D2.3 Architecture Requirements and Definition (v2)

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	2.0
Planned Delivery Date	M18 (February 2020)
Actual Delivery Date	M18 (February 2020)
Deliverable Name	D2.3 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 second version of the functional and non-functional architectural requirements of the AFarCloud platform and the design of the architecture.

Document History

Version	***Date***	***Contributing partner***	***Contribution***
1.1	10th December 2019	TECN, ROTECH	Document creation and ToC; Requirements summary (TECN) Blockchain technology (ROTECH)
1.2	20th December 2019	TECN, ESTE, MTECH, AVL-CD, ROVI	General architecture updated information, device manager and alarm processing & reporter (TECN); NCDX (MTECH); ISOBUS and tractor related information (ESTE, AVL-

			CD); IPP and water stress, weeds & dead plants detection (ROVI).
1.3	10th January 2020	TECN, AVL-CD, MDH, TST, TTC, UNIPR	Alarm Processing & Reporter, DDS Manager, Data flow diagrams (TECN), Secure Gateway related information (AVL-CD), Mobile MMT and ISOBUS systems (MDH), general update on available sensors and expected sensors functionalities (TST), IoT Gateway (TTC), Multiprotocol Gateway (UNIPR)
1.4	24th January 2020	ROTECH, AIT, PDMFC, ESTE, CNR, UPM, TECN, DAC	Blockchain requirements (ROTECH), Cyber-Security in AFarCloud (AIT), Multiprotocol Gateway (PDMFC), ISOBUS Converter and ISOBUS Gateway (ESTE, CNR), new architecture requirements based on D2.8 (UPM), Cloud Resources Monitoring and data flow diagrams (TECN), Real-time streaming of data and Stream Processing Engine (DAC, TECN)
1.5	4th February 2020	TECN, ROVI, SM, MTECH, HIB, AMS, AVL, UPM, INTRA	DDS topics update and data definitions (TECN), update of IPP diagram (ROVI), Image Catalogue and interfaces (SM), update of Nordic CDX (MTECH), update of Data Access Manager and DataQuery interfaces (HIB, TEC, UPM), firmware updates (AMS, AVL/AT), Data Access Manager API update (UPM), Mission Manager & Mission Processing and Reporter (UPM), System Configuration update
1.6	17th February 2020	AIT, TECN, DAC, PDMFC, QRT, ITAV, BEV	Cyber-Security in AFarCloud (AIT, TECN), requirements update (DAC), update in IPP (PDMFC), Image Processing Platform for Farm animal detection (QRT), section 1 (TECN), updates on DPP, DF, KE and ER (ITAV, BEV)
1.7	25th February 2020	SINTEF, TTC, TECN	Internal reviews and updates according to feedback. Final version.

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
ROTECH	Nadia Caterina Zullo Lasala	nadia.zullo@rotechnology.it
ROTECH	Diego Grimani	diego.grimani@rotechnology.it
ROTECH	Niccolo Cometto	niccolo.cometto@rotechnology.it
ROTECH	Lorenzo Bortoloni	lorenzo.bortoloni@rotechnology.it
ESTE	Giorgio Malaguti	malaguti@estetechnology.com
ESTE	Carlo Ferraresi	ferraresi@estetechnology.com
MTECH	Johanna Häggman	johanna.haggman@mtech.fi
AVL-CD	Daniel Puckmayr	daniel.puckmayr@avl.com
AVL-CD	Christian Hirsch	christian.hirsch@tuwien.ac.at
ROVI	Antonio José Siles	antonio.siles@rovimatica.com
ROVI	Patricio Alemany	patricio.alemany@rovimatica.com
ROVI	Fernando Palacios	fernando.palacios@rovimatica.com
DAC	Rafał Tkaczyk	rafal.tkaczyk@dac.digital
DAC	Krzysztof Radecki	krzysztof.radecki@dac.digital
MDH	Afshin Ameri	afshin.ameri@mdh.se
MDH	Baran Cürüklü	baran.curuklu@mdh.se
TST	Arturo Medela	amedela@tst-sistemas.es
TST	Pablo Pelayo	ppelayo@tst-sistemas.es
TTC	Martijn Rooker	martijn.rooker@tttech.com
UNIPR	Gaia Codeluppi	gaia.codeluppi@unipr.it
AIT	Erwin Kristen	erwin.kristen@ait.ac.at
AIT	Reinhard Kloibhofer	reinhard.kloibhofer@ait.ac.at
PDMFC	Álvaro Batista dos Ramos	alvaro.ramos@pdmfc.com

Partner name	Partner member	e-Mail
PDMFC	Francisco César Damião	francisco.damiao@pdmfc.com
UPM	Vicente Hernandez	vicente.hernandez@upm.es
UPM	Néstor Lucas Martínez	nestor.lucas@upm.es
UPM	Sara Lana Serran	sara.lana@upm.es
UPM	Mario San Emeterio	mario,sanemeterio@upm.es
UPM	Daniel Vilela García	daniel.videla@upm.es
UPM	José Fernán Martínez Ortega	jf.martinez@upm.es
MTECH	Johanna Häggman	Johanna.Haggman@mtech.fi
HIB	Tamara Martin	tmartin@hi-iberia.es
AMS	Johannes Loinig	Johannes.loinig@ams.com
SM	Daniel Åkerman	da@spacemetric.com
CNR	Massimiliano Ruggeri	ruggeri@estetechnology.com
BEV	Ricardo Sacoto Martins	ricardo.martins@beyond-vision.pt
BEV	João Matos Carvalho	joao.m.carvalho@beyond-vision.pt
ITAV	Joaquim Bastos	jbastos@av.it.pt
ITAV	Paul Marcel Shepherd	paul@av.it.pt
INTRA	Theofanis Orphankokudakis	theofanis.orphanoudakis@intrasoft-ilntl.com
INTRA	Dimitrios Skias	dimitrios.skias@intrasoft-ilntl.com

Internal Reviewers

Partner name	Partner member	e-Mail
SINTEF	Mariann Merz	Mariann.Merz@sintef.no
TTC	Martijn Rooker	martijn.rooker@tttech.com
DAC	Mateusz Bonecki	mateusz.bonecki@dac.digital
INTRA	Theofanis Orphankokudakis	theofanis.orphanoudakis@intrasoft-ilntl.com

Table of Contents

Table of Figures

Tables

Definitions and Acronyms

Acronym	*Definition*	*Remark*
AFC	AFarCloud	
AJA	Autonomous Job Analysis	
API	Application Programming Interface	
CAN	Controller Area Network	
CRUD	Create, Read, Update and Delete	
CWSI	Crop Water Stress Index	
DoW	Description of Work	
DB	DataBase	
DDBB	DataBases	
DDS	Data Distribution Service	
DLT	Distributed Ledger Technology	
DSS	Decision Support System	
FMS	Farm Management System	
FOV	Field Of View	
FPCM	Fat-Protein Corrected Milk	
FR	Foundational Requirements	
GDPR	General Data Protection Regulation	
GIS	Geographical Information System	
GPS	Global Positioning System	
GUI	Graphical User Interface	
GV	Ground Vehicle	
HMI	Human Machine Interface	
ID	IDentifier	
IDL	Interface Description Language	
IMU	Inertial Measurement Unit	
IoT	Internet of Things	

Acronym	***Definition***	***Remark***
IPP	Image Processing Platform	
JSON	JavaScript Object Notation	
KE	Knowledge Extractor	
MQTT	Message Queuing Telemetry Transport	
MQTT-SN	Message Queuing Telemetry Transport for Sensor Networks	
MMT	Mission Management Tool	
MW	Middleware	
M2M	Machine to Machine	
NDVI	Normalized Difference Vegetation Index	
PEF	Product Environmental Footprint	
PKI	Public Key Infrastructure	
QoS	Quality of Service	
REST	REpresentational State Transfer	
SIFT	Scale Invariant Feature Transform	
SfM	Structure for Motion	
SL	Security Level	
SL-C	Security Level Capability	
SLO	Service-Level Objective	
SL-T	Security Level Target	
SPARQL	SPARQL Protocol and RDF Query Language	
SQL	Structured Query Language	
SSL	Secure Sockets Layer	
SuC	System Under Consideration	
UAV	Unmanned Aerial Vehicle	Also mentioned as vehicle(s) dependent on the context
UDP	User Datagram Protocol	

Acronym	***Definition***	***Remark***
(U)GV	(Unmanned) Ground Vehicle	Also mentioned as vehicle(s) dependent on the context
USB	Universal Serial Bus	
UTF	Unicode Transformation Format	
WMS	Web Map Service	
WSN	Wireless Sensor Network	
XML	EXtensible Mark-up Language	

1. Introduction

This document contains the second version of the functional and non-functional architectural requirements of the AFarCloud platform, as well as the design of the architecture. Compared to the first version of this deliverable, this document contains updated functional and non-functional requirements, based on the first year experiences in the 11 demonstrators. In addition, all the components of the architecture have been defined in more detail, a solution based on Blockchain technology has been included for traceability and a cyber-security assessment has been carried out.

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. The second version of the architecture has also taken into account the results from the tests carried out in the 11 scenarios during Y1. Based on this feedback, the functionalities of some components have been enhanced or updated and new components have been included such as the blockchain API for traceability.

The AFarCloud platform consists of three main functional components: (i) the Farm Management System, (ii) the Semantic Middleware and (iii) the Hardware Layer. Besides, the AFarCloud platform interconnects with other data sources like third-party data and legacy systems databases.

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

The Semantic Middleware offers, among others, components for: data storage and retrieval from the Cloud; managing and cataloguing images; registration of assets in the farm (IoT devices, animals and vehicles); 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 Hardware Layer involves the functionalities related to unmanned aerial/ground vehicles, ISOBUS systems, actuators, sensors and other IoT devices.

1.1. Structure of the document

The document is organised as follows. Chapter 2 provides the updates on the list of architectural requirements defined in D2.2 (see Annex 1). Some new requirements have been identified and others have been adjusted based on the tests carried out in the 11 scenarios.

Chapter 3 explains the approach followed in the project for sharing of resources among farms; presents the three main functional components of the platform (i.e. *Farm Management System*, *Semantic Middleware* and *Deployed Hardware*) with their expected functionalities; lists the data protocols supported for information transmission; and describes the functionality for real-time streaming of data.

Chapter 4 details the benefits that *Blockchain* technology can offer to the agri-business and how this technology will be used in AFarCloud for traceability in scenarios AS05 and AS11.

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 the AFarCloud instance and of the 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, asset registry (formerly known as device registry), stream processing engine, device and mission managers, mission and alarm processing & reporters, environment reporter, data pre-processor, data fusion and knowledge extraction, image processing and catalogue, ISOBUS converter and DDS manager.

Chapter 7 explains the devices and vehicles in the *hardware layer* that interact with the AFarCloud platform, their expected functionalities in the different scenarios and the interfaces to control or exchange data with them. These devices and vehicles are: sensors, actuators, unmanned aerial and ground vehicles, tractors and ISOBUS systems.

Chapter 8 contains data flow diagrams depicting a) how missions are sent to UAVs, UGVs and ISOBUS systems, 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 or send a command to a MQTT device, d) how to register in AFarCloud repositories the static information of the assets in a farm, and e) how to manage food traceability using Blockchain technology.

Finally, Chapter 9 describes the cyber-security risk assessment done for AFarCloud architecture according to the security standard IEC 62443 and the cyber-security requirements and recommendations obtained from this assessment.

1.2. Updates and additions with respect to version 1

All of the components of the AFarCloud architecture have been reviewed and updated taking into account the results from the tests carried out in the 11 scenarios during Y1.

The main updates are listed below:

- Regarding the AFarCloud platform requirements, 8 new requirements have been added (section 2.1) based on the results from the scenarios and to support real-time analytics, 3 requirements have been updated (section 2.2) and 9 requirements have been included (section 2.3) to provide support for traceability based on Blockchain technology.
- Version 2 of the AFarCloud architecture provides a solution for real-time data processing, aggregation and analytics (sections 3.4 and 6.7) in addition to the batch processing capabilities.
- A new section has been included describing the benefits of Blockchain technology for agri-food and its application in AFarCloud scenarios (section 4).
- Updates in the Farm Management System (section 5), in its software interfaces (section 5.1.2) and in the System Configuration (section 5.3)
- InfluxDB will be used as NoSQL repository instead of MongoDB to facilitate integration with Grafana to visualize historical data
- Inclusion of the formula used by the Cloud Resources Monitoring to calculate availability (section 6.2.1)
- More detailed definition of the functionalities and interfaces of the middleware components.
    - Data Access Manager (section 6.4): update of the components diagram.
    - Data Query (section 6.5): update of the description, the components diagram and definition of the interfaces.
    - The Device Registry (section 6.6) has been renamed to Asset Registry as its scope has been extended to manage the registry not only of vehicles and IoT devices in the

farm but also of the farm itself (location, part fields, crops, etc) and of the customer or owner of the farm. It now accesses directly to the DAM instead of using the DataQuery. Besides, its interfaces have been included.

- The Streaming Engine (section 6.7) has been renamed to Stream Processing Engine (SPE) to better reflect its main functionality which is to provide real-time streaming data pipelines for reliable exchange of data between the AFarCloud Interfaces, third-party software and the end-users, i.e. the Farm Management System. Its scope has been focused on real-time data processing and analytics. The components diagram has been updated accordingly.
- The interfaces for the Device Manager (section 6.8.3) have been included
- The components diagram of the Mission Manager (section 6.9.2) and of the Mission Processing & Reporter (section 6.10.2) have been updated
- Alarm Processing & Reporter (section 6.11): update of the component description and inclusion of the components diagram and interfaces
- Environment Reporter (section 6.12): update of the description and of the components diagram
- Update of the descriptions of the DPP, DF and KE (sections 6.13, 6.14 and 6.15).
- The Image Data Manager has been renamed to Image Catalogue (section 6.16) and its description and interfaces have been provided in more detail
- Update of the description of the Image Processing Platform (section 6.17)
- Update of the description and components diagram of the ISOBUS Converter (section 6.18)
- Update of the DDS interface with UAVs/UGVs (section 6.19.3.2)

- 5 types of gateways have been described under section 7: Multiprotocol Gateway (MPGW), Cloud Multiprotocol Gateway (CMPGW), ISOBUS Gateway, Secure Gateway and IoT Gateway. Besides, the functionalities of the hardware components in the hardware layer have been updated.
- New data flow diagrams have been included in section 8
- A new section (section 9) has been included detailing the cyber-security assessment done and the recommendations for AFarCloud.

2. Scenarios Requirements Matrix

In deliverable D2.2, a list of architectural requirements for the AFarCloud platform was provided (see Annex 1. Requirements identified in D2.2) taking as input the list of demonstrator functionalities in D7.1, the list of user requirements defined in D2.1 and the DoW. This section contains changes to existing requirements and identification of new ones due to the inclusion of new functionalities in the platform (e.g. Blockchain technologies for traceability) or modifications based on the feedback from Y1 demos and the updated list of user requirements defined in D2.8.

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. Bold text formatting is used 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. The 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

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

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.
- BLC – Blockchain requirements. These are the requirements related to the implementation of the Blockchain technology for traceability.

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 D2.8 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.

2.1. New requirements from existing categories

Req no.	Req. Description	Source	Priority	Deadline	Responsible WP	Comment
HMI-18	The system **should** provide tools to support the end-user in the Product Environmental Footprint (PEF) monitoring process in the dairy supply chain, e.g. PEF in the dairy production or PEF in the transport of milk to dairies.	DoW	High	Y3	WP2, WP3, WP4, WP7	
INT-15	The system **should** provide support tools that allow the end-user to check how PEF is affected by changes in milk production (change in the production model, energy or water consumption, etc.) or changes in transport.	DoW	High	Y3	WP3	
INT-16	The system **should** support real-time analytics (based on Stream Processing Engine) and batch (static) data analytics (based on AFarCloud data repository) (i.e. the so-called “lambda architecture” for IoT data processing).	DoW	High	Y3	WP2, WP4, WP7	

Req no.	Req. Description	Source	Priority	Deadline	Responsible WP	Comment
SEN-12	The system **shall** create a WSN to track and monitor farm animals (cows).	DoW	High	Y3	WP2, WP5, WP7	
VEH-13	Solutions offered by AFarCloud for vehicles (UAVs, UGVs and tractors) **shall** be compliant with the regulation in the European countries where the demonstrations will take place.	UR35 (from D2.8)	High	Y3	WP2, WP3, WP6, WP7	
VEH-14	Workers managing vehicles **shall** always be able to recover manual control of autonomous vehicles.	UR36 (from D2.8)	High	Y3	WP2, WP3, WP7	
VEH-15	Vehicles intended to support collection of data from remote sensors **shall** carry the necessary equipment onboard.	DoW	High	Y2	WP6	The sensors referred to in this requirement are external to the vehicle
COM-24	The system **should** be able to send firmware updates from the cloud to a sensor through a gateway.	DoW	Low	Y3	WP6, WP2	COM-6 has been extended to cover software updates not only for tractors but also sensors.

2.2. Updated requirements from existing categories

Req no.	Req. Description	Source	Priority	Deadline	Responsible WP	Revision
GEN-2	Historical information **shall** be available in the system for further usage and as input to the DSS for additional knowledge extraction.	DoW, UR33 (from D2.8)	High	Y3	WP2	New UR33 states that historical data must be made available, not only concerning missions. Priority has been increased from medium to high.
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, rainfall, soil temperature, soil moisture, solar radiation, atmospheric pressure, wind speed and wind direction.	F15 (from D7.1), UR31 (from D2.8)	Medium	Y2	WP4, WP5	Leaf wetness is not considered as necessary to be monitored in vineyards due to the short periods when vineyards have the leaves.
SEN-8	The system **shall** monitor the environmental conditions in crops, the stable and the surroundings.	F1 (from D7.1), DoW, UR12 (from D2.8)	High	Y2	WP4, WP5	Environmental conditions need to be monitored also in crops.

2.3. Blockchain requirements

Req no.	Req. Description	Source	Priority	Deadline	Responsible WP
BLC-1	The system **shall** support the storage of a set of data using blockchain technology as blockchain provides cryptography, traceability and immutability of the data stored.	DoW, Portuguese farmers' input	High	M24	WP2
BLC-2	There **shall** be a server, reachable in cloud, that hosts a node of the blockchain.	DoW	High	M17	WP2
BLC-3	The node **shall** be part of a public blockchain based on Ethereum.	DoW	High	M19	WP2
BLC-4	A wallet containing Ether-type cryptocurrency **shall** be created in order to perform transactions.	DoW	High	M19	WP2
BLC-5	There **shall** be a database in the cloud server to store the data managed by blockchain technology.	DoW	High	M18	WP2
BLC-6	A contract on the blockchain **shall** be developed in order to collect data.	DoW	High	M19	WP2
BLC-7	There **shall** be APIs that expose the contract access methods outside the machine.	DoW	High	M20	WP2
BLC-8	There **shall** be APIs for retrieving the reference to the transaction within the database based on some specific parameters.	DoW	Medium	M20	WP2
BLC-9	All the Blockchain APIs **should** be accessible via ssl protocol.	DoW	Low	M21	WP2

3. General Architecture

3.1. Architecture approach for sharing of resources among farms

There are different alternatives when defining the type of data access for farms. Depending on the data sharing possibilities between farming entities, different types of architecture designs can be used:

a. Isolated repository: each scenario is isolated and there is neither sharing of information nor services among farms. Each scenario deploys its own AFarCloud instance, 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. Federated repository: in this case, some scenarios share information or services among them. Each scenario deploys its own AFarCloud instance, with its own database in an isolated and dedicated repository in the Cloud. Besides, some farms are federated, which implies some resource or data sharing. Replication services need to be implemented to add the information to be shared to all the databases that are federated.
c. Centralized repository: all the scenarios share the same AFarCloud database, which is deployed in the Cloud. Access to the information must be filtered by farm.

In AFarCloud, we propose a combination of the approaches "isolated repository" and "centralized repository": one AFarCloud instance per farm will be created, as it is illustrated in Figure 1. This means that each scenario deploys its own AFarCloud components, with its own data repositories, REST services and MQTT broker. The System Configuration of each AFarCloud instance needs to provide the means to configure:

- the needed parameters for each of the data repositories used per farm (SQL DB, NoSQL DB and ontology): e.g. DDBB endpoint, DDBB name, username and password;
- the URLs of the REST services;
- the MQTT broker endpoint, username and password;
- the farm name or identifier;
- the registration of the devices and vehicles in the hardware layer.

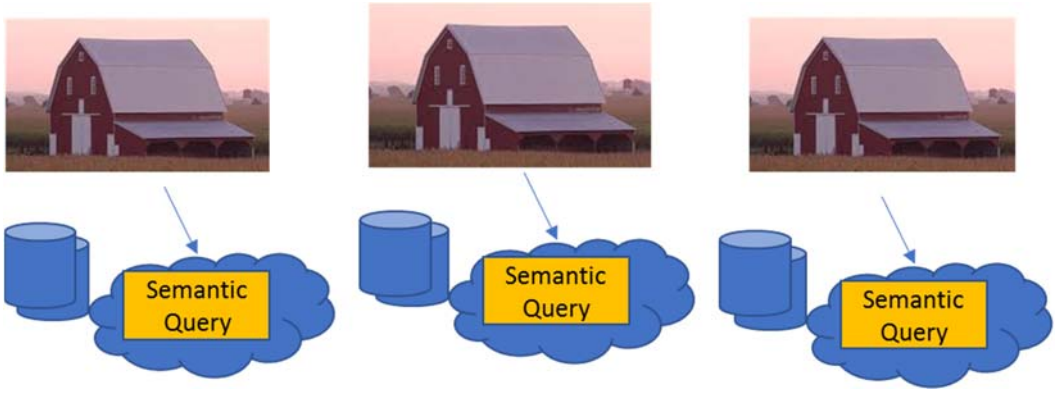

Figure 1: One AFarCloud instance per farm is created

This architecture requires a specific amount of processing power and storage capacity that is obtained with the help of cloud services. Bear in mind that although each farm has its own AFarCloud instance, all farms use a common cloud infrastructure.

In addition to this, AFarCloud offers the possibility of sharing data of common interest for all farms, such as patterns to detect diseases, etc. To allow this behaviour, a global AFarCloud repository where interested farms can store the particular data to be shared must be created, as described in Figure 2. To manage those data, specific global REST services must be created too.

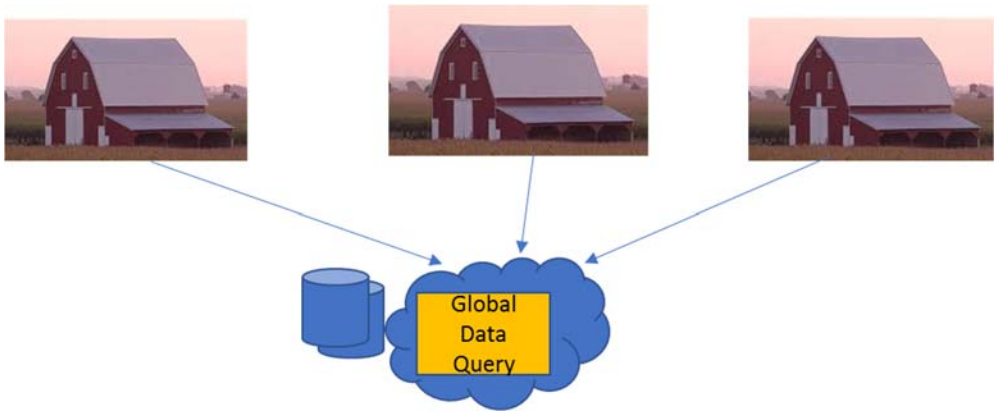

Figure 2: Global AFarCloud repository to allow data sharing

3.2. Functional and components architecture

The AFarCloud platform consists of three main functional components:

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

Besides the above-mentioned functional components, the AFarCloud platform interconnects with other data sources like third-party data and legacy systems' databases. **¡Error! No se encuentra el origen de la referencia.** depicts in more detail the functionalities and protocols covered in AFarCloud.

