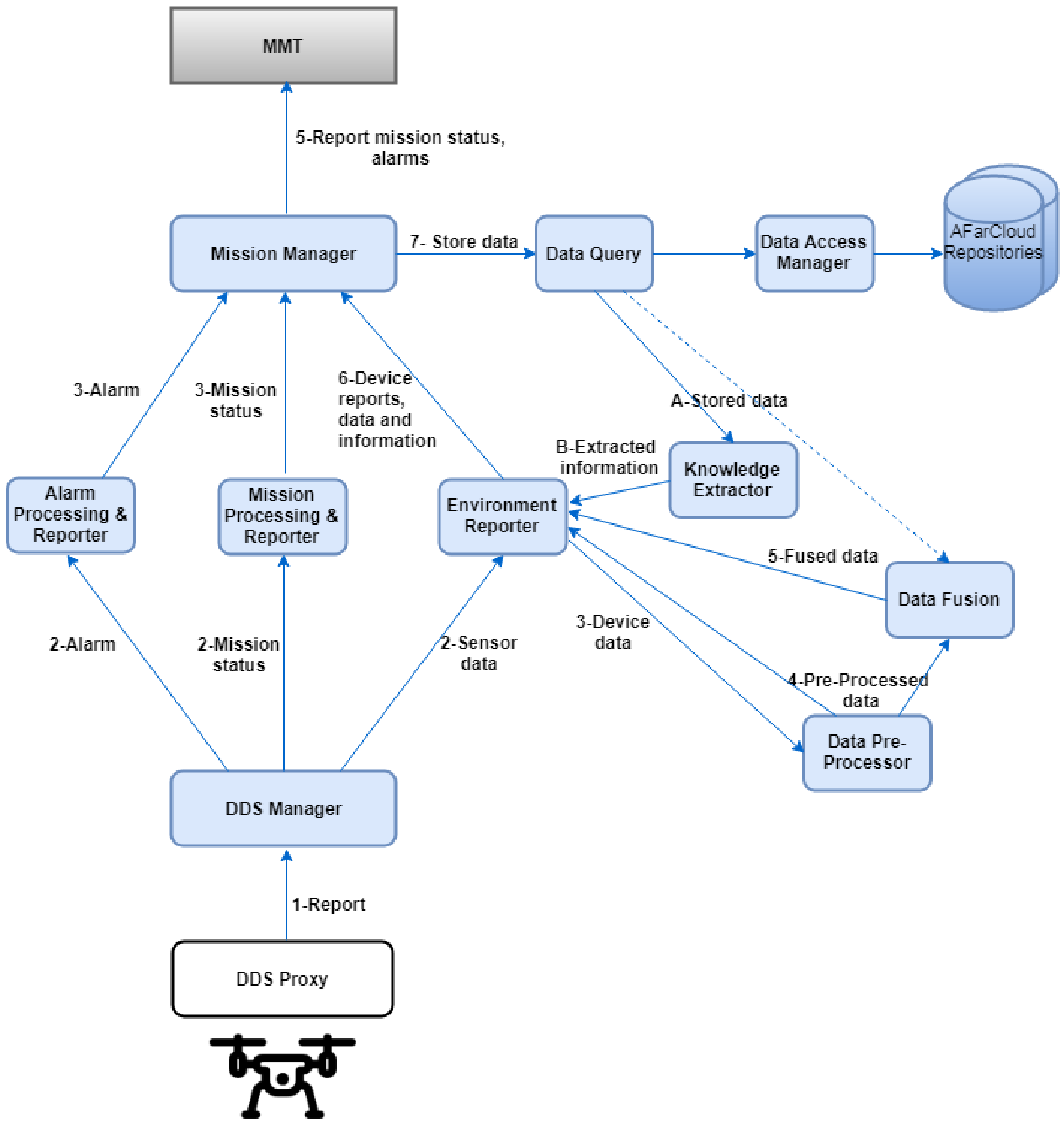

Figure 33: A UAV sends data to the FMS

7.3. Send a mission to a semi-autonomous ground vehicle

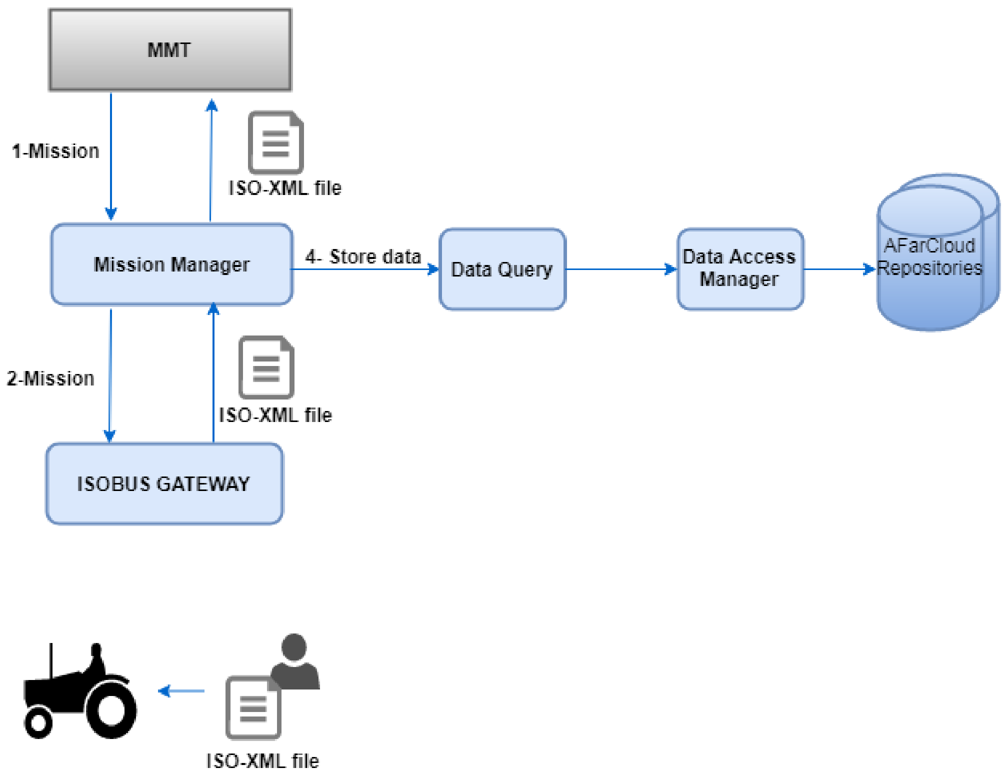

Figure 34: Send a mission to a tractor

7.4. Configure a sampling rate on a MQTT device

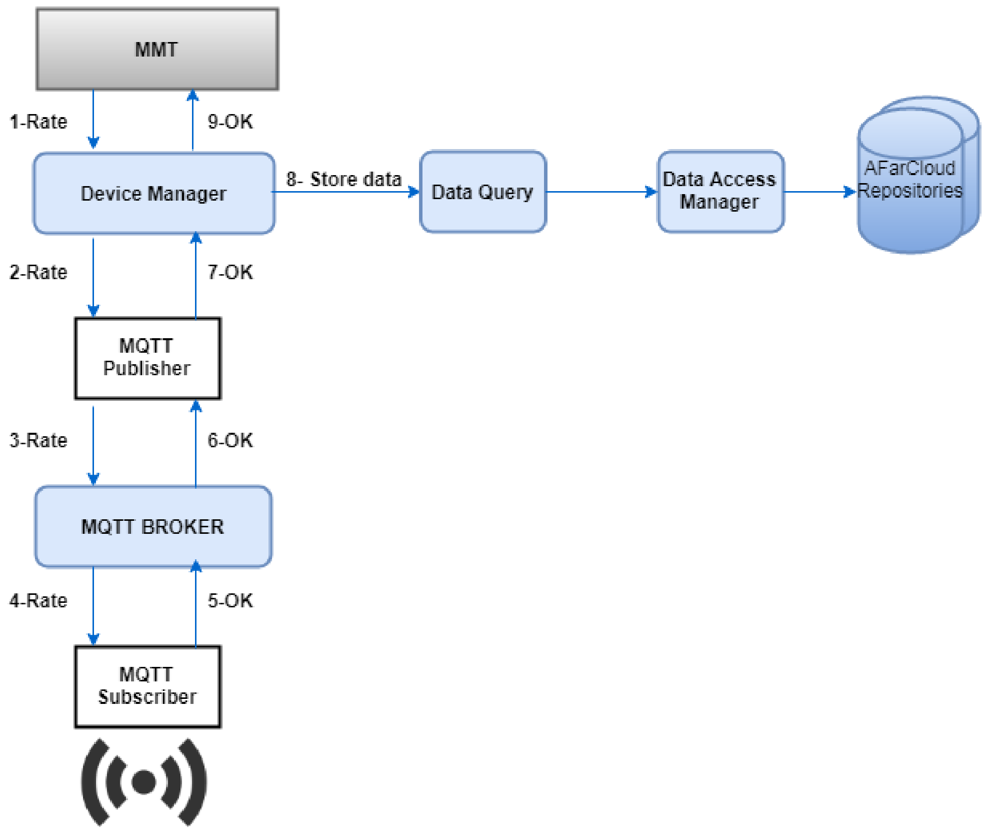

Figure 35: Configure a sampling rate on a MQTT device

7.5. Devices send data to the platform

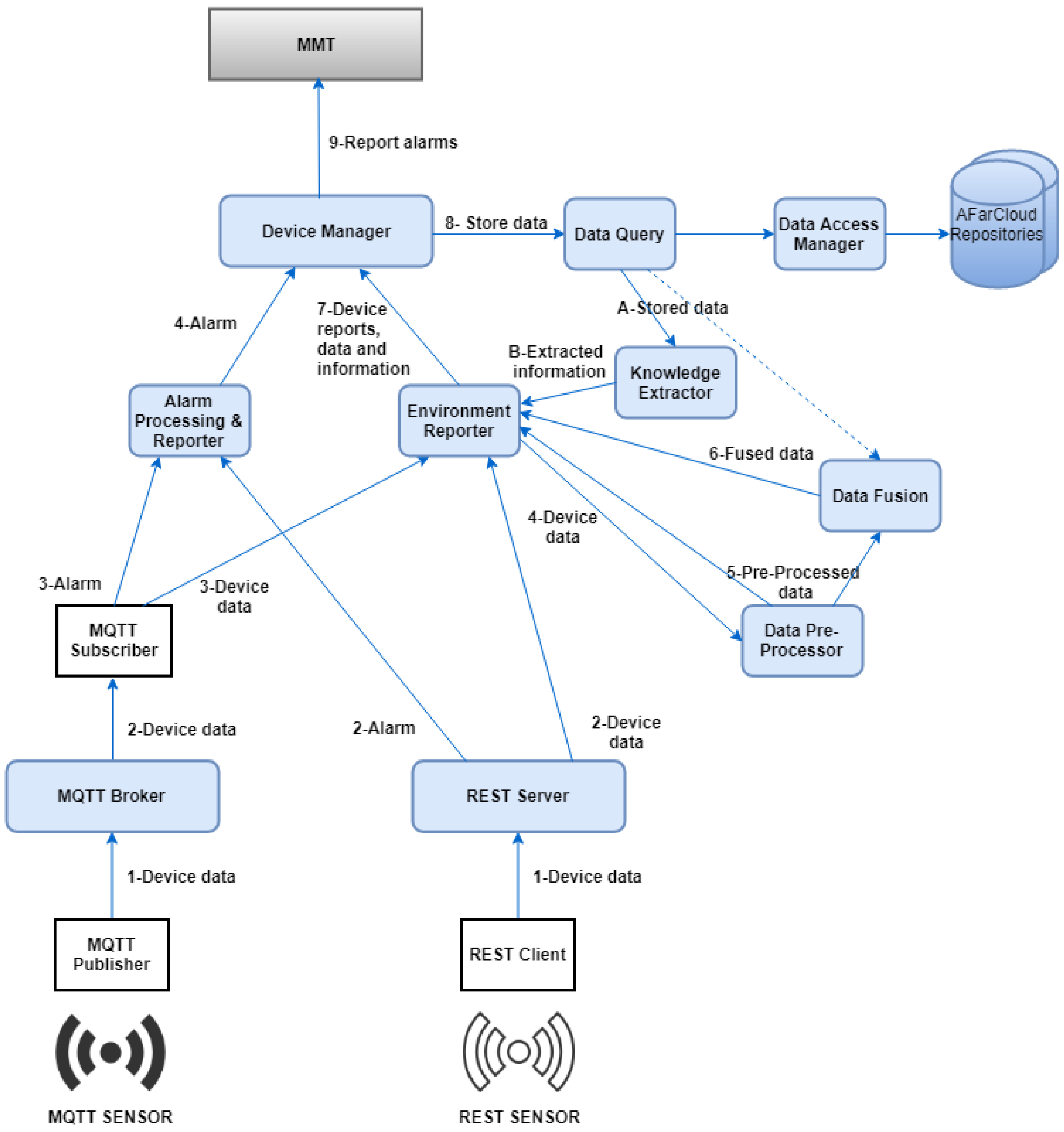

Figure 36: Devices send data to the platform

Annex 1. WSN Technologies

Available technologies for supporting the physical layer (radio interface) for the sensor networks include several established standards such as Zigbee, Bluetooth and WIFI, among others, for short range networks. Long range LPWAN (Low Power Wide Area Networks) standards and technologies such as LoRa, Sigfox and NB-IoT provide a solution when distance between the sensing nodes and the gateway/sinking node is high, in the range of hundreds of meters to kilometres.

Short range radios, in the context of the AFarCloud project, are expected to be used in limited areas, such as barns, farm buildings or limited open areas. Since radio range is short, to expand the area covered by such wireless sensor networks, the area covered is expanded by increasing the size of network, either by adding more nodes (to allow node to node hop communication), more gateways or both.

Short range networks allow higher bit rates and hence higher speeds in data transmission and also provide lower latency (the time between the data is collected until it is received by its destination) and so are ideal for low latency applications namely real time monitoring, such animal health or industrial process.

If the area to be covered is too large, a LPWAN based wireless sensor network provides a solution by deploying sensing nodes that only communicate with gateways that are distant at hundreds of meters to several kilometres by using more efficient radio protocols. Since LPWAN coverage is in the range of tens of kilometres on open space, it allows direct communication between the sensing node and the remote gateway without the need to deploy nodes as an intermediate hop to cover the larger distances, which is required on the short-range networks.

The LPWANs key objective is low power usage and a higher radio range, however this is achieved at the cost of lower bit rate, which impacts the latency, which is much higher than the latency of short-range networks. Due to this, LPWANs are not adequate to do real time monitoring or real time process control but are the ideal solution for low power and sparsely located remote sensing nodes.

Other possible alternative solutions to increase the area covered by the wireless sensor network, is to store locally and temporarily the sensor data on a central node/gateway, and collect it, sometime into the future, either by a fly by UAV that collects the stored data or by an agricultural vehicle equipped with the proper hardware. In such cases sensors relay data to one or more central gateways (in topological sense, not geographic) that accumulates the collected data and buffers/save it temporarily before it is downloaded to a mobile gateway (UAV or vehicle). As we can infer, latency is very high, and so such networks aren't adequate for real-time monitoring but adequate to long term monitoring.

Some parameters for the most common short-range network standards are compared in Table 9:

Table 9: Comparison of short-range network standards

	Zigbee/6LoWPAN	**Bluetooth**	**Wifi**	**3G/4G LTE**
Data Rate	20,40,250 kbps	1000-3000 kbps	300 Mbps	7.2Gbps/100Gbps
Range	10-100 m	50 m	100m	1-3 km
Battery Lifetime	1-2 years	Weeks to 1-2 years	1-2 days	1- 4 weeks
Standard	IEEE 802.15.4 Zigbee (Zigbee Alliance) 6LoWPAN (IETF Open)	IEEE 802.15.1	IEEE 802.11 (n)	GSM/UMTS/LTE