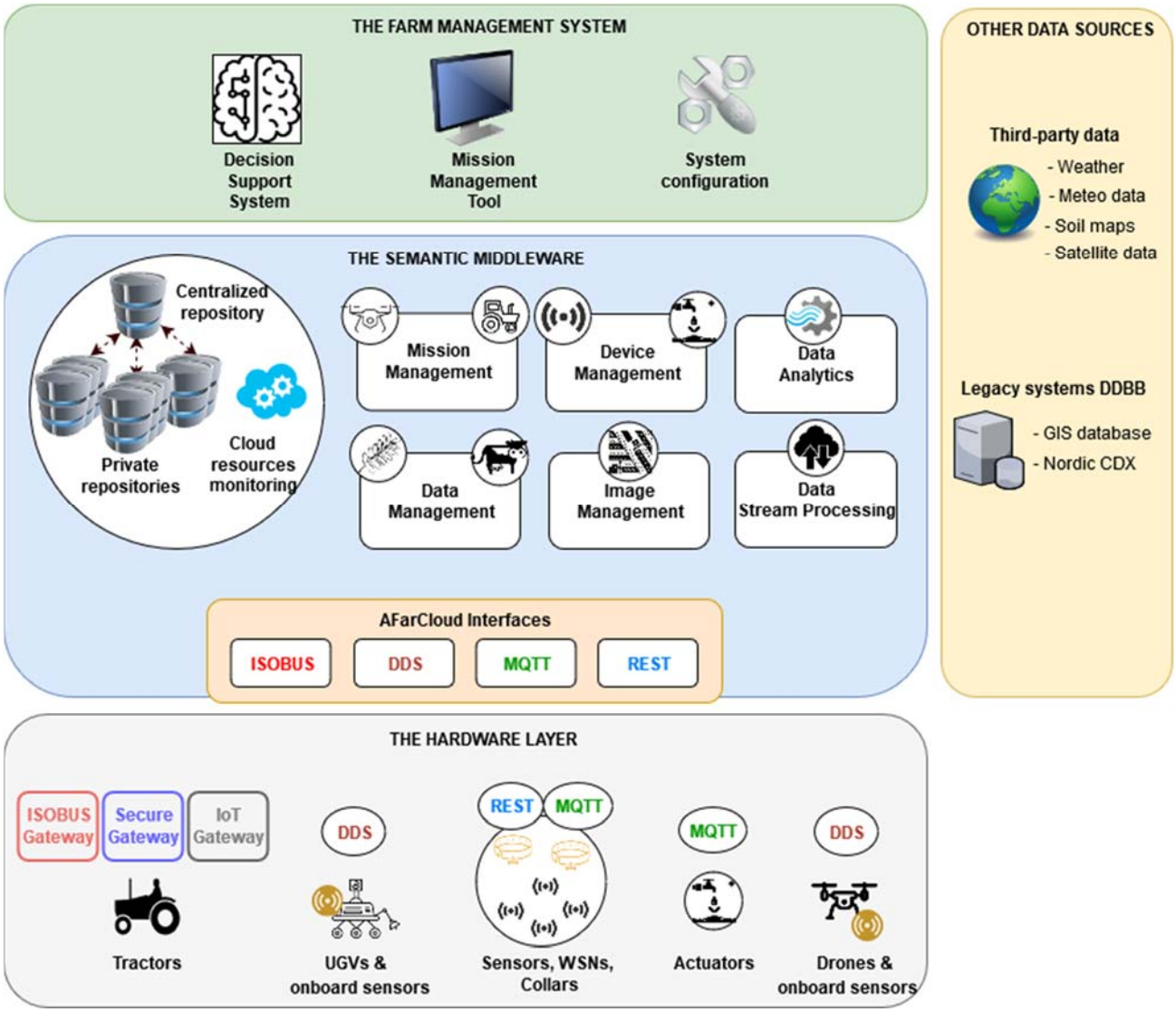

Figure 3: AFarCloud architecture

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

3.2.1. The Farm Management System

The Farm Management System offers:

- a Mission Management Tool (MMT) to define the conditions for cooperative missions involving Unmanned Aerial Vehicles (UAVs or drones) and Ground Vehicles (GVs) ranging from fully

autonomous Unmanned Ground Vehicles (UGVs) to ISOBUS systems. The various MMT solutions will allow the user to access both services in the MMT as well as those belonging to the Decision Support System and System Configuration modules. There will be a Main MMT for the operators that process data, plan a mission, monitor a mission, evaluate the results and work with the Decision Support System, and two specific Mobile MMT versions for monitoring vehicles' missions, the UAV-MMT, for the pilots of the UAVs, and the Tractor MMT, dedicated to in-vehicle usage.

- a Decision Support System (DSS) to make decisions pre-, during-, and post-mission;
- a System Configuration to configure both the AFarCloud instance of each farm and the above-mentioned systems, to provide for the registration of the Farm inventory available such as Ground Vehicles (UGVs and ISOBUS systems), UAVs and devices (sensors and actuators) and to perform pre-mission status control of the vehicles that are involved in a mission;
- applications for the user to manage and monitor the whole system.

More extended information about the Farm Management System can be found in section 5.

3.2.2. The Semantic Middleware

A middleware is a software layer used to hide the underlying complexity of hardware and services in distributed systems, so that application layers can access resources in a unified way. The AFarCloud middleware uses semantic models, specified by an ontology to abstract the heterogeneity of the underlying resources, and to ensure that all information is stored according to a common information model that guaranties interoperability. The semantic middleware acts as a communication centralizer, disseminating messages between the Farm Management System and the Hardware Layer. The Semantic Middleware is in charge of unifying, processing and analysing data coming from, or directed to, different types of cyber-physical systems deployed in the AFarCloud scenarios.

More extended information about the Semantic Middleware can be found in section 6.

3.2.3. The Hardware Layer

The Hardware Layer can be divided into the following categories:

1. Sensors:
    a. Standalone sensors;
    b. Collars: smart neck collars for monitoring cows;
    c. WSNs: groups of spatially dispersed and dedicated sensors for monitoring and recording the physical conditions of the environment.
2. Actuators;

3. Ground Vehicles: three different types of GVs are covered by the AFarCloud architecture:
    a. UGVs: GVs able to autonomously execute missions defined by the Farm Management System that consist of a list of commands. Missions involve the autonomous movement of UGVs around certain areas of the demonstrators.
    b. Legacy systems: GVs able to autonomously execute a list of agricultural tasks defined by the Farm Management System. It is not the goal of those missions to implement autonomous navigation (due to the lack of auto-steer abilities of tractors, e.g., go to a waypoint). Two different types of legacy GVs are considered in the AFarCloud architecture:
        i. ISOBUS systems: legacy GVs able to semi-autonomously execute agricultural tasks carried out by the ISOBUS compliant implements of ISOBUS tractors.
        ii. Tractors: legacy GVs able to execute/configure specific commands/parameters of the CAN bus (based on the J1939 standard) of the tractor.
4. UAVs: drones able to autonomously execute missions defined by the Farm Management System, that consist in a list of commands. Missions involve the autonomous movement of UAVs around certain areas of the demonstrators.

More extended information about the Semantic Middleware can be found in section 8.

3.2.4. Other Data Sources

3.2.4.1. Third-party data

Third-party data provides information for understanding the environment surrounding the ground/aerial vehicles and sensors. These data are accessed from the Farm Management System of AFarCloud. The information of interest for the project is:

- 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

3.2.4.2. Legacy systems databases

The AFarCloud platform has access to the data provided by two legacy systems databases:

1. Nordic Cattle Data eXchange (CDX):

Nordic CDX is a REST API that provides data of milking stations and robots taken from the national milk recording, insemination and breeding information systems of Finland, Sweden, Denmark and Norway.

Within the AFarCloud project, access has been allowed to the Finnish implementation of CDX. As the Finnish farm in AS09 does not have agreement with CDX (at least not currently), we cannot get the individual cow milking event data for this farm. However, the AFarCloud project will use basic information from milking robots and, optionally, also weight data in carbon footprint calculation (WP4). For example: feed use and consumption information.

2. Geographical Information System (GIS) database:

The GIS 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.

3.3. Data protocols for information transmission

Considering the heterogeneous nature of the elements deployed in the Hardware Layer, the Semantic Middleware offers different types of interfaces as described in the following sub-sections.

3.3.1. Data Distribution Service (DDS)

DDS is available for autonomous vehicles able to perform autonomous navigation (i.e., UAVs and UGVs). 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. For example, imagine an alarm is sent to a UAV to avoid a collision but at that moment, the UAV loses connectivity. DDS allows that as soon as the connectivity is established, the UAV receives 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 the Hardware Layer) related to the command and control of vehicles able to execute autonomous navigation, 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, allows the re-planning of ongoing missions if necessary. Thus, this DDS interface is used by all UAVs and UGVs able to carry out autonomous AFarCloud missions defined by the Farm Management System and to send the data gathered during those missions back to the middleware. All data exchanges are compliant with the AFarCloud data model.

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

3.3.2. ISOBUS

ISOBUS is available for legacy systems that are ISOBUS compatible (i.e., ISOBUS tractors & implements). As already mentioned, it is not the goal of these vehicles to implement autonomous navigation, but to autonomously execute agricultural tasks carried out by the ISOBUS compliant implements attached to tractors, instead. These special missions can also be defined by the Farm Management System of AFarCloud. The communication between ISOBUS systems and the Farm

Management System is standardized and simplified through the use of ISO-XML files. The ISO-XML format (described in the ISO 11783-Part 12 standard) is the only standardized way defined by ISOBUS for exchanging information between ISOBUS compatible entities. The semantic middleware offers an interface able to generate an ISO-XML file to define each of these special missions (e.g., the prescription map of a treatment). ISO-XML files are loaded to the ISOBUS system through the Tractor MMT. The ISOBUS interface also offers means to load and convert offline ISOBUS log data collected from the ISOCAN bus of ISOBUS systems once the mission has finished (e.g., ex-post telemetry of the treatment applied), to store it in the AFarCloud repositories.

3.3.3. MQTT

MQTT (Message Queuing Telemetry Transport) is available for collecting measurements from/implementing actions on IoT compatible devices (WSN, standalone sensors, actuators, etc.). MQTT is also used in AFarCloud by tractors and ISOBUS systems, to send real-time telemetry data during the execution of missions. MQTT is a publish/subscribe communication protocol standardized by ISO (ISO / IEC PRF 20922). It is light-weight, open, simple, and designed to be easy to implement. These characteristics make it ideal for its 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. Other features of MQTT are the following:

- Publisher is decoupled from subscriber: client connections are always handled by an MQTT broker. The MQTT broker uses the topic of a message to decide which client receives which message.
- The term “topic” refers to an UTF-8 string that the broker uses to filter messages for each connected client. A topic consists of one or more topic levels.
- Subscribers of a topic 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.
- Different Quality of Service (QoS) levels to guarantee the delivery messages.

When a client subscribes to a topic, it can subscribe to the exact topic of a published message or it can use wildcards to subscribe to multiple topics simultaneously. There are two different kinds of wildcards: single-level (+) and multi-level (#).

A single-level wildcard replaces one topic level (e.g., `afc/AS05/waterManagement/soilSensor/+/measure`). Any topic matches a topic with single-level wildcard if it contains an arbitrary string instead of the wildcard. On the other hand, the multi-level wildcard (#) covers many topic levels (e.g., `afc/AS05/waterManagement/#`). For the broker to determine which topics match, the multi-level wildcard (#) must be placed as the last character in the topic and preceded by a forward slash. When a client subscribes to a topic with a multi-level wildcard, it receives all messages of a topic that begins with the pattern before the wildcard character, no matter how long or deep the topic is.

3.3.4. REST

REST (Representational State Transfer) is available for collecting measurements from IoT compatible devices (WSN, standalone sensors, etc). REST is a client/server software architecture style used for creating web services. REST stands out due to its scalability, flexibility, portability, simplicity and low use of resources.

3.4. Handling of data streams in real-time

The Stream Processing Engine (see section 6.7 for more details on SPE architecture) is used to provide real-time processing capability for internal platform components to external applications, including decision support applications available through DSS.

SPE is an implementation of Apache Kafka distributed streaming platform which offers a publish-subscribe mechanism for streams similar to typical message queues. Firstly, Kafka suite facilitates implementation of data pipelines to transfer data from entry end-points (producers or publishers, using *Producer API*) to other systems or applications (consumers or subscribers, using *Consumer API*) in real-time and in a fault-tolerant manner. Secondly, Kafka supports the development of real-time processing applications which react to data streams flow (e.g. real-time responses to pre-defined events).

Thanks to SPE, AFarCloud developers can implement their own consumer applications for data analytics or preprocessing and deploy it on Kafka data stream pipeline. Depending on the requirements, the results of analytics or preprocessing can be ingested as additional topics (through *Streams API*) to be consumed by other clients. The communication between Kafka clients and servers is handled with TCP protocol.

Furthermore, since AFarCloud platform is intended to offer an architecture that covers functional requirements of various size farms, Kafka-based SPE could eventually enable high-throughput and low-latency streaming of big data volumes, if required, which is an additional advantage of the platform in terms of future development, scalability and exploitation potential.

4. Blockchain technology in AFarCloud

4.1. Blockchain for traceability

Distributed ledger technologies and, in particular Blockchain, represent the state of the art in terms of transaction validation mechanisms.

Thanks to its architecture, the blockchain provides cryptography, traceability and immutability of the data stored into one of its “blocks”. Once the data has been approved and entered into the blockchain, it is virtually impossible to tamper with it; every transaction (which can be custom designed according to the need of the traceability scenario) has a timestamp associated with it and needs to be confirmed by more than fifty percent of the nodes in the chain. All these reasons make blockchain a good choice in terms of data traceability.

4.2. How blockchain can help the agri-business

Food traceability has become a main concern nowadays; transparency and safety of the product is a guarantee of quality and trust for the consumer, and credibility for the producer. The benefits of data traceability and integrity can also have an important impact on farmers’ strategies. Tracking the animals’ diet in an inalterable manner, for instance, can constitute a useful aid in the decision making about the best treatment of farm animals in order to have the best possible outcome. The same goes for fruit and vegetables production, i.e. tracking data about environmental situation and products -if any- used on the plants is not only a useful information to build customers’ trust in the reliability of the producer, but also an immutable and trustful diary, for the producer, to keep track of the strategies and the changes in the production.

4.3. Application in AFarCloud scenarios AS05, AS11

Blockchain technology could be applied in AFarCloud in order to store a set of data such as meat (AS11), milk, fruits and vegetables (AS05) production information. Storing this information into the blockchain can also guarantee data integrity, security and availability.

Sensors placed in strategical positions within the farm and/or on the animals shall provide the data to be collected; the subsequent step is storing this data into the blockchain. This will only happen after the block has been validated and, once it is, its content will be immutable.

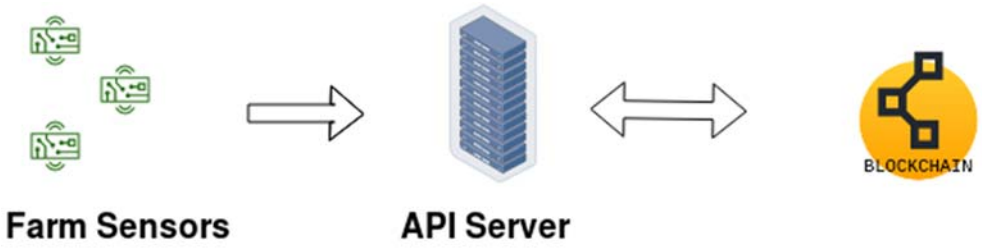

Figure 4: Blockchain architecture definition

While a blockchain implementation of data traceability can follow the whole supply chain (from the farm to the end user, e.g. the consumer), it doesn't necessarily have to, in order to reach its goal. The production steps to store into the blockchain shall not exit AFarCloud scope, and they can be retrieved from the blockchain and visualized on the user interface to raise awareness and fulfil the quality requirements of production.

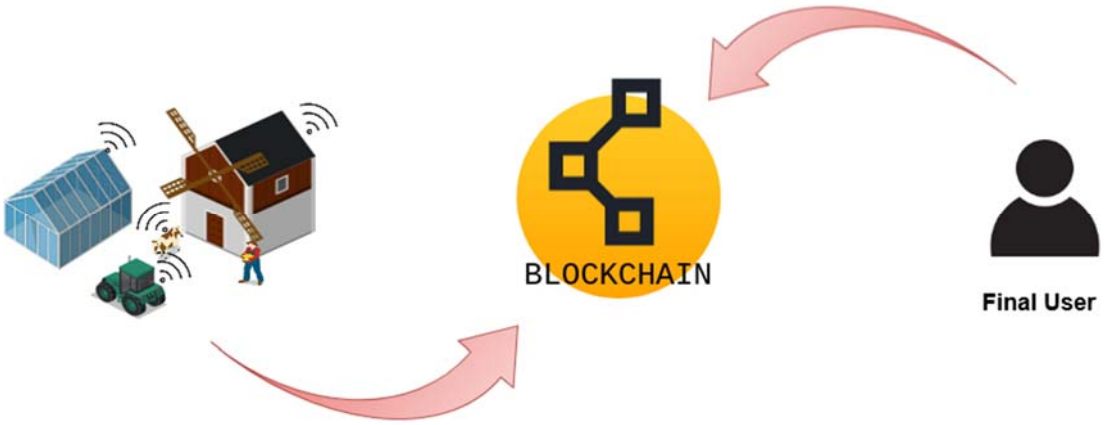

Figure 5: Blockchain technology in AFarCloud scenarios AS05, AS11

There are many possibilities for a blockchain development, from public platforms such as Ethereum and Hyperledger, to private and/or permissioned ones. A clever move might be to choose the technology according to what the majority of food distribution companies, which are currently looking into DLT (Distributed Ledger Technology) to track the supply chain for their end users, are choosing, in order to leave the door open for a possible future integration which will cover the product's first production steps, its development (transport, analysis) and its distribution to the final consumer.

4.4. Blockchain API Definition

In this section the data access method will be described. There will be a NodeJs server exchanging JSON data through express library.

4.4.1. writeData

Write data into the chain.

Request: Method: Post Type: Application/Json Path: /writeHash
Parameters: 1. Data (String) : Json Stringify holding the data we want to store into the blockchain (Required).
Returns: The method will return the transaction id (TxId) after the blockchain validation.

Example:
`Request:`
`{`
`“Data”:” {“ProductionData”: 1/09/2019,`
`“Producer”: “Azienda San Rossore”,`
`....`
`}”`
`}`
`Response:`
`{`
`“TxId”:“0xfd2006f6b50144a74fa3a65ec2873deb4967cfd6ab3e1ab45f598a294724033f"`
`}`

4.4.2. getData

This method will return the data written into the Blockchain.

Request: Method: Post Type: Application/Json Path: /getData
Parameters: The Id of the transaction holding the data we want to retrieve.
Returns:

This method will return the data written into the Blockchain and all the transaction information. Some of the data that could be packaged into JSON is the following:

- blockHash 32 Bytes - String: Hash of the block holding the transaction
- blockNumber - Int: Number of the block holding the transaction.
- hash - String: Hash of the transaction.
- gas - Int: gas spent in order to deploy the transaction.
- input- String: Hexadecimal String of the JSON insert into the blockchain.
- timestamp: timestamp of block mining.

Example:

```
Request:
{
“txId”:"0x85bE940C87647706Dc0f0fE82e7C07Fb02b20C40"
}

Response:
{
    "blockHash":
"0x313d4deda85dd9c969b582e43e8ca18c1e7900dfc21cff0a52a87946a223beff",
    "blockNumber": 5237525,
    "gas": 80000,
    "hash":
"0x5bcb46bedbae7d471516e7792b6f46845265e78b35faf534fc0b97fddd446bfb",
    "input":
"0xb1498e29000000000000000000000000000000000000000000000000000000000000002000000000000000000000000000000000000000000000000000000000000000203138396262626230306335663166623766626139616439323835663139337232",
    "timestamp": 1570701914,
}
```

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 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. This approach to develop the MMT in a modular manner means also that the main categories of services provided by the MMT will be implemented in T3.2 - T3.5. One important remark is that the MMT is not one configuration. The original MMT, referred to as the “main MMT”, and its derivatives aim at solving different needs, in various contexts. Information is critical in precision farming, and who should have access to what information is not a trivial problem. The initial classification of versions/interfaces as described in the DoW are as follows (implementation of the associated GUIs, except for the main MMT, is T3.4):

1. **Main MMT:** this version is the only MMT which is needed for the missions. It is for the operators that process data, plan a mission, monitor a mission, evaluate the results and work with the decision-support system.
2. **Mobile MMT:** This is the stripped-down version of the main MMT to be used at the mission site for monitoring purposes. This solution will be implemented as a web-based service as well.
3. **UAV-MMT:** Dedicated to the pilots of the UAVs. This configuration is derived from the mobile MMT.
4. **Tractor MMT** is dedicated to in-vehicle usage, i.e., in a tractor, and also with UGVs. This configuration is derived from the mobile MMT.

Some of the services (i)-(vi) are provided by various software solutions within the core of the main 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 middleware. 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 ISOBUS systems. This solution means also that the user will not interpret a DSS algorithm of the services within the System Configuration stage as separated from any of the MMT solutions.

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 a selected set of services when mobility near the mission site is required. In this context, three different configurations 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 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 a list of features of the vehicles such as, their status, properties, capabilities and equipment. 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 middleware to receive status updates from vehicles;
10. During a mission, providing the operator with messages and alerts received from the vehicles;
11. Allowing the operator to abort or re-plan a mission and execute it (the mobile MMT will have a sub-set of functionalities).

Figure 6 shows an overview of the MMT and its connections with the vehicles 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, ISOBUS systems, as well as non-moving equipment (which can be subject to hierarchical planning).

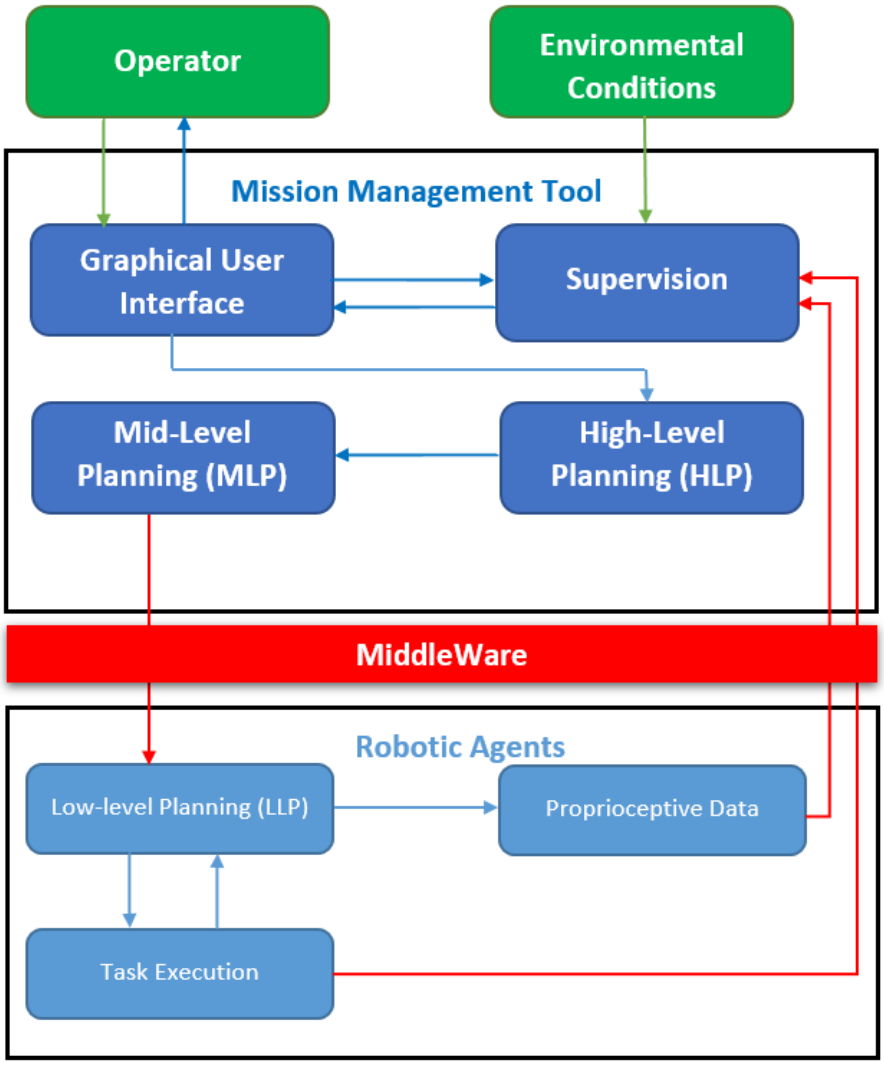

Figure 6: Overview of the MMT and its connections with the vehicles (Robotics Agents) through the MW

The MMT provides the operators with a Graphical User Interface (GUI) for services 1-11 above. This GUI provides 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 is possible to visualize measured data, trajectories of the vehicles, as well as other relevant data such as UAVs battery time/consumption. The MMT also provides the operators with planning capabilities. 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. Planning process can be divided into levels, or a hierarchy, as follows: (a) High-Level Planning (HLP), which, based on the information entered by the operator in the GUI, produces a plan, i.e., a list of ordered tasks assigned to vehicles; (b) Mid-Level Planning (MLP), which computes waypoint sequences based on an optimization approach according to some predetermined criteria; and (c) Low-Level Planning (LLP), which deals with the motion planning and management of actuators or equipment on-board. Finally, the MMT provides the operators with supervision of missions and vehicles, showing the operator 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 the Farm Management System and accessed by the MMT. More information about the MMT can be found in the deliverables related to WP3.

5.1.2. Software Interfaces

The MMT provides interfaces for development of tools and external components that need to be integrated with the MMT and its GUI (see Figure 7). These interfaces allow the AFarCloud partners to develop plugins for the MMT that can easily be added to it and provide new functionalities and user interfaces. In cases where a plugin interface cannot be used, Apache Thrift will preferably be employed.

5.1.2.1. Interfaces for Farm Management System

The DSS and the System Configuration are other parts of the Farm Management System that require to be accessible through the MMT’s GUI. They 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 the 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. 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 the MMT via web-components (and potentially an IFRAME). This integration will further require sharing of authentication/authorization tokens from the 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 (Web Map Service, a standard of Open Geospatial Consortium) protocols and presents the final image on the 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. Semantic Middleware

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

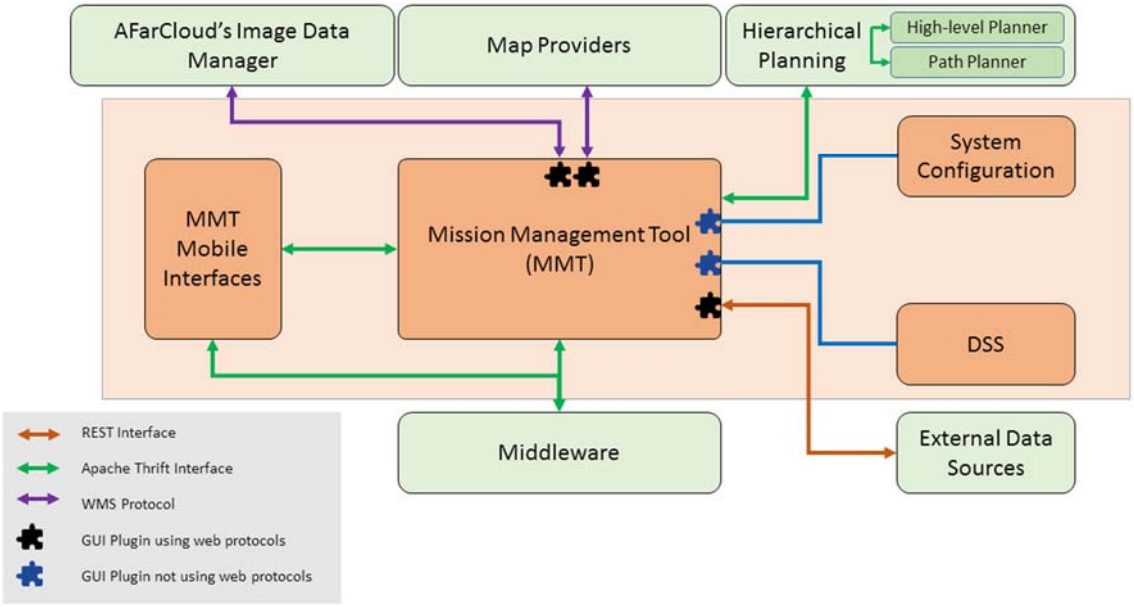

Figure 7: Interfaces of the Mission Management Tool

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. The DSS complies with criteria of scalability and adaptability, as users' requirements are different in each scenario (see Figure 8).

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" can be based on: (i) soil humidity in several points, (ii) solar radiation, (iii) amount of watering, (iv) rainfall, (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 for the 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 stored 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 the DSS need to implement an API to listen to 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, an API service is proposed 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 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 MQTT publish/subscribe mechanism.

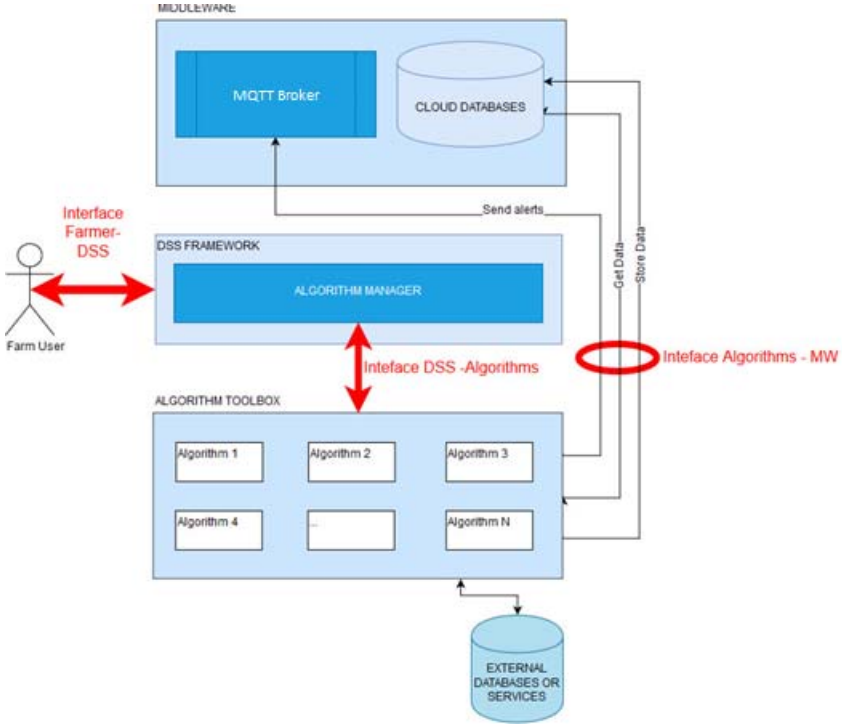

Figure 8: The AFarCloud DSS architecture and interfaces

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 what 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 repositories (for visualization purposes or for further analysis), and optionally, they can generate alerts via MQTT. This mechanism will publish relevant alerts, warnings or errors to interested subscribers (users or other components).

5.3. System Configuration

The System Configuration (SC) is responsible for a) the AFarCloud instance configuration; b) it facilitates the registration process of devices (collars, sensors and actuators) and vehicles (UAV, UGV and ISOBUS systems) through its graphical user interface and gathers information related to the Farm and the Customer. Moreover, c) it performs the pre-mission status control on the vehicles that are part of a specific mission, once requested by the MMT. Finally, d) it enables firmware updates' notification process, and e) provides for the configuration of sensors' and actuators' properties by sending the appropriate commands. Its main functionality is described in the section below in more detail.

5.3.1. Functionality

Registration process: for the registration process the SC will retrieve the required data for the device and vehicle registration from the SC Graphical User Interface. In addition, the SC GUI will gather farm and customer data relevant to that particular AFarCloud instance. Following that, the SC will forward the data in the predefined format that will be described in D2.6 to the Asset Registry component of the Middleware.

AFarCloud Instance configuration: for the AFarCloud instance configuration process the SC will interact with the AFarCloud cloud infrastructure orchestrator REST API (based on Kubernetes API, see D3.14 for more information). Through that interface it will be possible to:

- Parameterize each of the data repositories used per farm (SQL DB, NoSQL DB and ontology) and define their endpoint addresses and DB name, username and password.
- Parameterize the URLs of the REST services provided by AFarCloud components following a universal pattern e.g.: *http://afc/[scenario_name]/service/[entity_id]* etc.
- Parameterize the MQTT broker attributes such as their scenario name, username, password and endpoint address e.g. *http://afc/[scenario_name]/service/[sensor_id]* etc.
  The parameterization is going to be based on the farm, scenario and customer specific information that were gathered through the SC GUI.

Device Configuration: the SC will communicate to the Device Manager and send appropriate commands to set or update the parameter values that are requested to be changed, e.g. sample rate or actuator command.

Firmware update notifications: in case the middleware mediation is necessary for this process (the final solution will be designed in D2.4), the SC will facilitate the firmware update notification process by forwarding the relative command to the Device Manager.

Pre-mission status control: one of the core functionalities of the SC is to check the availability of the vehicles involved in the mission. The SC will perform all the necessary status control activities in order to ensure that the requested mission is possible to be executed by the AFarCloud platform. The SC will communicate with the Mission Manager component in order to request vehicles data in real-time.

5.3.2. Interfaces

The SC provides a Northbound interface with the Mission Management Tool (MMT) and a Southbound interface with the Device Manager, the Mission Manager, the Asset Registry and the Data Query components. Both interfaces are realized as RESTful APIs. Regarding the AFarCloud instance configuration functionality that the SC provides, it communicates with Kubernetes API (Kubernetes.io)[3]. The high-level interoperability schema is presented in Figure 9.

[3] https://kubernetes.io/docs/reference/generated/kubernetes-api/v1.10/

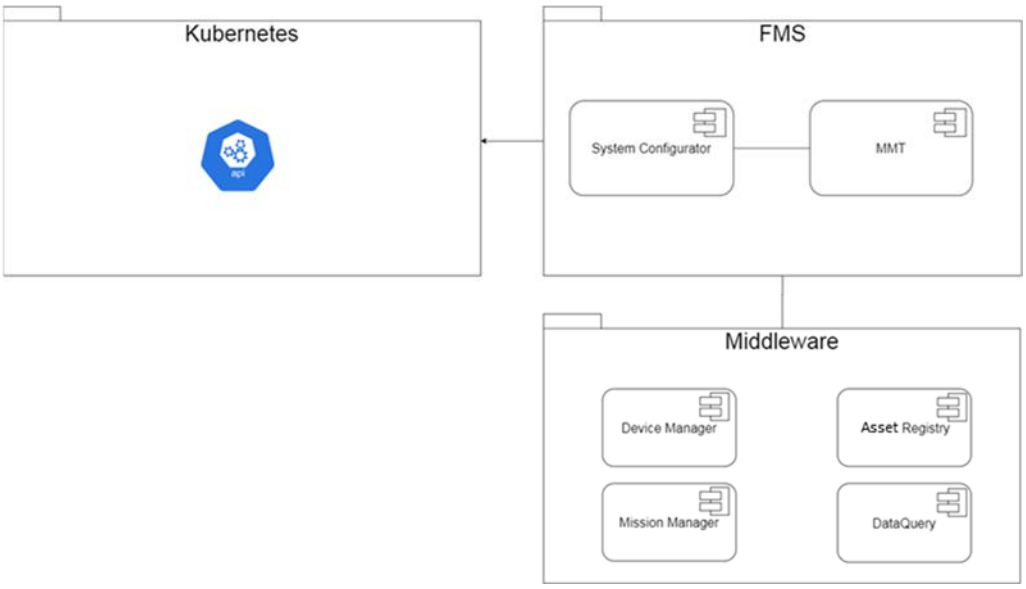

Figure 9: System Configuration interoperability schema

To this end the following high-level architecture of the SC is foreseen as shown in Figure 10. System Configuration modules are listed and described below:

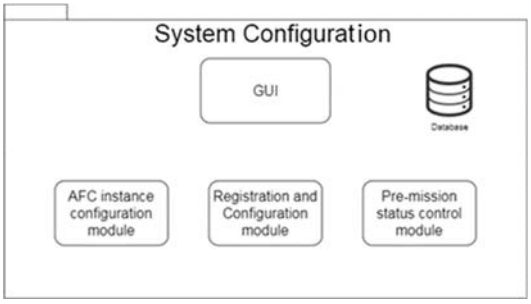

Figure 10: System Configuration High-level architecture

- The Graphical User Interface - GUI will realize the graphical interface to enter information regarding the user, the farm, the vehicles and devices that will be included on the AFarCloud instance. More specifically, the SC GUI will provide support for the:
    - Registration process of devices and vehicles by gathering and manipulating, if necessary, the information that are imported to the UI;
    - Gathering of Farm, User and Scenario information of the specific AFarCloud instance;
    - Configuration of sensors and actuators parameters (e.g., sampling rates);

- Firmware dispatch notification of available firmware updates for sensors and tractor controllers (in case the middleware mediation is necessary for firmware updates).

- The Registration and Configuration module will make any necessary data format adaptations in order to transform configuration and registration information received from the SC GUI to the relevant AFarCloud data format and transmit this data towards the underlying AFarCloud platform components.

- The Pre-mission Status Control module will perform the necessary status controls of the vehicles in order to evaluate which vehicles in the farm are available to be part of a mission. This request will be triggered through the MMT. This module will request from the Mission Manager the latest status vector of all UAVs/UGVs in the farm and will reply to the MMT with the information.

- The Instance Configuration module will implement the management logic that will allow the configuration of AFarCloud components' parameters, based on the information gathered through the SC GUI and communicate these configuration details to the underlying AFarCloud platform infrastructure container orchestrator (Kubernetes) via its RESTful API.

- The System Configuration Database will store internally data that are to be needed by the above-mentioned modules of the SC. This will help accelerate the provision of any necessary SC specific service and it will reduce the necessary response time of the component.

6. The Semantic Middleware

Figure 11 depicts the components and interfaces of the Semantic Middleware.

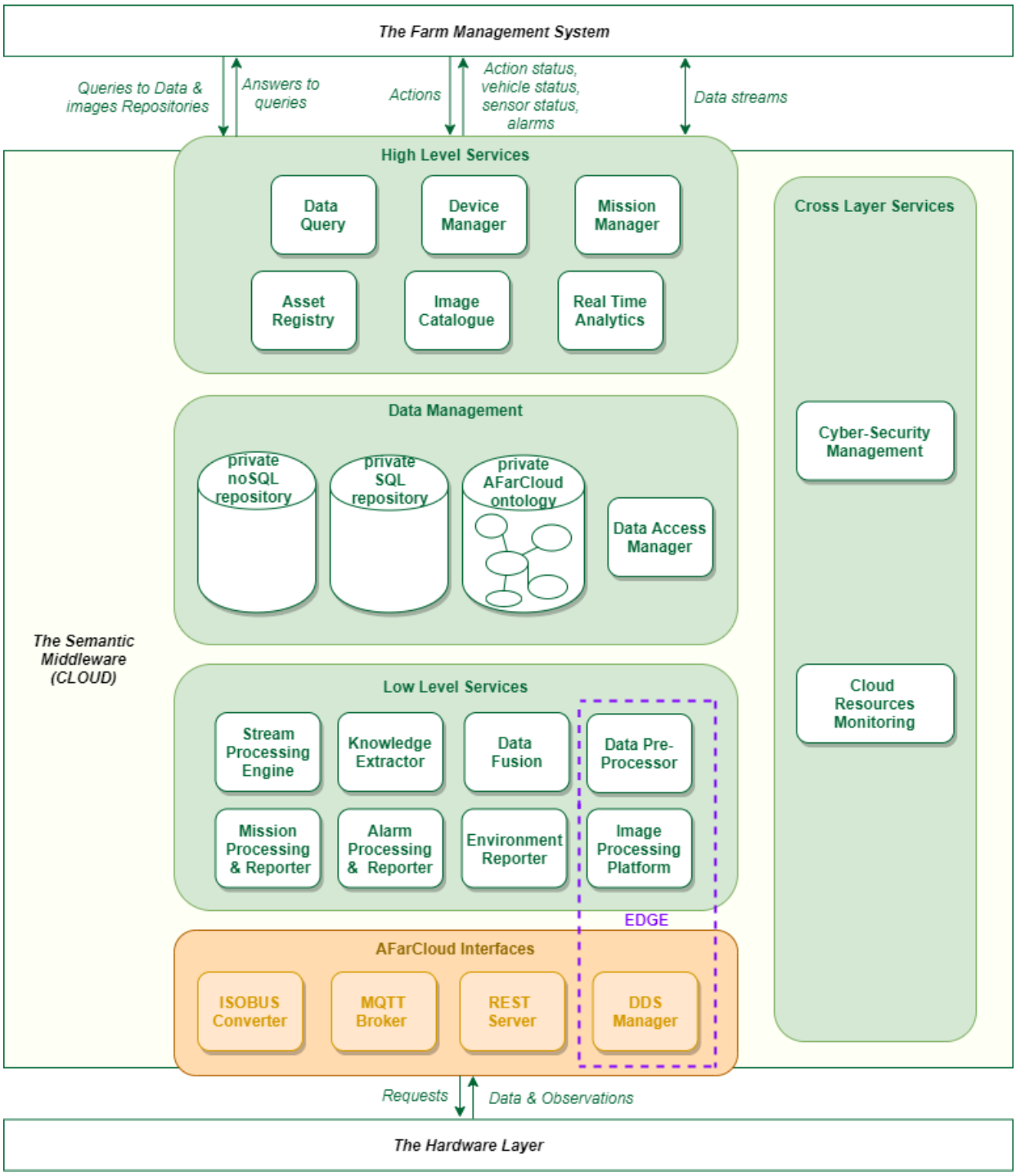

Figure 11: Components of the Semantic Middleware

Most of the AFarCloud middleware components are hosted by the cloud infrastructure 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.

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

- The Image Processing Platform: due to the large size of the images processed by this component and taken by UAVs, the loading of these images is carried out offline (i.e., through a memory stick), to minimize errors and communication costs in the transmission of files and speed up the process of loading.
- The DDS Manager: as this module is responsible for processing the real-time communication with UAVs, it is deployed as close as possible to the place where data is generated, to minimize latencies.
- The Data Pre-Processor: for complex sensors, pre-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, the processing time and the cost of both transmission and storage. For simpler sensors that don't have pre-processing, it will be done in the cloud.

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: for querying AFarCloud repositories, sending missions to vehicles, sending commands to actuators and collecting mission results, sensor measurements and alarms.
    - **Web Map Service** interface: for retrieving images from the Image Catalogue.
    - **Apache Kafka** interface: for the DSS to consume streaming data in real-time and enable real-time analytics.
- Interface with the Hardware Layer:
    - **DDS** interface with UAVs and UGVs: for sending commands and collecting results and alarms. DDS communications are managed in real-time.

- **ISOBUS** interface with ISOBUS systems: for sending agricultural tasks and collecting results.
- **MQTT** interface with IoT compatible devices: for collecting measurements from sensors, sending commands to actuators, collecting telemetry from tractors. MQTT communications are managed in real-time.
- **REST** interface with IoT compatible devices: for collecting sensors measurements.

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 the 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 can be used to store the latest 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 in transactions.

Finally, NoSQL repositories (InfluxDB) will be used for storing observations from IoT devices and sensors. It is expected that this volume of data will be large and grow in the future and measurements coming from different sensors are not related 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 are behaving in accordance to the following expected non-functional requirements (NFR): availability, workload job processing performance, overall performance, and response time.

In the context of AFarCloud, availability is defined by the following formula:

$$Availability\ (A) = MTBF/(MTBF+MTTR)$$

For this metric, two parameters have been defined:

- MTBF: Mean time between failures – the amount of time that happens in between failures and it is measured as downtime - uptime
- MTTR: Mean time to recover, which is the amount of time required to restore a system back to its full functionality.

The overall goal of the workload performance monitoring is the intelligent scheduling of the workload processing, such that soft guarantees can be given for throughput or processing time while optimizing the effective utilization of resources in a cloud. To allow the development of an intelligent workload scheduler that plans workloads to individual computing nodes available in the cloud, it is mandatory to monitor workloads' performance, i.e., how much the individual resources are used by particular jobs.

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 "D4.7 Cloud storage infrastructure v1"):

- 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 for 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 in a different database, namely this one, for performance issues on the queries.
- UserManagement: this is a generic module to manage the users that have 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

The Data Access Manager is the Semantic Middleware component that manages data storage and retrieval. This component provides interfaces able to insert/retrieve/update information in the different AFarCloud repositories: ontology, 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. The non-relational database will be implemented using InfluxDB.

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:

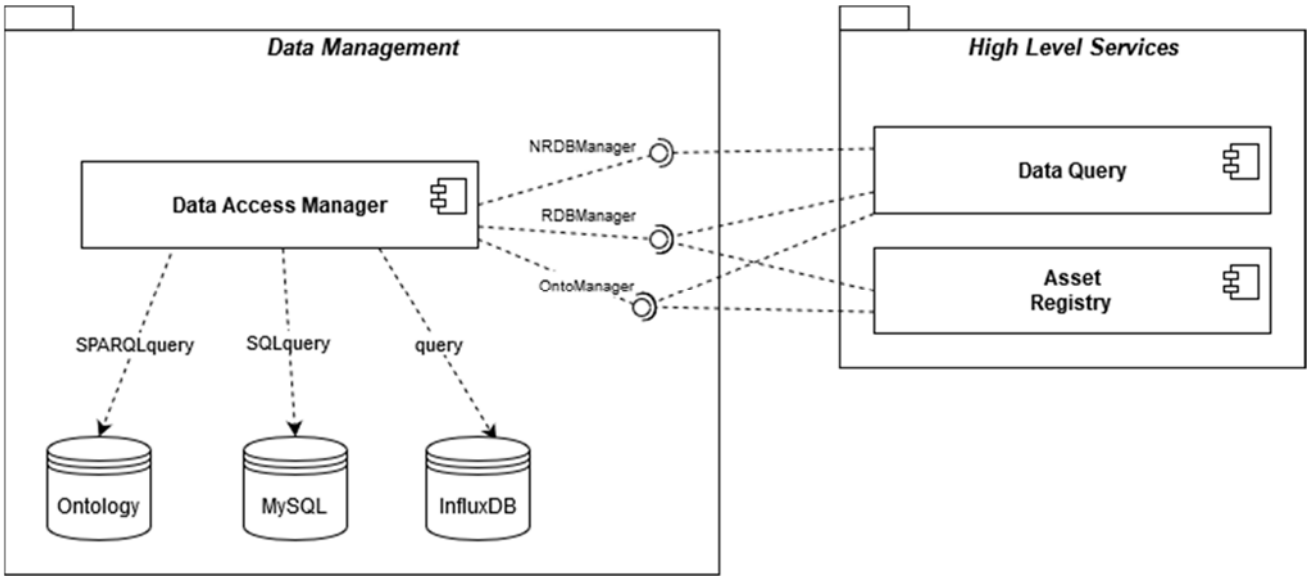

Figure 12: Data Access Manager Components Diagram

6.4.3. Interfaces

The Data Access Manager (DAM) defines three internal interfaces: OntoManager (in Table 1), RDBManager (in Table 2) and NRDBManager (Table 3). 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 and the Asset Registry for CRUD operations to the AFarCloud repositories.

6.4.3.1. OntoManager

Table 1. 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) **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.

6.4.3.2. RDBManager

Table 2. RDBManager interface

(*public*) **queryDatabase**(query)	This method provides the endpoint to query the database.
(*public*) **openDBConnection**()	This method establishes a database connection. Credentials and endpoint url are taken from the configuration file.
(*public*) **closeDBConnection**()	Closes the connection to the database.
(*public*) **insertDatabase**(query)	This method provides the endpoint to insert data to the database

6.4.3.3. NRDBManager

Table 3. NRDBManager interface

(*public*) **queryDatabase**(query)	This method provides the endpoint to query the database.
(*public*) **openDBConnection**()	This method establishes a database connection. Credentials and endpoint url are taken from the configuration file.
(*public*) **closeDBConnection**()	Closes the connection to the database.
(*public*) **insertDatabase**(data<key,value>)	This method provides the endpoint to insert data to the database

6.5. Data Query

This component processes any query made to manipulate data in the AFarCloud repositories. The queries are the mechanisms used by the rest of the AFarCloud components to insert, consult or update any information from the AFarCloud repositories.

This module will provide REST and Thrift services for predefined queries (e.g., for retrieving information from vehicles, querying data from a mission, inserting observations from sensors, etc.) and will translate them to the repository's 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. Next, we provide, as example, the interfaces of some of these services.

6.5.1. Components Diagram

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

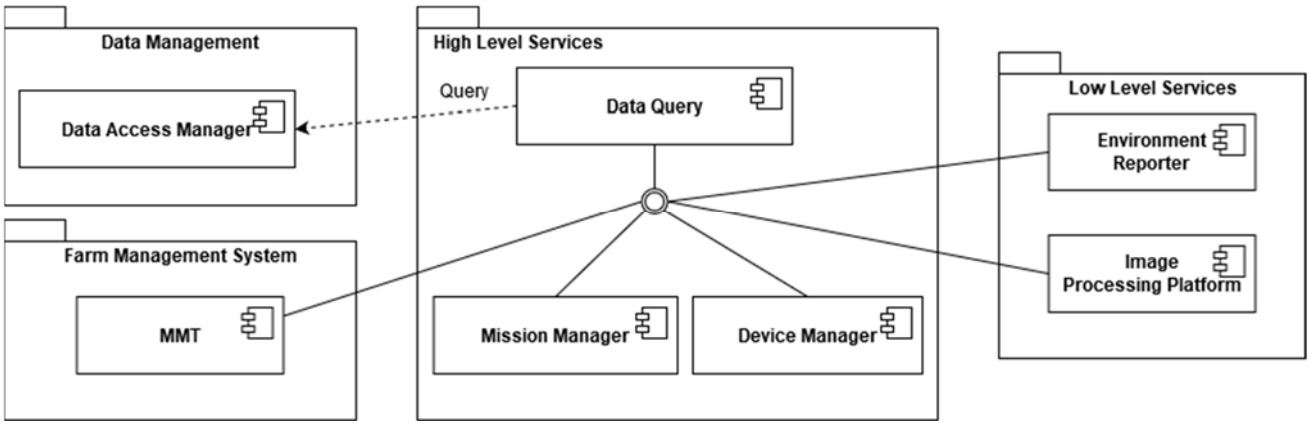

Figure 13: Data Query components diagram

6.5.2. Interfaces

6.5.2.1. Thrift services for mission management and data visualization in the MMT

The Data Query (DQ) exposes Thrift services that the MMT uses when planning a mission and when visualizing data. These services allow the MMT, for example, to know the last updated status of the vehicles available or involved in a mission, the last measurements acquired by the sensors and IoT devices, the events or alarms generated in the farm and the historical values in a period of time.