RF Band	ISM Bands	ISM Bands	ISM Bands	Licensed
Optimized for	Reliability Low Power Low Cost	Low Cost Convenience	Speed	Speed Reach
Network	Mesh, Star, Cluster, Tree Ad-Hoc Networks	Master – Slave	Star, Ad-Hoc	Star
Applications	Industrial control & monitoring WSN Home Automation	Personal data connectivity	Wireless LAN access	Mobile Wireless WAN access
Node Hardware	Available	Available	Available	Available
Gateway Hardware	Available	Available	Available	Not Available

Wireless sensor networks based on short range radios, such as Zigbee and 6LoWPAN (IPv6 low power wireless personal network), depend on the capacity of adding nodes that communicate with each other to build networks that have a greater coverage than one single node can cover. The capacity of adding nodes to the network means that these protocols allow the construction of different network topologies, which allow the network to be adapted to the real-world use case as needed.

Other solutions impose the network topology, such as the master-slave for Bluetooth and so can't solve the range solution on their own except by adding more gateways to increase coverage.

Both Zigbee and 6LoWPAN offer a standardized solution for building wireless sensor networks, mainly because of the 802.15.4 standard that defines exactly how access to the radio spectrum is done by the radios.

Zigbee protocol is a Zigbee Alliance closed specification that specifies how Zigbee based devices behave and they construct the sensor network. This standard must be used exactly to build a Zigbee based network so that compatibility between different hardware providers can be assured. Specifically, Zigbee specification defines the following three roles:

Zigbee coordinator → Creates the network and allows routers and end-devices to join the created network. Since it is a core function, it can't sleep for saving power and so must be permanently powered on (Mains power).