Table 4. Thrift services

list <Vehicle> **getAllVehicles**()	Returns all vehicles that are available for a mission.
list <MissionTag> **getAllMissions** (),	Returns all the missions stored in the repository
list <MissionTag> **getOngoingMissions** (),	Returns a list of all missions that contain unfinished tasks.
Vehicle **getVehicle** (1: i32 vid)	Returns the last status of a vehicle or null if no vehicle was found with the given id.
oneway void **queryStateVector** (1: i32 requestId, 2: i32 vehicleId, 3: i32 startTime, 4: i32 endTime),	Retrieves the last known State Vector for given vehicle at given time period. If no time period is given, then the last known State Vector ever.
oneway void **querySensorData** (1: i32 requestId, 2: Region region, 3: i32 startTime, 4: i32 endTime, 5: SensorType sensorType)	Retrieves the last observations of sensors in a region and of a certain type at a given time period.

oneway void **queryHistoricalStateVectors** (1: i32 requestId, 2: i32 vehicleId, 3: i32 startTime, 4: i32 endTime)	Retrieves the State Vectors for given vehicle at given time period.
oneway void **queryHistoricalSensorData** (1: i32 requestId, 2: Region region, 3: i32 startTime, 4: i32 endTime, 5: SensorType sensorType)	Retrieves all the observations of sensors in a region and of a certain type measured in the given time period.
oneway void **storeEvent** (1: i32 requestId, 2:i32 missionId, 3:i32 vehicleId, 4:i32 subtype, 5:string description, 6:i64 timeReference)	Reports an event to the middleware

6.5.2.2. REST services for storage of regions

The Data Query offers REST services for the Image Processing Platform (IPP) to store the outcome of its algorithms. This outcome will be the regions with water stress, weeds and dead plants in the vineyard.

POST **/region/measure** Store a new region measurement from IPP
Parameters: name: body schema: ref: "#/definitions/RegionData" (defined in D2.6)
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" (defined in D2.6)
Responses: 200: "Successful operation" 405: "Invalid input"

6.5.2.3. REST services for managing observations from devices

The Data Query component also has to provide means for storing alarms and observations from devices, as well as for retrieving all this information when required from other components. This

section provides a first version of the Data Query interface for these two operations, using the REST protocol. In D2.6, readers will find the complete final version of the API for these two operations: storing and retrieval of information from devices.

The AFarCloud data model, included in D2.6, establishes JSON format as the preferred representation format for data exchange. Thus, the Data Query component just needs the JSON representation for storing data:

POST /store/sensorTelemetry Stores measurements from sensors, either aggregated or not. Data must comply with AFarCloud data model in D.2.6.
Parameters: name: body schema: #SensorTelemetryData (defined in D2.6)
Responses: 200: "Successful operation" 405: "Invalid input"
POST /store/collarInfo Stores information from a collar or from a list of collars. Data must comply with AFarCloud data model in D.2.6.
Parameters: name: body schema: #CollarData (defined in D2.6).
Responses: 200: "Successful operation" 405: "Invalid input"
POST /store/vehicleStatus Stores information related to vehicle status (vehicle telemetry, state vector, etc.). Data must comply with AFarCloud data model in D.2.6.
Parameters: name: body schema: #VehicleStatusData (defined in D2.6).
Responses: 200: "Successful operation" 405: "Invalid input"

The type of information to be retrieved from the AFarCloud repositories, or more precisely, the queries for information that different components may require, are not completely defined so far. But, as it can

be concluded from the Thrift services listed above, the envisioned information that other architecture components may need to retrieve is the latest or the historical telemetry from sensors (including collars) or from vehicles, that meet specific constraints like location, type, etc. Readers can get the final version of the interface in D2.6.

GET /getSensorTelemetry/latest Retrieves the latest telemetry from sensors that meet specific constraints.
Parameters: name: query schema: JSON. The requester must provide the time interval for narrowing the search down, as well as the conditions that must meet sensors to be selected: sensor location is in a specific region, sensor type, etc.
Responses: 200: "Successful operation". A JSON with the latest observation for each of the sensors that meet the specific conditions. 405: "Invalid input"
GET /getVehicleStateVector/latest Retrieves the latest state vector of a vehicle.
Parameters: name: query schema: JSON. The requester must provide the time interval for narrowing the search down and the vehicle identifier.
Responses: 200: "Successful operation". A JSON with the latest state vector of the vehicle with the provided identifier. 405: "Invalid input"
GET /getSensorTelemetry/historic Retrieves all the telemetry in a time interval from sensors that meet specific constraints.
Parameters: name: query schema: JSON. The requester must provide the time interval for narrowing the search down, as well as the conditions that must meet sensors to be selected: sensor location is in a specific region, sensor type, etc.
Responses: 200: "Successful operation". A JSON with the all the observations for each of the sensors that meet the specific conditions. 405: "Invalid input"

GET **/getVehicleStateVector/historic** Retrieves all the state vectors in a time interval of a vehicle.
Parameters: name: query schema: JSON. The requester must provide the time interval for narrowing the search down and the vehicle identifier.
Responses: 200: "Successful operation". A JSON with all the state vectors of the vehicle with the provided identifier. 405: "Invalid input"

6.6. Asset Registry

6.6.1. Description

The Asset Registry registers the static information about the farm, the customer and the assets in the farm (sensors, collars, tractors, UAVs, etc.) in the AFarCloud data repositories. This includes information about the location of the farm, its part fields and crops, customer information and the list of IoT devices in the farm. 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 and UGVs, it includes the equipment onboard, the type of commands that the vehicle can understand, etc. For tractors, it contains the tasks it can perform and the implements on-board.

6.6.2. Components diagram

The Asset Registry uses the Data Access Manager to store and update the static information in the AFarCloud data repositories. This component offers the registration functionality as REST services that can be invoked by the MMT or System Configuration in the FMS. This service hides the complexity of the registration process from the farmer.

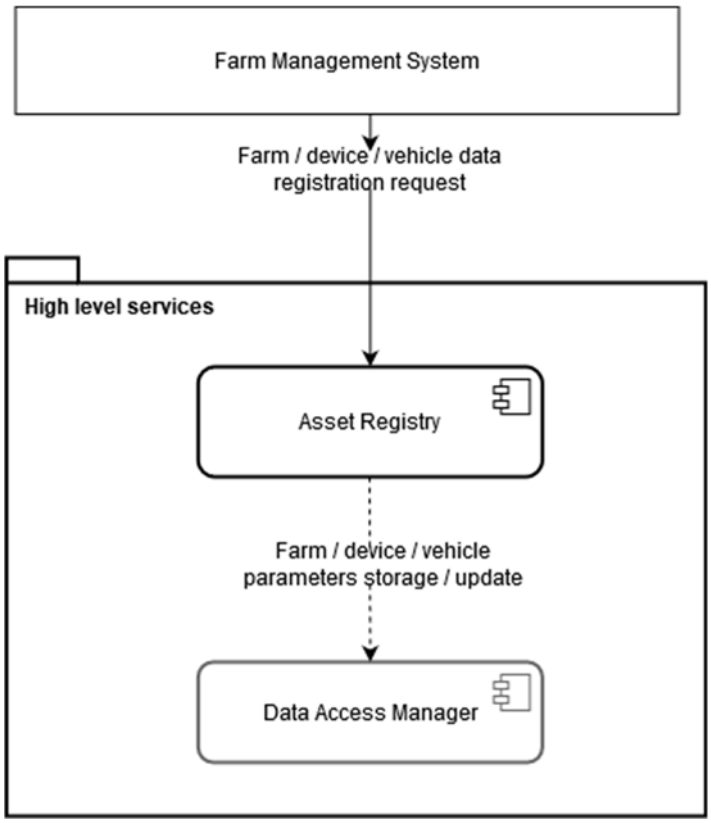

Figure 14: Asset Registry component diagram

6.6.3. Interfaces

The Asset Registry offers REST services to retrieve, store and delete the farm's and assets information from the repository. In the case of POST operations, the response will be:

- 200: "Successful operation"
- 405: "Invalid input"

The data schema for sensors, vehicles, the farm and the customer can be found in deliverable D2.6.

- "#/definitions/VehicleData"
- "#/definitions/LivestockData"
- "#/definitions/SensorStaticData"
- "#/definitions/FarmData"
- "#/definitions/CustomerData"

Table 5. Interfaces provided by the Asset Registry

	Method	Path	Input/output parameter	Description
Sensors and actuators	POST	/registry/addSensor	I: SensorStaticData O: 200 / 405	Register a new sensor and its observation properties
	POST	/registry/delSensor/{sensorID}	I: Sensor identifier O: 200 / 405	Deletes a sensor
	GET	/registry/getSensor/{sensorID}	I: Sensor identifier	Retrieve the information from a sensor
Vehicles	POST	/registry/addVehicle	I: VehicleData O: 200 / 405	Register a new vehicle
	POST	/registry/delVehicle/{vehicleID}	I: vehicle identifier O: 200 / 405	Delete a vehicle from the repository
	GET	/registry/getVehicle/{vehicleID}	I: vehicle identifier O: VehicleData	Retrieve the information from a vehicle
Collars	POST	/registry/addCollar	I: LivestockData O: 200 / 405	Register a collar and the animal's information
	POST	/registry/delCollar/{legal_id}	I: legal id of the animal O: 200 / 405	Delete a collar and the animal's information
	GET	/registry/getCollar/{legal_id}	I: legal id of the animal O: LivestockData	Retrieves the information of the collar and animal related to that legal identifier.
Farm	POST	/registry/addFarm	I: FarmData O: 200 / 405	Register a farm
	POST	/registry/delFarm/{farmId}	I: id of the farm O: 200 / 405	Delete a farm and its information
	GET	/registry/getFarm/{farmId}	I: id of the farm O: FarmData	Retrieves the information of the farm.
Customer	POST	/registry/addCustomer	I: CustomerData O: 200 / 405	Register a customer

	POST	/registry/delCustomer/{cId}	I: id of the customer O: 200 / 405	Delete a customer
	GET	/registry/getCustomer/{cId}	I: id of the customer O: CustomerData	Retrieves the information of the customer.

6.7. Stream Processing Engine (SPE)

This section introduces the Stream Processing Engine (SPE) as a part of the Real-Time Analytics layer of the AFarCloud platform. This component was formerly named in deliverable D2.2 as Streaming Engine within the High-Level Services layer. The name has been updated to better reflect the main functionality of the component. The SPE provides real-time streaming data pipelines for reliable exchange of data between the AFarCloud Interfaces, third-party software and the end-users, i.e. Farm Management System. A preliminary SPE concept was presented in D2.2[4]. The aim of this section is to present the role of the SPE within the AFarCloud platform, and the correlation between the SPE and other AFarCloud components.

6.7.1. Description

The SPE allows for real-time data processing and analytics based on Lambda Architecture[5][6], i.e. a generic, scalable and fault-tolerant data processing architecture. This architecture is based on an append-only and immutable data source; thus, the serving layer is decoupled from data (events) storage and processing.

The aim is to **aggregate and perform data analytics on** the data inbound from heterogeneous data streams (data sources, data from software systems and devices such as sensors) under real-time constraints.

Apache Kafka libraries will be used to process the data streams. This component will use pre-processing algorithms from WP4 (T4.1) and will be used by WP3 for real-time analytics. More details can be found in deliverables D3.7 and D4.1.

[4] AFarCloud consortium, "D2.2 Architecture Requirements and Definition", public report
[5] M. Hausenblas, N. Bijnens, Lambda Architecture (url: http://lambda-architecture.net/)

[6] A. Storm, "The Lambda Architecture, simplified", Jul 2017 (url: https://medium.com/@ajstorm/the-lambda-architecture-simplified-a28e436fa55e)

6.7.2. Components diagram

SPE utilizes the Apache Kafka[7] platform to implement a Data Broker (see Figure 15). This is the core element that manages the inbound data. Kafka provides tools for managing real-time data pipelines and creating provider, consumer, and streaming applications.

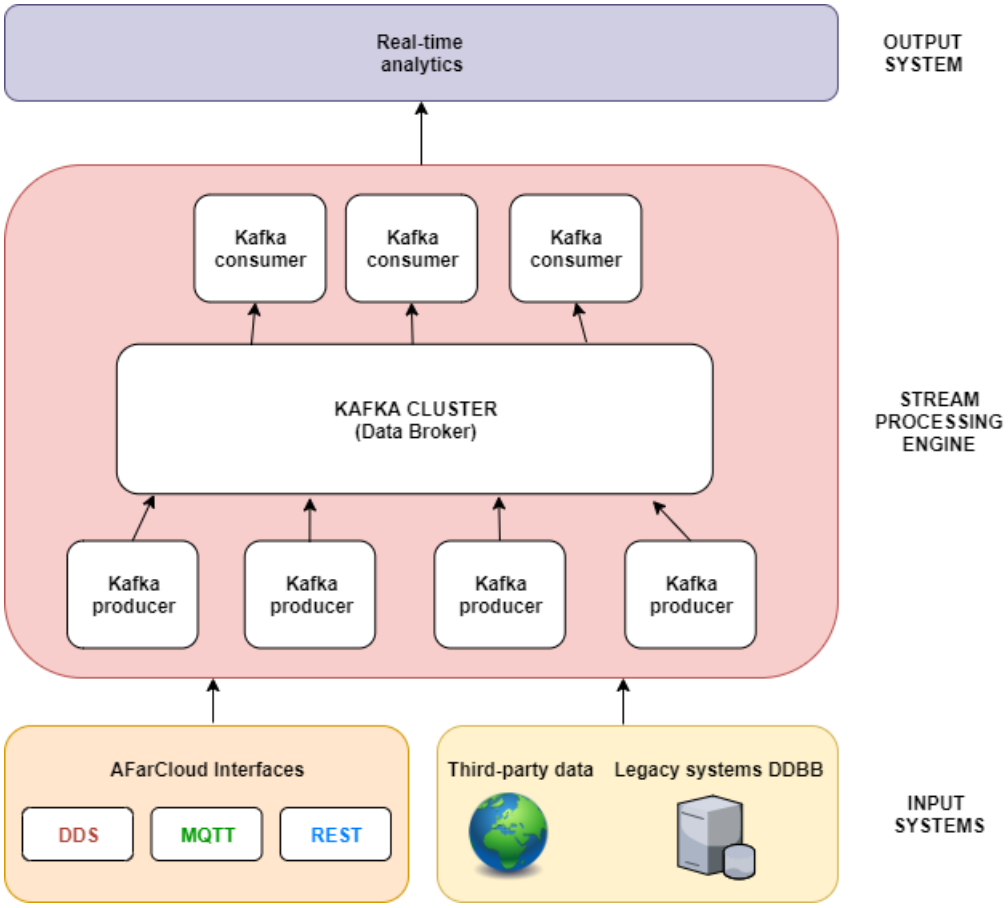

Figure 15: Stream Processing Engine components diagram

Figure 16 presents SPE within the AFarCloud architecture as the middleware solution for real-time data processing, aggregation and analytics in addition to the batch processing capabilities. For the SPE, the data producers acquire data from heterogeneous inputs systems (deployed hardware through the AFarCloud interfaces, third-party data, legacy systems and databases) and publish it into the Kafka cluster. Output systems, such as real-time analytics applications, can subscribe to a specific topic providing data, so they are notified as soon as new data is available and can retrieve these data as Kafka consumers.

[7] Apache Software Foundation, "Apache Kafka, A distributed streaming platform" (url: https://kafka.apache.org/)

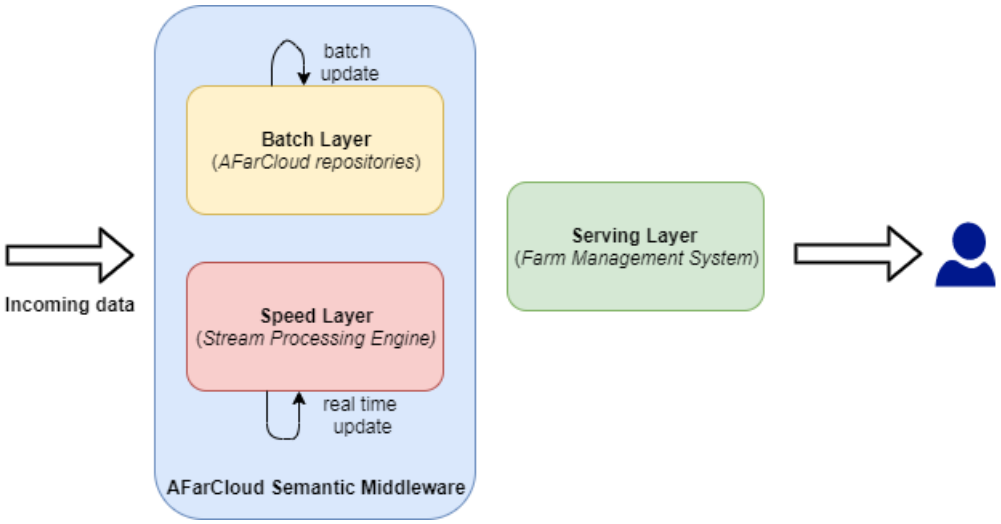

Figure 16: AFarCloud is based on Lambda architecture

As in Figure 16, the Farm Management System of AFarCloud can benefit both from the batch and the real-time layer of the architecture. For cases in which there is a need of aggregated real-time data (e.g., data grouped for specific type or domain or intermediate values calculation), the SPE is capable of performing the needed data fusion for further processing by third-party applications or the Decision Support System. For example, for monitoring of cow breeding zone, the FMS can consume data from the SPE in order to provide all necessary information about breeding, such as detecting abnormalities, planning future breeding, calculating the costs of infrastructure and herd maintenance, etc.

6.8. Device Manager

6.8.1. Description

This component is responsible for the management of standalone devices (sensors, actuators, etc.) and groups of devices (e.g., WSNs) connected to the AFarCloud platform. The management operations covered by the Device Manager are the following:

a) Setting the sampling rate of sensors (if possible);
b) Sending notifications on firmware updates to tractor controllers and sensors (in case the middleware mediation is necessary for firmware updates);
c) Sending commands to actuators;
d) Receiving alarms generated by sensors and actuators. The Device Manager will forward these alarms to the MMT;

Operations (a), (b) and (c) will be triggered by the farmer or system operator through the FMS.

6.8.2. Components diagram

The Device Manager manages all requests on devices done by the farmer or system operator through the FMS. These settings and commands can be stored in the AFarCloud repositories through the Data Query. For MQTT devices, requests to change the sampling rate of sensors (if possible), actuator commands and notification of firmware updates are published through the MQTT Broker, that publishes messages on specific topics addressed to the involved MQTT devices.

The Alarm Processing & Reporter transfers alarms coming from devices to the Device Manager, that forwards them to the MMT in the FMS.

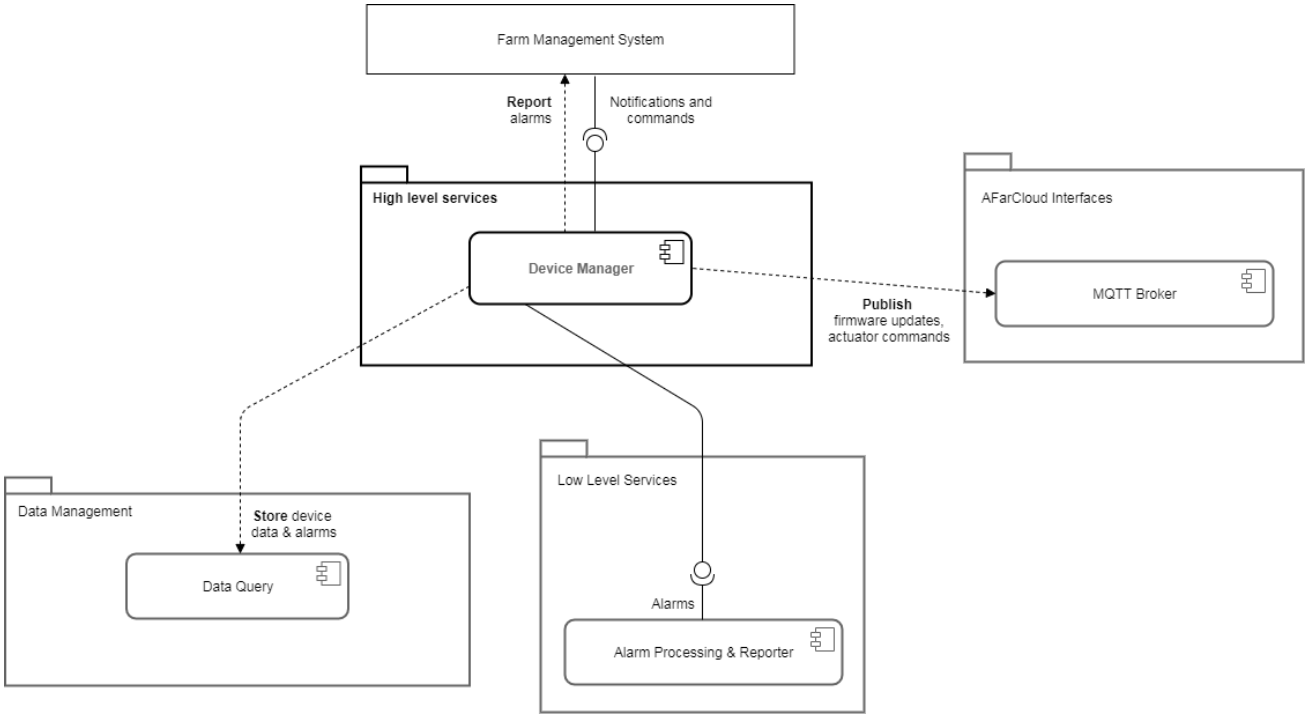

Figure 17: Device Manager components diagram

6.8.3. Interfaces

6.8.3.1. Thrift interface for sending commands to IoT devices

The Device Manager exposes the following Thrift services to the FMS:

Table 6. Thrift services exposed by the Device Manager

setSamplingRate (entityName, period)	The FMS must invoke this method to set a new sampling rate on a device. Input parameters: identification of the device (entityName) and new sampling rate (period) in minutes.

sendCommand (entityName, command, [params])	The FMS must invoke this method to send a command to an actuator. Input parameters: identification of the device (entityName), command and list of possible parameters of the command.
newFirmwareUpdate (entityName, [params])	In case the middleware mediation is necessary for firmware updates, the FMS must invoke this method to notify tractor controllers or sensors about a new firmware update. Input parameters: identification of the device (entityName), needed parameters (params).

6.8.3.2. REST interface to report alarms

The Device Manager exposes the following REST service to the Alarm Processing & Reporter:

Table 7. REST service exposed by the Device Manager

alarm (entityName, alarmCode, description)	The Alarm Processing & Reporter must invoke this method to report on any new alarm generated by the devices (i.e., sensors and actuators). Input parameters: identification of the device (entityName), alarm code and alarm description.

6.8.3.3. Publishing commands for IoT devices through MQTT

In AFarCloud, the management of sensors and actuators is done for MQTT compatible devices. The list of MQTT topics to be used by the Device Manager for publishing operations triggered by the FMS (i.e., a-, b- or c- described in section 6.8.1), is provided below. The JSON data formats for publishing any of these operations is defined in deliverable D2.6 Semantic Middleware. As described in D2.6, all devices are uniquely identified in an AFarCloud instance by means of their resource ID:

```
urn:afc:[scenario]:[service]:[provider]:[type]:[entityName]
```

A. SENDING COMMANDS TO ACTUATORS OR SETTING SAMPLING RATES TO SENSORS

Proposed topic structure:

`afc/[scenario]/[service]/[deviceType]/[entityName]/action`

where:

- [scenario] must be replaced by the scenario name defined at configuration time by the System Configuration of the FMS
- [service] represents an application domain, for example water management or environmental observations
- [deviceType] is a field used to identify the kind of sensor or actuator
- [entityName] must be replaced by the entityName of the sensor or actuator defined at configuration time by the System Configurator

Proposed data model:

```
{
   "resourceId":
"urn:afc:AS03:cropsManagement:RISE:soilSensor:afc_node_0100_0",
  "sequenceNumber": 123,
  “action”: “set_sampling_rate”,
  “value”: “20”
}
```

Where:

- Action is the command to be performed by the sensor or actuator. For example:
  - NTP device: Alarms_reset (True), ionization_module left (True/False), ionization module right (True/False)
  - Irrigation valve: enabled (True/False)
  - Sensor: set_sampling_rate
- Value is the parameter needed to perform the command

B. INFORMING ABOUT FIRMWARE UPDATES TO DEVICES

In case the middleware mediation is necessary for performing firmware updates, an MQTT based mechanism will be defined. This functionality will be documented in D2.4.

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 Management 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 (e.g., command to abort a mission) sent by the middleware that should be considered by vehicles or other IoT devices. Events will be generated by the Farm Management System as a result of an analysis of data.

More information on mission types can be found in deliverable D2.6.