Zigbee routers → Joins a Zigbee network previously created by the coordinator and receives, transmits and assists in routing data. It allows other routers and end-devices to join the Zigbee network. As the coordinator it cannot sleep and so it must have some form of permanent power (Mains power).

End-Device → Must join a Zigbee network before being able to transmit or receive data through its parent (Coordinator or router). Cannot assist on routing data but it can sleep to conserve power.

Zigbee protocol stack depends on 802.15.4 based radios and are available in devices that already have the protocol stack embedded or offloaded to a controller micro controller. In the latter case, the protocol stack is not freely available to be embedded or modified but an open source implementation exists. Zigbee nodes aren't directly accessible by other non-Zigbee devices, since a Zigbee bridge/gateway is necessary to bridge the non-IP (Internet Protocol) Zigbee network with the standard IP network used by computers and the Internet.

An alternative to the ZigBee based wireless sensor networks protocol is the open standard 6LoWPAN (IPv6 Low power Wireless Personal Network), an IETF standard also based on the 802.15.4 standard, that allows the establishment of wireless sensor networks but by using a compressed version of the

IPv6 protocol standard. Since it uses the IPv6 protocol, bridging the 6LoWPAN networks with the standard wireless/cabled IP based computer network is much simpler and avoids the existence of a more complex Zigbee to IP bridge device. Another advantage to the 6LoWPAN based networks is that nodes behave as standard network device, and so can be accessed and controlled from anywhere, namely controlled directly from the AFarCloud Servers.

The 6LoWPAN network topology is simpler than the Zigbee:

Gateway → Relays and routes end-node data from the IPv6 based 6LoWPAN network to the backbone cable/wireless IPv4/IPv6 network, allowing it to reach the back-end services such the cloud server.

End-Node → Devices that gather sensing data and transmits it to neighbouring nodes and/or the gateway. End-nodes can create between them a mesh network that allows information to be routed across the sensor network until it reaches the gateway and avoiding network areas that are out of reach, route to nowhere or have failed.

While CoTS (Commercial off The Shelf) based hardware that implements the 802.15.4 for 6LoWPAN and Zigbee standard are readily available, alternatives for accessing the radio spectrum also exist so that only access to the radio interface is provided and have no associated standard (An example is TI FSK sub GHz CC1101 transceiver). Such radios are useful if the Zigbee or 6LoWPAN based WSNs are too complex or expensive to implement, but on the other hand requires that all components for the WSN be specifically implemented namely the mesh network management protocols.

When the area needed to be covered is higher than that is possible through a low range sensor network, a solution based on LPWAN or a solution based on data store and forward are more adequate than the short range based WSNs.

LPWAN networks have a high coverage range at the cost of data-rate and latency while requiring very little amounts of power. In fact, the transmission of data through radio can be characterized by the following characteristics: signal range, data throughput and energy conservation. Of the three characteristics, it is only possible to choose two of them at any time. As an example, if high data throughput and signal range is needed, it is not possible to have low power usage. If low power and signal range are needed, data throughput must be compromised.

Table 10 compares three of the LPWAN implementations.

Table 10: Comparison of LPWAN implementations

	LoRa	Sigfox	NB-IOT
Data-rate	1-50 Kbps dep on range	100bps	250 kbit/s (multi-tone) 20 kbit/s (single-tone)
Range	15 Km (Rural) at 1kbit/s 5 Km (Urban)	30Km (Rural) 10Km (Urban)	15 Km
Battery Lifetime	10 years	10 years	10 years
Standard	LoRa Alliance	Proprietary	3GPP Rel 13
RF Band	ISM	ISM	Licensed
Optimized for	Low power Range	Low power Range Payload ≥ 12 bytes	Low power Bandwidth
Network	Star	Star	Star
Applications	IoT Smart Devices Smart Agriculture	IoT Smart Devices	IoT Smart Devices
Duty Cycle	1% / Hour (36s)	1% / Hour (36s), max 140 messages per day, and/or 6 mess/h	100%
Bidirectional	Yes	Yes, limited to 12 msg/day @ 8 bytes	Yes
Node Hardware	Available (5€-10€)	Available (5€-10€)	Available (10€-15€)
Gateway Hardware	Available (100€-1000€)	Not Available	Not Available
Depends on	Free implementation	Sigfox Network Operator	Mobile Network Operator

Competing technologies to the ones described do exist e.g., Weightless and DASH7, however, choice was made between the technologies that have readily available sensor node hardware and connectivity, so that an AFarCloud wireless sensor network can be deployed without being restrained by the lack of wireless radio hardware. Note that 5G NB-IOT, from 3GPP Rel 16 is not considered, due to low general availability hardware and operators.

Of the three compared technologies, LoRa is the only one that allows the building of a complete wireless sensor network without depending on an external network operator. While with LoRa there is no dependence on the network operator service and coverage, it also means that the network may need to be built from the ground up by providing the LoRa gateways that allow to cover the required areas and provide the required service level. With the Sigfox and NB-IoT, coverage and service level are dependent on the network operator and their back-end services but is not another additional component to the network.

Multiple LoRa gateways can be deployed to allow a greater coverage range and availability, but also need to relay the received data to the AFarCloud backend services. LoRa gateways that relay the collected data either to the AFarCloud private servers located at the farm, or to the AFarCloud cloud servers, require a backhaul data connection, either by using mobile services (3G/4G) and so dependent on the availability of a mobile operator network with the associated costs, or cable (Ethernet) connectivity to either the local network or to an internet access point. In either case, such relaying may not be possible either due to the impossibility of extending an Ethernet cable or by lacking mobile network coverage or due to low mobile network signal.

Specifically, for solving this issue, the LoRa gateway can store and hold the data until a fly by UAV or terrestrial vehicle comes in range and sinks the data from the gateway, so it can be delivered to either a backhaul connected gateway or to the farm network. Since LoRa gateway hardware is available, as specialized gateway with such functions can be built, which is not possible with other LPWAN solutions.