6.9.2. Components diagram

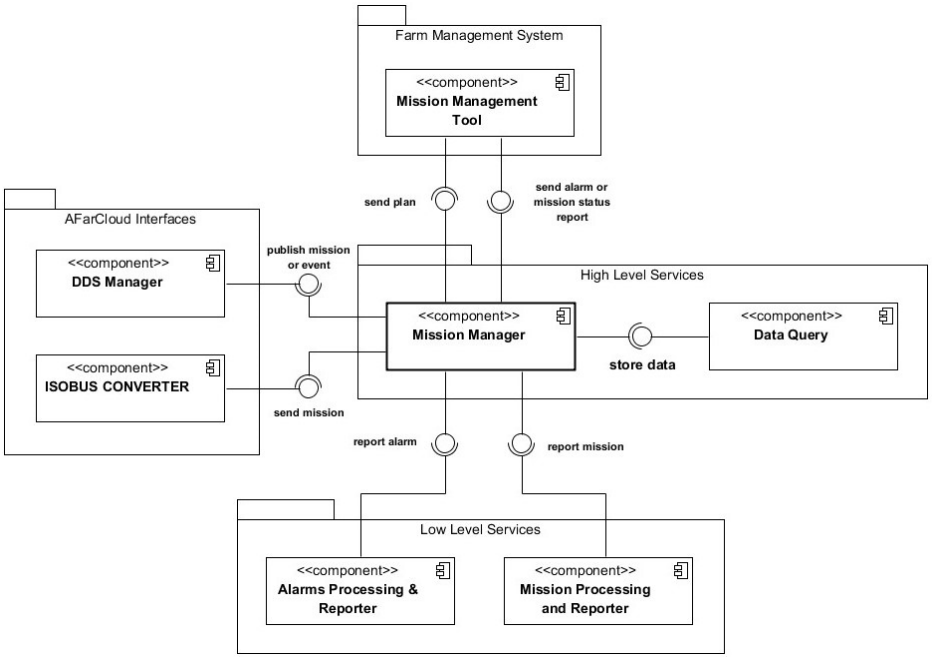

Figure 18: Mission Manager components diagram

The Figure 18 depicts the Mission Manager components diagram. The Mission Manager component receives the mission plan for each vehicle from the Mission Management Tool through the *send plan* interface. The Mission Manager processes the mission plan and commands vehicles indirectly through the DDS Manager (in the case of UAVs and UGVs) and the ISOBUS Converter (in the case of ISOBUS systems) and their respective interfaces *publish mission or event* and *send mission*. During the mission, any vehicle alarm is reported through the Alarms Processing & Reporter component using the *report alarm* interface. Mission reports are to be reported through the Mission Processing & Reporter using the *report mission* interface. Any significant data is stored in the AFarCloud repositories using the Data Query component.

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 Figure 19 depicts the Mission Processing & Reporter components diagram. The Mission Processing & Reporter component gets information about all the missions in the system through the interface *report mission status* that this component indirectly offers to vehicles carrying out missions (through the MQTT Subscriber in the case of tractors and ISOBUS systems, and through the DDS Manager in the case of UAVs and UGVs). All reports are sent to the Mission Manager through its *report mission* interface.

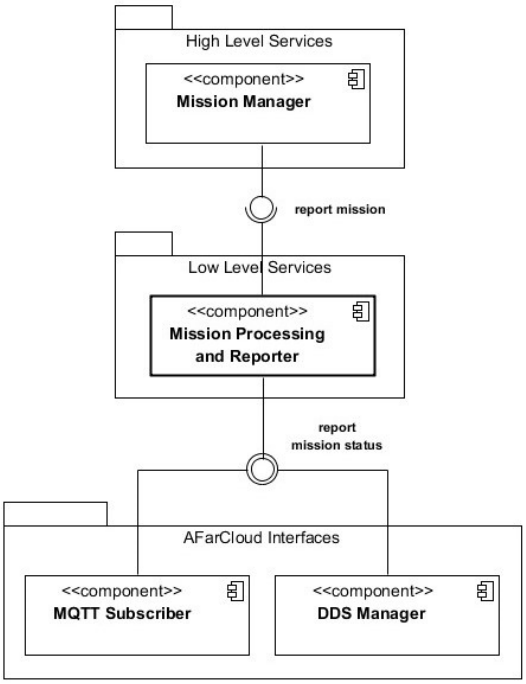

Figure 19: Mission Processing & Reporter components diagram

6.11. Alarm Processing & Reporter

6.11.1. Description

The Alarm Processing & Reporter processes the alarms triggered by the devices, validating the data received to ensure that the format is clean and correct, and forwards them to the Farm Management System. We consider as alarms any message sent from a sensor (i.e., standalone sensor, collar, WSN, etc.), actuator, UAV or GV 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, collar battery warning, etc. 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.

Drone alarms could be:

- low battery;
- low number of GPS satellites;
- failed calibration – including detail on the cause of the failure (e.g., no GPS fix, fail in accelerometer calibration);
- failed communication with any sensor equipped (e.g., no camera communication);
- low memory on the on-board computer.

The collars include a set of alarms that can be sent to notify the farmer. These notifications can be modified by the user for alarms he/she want to receive. These notifications include:

- battery warning: alarm when the battery level is low or when the device is indoors, and it is consuming more than it should.
- coverage recovered: this rule allows getting alerts when a device recovers the coverage
- device lost: this rule sends notifications when the device is not in the animal.

Besides, the Alarm Processing & Reporter also processes the notifications/warnings triggered by the Decision Support System, forwarding this information to the Farm Management System.

6.11.2. Components diagram

The Alarm Processing & Reporter deals with alarms arriving through the MQTT, REST or DDS interfaces. This information is transmitted as feedback to the Farm Management System (via the Device Manager), responsible both to plan each of the missions and to monitor the global system.

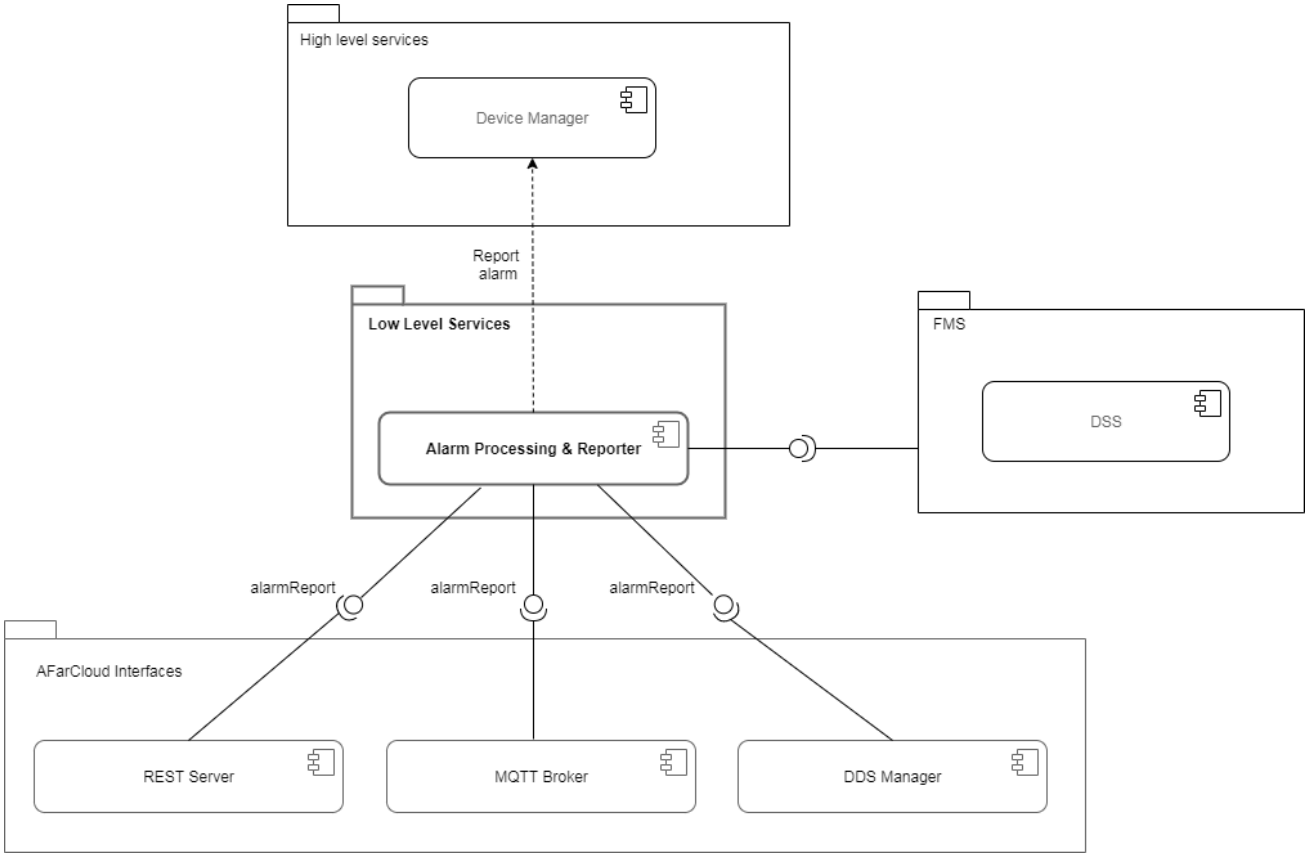

Figure 20: Alarm Processing & Reporter components diagram

6.11.3. Interfaces

AFarCloud offers three different interfaces to devices connected in the Hardware Layer, so they can send alarms triggered to the platform:

- MQTT interface to collect alarms triggered by MQTT compatible sensors (e.g., standalone sensors, WSNs), actuators, tractors, ISOBUS systems and the DSS.
- REST interface to collect alarms triggered by REST compatible sensors (e.g., collars).
- DDS interface to collect alarms triggered by UAVs and UGVs.

6.11.3.1. MQTT interface for reporting alarms

The MQTT topic for reporting alarms from sensors, actuators, tractors, ISOBUS systems and the DSS, is provided below:

A. REPORTING ALARMS SENT BY A DEVICE/DSS TO THE MIDDLEWARE

Proposed topic structure:

`afc/[scenario]/[service]/[type_of_device]/[entityName]/alarm`

where:

- [scenario] must be replaced by the scenario name defined at configuration time by the System Configuration of the FMS
- [service] represents an application domain, for example water management or environmental observations
- [type_of_device]: is a field used to identify if the alarm has been generated by a sensor, actuator, tractor, ISOBUS system or the DSS
- [entityName] must be replaced by the entityName of the actuator defined at configuration time by the System Configuration

Proposed data model:

#/definitions/Alarm

```
{
        "alarmCode":"GREENHOUSE_IRRIGATION_FLOW",
        "message": “irrigation valve is open but no water flow detected",
        "resourceId":"este_irrigation_valve_0"
}
```

Where:

- alarmCode should be one of the alarm codes defined in AFarCloud ontology
- message is a free text describing the alarm. It will be used to inform the operator.
- resourceId is the entityName of the device that generated the alarm

6.11.3.2. REST interface for reporting alarms

The following REST service is available for reporting alarms from REST devices to the Alarm Processing & Reporter.

POST **/alarm** report an alarm from a REST device
Parameters: name: body schema: #/definitions/Alarm (same as above)
Responses: 200: "Successful operation"

405: "Invalid input"

6.11.3.3. DDS interface with UAVs/UGVs

The DDS topic available for reporting alarms from UGVs and UAVs is *alarm,* associated to the data model defined in the IDL file Alarm. All information related to this DDS interface is defined in D2.6 Semantic Middleware.

6.12. Environment Reporter

6.12.1. Description

The Environment Reporter (ER) manages data provided by sensors in the AFarCloud hardware layer (e.g., embedded in standalone devices, devices on semi-autonomous ground vehicles or on aerial vehicles) arriving through its MQTT, REST or DDS interfaces. The core function of the ER is to validate and report any data or information related to environment, allowing it to be stored in AFarCloud repositories (in the cloud). Such data can be provided directly by the sensors, either in a complete or simplified format, as raw or effective data. The data received by the ER can also be pre-processed data, fused data or even extracted information (on environment, crops and livestock) provided by the AFarCloud components responsible for such operations, respectively. In the two former cases, such components have been previously fed with raw or incomplete effective data by the ER, while in the latter case, the component in question requests stored data, to the Data Query, through queries. The ER checks basic aspects of the received data, such as their format against the defined schemas, as well as validating them to be within to the expected value ranges, respectively, considering the specifications of the sensor in question. These operations can occur, in the former example, before forwarding such data to the Data Pre-Processor (DPP) in the cloud or to the Data Fusion (DF) components, as well as, in the latter example, before such data or information are eventually sent for storage to AFarCloud repositories, either during missions (online), or pre- or post-mission (offline). Before dispatching received data, the ER verifies if it is raw or incomplete and in such cases it forwards that data for pre-processing or for data fusion, respectively, e.g. in case of missing georeferenced metadata associated to sensors measurements, in the latter case, or for appropriate translation into effective or readable data, in the former case.

Immediately before sending complete effective data (if necessary parsed into individual observed properties to the Data Query component) to be appropriately stored in the AFarCloud repositories, the ER validates the data value against the respective sensor ranges, which the ER retrieves by

querying the Asset Registry component. In case of invalid data values, or other anomalies, the ER can report it to the Alarm Processing and Reporter (APR), which should follow the predefined policies to address any of such eventualities.

Summarizing, the ER parses, dispatches and validates sensor data, as well as it can report regarding their validity, eventually also time and georeference wise. These reports, notifications or alarms can be used by the APR, the MMT or the DSS, in the Farm Management System, to allow awareness of any sensor failure(s) and react upon it if necessary. Moreover, the ER validates and forwards extracted environment-based information or knowledge associated to livestock and crops, to the appropriate AFarCloud repositories, to be exploited in the cloud, namely by the MMT and the DSS.

6.12.2. Components diagram

The ER component interacts with other components, receiving and providing data, as follows. First, it receives data according to the defined AFarCloud formats, as in D2.6, via DDS Manager in case such sensor data comes from UAVs, or via MQTT or REST in case the data comes from ground vehicles' onboard sensors or from deployed sensors in the farm. Secondly, the ER verifies whether the received data requires pre-processing, either because it is raw data or if data should be subjected to duplicates check and removal, as well as if data requires data fusion because is not complete, and dispatches such data to the DPP or DF respectively. In order to be possible for the ER to assess if data is raw and requires pre-processing, or what are the value ranges of the sensor in question, it needs also to interface with the Asset Registry.

The DPP sends the outcome of pre-processing data either to the DF component if data fusion is also necessary, or directly to the ER otherwise. In other cases, uncorrelated to DPP operation, the DF and the Knowledge Extractor (KE) components send fused or completed data, or extracted information/knowledge, respectively, to the ER for it to finally validate and forward to the Data Query for storage at the appropriate AFarCloud repository. On the other hand, the ER can also report on the received data, through notifications, alerts or alarms to the Alarm Processing and Reporter. In order for both outputs to take place, ER's *postSensorReport* and *envAlarmReport* methods are used, respectively. For querying the Asset Registry regarding active sensors or devices the ER uses its *getSensor* method. Figure 21 presents the components diagram of the ER.

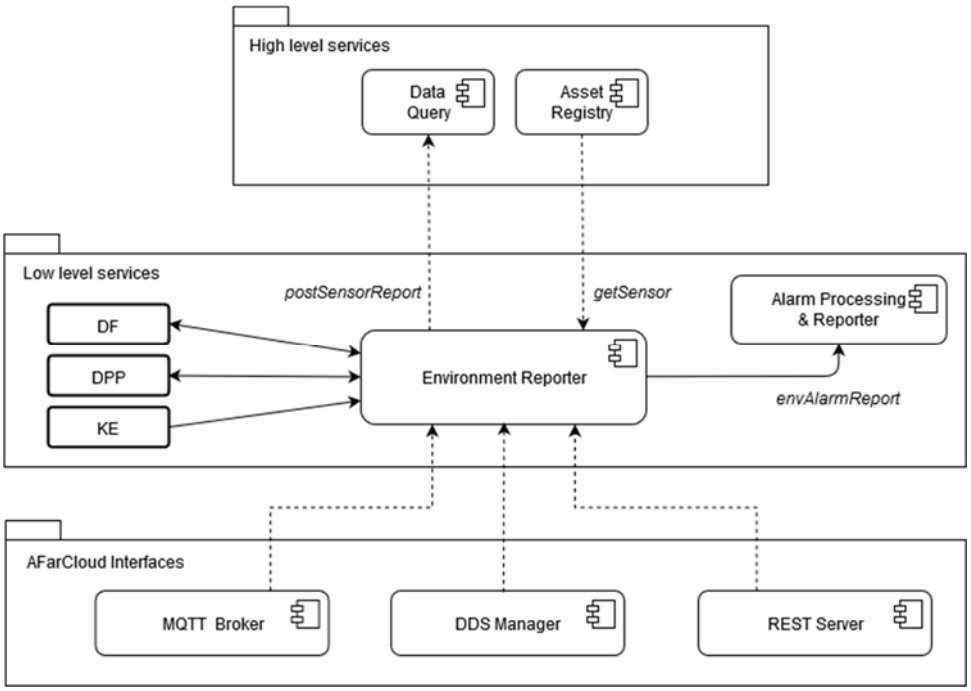

Figure 21: Environment Reporter components diagram

6.12.3. Interfaces

AFarCloud offers three different interfaces to devices connected in the Hardware Layer, so they can send observations taken to the platform:

- MQTT interface to collect observations from sensors (i.e., standalone sensors, WSNs, telemetry from GVs, etc.).
- REST interface to collect observations from sensors, collars and regions detected.
- DDS interface to collect measurements taken by UAVs and UGVs.

The list of MQTT topics and REST services to which the Environment Reporter must subscribe to receive measurements taken by sensors, is provided below.

6.12.3.1. MQTT interface for reporting observations

The MQTT topics for reporting observations from IoT devices, are provided below:

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

Proposed topic structure:

`afc/[scenario]/[service]/[deviceType]/[entityName]/measure`

where:

- [scenario] must be replaced by the scenario name defined at configuration time by the System Configuration of the FMS
- [service] represents an application domain, for example water management or environmental observations
- [deviceType]: is a field used to identify the IoT devices that reports the observation
- [entityName] must be replaced by the entityName of the IoT device defined at configuration time by the System Configuration

Proposed data model:

`ref: "#/definitions/Measurement"` (see D2.6 Semantic Middleware)

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

Proposed topic structure:

`afc/[scenario]/[service]/[deviceType]/[entityName]/measureList`

The meaning of each of the tags in the topic structure is described in the section A.

Proposed data model:

`ref: "#/definitions/SensorDataList"` (see D2.6 Semantic Middleware)

6.12.3.2. REST interface for reporting observations

The following REST services are available for reporting observations from sensors and data from collar devices to the Environment Reporter.

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
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"

6.12.3.3. DDS interface with UAVs/UGVs

The DDS topics available for reporting observations from sensors onboard UAVs and UGVs are the following: *pose, observation, battery, image* and *state_vector,* associated to the data models defined in the following IDL files: Pose, Observation, Battery, Image and Statevector. All information related to this DDS interface is defined in D2.6 Semantic Middleware.

6.13. Data Pre-Processor

The Data Pre-Processor (DPP) component performs essentially raw data translation or transformation into effectively readable data, as well as the assessment of duplicated values in accumulated sensor data. Such processing is done online (during missions), with some latency. To perform its tasks and algorithms, the DPP communicates with the Asset Registry component, which it can query for retrieving key static information or characteristics of the sensor in question, e.g. its accuracy.

Pre-processing is typically done at the edge for complex sensors (before the gateway), but in the case of simpler sensors that don't have pre-processing, their raw data is received from the Environment Reporter (ER), which previously checks the appropriate data format.

This module will receive a simplified JSON (as defined in deliverable D2.6) from the ER with raw values, verify the sensor model and request the Asset Registry for the name of the python script associated with that sensor. It will search for the python script within the module if said script isn't in the module, a request for a download will be made. After running the script, the raw values will be converted to readable values. Having the information about the python script in the Asset Registry will increase the number of different type of sensors that the DPP can pre-process data from.

After the pre-processing, this data will be sent to the Data Fusion (DF) module so it can be converted to a complete JSON, and eventually be forwarded to the ER for validation.

Another functionality of the DPP is when it receives aggregated JSONs (as in D2.6), it will verify if there are duplicates and remove them. After that, since aggregated JSONs should be already complete, the JSON will be sent back to the ER module for final validation.

A more detailed description of the Data Pre-Processor component is available in deliverables D4.1 and D4.2 "Data fusion server".

6.14. Data Fusion

Data Fusion (DF) objective is to aggregate data from multiple sources in one single form, making data more readable, consistent and accurate. Such data fusion could be associating vehicles proprioceptive data to exteroceptive data acquired by embarked environmental sensors, e.g., associating location (from vehicles own GPS chipset/sensor), as metadata, to the environment sensor readings, namely exploiting timestamps and sensor IDs, respectively.

Data fusion is in most cases done after the data has been pre-processed. Since Data Fusion needs to combine information from several sources it's 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, since some sensors already send the complete JSON (as defined in deliverable D2.6) to the ER, in such cases DF is not needed, but in cases that sensors send a simplified JSON with missing fields, data fusion is needed. After the DF receives data in a validated JSON format (potentially with missing fields), the DF will request the Asset Registry for the missing information to fuse the data and obtain a complete JSON ready to be forwarded back to the ER for final validation.

For more complex data fusion this module is able to communicate with data query to aggregate different types of data, from different tables in the AFarCloud repositories, providing a manner to correlate data from different sources.

A more detailed description of the Data Fusion component is available in deliverables D4.1 and D4.2 “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 has been previously stored in the AFarCloud cloud repositories, for analysis and extraction of information or knowledge. The KE needs, therefore, to interface with the Data Query component in order to request or query for relevant data or information from the AFarCloud repositories, to be processed through its KE algorithms.

The outcome of this kind of processes should also be stored at the appropriate cloud repositories allowing it to be eventually used by the MMT and/or the DSS if necessary, standalone or together with data previously pre-processed and/or fused by the DPP and the DF components, respectively. All resulting extracted data, information or knowledge produced by the KE should be validated by the ER.

Further detailed description of the KE is available in deliverables D4.4 and D4.5 on “Livestock and crop quality assessment framework”.

6.16. Image Catalogue

6.16.1. Description

The purpose of the Image Catalogue (previously named Image Data Manager in D2.2) is to store images, image data and catalogue the metadata, so it is possible to find and make use of the images when needed. The Image Catalogue serves as the image layer source for the MMT.

The Image Catalogue is based on the image data management COTS Keystone, a Spacemetric's software data hub for handling, processing and serving geospatial imagery.

A. Catalogue

The catalogue part of the Image Catalogue stores metadata of each uploaded image. The data contains exposure or production time, geometric model (the description of the geometric relation between each pixel and the ground), location in the filesystem, camera and sensor information, tags and labels, quicklook and thumbnail images (small overview images) among other information.

The catalogue can then be searched to find the relevant images. A typical search query in AFarCloud is expected to use “farm”, “mission”, area (polygon or point), sensor type and dates. Examples can be “*find all images from today covering the point with coordinates X, Y*”, “*find all images from farm A from flight mission P2023-05-06 using a thermal sensor*” or “*find the hyperspectral latest images covering the area defined by polygon P and the hyperspectral images for the same area that is seven days older*”

B. Image data storage

The images uploaded to the Image Catalogue can be shown in the MMT map view, as long as they have a known geometry and cover the ground.

The Image Catalogue can be used for long-term storage of images data. A typical application is longitude image comparations, where for instance the latest image over an area is compared with the image acquired a week earlier.

However, the storage quota is never unlimited, and imaging sensors tend to create a large amount of data, so the selection of what to store in the Image Catalogue should be somewhat restrictive. In some cases, it will be better to store the resulting maps rather than the original image data and in other, the best strategy will be to save all images from the mission.

6.16.2. Interfaces

The Image Catalogue will be a Docker-based REST service wrapping the Keystone server. By this wrapping, the Keystone server, and its native APIs and interfaces, will not be exposed to the rest of the AFarCloud system.

The Image Catalogue uses a REST-based API and a WMS service for exposing the image data in, for example, the MMT.

6.16.2.1. MMT image layer provider

The OGC OpenGIS Web Map Service Interface Standard (WMS)[8] is used for retrieving image data from the Image Catalogue. The interface is HTTP based and, to some extent, REST-like. The MMT map view uses WMS calls to request the image data to show as image layers on the map. The calls are often made tile-wise and in the desired resolution.

The Image Catalogue WMS Service makes sure that image data retrieved from the Image Catalogue uses the format and the map projection specified in the WMS request. The uploaded images can be in a variety of file formats and can be either a map-projected image in any map projection supported in the EPSG coordinate list, or using the original image geometry from a range of satellite image sensor geometries, or frame camera geometry.

A WMS service shows the image data in layers, where each layer shows one specified image or several images simultaneously. The group of images can be a number of map-sheets that together cover a larger area, or a stack of images more or less covering each other. To decide what image pixels to show, rules are applied. A rule may be, for instance, to show the latest acquired image data for each pixel. The images that will be included in the layer can be decided by a search query in the catalogue, like "*find all images from today in farm A".*

6.16.2.2. Management of image files and image metadata

The integration between the Middleware components (e.g., Image Processing Platform) and the Image Catalogue is done through a REST based interface.

Typical interactions include searching for images, retrieving metadata for specific image, uploading images to the Image Catalogue, retrieving overviews of the images, full resolution image data and orthorectified (map projected) image data.