Antenna placement and geometry is central to the performance and coverage of LoRa WAN networking. For areal platforms, this is especially interesting with the population of UAV in the AFarCloud project. The record of receiving a LoRa packet (albeit from an elevated balloon is 700+ km.[https://www.thethingsnetwork.org/article/ground-breaking-world-record-lorawan-packet-received-at-702-km-436-miles-distance]

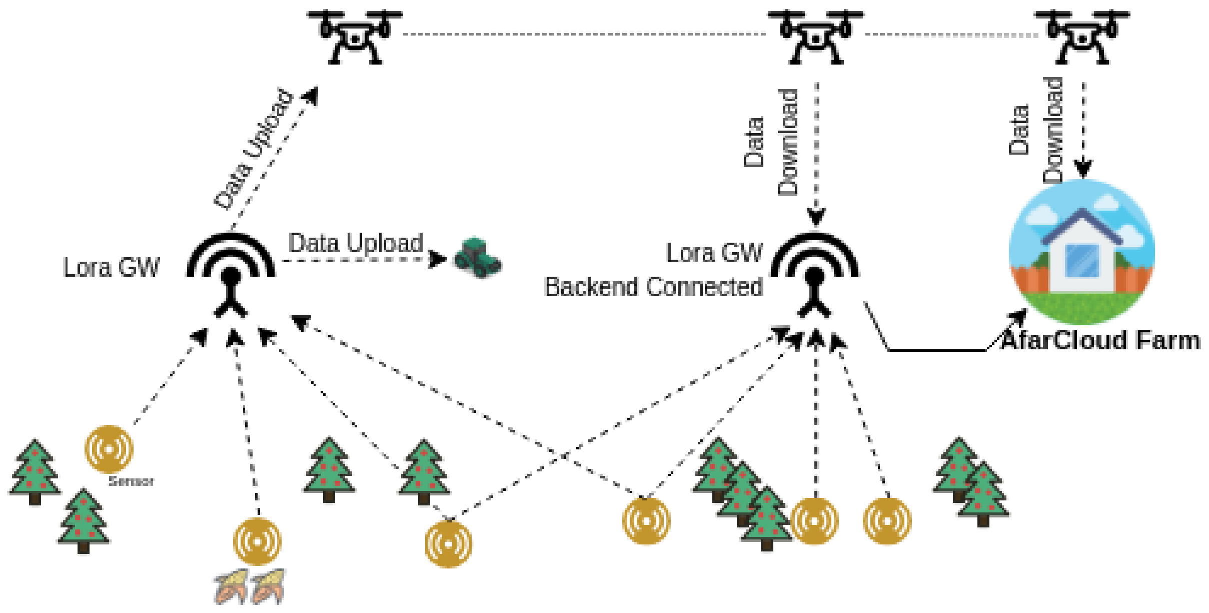

Figure 37: Data collection through UAVs for low range WSNs

A single gateway can be a single point of failure (SPOF) either if they fail (hardware issue, power issue, software issue) or because they lose connectivity to the backhaul.

As such, the requirement for resilient and high availability WSN is the necessity of availability of several LoRa gateways with overlapping coverage areas.

Store and forward solutions as depicted by the UAV fly in or the tractor being in range of a gateway can be applied to both LPWAN based networks and short-range networks. In both cases latency is very high (data takes a long time to reach the AFarCloud Servers), however allows the existence of stand-alone remote isolated networks which data can be collected. In either case, the gateway without the backhaul connection, either by design or due to backhaul connection failure, recognizes the presence of the gathering mobile gateway on the UAV or vehicle/tractor, packets the data and off loads the data. In a future UAV fly by or vehicle proximity, the mobile gateway signals that the previously packet data was delivered successfully, and it can now be deleted safely from the internal storage.

Data gathering and relaying

The establishment of a sensor network topology, either by deploying a closely grouped sensors with protocols such as Zigbee or 6LoWPAN, or a loosely spaced LPWAN network, establishes the supporting wireless protocols that allows the data to be transmitted from one place to another. Still, the sensed data must be collected, possibly stored temporarily and finally delivered to the backend services.

Data collection from end devices/nodes is half the equation since data can also flow from the AFarCloud servers to the end-nodes to act over equipment (water pump, water valve, light…) either based on operator command or through a decision from the DSS (Decision support system).

Short-range networks have symmetrical bidirectional communication, which means that the data rate and range are equal in both directions: receiving and sending data. On the other end, LPWAN based networks have some limitations since they were designed to be more efficient on one direction: For example, on Sigfox, data from the gateways to each node is limited to 12 messages per day. Without entering into too much details, LoRa range on the download link can be the same or lower than the upload link range.

Independently of the wireless protocol used to communicate, if the end device is always on, it is ready to receive commands at any time, but battery powered nodes must conserve energy and so are not active and available for data command reception at any time. The standard solution for enabling the reception of command data on a node that is saving power is to, when the node transmits data to the gateway, it keeps its radio on for some time after the transmission (receive window) to allow the gateway to relay any command(s) that are queued for the node.

The following selected higher-level protocols MQTT, MQTT-SN and CoAP all allow the data flow between end devices and central services but mainly decouple the data transmission process from the state and identification of the device. Decoupling the data flow from the device state by using the above protocols allows to hold data temporally for a device until it becomes available and is ready to consume/receive the data. There are also other protocols including AMQP (Advanced Message Queuing Protocol) and Data Distribution Service (DDS)

A typical use case is a client subscribing to a subject, and given the right credentials appends data into a time-series or relational database. In reality, this corresponds to a few lines of code and can be authored with a wide range of components and languages

These are some of the most common protocols for data aggregation and relaying:

MQTT → Message Queue Telemetry Transport runs over TCP/IP and works by providing higher-level protocol based on the topic publish-subscribe paradigm. A topic defines an item of information of interest, for example *Barn_Temperature*. Any sensor that has data related to this information of interest, publishes information on the topic independently of the availability and identification of the subscribers. Subscribers to the topic receive the message from the topic automatically when the data is available and depending on the quality of the service associated to the topic, data can be stored until all subscribers have received the message. This allows to send data/commands to devices that subscribed to certain control topics but are normally asleep without any loss of data. MQTT is data agnostic, and can send data of any encoding or encryption, typically any data in binary format.