[8] https://www.opengeospatial.org/standards/wms

In the following examples the URL *http://ic.afc.com/REST/1.0* will be used, the name *Ogesta* will be used as an ID for the farm and *K1234* will be used as a mission ID. The mission can be a specific flight or a specific data collection event.

Searching for images

A GET request to /{farm}/images will list all the images accessible in the catalogue. These can be further filtered with additional request parameters.

Example, where the service will respond with three of the found images, starting on the thirteenth. The search is limited to a specified area and time slot. The images must start with K1234.

```
http://ic.afc.com/REST/1.0/ogesta/images?
limit=3&offset=12&
geometry=POLYGON%20((30%2010,%2040%2040,%2020%2040,%2010%2020,%2030%2010))&
crs=3006&
startTime=2021-07-18T12:21:43.700Z&stopTime=2021-07-18T13:21:43.800Z&
imageNames=K1234*
```

A typical JSON response can look like this:

```
{
  "images" :
  [ {
    "id" : "I29642",
    "node" : "AFC01",
    "name" : "S2AT33VVF160614L1C160615165550MSBGRN",
    "productType" : "OCN",
    "stripInstance" :
    {
      "platformName" : "SENTINEL 2A",
      "platformVersion" : null
    },
    "created" : 1482063703700,
    "modified" : 1482063704700
  } ],
  "warnings" : [ ]
}
```

Retrieving an overview

It is possible to request for an orthoprojected overview of an image using a GET request. The images are typically returned as georeferenced GeoTIFF images.

Call: `/v1/{farm}/images/{id}/overview`

Example:

```
http://ic.afc.com/REST/1.0/ogesta/images/K1234_0001/overview?size=400&crs=4326
```

Retrieving an image tile

It is possible to request for an image tile of an image using a GET request. Image tiles are typically returned as stretched 3 bands png images. Any part of the image can be retrieved in any scale.

Call: `/v1/{farm}/images/{id}/zoom/{zoomlevel}/{x}/{y}.png`

Example, where a tile with size 512x512 pixels will be cut with the centre coordinate (col: 759, row: 833) in full resolution:

```
http://ic.afc.com/REST/1.0/ogesta/images
K1234_0001/zoom/0/750/833.png?width=512&height=512&interpolation=CC
```

Uploading images to the Image Catalogue

Images are uploaded using a POST request. GeoTIFF files are supported. Frame camera format images are supported if the frame camera format is given. If there is a need for uploading large files there will be an FTP service available. The upload request will then include the FTP directory used.

The upload request call is asynchronous. The service will respond with an upload ID and an image ID that can be used to request the status of the upload.

Call: `/v1/{farm}/upload`

Example, where the image data is posted in the message body:

```
http://ic.afc.com/REST/1.0/ogesta/upload?mission=K1234
```

Response:

```
{
  "uploadprocessid" : "UL3774",
  "imageid" : "ee0ec9e6-f148-443b-9c91-8a85eaef43fd"
}
```

6.17. Image Processing Platform

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 the possible customizations of the IPP, are presented below.

6.17.1. Detection of water stress, weeds and dead plants

The objective of this customization of the IPP is to analyse the quality of vineyard cropping through the generation of georeferenced mosaics provided by UAVs, which are obtained by the pre-processing of a georeferenced images set of the field. The outcome of the IPP will be the regions with water stress, weeds and dead plants in the vineyard.

As a source of the analysis, georeferenced images taken by a UAV flight with a multispectral camera and a thermal camera onboard are used. Moreover, GPS and IMU data are also collected. This data is stored on a memory stick or SD card, and downloaded to the IPP component. 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 Euclidean 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:

- **Water stress:**

Water stress is based on the calculation of the crop water stress index (CWSI) by measuring the difference between the canopy and air temperature[9]. CWSI vary in ranges between 0 and +1 where lower value means well-watered crop and +1 water stress. The CWSI is calculated as:

$$\mathrm{CWSI} = \frac{(T_c - T_a) - (T_c - T_a)_{LL}}{(T_c - T_a)_{UL} - (T_c - T_a)_{LL}}$$

Where $T_c - T_a$ is canopy-air temperature difference; LL is the $T_c - T_a$ values for lower limit, UL is the upper limit of the same. $T_c - T_a$ is a linear function of vapour pressure deficit (VPD). The IPP uses a thermal camera to calculate the temperature of the canopy; $(T_c - T_a)_{LL}$ and $(T_c - T_a)_{UL}$ are estimated based on the VPD and T_a (air temperature) of the vineyard region using an statistical model. Finally, water stress regions are considered as regions with CWSI > 0.7. The outcome of the IPP for water stress will be the location (latitude, longitude) of the boundaries of the region with high value of CWSI.

- **Dead plants and weeds detection:**

These parameters are evaluated with the combination of the vigour index and the training of Artificial Intelligence algorithms. Crop 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. The vigour increases from the minimum value to maximum. The goal of NDVI in this task is the vegetation segmentation, which means, the discrimination between pixels that represent green vegetation and the ground. This process enables to simplify and speed up the subsequent plant detection and classification subtask, providing a mask image. Pixels that belong to vegetation and no vegetation need to be classified between crop and weeds. Dead plants will be detected as gaps in the crop rows when the vigour is maximum. The outcome of the IPP for weeds and dead plant detection will be the location (latitude, longitude) of the boundaries of the regions with dead plants and weeds.

Based on the previous description, the NDVI and CWSI indexes are used as a tool for calculating the parameters of interest and not as an output of the IPP. A data flow diagram for this customization of the IPP is shown below:

[9] J. Bellvert, et.al, “Mapping crop water stress index in a Pinot-noit vineyard: comparing ground measurements with thermal remote sensing imagery from an unmanned aerial vehicle”, Precision agriculture, vol. 15, no. 4, 2013

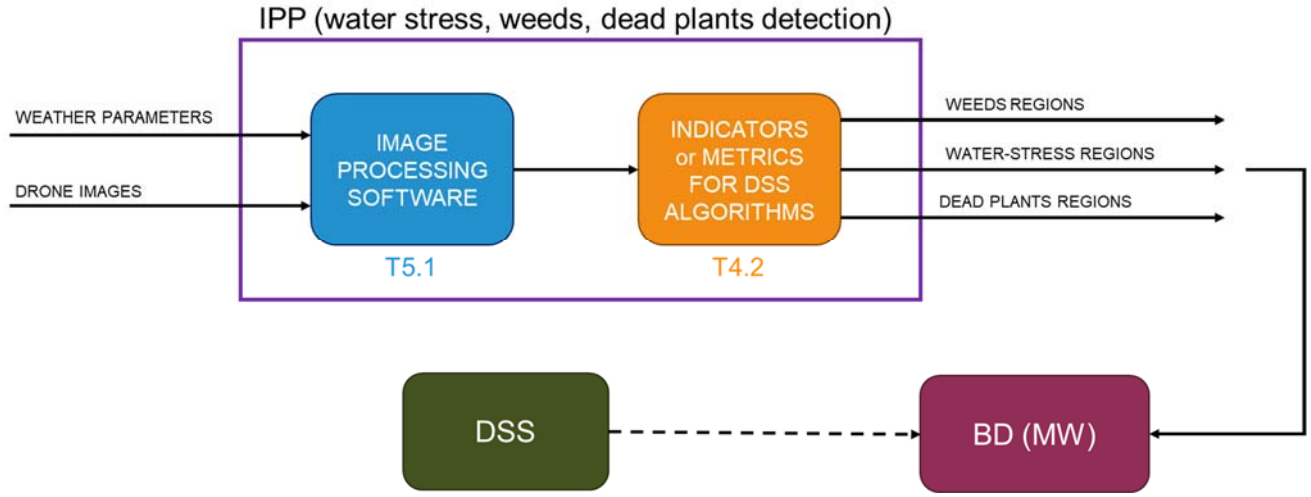

Figure 22: IPP for water stress and weeds & dead plants detection

6.17.2. Surface maps of a terrain

In AFarCloud, the surface mapping of a terrain is done using image processing in conjunction with data from gyroscopes, compasses and GNSS receivers. In order to know the metric dimensions of an image, the algorithm has to relate the altitude of the UAV with the resolution of the image taken and the FOV (field of view) of the camera lens. Associating that information with the data from the UAV pose, it is possible to create a virtual map of the territory by merging the images. That map will be automatically updated according to the UAV's movement. Using thermal and multispectral cameras it's possible to extract information about the crops and animals from the created maps. This topic is further explored in D6.5.

6.17.3. Image Processing Platform for Animal Detection

The objective of this component of the IPP is to develop and train a machine learning algorithm for processing of RGB images of farm animals (cows in this case) that are captured by drones. This machine learning model will be used to support farmers with daily inspection of animals in large grazing areas, and also to locate lost or new-born animals.

For training machine learning models, collection and annotation of RGB images are required. During 2019, about 200 drone RGB images have been collected and used for training purposes. A data processing pipeline has been developed to partially automate the annotation and labeling of the image data. The image data and metadata will be stored in the Image Catalogue. Metadata can be used to enable search for certain images.

The AI image processing is currently performed on local servers deployed at the edge, and the detection results (JSON messages that contain coordinates of each detected object) will be sent to AFarCloud repositories.

6.17.4. Image Processing Platform for Image Catalogue

As an extension to the Image Catalogue, an Image Processing Platform is available to run locally on the farm. The purpose of the platform is to provide a tool to upload common frame format images and GeoTIFF images. The platform can open images locally and can upload them to the Image Catalogue. The graphical interface will be integrated in the MMT.

6.18. ISOBUS Converter

6.18.1. Description

The ISOBUS Converter is placed in the AFarCloud Interfaces of the Semantic Middleware (see Figure 11). The functionalities of this component are:

1. to convert the prescription map that the Mission Manager creates to the standard ISO-XML file format;
2. to convert the log (in an ISO-XML format) that the ISOBUS system creates after the treatment is done, to the AFarCloud format.

The converter can be seen as a stateless component that converts ISO-XMLs files forth and back. During the conversion of the prescription map (functionality 1), when the ISOBUS converter is given the mission by the Mission Manager, it (a) stores the converted ISO-XML file in the AFarCloud user's private repository and (b) sends the ISO-XML file to the mobile MMT, whenever the file is requested from it. The Mobile MMT is then able to connect with the ISOBUS system in an ISOBUS standard way, as depicted in Figure 29.

During log conversion (functionality 2), the Mobile MMT gets the logs of the telemetry from the ISOBUS system and sends them to the ISOBUS Converter, which will (a) store the logs in the AFarCloud user's private repository and (b) convert the logs into AFarCloud mission format (see D2.6) for further processing and/or analysis, as depicted in Figure 31.

6.18.2. Components diagram

The component will be delivered as a docker and its functionalities are the following:

- get the mission file created by the Mission Manager (3 in Figure 29);

- store in the AFarCloud repositories the ISO-XML file of the treatment to be applied (5 in Figure 29);
- send the ISO-XML file to the Mobile MMT (4 in Figure 29);
- get from the Mobile MMT the ISO-XML file of a treatment applied (1 in Figure 31);
- store in the AFarCloud repositories the ISO-XML file containing the log of the treatment and the telemetry that has been converted from the ISO-XML file (2 in Figure 31);
- notify the MMT about new telemetry data available (3 in Figure 31).

6.19. DDS Manager

6.19.1. 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 DDS messages with UAVs and UGVs. By means of this DDS interface, the Middleware is able to: (a) send actions to UAVs or UGVs; (b) collect the status of these actions; (c) collect data from sensors onboard UAVs or UGVs; and (d) collect alarms generated by UAVs or UGVs.

Conceptually, DDS manages a global data centric space, that we call the AFarCloud DDS dataspace. This 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. In AFarCloud, we create a different DDS partition for each of the scenarios to be deployed. For a particular scenario, the DDS Manager publishes actions for UAVs or UGVs in the DDS partition of the scenario and subscribes to data published in this partition by the DDS Proxy of each of the UAVs or UGVs. Different topics are created for the different data types to be managed in the partition.

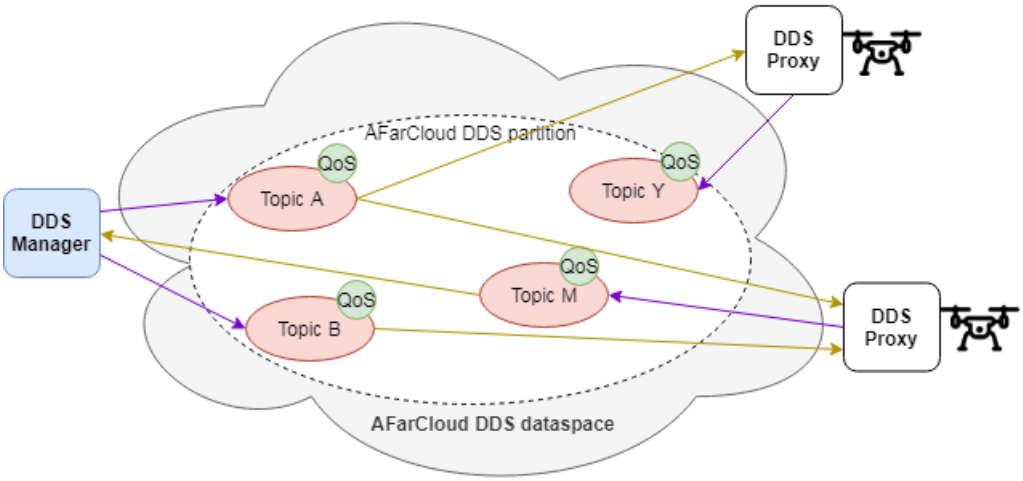

Figure 23: AFarCloud DDS dataspace

6.19.2. Components diagram

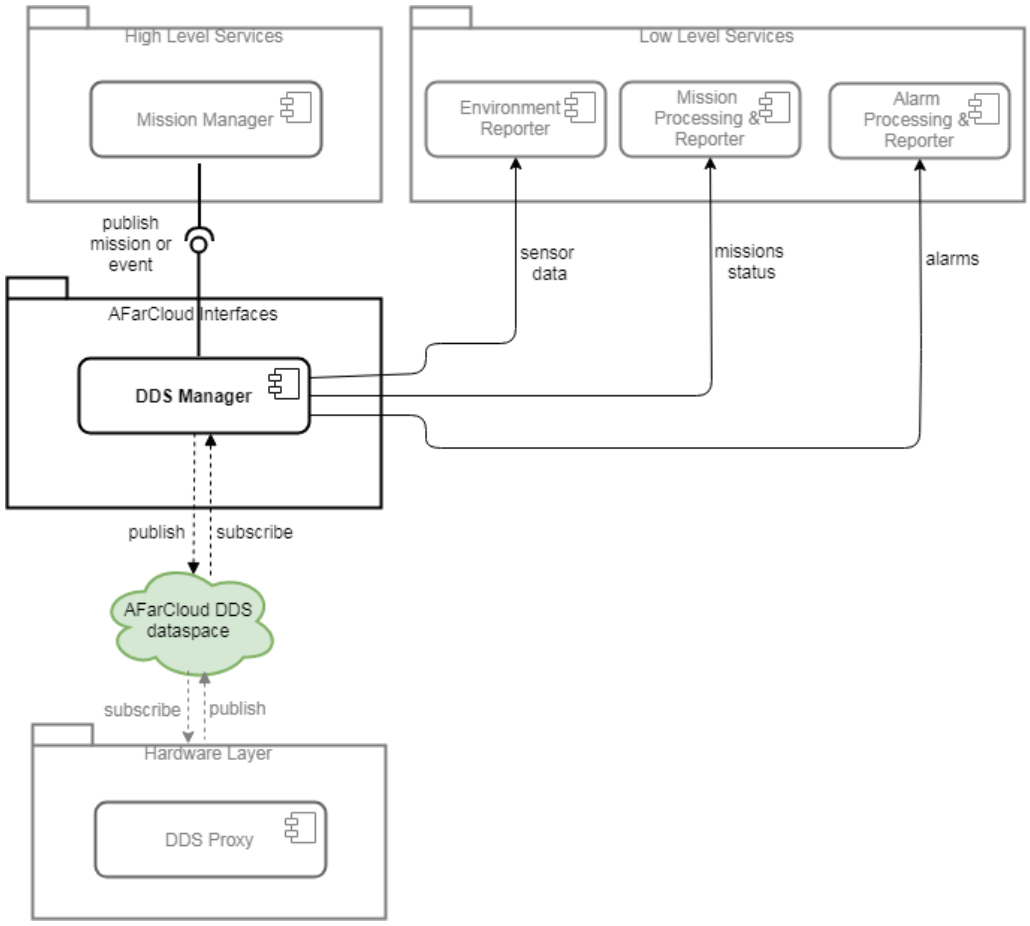

Figure 24: DDS Manager components diagram

6.19.3. Interfaces

6.19.3.1. REST interface with the Mission Manager

Table 8. DDS Manager interface with the Mission Manager

(*public*) **publish** (topic, data)	The Mission Manager must invoke this method to publish data in the DDS partition managed by the DDS Manager, for the specified topic. Data to be published must be compliant with any of the AFarCloud's IDLs (Interface Definition Language) defined in D2.6.

6.19.3.2. DDS interface with UAVs/UGVs

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: `afc_as01`. Each of these partitions manages the same set of AFarCloud topics. The list of DDS topics to be used in the AFarCloud scenarios are the following:

a) Data sent from the DDS Manager to all DDS Proxies:
   - Topic to publish **new missions**: "*mission*"
   - Topic to publish **new events**: "*event*"
   - Topic to publish **get latest state vector**: "*latest_state_vector*"

b) Data received by the DDS Manager from any DDS Proxy:
   - Topic to subscribe to **mission reports**: "*mission_report*"
   - Topic to subscribe to **alarms**: "*alarm*"
   - Topic to subscribe to **observations**: "*observation*"
   - Topic to subscribe to **locations and orientations**: "*pose*"
   - Topic to subscribe to **battery** data: "*battery*"
   - Topic to subscribe to **location, orientation, battery** and **speed** data: "*state_vector*"
   - Topic to subscribe to **image** data: "*image*"

More information related to this DDS interface is defined in D2.6 Semantic Middleware.

Different QoS settings can be applied to each of these topics. 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 → Transient_local

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_local

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 → Persistant

History →KeepAll

DestinationOrder→SourceTimestamp

Liveliness →Fixed timeout

7. The Hardware Layer

7.1. Sensors

Under this category in AFarCloud we consider standalone sensors and groups of dedicated sensors (e.g., WSNs, collars).

A. Functionalities and features of sensors:

- Software updates management: allow operators/administrators to deploy secure firmware updates to sensors.
- Alarms management: sensor error states detection and reporting to the Semantic Middleware.

B. Interfaces of sensors:

- REST interface: communication with the REST services of the MW.
- MQTT interface: communication with the MQTT Broker of the MW.
- Multiprotocol Gateway (MPGW): this device aims to enable sensor data coming from farm and field to be forwarded to the Cloud, performing operations such as: data aggregation, data fusion and protocol translation. Furthermore, a key feature of the Multiprotocol Gateway is to be enabled with more than one connectivity type, which means, from a practical point of view, that the gateway can collect sensor data from a farm with one or more type of communication protocols, which can be both low-medium range (e.g., BLE, IEEE 802.15.4, IEEE 802.11) or long-range (e.g., LoRa), but can forward them to the AFarCloud platform with a different connectivity (e.g., 3G/4G, Wi-Fi). This is of great importance since sensor nodes (namely, devices equipped with sensors) usually do not all support the same type of connectivity but all need to be connected to the Cloud: the multiprotocol gateway solves the problem. Besides the protocol translation and Internet single access point functionality, as anticipated, the multi-protocol gateway can also perform data aggregation and data fusion, in order to reduce and optimize traffic flow to the Cloud.
- Cloud Multiprotocol Gateway (CMPGW): similar to the previous, but with focus in a cloud-based architecture. Hence, beside integrating heterogeneous protocols in AFarCloud, it is also able to scale from a single instance to a Cloud/multiple deployment (i.e., Kubernetes) for higher resiliency and throughput. It also supports Edge Gateways (that aggregate data locally), feeding their data to the single instance/Cloud-Kubernetes deployment. It allows multi tenancy (multiple applications, multiple purposes), and it is configurable through a Web UI. No data

processing or fusion is done by this gateway (although it is possible, if needed). The architecture of the Cloud Multiprotocol Gateway is the following:

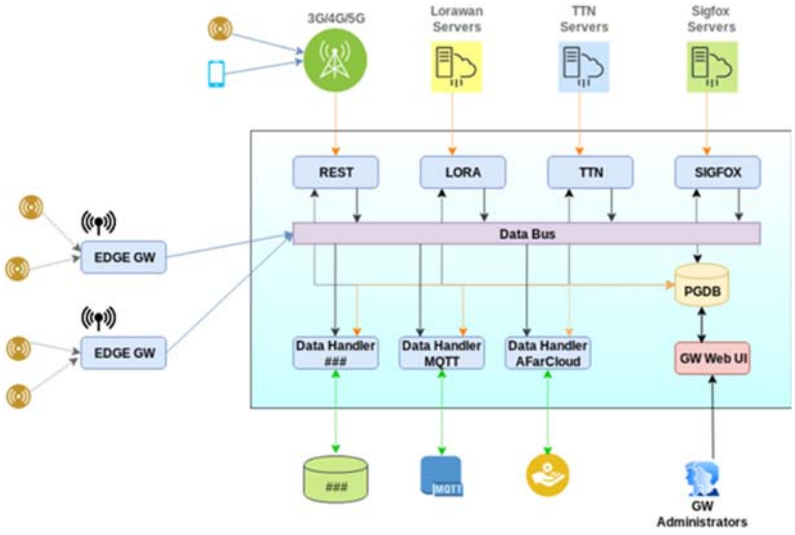

Figure 25: Architecture of the Cloud Multiprotocol Gateway

- Communication between services use a high-performance Data Bus for microservices (Nats.IO)
- Components can be single instance or 1..N instance (K8s Scaling + Nats.IO)
- Input Services: (i) REST: Allows any device to inject data using a POST JSON request. Used by WiFi, 3G, 4G devices or any other REST capable component. The end point requires authentication using an API Key; (ii) TTN: The Things Network connector uses MQTT and TTN API to receive data. TTN Access credentials used are configured through the WEB UI; (iii) Sigfox: Provides a REST end point callable from the Sigfox backend. The end point requires authentication; (iv) LoRa.
- Output Services: (i) AFarCloud data handler: Handles data from any associated input service and delivers it to the AFarCloud services using the REST protocol; (ii) Other output services can also be coded such as MQTT outputs or RDBMS outputs.

C. Available sensors

A summary of the available sensors within the project is shown in Table 9.

Table 9. Overview of sensors used within the project

Sensors type
Multispectral sensor, hyperspectral camera, thermal/IR camera (crops, soil information) – FIXED / Handheld Multispectral sensor, thermal camera (crops, soil information) – MOBILE / UAV NIR (silage information) - Handheld
Soil (electrical detection - humidity, temp, conductivity) – FIXED Soil (spectral detection, humidity, temp) – MOBILE
Environmental: air T, air Humidity, wind, CO2, light intensity, 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)
Sensor to perform real time location – Smart Collar (cow)

The proposed sensors have different degrees of development: some are commercial, some were already developed by the partners during the early stages of AFarCloud, and some will be further developed during the project's duration. Depending on the utilized platform, there are sensors that will be mounted 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).

7.2. Actuators

Under this category in AFarCloud we consider standalone devices responsible for moving and controlling a mechanism or system, e.g., opening a valve.

A. Functionalities and features of actuators:

- Actuator interface management: translation of generic actuator 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 Semantic Middleware)

- Alarms management: actuator error states detection and reporting to the Semantic Middleware.

B. Interfaces of actuators:

- MQTT interface: communication with the MQTT Broker of the MW.

C. Available actuators:

The main actuators to be developed within the project are shown in Table 10.

Table 10. 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
Greenhouse rooftop inflator	AS05 features a greenhouse with inflatable, which will be able to remotely control	On rain/hail weather alert can automatically inflate the roof and protect the greenhouse F17 G7: Reducing water waste and cost in horticulture w/Automatic actuation on rooftop (open, close)
Greenhouse irrigation system	The AS05 greenhouse is equipped with an irrigation system with flow control. A valve will also be deployed to control the flow remotely (via DSS)	F19 G7: Using actuators, irrigate with correct amount and location

7.3. Unmanned Ground Vehicles

UGVs perform different types of missions 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 an internet connection and forward it to the Middleware; (c) carry out other missions defined by the Farm Management System.

As described in section 3.3.1, all communications related to the management of vehicles able to implement autonomous navigation (like UGVs), go through the DDS interface of the Semantic Middleware.

A. Functionalities and features of UGVs:

- Vehicle interface management: translation of the AFarCloud generic missions to the specific language of the UGV. Translation of the data collected by the UGV to the data format defined in AFarCloud. The vehicle interface functionality is implemented by the DDS Proxy of the UGV, that provides a DDS compatible communication channel with the Semantic Middleware. DDS messages are based on the data format for UAVs defined in D2.6 Semantic Middleware.
- Alarms management: UGV error states detection and reporting to the Semantic Middleware.

B. Interfaces of UGVs:

- On-board DDS Proxy: interface with the Semantic Middleware.