MQTT has several message types where Publish, Subscribe and Unsubscribe are most central. A published message can also be retained on a channel, meaning that a subscriber will receive the most recent available message, when subscribing later than when message was actually published. There is another special message type called Last Will and Testament (LWT) to notify other clients about an ungracefully disconnected client. It allows publishing clients to send information like “I’m out of range”, or “My battery has run out” which is triggered after a reasonable timeout. This is convenient for maintenance on battery-powered devices.

There is no security built into MQTT as such, while both payload encryption as well as channel encryption in the TCP connection using SSL/TLS.

MQTT-SN → MQTT for Sensor Networks works the same way as standard MQTT but is streamlined to be more efficient for sensor networks that normally use more constrained devices. MQTT-SN uses UDP instead of TCP and has features that allow the efficient use of the underlying UDP protocol. It doesn’t run over the TCP/IP stack, but rather over serial e.g., using Bluetooth, Zigbee and needs a bridge to convert into proper MQTT messages.

CoAP → Constrained Application Protocol uses the Client/Server model over UDP, where the server is the end-node, and the client is the consumer (The AFarCloud Servers). CoAP allows the client to discover services from server (the device with the sensors) and monitor changes by a process called observing where the client requests to be notified of changes of an observed client resource.

While MQTT and MQTT-SN require the deployment of a MQTT Broker (server), it allows many-to-many communication and supports the temporary storage of data destined for end devices that sleep, CoAP on the other hand as a Server/Client protocol, only allows a one-to-one communication, and support for sleeping end devices depends on the CoAP client implementation.

A sample block diagram related to data gathering, relaying and storage of measures collected by a multi-hop WSN and integrated in a typical IoT architecture is depicted in the figure below:

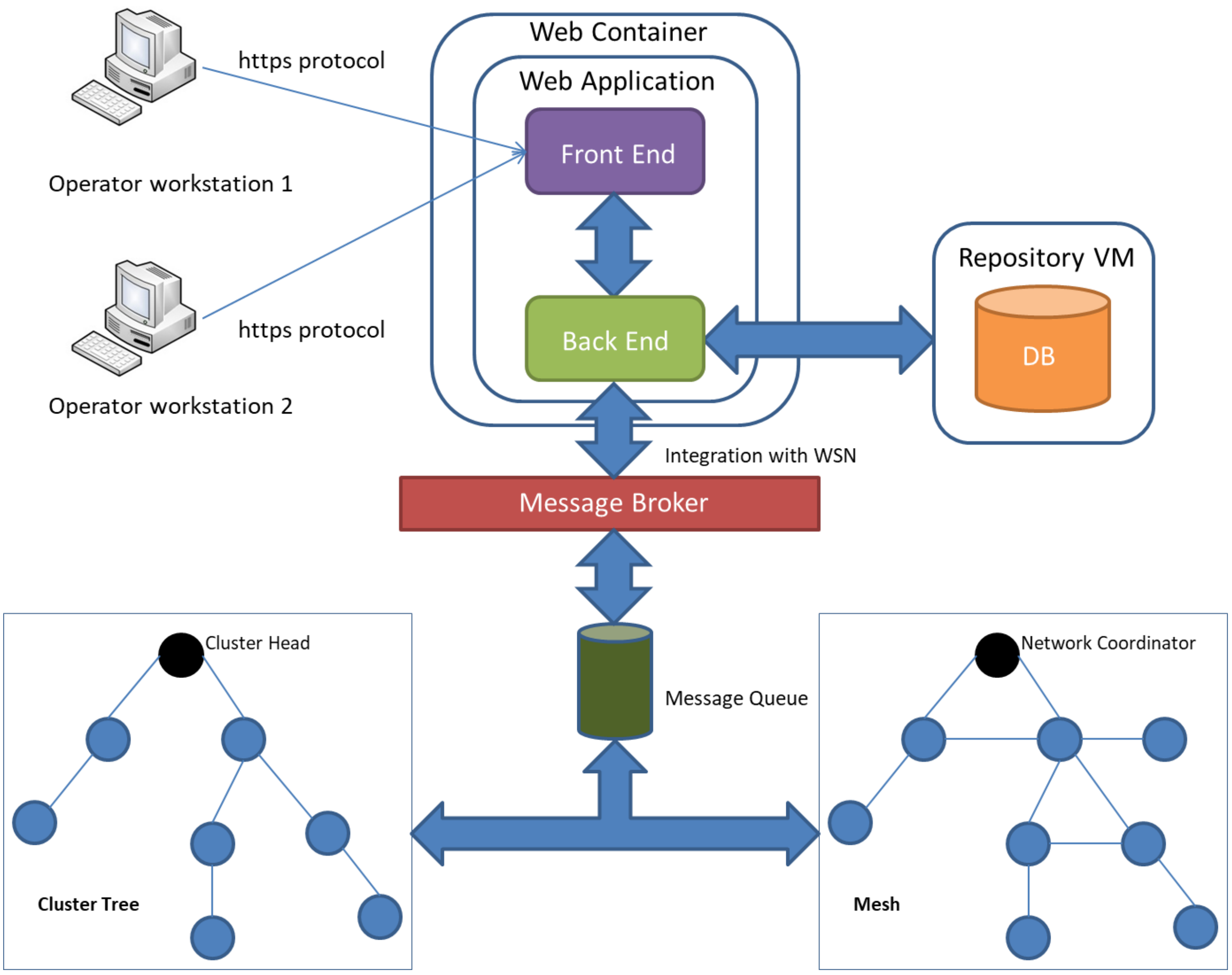

Figure 38: Collection of measures by a multi-hop WSN and integration in a typical IoT architecture

IoT devices, or so-called *motes* in the context of WSN, are equipped with sensors to sample environmental parameters like temperature, humidity, pressure, visible light, etc. Data collected by each device are then transmitted through the radio channel (e.g., Zigbee) to a central node called *sink*; it's the cluster-head in Cluster Tree topologies, or a particular device in mesh topologies designated as message receiver of the overall network. It's reasonable to provide sink motes with long-range connectivity to reach cloud infrastructure too, thus acting as a Gateway. Sensor data received by Gateway will be packed as MQTT messages and published on an MQTT queue read by an application.