7.4. ISOBUS tractors & implements (ISOBUS system)

Under this category in AFarCloud we consider legacy ground vehicles that are not equipped with an autosteer system, (i.e., unable to perform autonomous navigation), but capable of executing special missions in which tasks are carried out autonomously by their ISOBUS compliant implements (e.g., apply fertilizer in certain positions).

As described in section 3.3.2, ISOBUS systems make use of information represented as ISO-XML files. In AFarCloud, any ISOBUS system uses two types of ISO-XML files: (a) the ISO-XML file of a prescription map that the ISOBUS system must load to carry out a treatment mission; (b) the ISO-XML file with the treatment ex-post telemetry that the ISOBUS system generates once the treatment mission is finished.

An ISOBUS system shall feature: (a) One or more ISOBUS Virtual Terminal(s); (b) One or more ISOBUS Task Controller Server(s) version 3 or higher; (c) One or more ISOBUS-compliant mounted/towed/trailed implement(s) that support Task Controller Client functionalities version 3 or higher. These specifications enable “precision farming” capabilities.

A. Functionalities and features of ISOBUS systems:

- Vehicle interface management: load/download of ISO-XML files containing missions/results of missions, load commands/parameters to be applied to the vehicle CAN bus, and send vehicle CAN data to the Semantic Middleware. The data format to be used in each of these cases is defined in D2.6 Semantic Middleware.
- Software updates management: allow operators/administrators to deploy secure firmware updates to tractor controllers.
- Alarms management: ISOBUS system error states detection and reporting to the Semantic Middleware.
- User interface: show information about missions and alarms to the ISOBUS system operator.

B. Interfaces of ISOBUS systems:

- Mobile MMT: acts as an intermediary between the ISOBUS converter of the Semantic Middleware and the ISOBUS system, and is responsible for both loading the ISO-XML file with the prescription map into the ISOBUS system, and for sending the ISO-XML file with the ex-post telemetry to the Semantic Middleware.
- ISOBUS Gateway: reads runtime ISOBUS data of the tractor (specifically the Data Dictionary Identifier (DDI)) and logs them via MQTT in the Semantic Middleware.
- Secure Gateway: connects the vehicle with the Semantic Middleware and establishes a secure and encrypted bi-directional data communication of sensible vehicle data. The Secure Gateway will be connected via CAN interface to the vehicle and transmits data continuously via internet connection.

7.5. Tractors

Under this category in AFarCloud we consider legacy ground vehicles that are not equipped with an autosteer system, (i.e., unable to perform autonomous navigation), but capable of executing/configuring specific commands/parameters of the CAN bus (based on the J1939 standard) of the tractor.

A. Functionalities and features of tractors:

- Vehicle interface management: send vehicle CAN data to the Semantic Middleware. The data format to be used is defined in D2.6 Semantic Middleware.
- Alarms management: tractor error states detection and reporting to the Semantic Middleware.

B. Interfaces of tractors:

- Secure Gateway: see section 7.4.
- IoT Gateway: The IoT Gateway is a ruggedized connectivity device, enabling connections between agricultural vehicles (i.e. tractors, harvesters, etc.) and the cloud using wireless or cellular interfaces. It can be integrated seamlessly into existing electronic architectures of mobile machines, enabling remote access to all data available on the vehicle's CAN busses. It is equipped with standard in-vehicle interfaces, such as Ethernet, USB and CAN. The gateway is provided with a seamless connection to cloud platforms and enables applications that range from simple machine data to monitoring and logging to more advanced functionalities such as prognostics and predictive maintenance. The IoT Gateway is not compatible with ISOBUS, but supports among others Ethernet, CAN, WLAN, LTE, etc.

7.6. Unmanned Aerial Vehicles

UAVs 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.

As described in section 3.3.1, all communications related to the management of vehicles able to implement autonomous navigation (like UAVs), go through the DDS interface of the Semantic Middleware.

A. Functionalities and features of UAVs:

- Vehicle interface management: translation of the AFarCloud generic missions to the specific language of the UAV (it could be based on ROS or any standard solution used by the UAV). Translation of the data collected by the UAV to the data format defined in AFarCloud. This vehicle interface is implemented by the DDS Proxy of the UAV, that provides a DDS compatible communication channel with the Semantic Middleware. DDS messages are based on the data format for UAVs defined in D2.6 Semantic Middleware. Apart from being the link with the Middleware, the DDS Proxy could also provide a DDS link with the DDS Proxy of other UAVs (if a direct UAV to UAV communication is needed).
- Alarms management: UAV error states detection and reporting to the Semantic Middleware.

B. Interfaces of UAVs:

UAVs should implement a DDS Proxy in order to interface with the Semantic Middleware. The DDS Proxy implementation depends 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.

- Interface with the Semantic Middleware for open vehicles: DDS Proxy onboard the UAV.

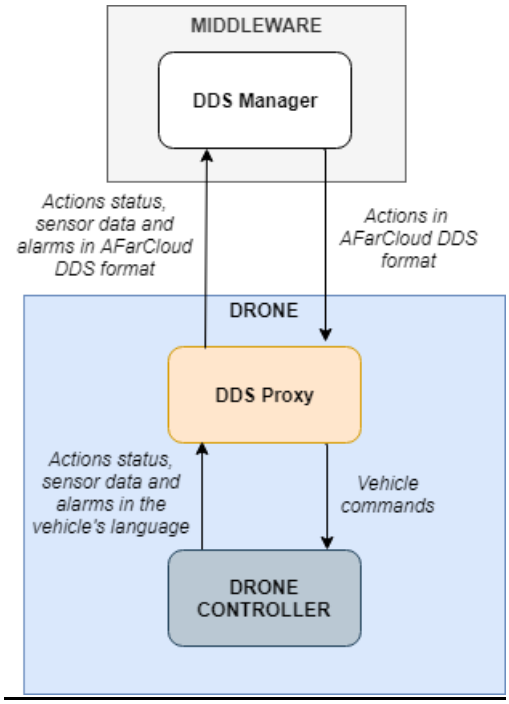

Figure 26: Interface with the Semantic Middleware for open vehicles

- Interface with the Semantic Middleware for proprietary vehicles: DDS Proxy of each vehicle installed in the Ground Control Station (or mobile application) that manages the UAV.

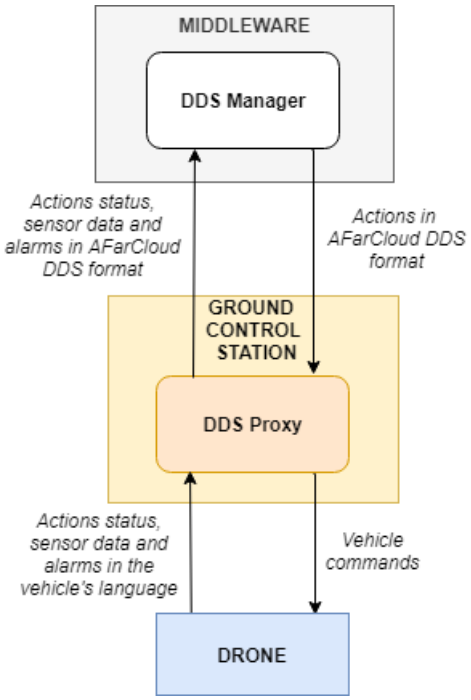

Figure 27: Interface with the Semantic Middleware for proprietary vehicles

8. Data Flow diagrams

8.1. Send a mission to a UAV / UGV

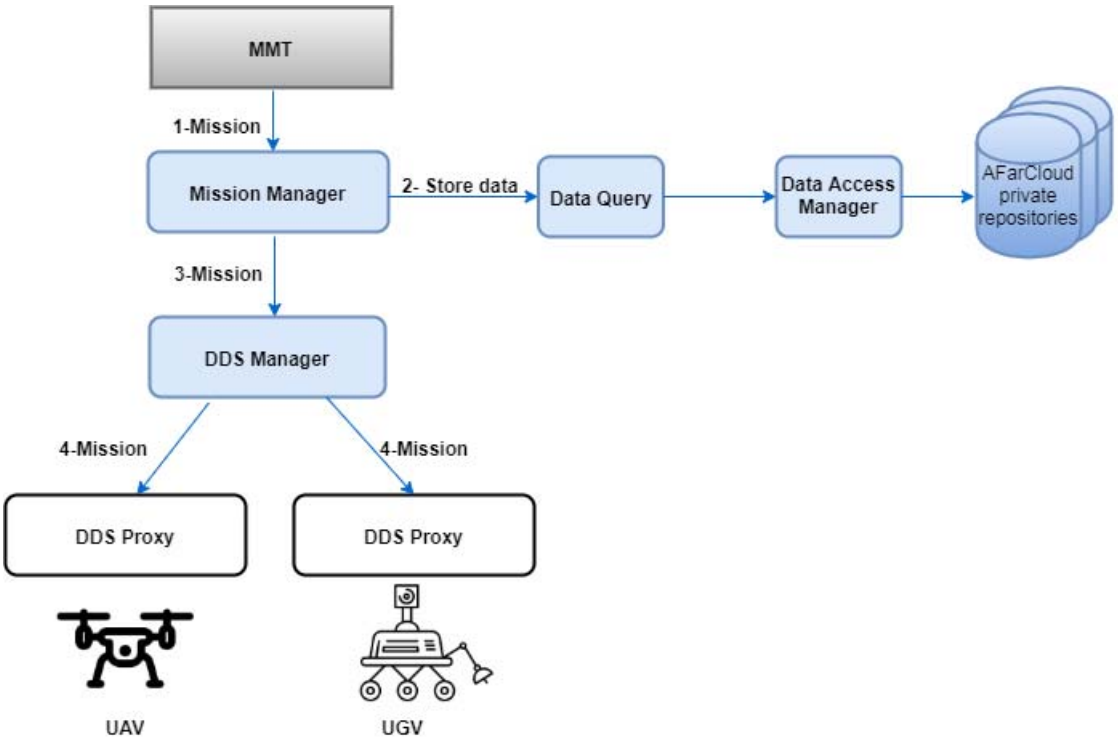

Figure 28: Send a mission to a UAV / UGV

8.2. Send a mission to an ISOBUS system

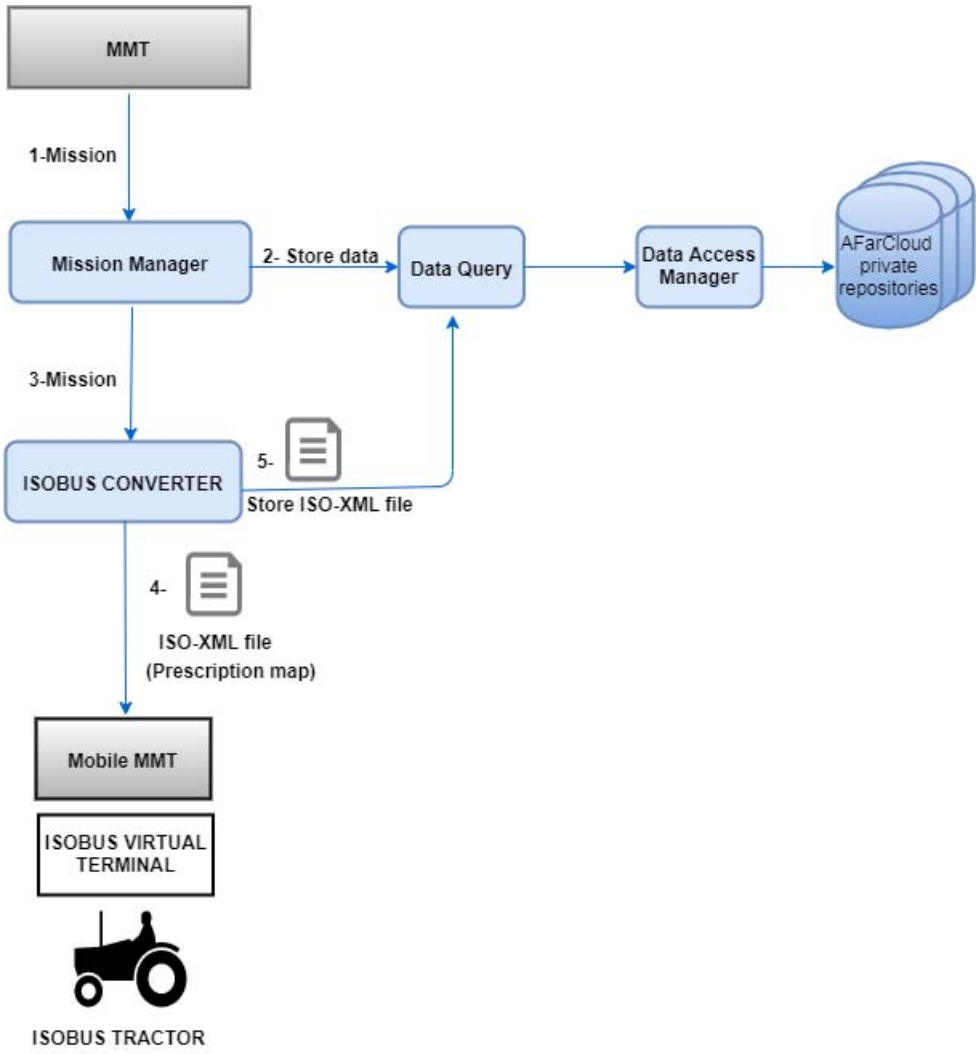

Figure 29: Send a mission to an ISOBUS system

8.3. A UAV / UGV sends data to the MMT

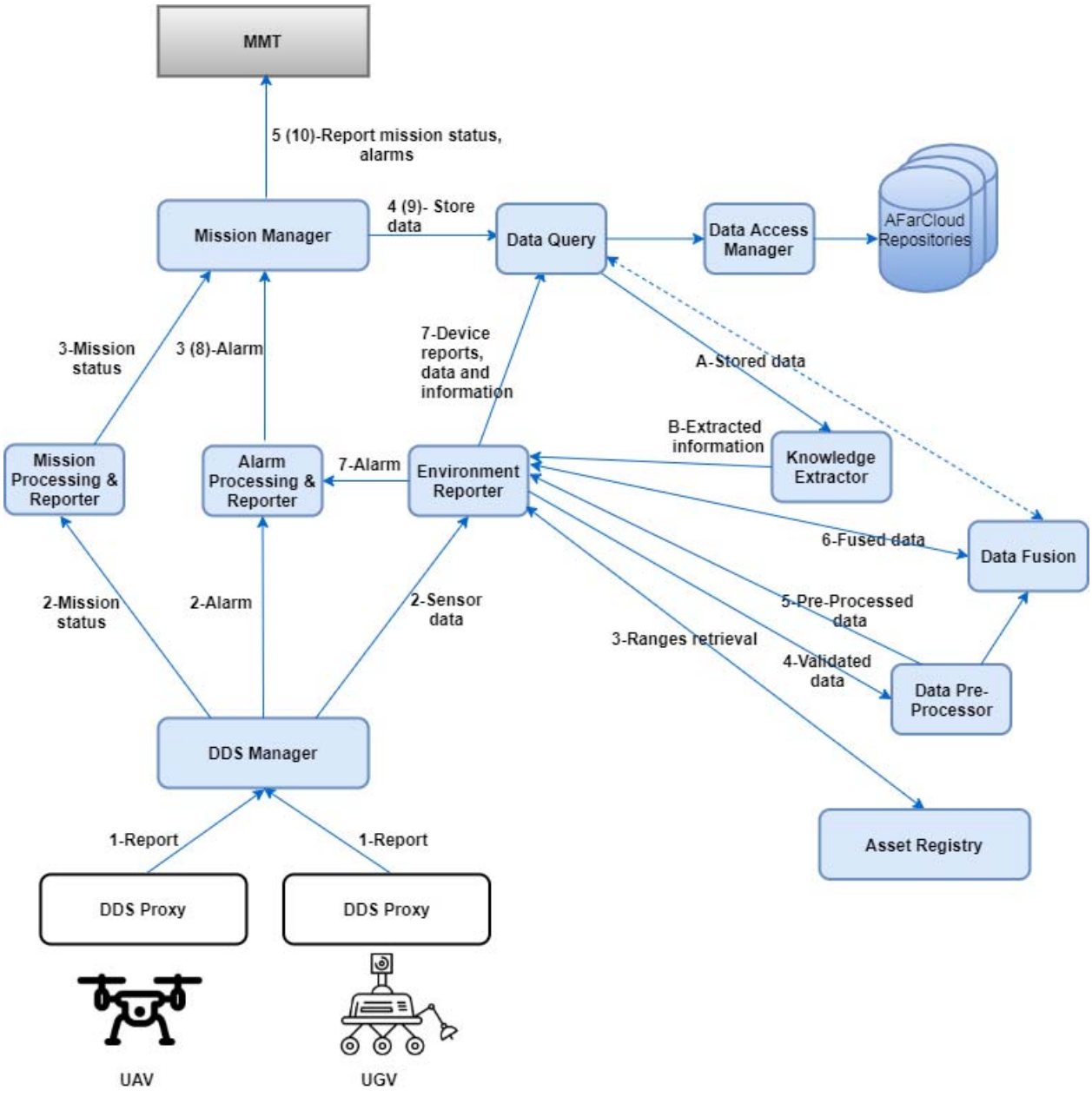

Figure 30: A UAV/UGV sends data to the MMT

8.4. An ISOBUS system sends offline data to the MMT

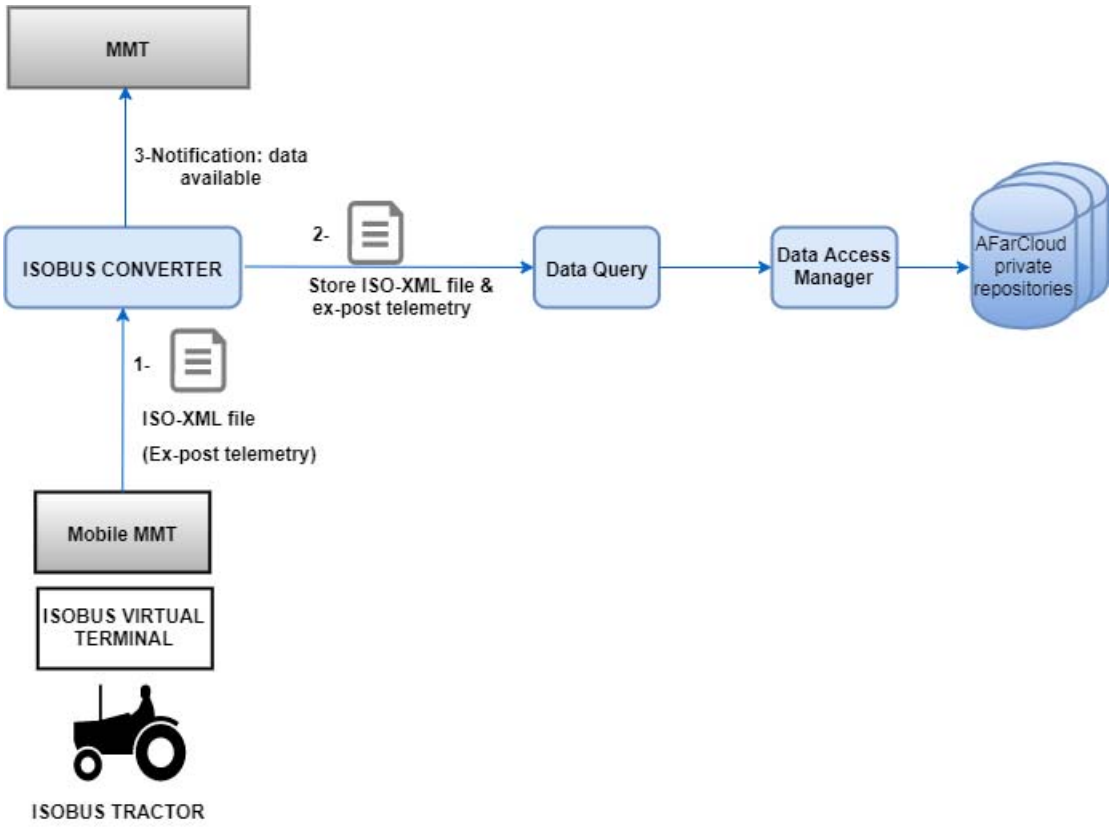

Figure 31: An ISOBUS system sends offline data to the MMT

8.5. A tractor / ISOBUS system sends real time data to the MMT

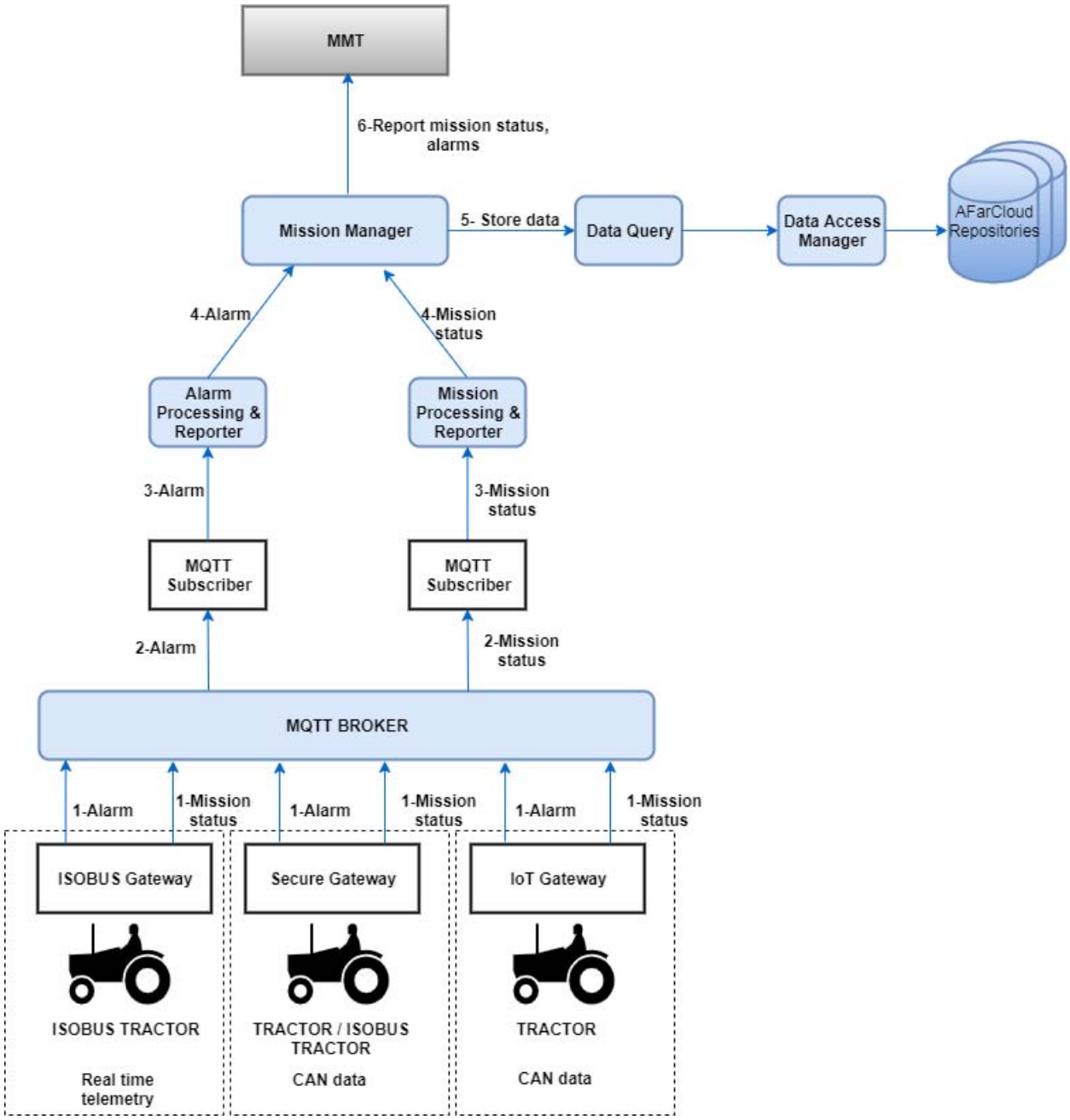

Figure 32: A tractor / ISOBUS system sends real time data to the MMT

8.6. Command sending to MQTT devices

Next figure represents the case in which a command is sent to an MQTT device. The following situations are covered:

- Sending notifications on firmware updates to sensors and tractor controllers (in case the middleware mediation is necessary for firmware updates);
- Setting sampling rates to sensors;
- Sending commands to actuators.