Annex 2. Data Definitions

Data definitions of the models used in the REST services and by the MQTT clients (publishers and subscribers).

Definition	**Properties**	**Type**	**Format**
Location	latitude	number	double
	longitude	number	double
	altitude	number	double

Result	value	number	float
	uom	string	example: "http://qudt.org/vocab/unit#DegreeCelsius"
	variance	number	float

Observation	resourceId	string	example: "urn:afc:AS04:environmentalObservations:TST:airTemperatureSensor0012"
	observedProperty	string	example: "http://environment.data.gov.au/def/property/air_temperature"
	resultTime	integer	int64 (Epoch time in seconds)
	result	ref: "#/definitions/Result"	

Measurement	sequenceNumber	integer	int64
	location	ref: "#/definitions/Location"	
	observation	ref: "#/definitions/Observation"	
Example:			

```
{
  "sequenceNumber": 123,
  "location": {
    "latitude": 45.45123,
    "longitude": 25.25456,
    "altitude": 2.10789
  },
  "observation": {
    "resourceId":
"urn:afc:AS04:environmentalObservations:TST:airTemperatureSensor0012",
    "observedProperty":
"http://environment.data.gov.au/def/property/air_temperature",
    "resultTime": 1558086914,
    "result": {
      "value": 3.2,
      "uom": "http://qudt.org/vocab/unit#DegreeCelsius",
      "variance": 0
    }
  }
}
```

Acceleration	accX	number	float
	accY	number	float
	accZ	number	float

Anomalies	locationAnomaly	boolean	
	temperatureAnomaly	boolean	
	distanceAnomaly	boolean	
	activityAnomaly	boolean	
	positionAnomaly	boolean	

Collar	resourceId	string	example: "urn:afc:AS06:livestockObservations:SENSO:collar0034"
	location	ref: "#/definitions/Location"	
	resultTime	integer	int64 (Epoch time in seconds)
	resourceAlarm	boolean	
	anomalies	ref: "#/definitions/Anomalies"	
	acceleration	ref: "#/definitions/Acceleration"	
	temperature	number	float

CollarData	sequenceNumber	integer	int64
	collar	ref: "#/definitions/Collar"	

Example:

```
{
  "sequenceNumber": 123,
  "collar": {
    "resourceId": "urn:afc:AS06:livestockObservations:SENSO:collar0034",
    "location": {
      "latitude": 45.45123,
      "longitude": 25.25456,
      "altitude": 2.10789
    },
    "resultTime": 1558086914,
    "resourceAlarm": true,
    "anomalies": {
      "locationAnomaly": true,
      "temperatureAnomaly": true,
      "distanceAnomaly": true,
      "activityAnomaly": true,
      "positionAnomaly": true
    },
    "acceleration": {
      "accX": 0,
      "accY": 0,
      "accZ": 0
    },
    "temperature": 36.5
  }
}
```

CollarDataList	sequenceNumber	integer	int64
	collars	array	items: ref: "#/definitions/Collar"

Example:

```
{
  "sequenceNumber": 123,
  "collars": [
    {
      "resourceId": "urn:afc:AS06:livestockObservations:SENSO:collar0034",
      "location": {
        "latitude": 45.45123,
        "longitude": 25.25456,
        "altitude": 2.10789
      },
      "resultTime": 1558086914,
      "resourceAlarm": true,
      "anomalies": {
        "locationAnomaly": true,
        "temperatureAnomaly": true,
        "distanceAnomaly": true,
        "activityAnomaly": true,
        "positionAnomaly": true
      },
      "acceleration": {
        "accX": 0,
        "accY": 0,
        "accZ": 0
      },
      "temperature": 36.5
    }
  ]
}
```

SensorDataList	resourceId	String	example: "urn:afc:AS04:droneMissions:BEV:drone001"
	sequenceNumber	Integer	int64
	location	ref: "#/definitions/Location"	
	observations	Array	items: ref: "#/definitions/Observation"

Example:

```
{
  "resourceId": "urn:afc:AS04:droneMissions:BEV:drone001",
  "sequenceNumber": 123,
  "location": {
    "latitude": 45.45123,
    "longitude": 25.25456,
    "altitude": 2.10789
  },
  "observations": [
    {
      "resourceId":
"urn:afc:AS04:environmentalObservations:TST:airTemperatureSensor0012",
      "observedProperty":
"http://environment.data.gov.au/def/property/air_temperature",
      "resultTime": 1558086914,
      "result": {
        "value": 3.2,
        "uom": "http://qudt.org/vocab/unit#DegreeCelsius",
        "variance": 0
      }
    }
  ]
}
```

SamplingPeriod	resourceId	String	example: “urn:afc:AS04:environmentalObservations:TST:airTemperatureSensor0012”
	sequenceNumber	integer	int64
	samplingRate	integer	int32 (in minutes?)

Example:

```
{
  "resourceId":
"urn:afc:AS04:environmentalObservations:TST:airTemperatureSensor0012",
  "sequenceNumber": 123,
  "samplingRate": 20
}
```

Definition	**Properties**	**Type**	**Format**
LocDimRegion	latitude	number	double
	longitude	number	double
	width	number	double
	length	number	double

Region	resourceId	string	enum: - Weed - DeadPlant - WaterStress
	resultTime	integer	int64 (Epoch time in seconds)
	locationDimension	ref: "#/definitions/LocDimRegion"	

RegionData	sequenceNumber	integer	int64
	Region	ref: "#/definitions/ Region "	

Example:

```
{
  "sequenceNumber": 123,
  "region": {
    "resourceId": "Weed",
    "resultTime": 1558086914,
    "locationDimension": {
      "latitude": 37.392529,
      "longitude": -5.994072,
      "width": 2.38,
      "length": 1.25
    }
  }
}
```

RegionList	sequenceNumber	integer	int64
	regions	array	items: ref: "#/definitions/ Region"

Example:

```
{
  "sequenceNumber": 123,
  "regions": [
    {
      "resourceId": "Weed",
      "resultTime": 1558086914,
      "locationDimension": {
         "latitude": 37.392529,
         "longitude": -5.994072,
         "width": 2.38,
         "length": 1.25
    },
    {
      "resourceId": "DeadPlant",
      "resultTime": 1558086920,
      "locationDimension": {
         "latitude": 37.393529,
         "longitude": -5.995072,
         "width": 1.38,
         "length": 1.42
    }
  ]
}
```

Annex 3. Scenario Functionalities (D7.1)

ID	FUNCTIONALITY DESCRIPTION
F1	Monitor environment: temperature (ambient, and the plant), wind, and weather forecast.
F2	DSS for deciding about if it will be frost or not.
F3	Detection of cereals nutrients composition (energy, protein and humidity analysis)
F4	Using DSS take decision regarding when and where to harvest
F5	Monitor NKP (sensors or imagery)
F6	Measure the needs of fertilization with high spatial precision
F7	DDS for decision about when to fertilize
F8	Outdoor livestock location tracking
F9	Detection of livestock heat
F10	Detection of livestock calving
F11	Detection of livestock rumination and eating
F12	Determination of livestock growth rate
F13	Inference of the livestock habits patterns for health and reproduction
F14	Measure field water content/vigour
F15	Measure water stress
F16	DSS for decision about how much water
F17	Automatic actuation on rooftop (open, close)
F18	monitor greenhouse temperature and humidity
F19	Using actuators, irrigate with correct amount and location
F20	Detect plant illness (imaginary near infrared)
F21	Monitor Gases
F22	NIR silage analysis
F23	Livestock indoor positioning
F24	Livestock identification

F25	Nutrition monitoring through rumen scanning
F26	Extract and analyze data from milky robots
F27	Livestock digestion monitoring
F28	Fleet management: tracking of farm vehicles

Annex 4. User Requirements (D2.1)

ID	REQUIREMENT DESCRIPTION
UR1	Domain specific decision support systems (DSS) are desired by the end-users. Everything from a specific process to a larger process such as dairy supply chains.
UR2	AFarCloud solutions should be compatible with ISOBUS tractors, and other equipment in a farm. Many farms have already well-functioning equipment, which cannot be omitted.
UR3	The system should be secure for workers driving or using the machinery.
UR4	The system should offer user-friendly solutions e.g., specialized HMI. Remember also that the environment is specific (hazardous, dusty, etc.).
UR5	The system should offer vehicle information (e.g., maintenance parameters, distance driven, operational hours, etc.)
UR6	The system should allow certain degree of automation in daily inspection tasks in order to reduce time and costs.
UR7	AFarCloud should be interoperable with the current systems in the farm.
UR8	Communication is important and sometimes a challenge in rural locations. Thus, different communication solutions, which provide a redundant solution is important.
UR9	The system should provide support to process NDVI as an agricultural index.
UR10	The system should be able to visualize information related to crops and livestock that allow farmers to diagnose current situation in the farm, predict future diseases or problems and make decisions.
UR11	The Hyperspectral image system should have highly accuracy. The type and quality of data must be discussed with the end-users before deployment (important measures are: grass height, illumination conditions, spectral data, etc.)
UR12	Weather, and other environmental data are important for the DSS.
UR13	Offer Environment footprint calculation (EFC), a solution that estimates environmental impact of the production for a single product.
UR14	Farm size distribution, production farm types of each class and common practices in different classes are required to improve current, and develop new services.
UR15	The system should provide information for Phenological status, disease/pest diagnosis of the crops, taking care to an extent of each crop specific needs.

UR16	The system should detect weeds in cereals and grass.
UR17	The system should help to know the precise moments of harvesting of both maize and grass
UR18	The system could acquire real-time information about crops including gravimetry, NPK, humidity, temperature and control of pesticides, temperature, load and cycle detection, use of water, illumination conditions.
UR19	The system should help monitoring animal health and activity.
UR20	The system should allow in heat detection of animals.
UR21	The system should allow the measurement of ruminal conditions of dairy cows by non-invasive methods. Also, the geometry of paralumbar fossa area for determining rumen fullness.
UR22	The system should be able to retrieve measured data from the rumen (pH, volatile fatty acids, ammonia) and to compare them with other type of data (milk production, milk quality, time of feeding and rumination).
UR23	The system should allow knowing the reproduction rates of cows.
UR24	The system should allow locating animals at any time.
UR25	The system should allow prediction the calving dates of animals. A DSS may be needed in this case.
UR26	The system must be able to detect animals that may pose a threat during harvest (deer, rabbits) or farm animals (wild boar). The former group can destroy the equipment, contaminate the crops, strass the livestock, etc. The latter group can be a great danger to the livestock, since attacking the livestock is part of their behaviours.
UR27	The system should be able to retrieve measured data from the rumen (pH, volatile fatty acids, ammonia) and to compare them with other type of data (milk production, milk quality, time of feeding and rumination).
UR28	The system should be able to identify livestock individually, as well as provide information about parameters such as position/tracking and location or battery lifetime for the tracking functionalities.
UR29	The system must provide real-time nutrient analysis for the help of ration mixing; at least dry matter and protein content are needed; other parameters give additional value.
UR30	The system should provide support for radiation frost detection and leaf temperatures.
UR31	The system must acquire real-time information about the grapes, mainly soil humidity, vigour and water stress to allow watering optimization and water flow information.
UR32	The system should be able to obtain information from leaves so health information can be inferred, and a classification can be established.