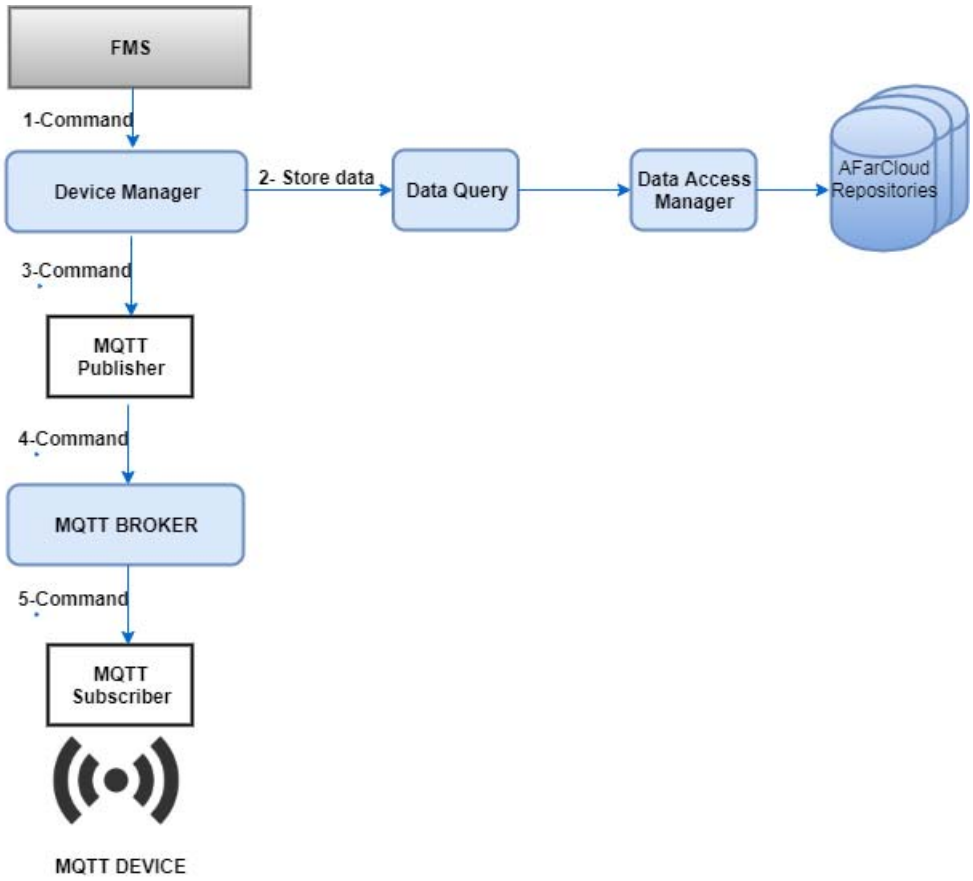

Figure 33: Command sending to MQTT devices

8.7. Devices send data to AFarCloud

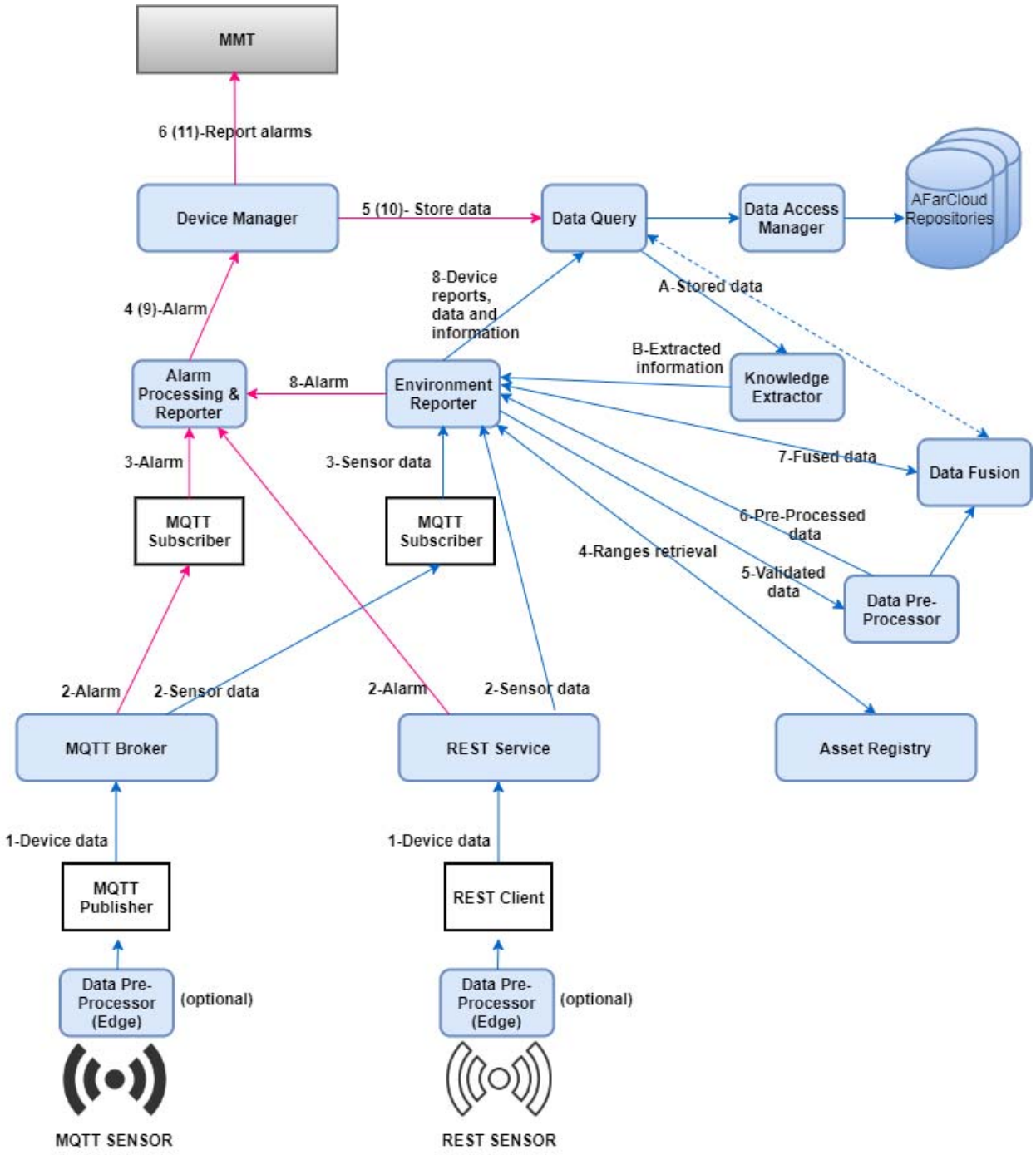

Figure 34: Devices send data to AFarCloud

8.8. Register a new asset in AFarCloud

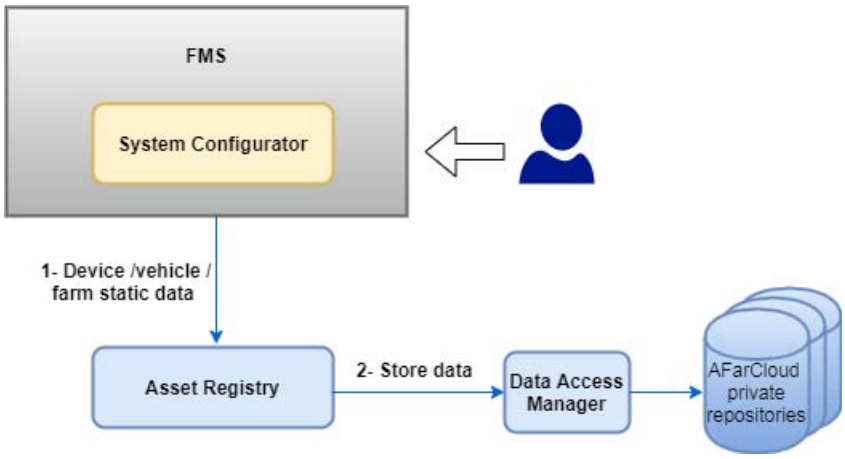

Figure 35: Register a new asset (e.g. collar, sensor or vehicle) in AFarCloud

8.9. Food traceability using blockchain

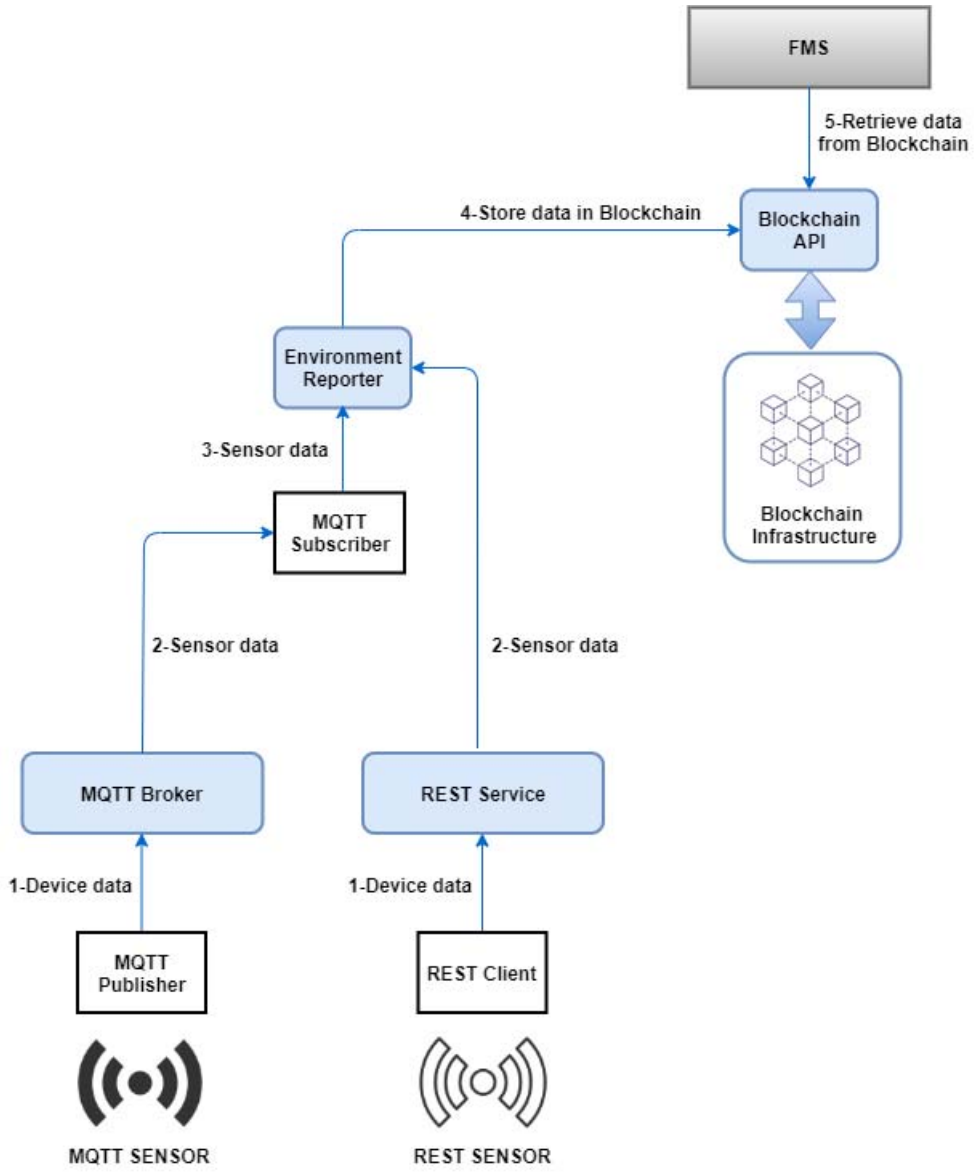

Figure 36: Food traceability using Blockchain

9. Cyber-security in AFarCloud

9.1. Cyber-security assessment

There is currently a need to define cyber-security guidelines in the European Union for modern agriculture (Agriculture 4.0), similar to those developed for industrial control systems. While in the USA the United States Department of Homeland Security (DHS) has carried out research during the last years to identify potential cyber-security vulnerabilities for agriculture, in Europe, however, a similar investigation does not seem to be recognizable. The paper “European Cybersecurity Centres of Expertise Map - Definitions and Taxonomy” from 2018, focuses on many industries to show the risks and the need of monitoring support to ensure cyber-security; but the modern agriculture domain is not included. Even in the EU publication entitled “Study on risk management in EU agriculture”, from Q4 2017, smart farming and cyber-security are not mentioned.

To address this gap, AFarCloud proposes an approach to use the cyber-security standard IEC 62443 as a base to assess the security vulnerabilities in this domain, as this standard describes a security design flow during system development and the assessment of existing systems. A detailed explanation of the standard IEC 62443 can be found in the deliverable D6.2.

IEC 62443 describes the following mandatory steps of a security assessment process:

Table 11. Mandatory steps of a security assessment

1	Identification of the System under Consideration (SuC)	see chapter 9.2
2	High-level cyber-security risk assessment	see chapter 9.3
3	Split in zones and conduits / Detailed cyber-security risk assessment	see chapter 9.4
4	Cyber-security requirements and recommendations	see chapter 9.5

This deliverable covers all these steps. The cyber-security assessment is based on software tools provided by AIT (*GSFlow* and *ThreatGet*), which have been adapted for the AFarCloud agriculture domain by implementing related security standards. Further work contributions and outputs will be documented in the deliverable D7.11 2nd holistic demonstrator report.

9.2. Step 1: SuC Identification

The System under Consideration (SuC) definition identifies the system architecture, the system borders and all access points to the system. In order to perform the cyber security analysis, the overall AFarCloud system architecture has not been considered as one general SuC, but it has been divided into sub SuC's, where each sub SuC has a specific focus on a different security domain.

The defined sub SuC's and the allocated security domains are listed in Table 12.

Table 12: SuC security focus and domains

System under Consideration	Security assessment and analysis focus
SuC-1: UAV – Middleware - MMT	UAV security (UAV communications to cloud)
SuC-2: UGV – Middleware - MMT	UGV security (UGV communications to cloud)
SuC-3: Sensor – Middleware - MMT	Field sensor security (sensor communications to cloud)
SuC-4: Tractor – Middleware - MMT	Tractor security (tractor communications to cloud)
SuC-5: Firmware updates (FMS) – Middleware – Tractor / Sensor	Firmware update security (communications to original equipment manufacturer (OEM)) (in case the middleware mediation is necessary for firmware updates only)

SuC borders definitions: third-party data and legacy systems DDBBs (identified in AFarCloud architecture as Other Data Sources) are external systems not covered by the cyber-security assessment.

SuC external access points: the only external access point identified is the OEM firmware repository.

The architecture, borders and access points of each of the sub SUC's identified, are defined below.

9.2.1. SuC-1: UAV – Middleware – MMT

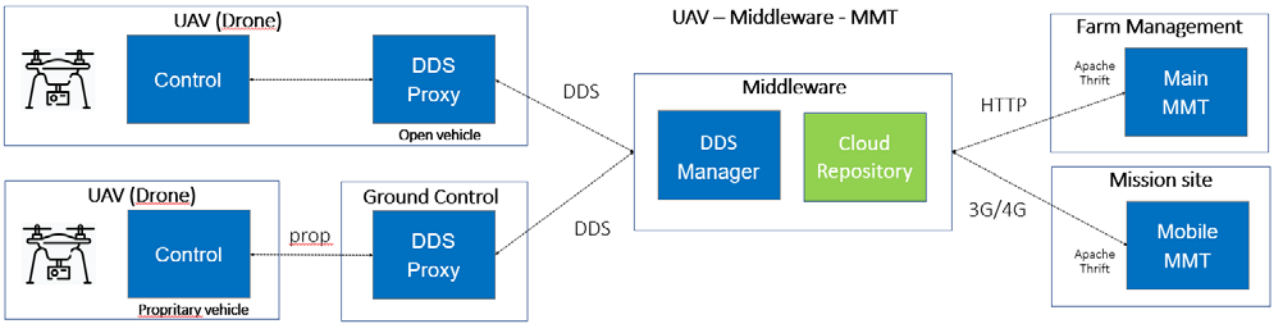

Figure 37: SuC-1: UAV – Middleware – MMT

SUC-1 defines the architecture components and data flows of UAV communications. The mission management (main MMT or/and mobile MMT) defines missions that are sent to the UAV through

DDS. Both “open“ UAVs (which allow the integration of the DDS proxy onboard the vehicle) and “proprietary” UAVs (which communicate with a proprietary protocol over a stationary ground control DDS proxy) are supported. UAV status is stored in the cloud repository for post processing and decision support.

Asset List: UAV (open vehicle: DDS proxy onboard, proprietary vehicle: DDS proxy located in a Ground Control Station), Middleware (DDS Manager, Cloud repository), Farm Management (Main MMT), Mission Side (Mobile MMT).

The cyber-security assessment for this SuC focuses on the UAV.

9.2.2. SuC-2: UGV – Middleware – MMT

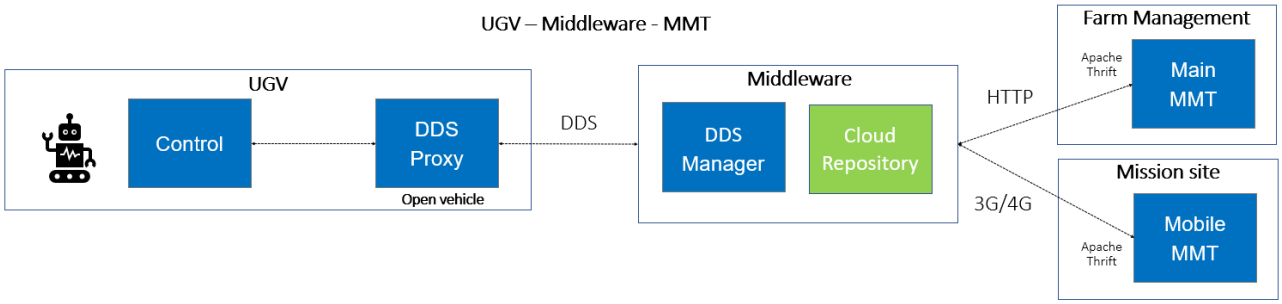

Figure 38: SuC-2: UGV – Middleware – MMT

SUC-2 defines the architecture components and data flows of UGV communications. The UGV is connected via a DDS proxy to the middleware and can receive missions from the mission management (main MMT or/and mobile MMT). UGV status is stored in the cloud repository for post processing and decision support.

Asset List: UGV (DDS proxy onboard), Middleware (DDS Manager, Cloud repository), Farm Management (Main MMT), Mission Side (Mobile MMT).

The cyber-security assessment for this SuC focuses on the UGV.

9.2.3. SuC-3: Sensor – Middleware – MMT

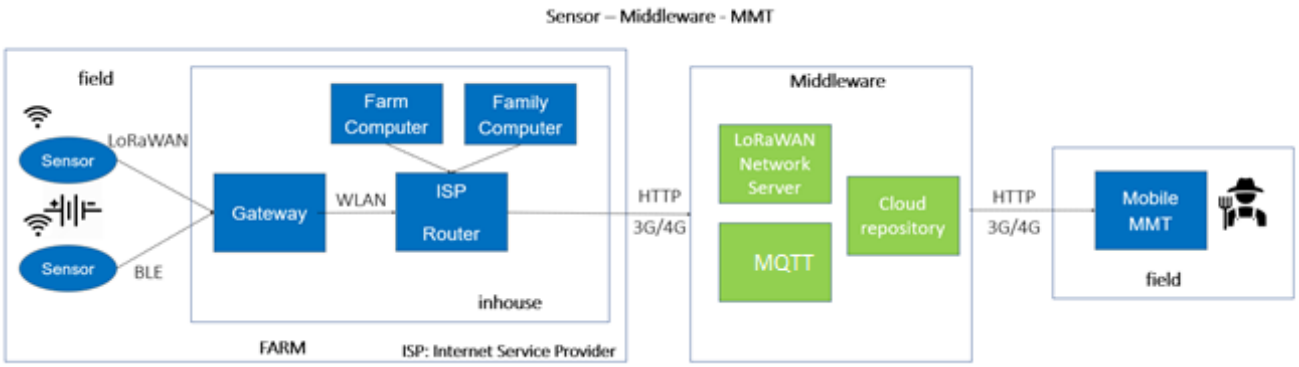

Figure 39: SuC-3: Sensor – Middleware – MMT

SUC-3 defines the architecture components and data flows of sensor communications. It defines also the sensor to the middleware and the middleware to the mobile MMT data communication. It provides a set of sensors in the field zone which transfer sensor data via LoRaWAN or by using BLE (Bluetooth Low Energy) transmission protocol to the cloud repository in the middleware. The farmer on the field side can request post processed environment data via a telecommunication link on a mobile MMT device (smart mobile phone, tablet with 3G/4G functionality).

Asset List: Field (Environment Sensor: LoRaWAN communication protocol, Soil sensor: BLE communication protocol, Mobile MMT: Tablet Computer - Data / Mission Display), Farm house (LoRaWAN / BLE Gateway, Router, Farm Computer, Family Computer), Middleware (LoRaWAN Network server, Cloud repository, DDS).

The cyber-security assessment for this SuC focuses on field sensor security.

9.2.4. SuC-4: Tractor – Middleware - MMT

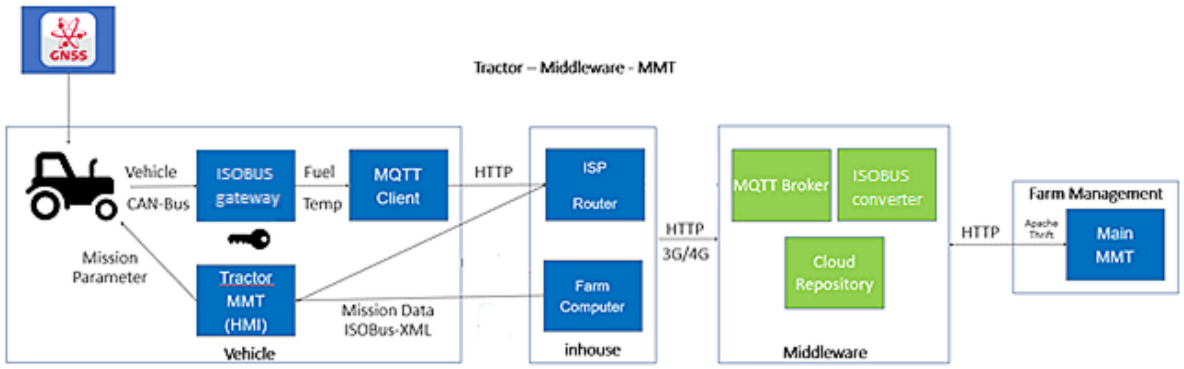

Figure 40: SuC-4: Tractor – Middleware – MMT

SUC-4 defines the architecture components and data flows of tractor communications. It defines the tractor to middleware and the middleware to tractor HMI data communication path for the security assessment. It consists of a tractor on the field zone, which collects continuously operation parameters and geographical position (from GNSS service). These data are sent in packets via the Secure Gateway over 3G to the cloud repository in the middleware. Using the Main MMT, the missions are planned and are prepared for the vehicle operator. The operation data are visualised on an ISOBUS virtual terminal on the tractor MMT (smart phone or a tablet computer). The mission data (an ISOBUS-XML file) is transferred to the tractor through the ISOBUS Converter and the Mobile MMT.

Asset List: Tractor (ISOBUS Gateway, Tractor MMT / HDMI ISOBUS Info Monitor / Mobile MMT, MQTT Client), Edge (ISP Router – Internet Access Point), Middleware (MQTT Broker, ISOBUS Converter, Cloud repository), Farm Management (Main MMT)

The cyber-security assessment for this SuC focuses on tractor security.

9.2.5. SuC-5: FW Update – Middleware – Tractor/Sensor

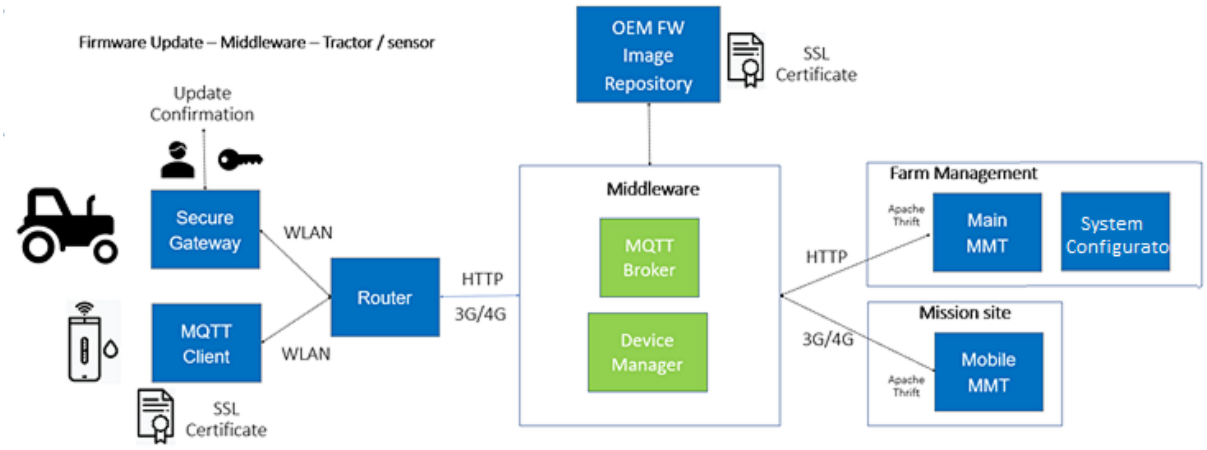

Figure 41: SuC-5: OTA Firmware Update

SUC-5 defines the architecture components and data flows of OTA firmware updates for sensors and for tractor controllers (in case the middleware mediation is necessary for firmware updates). The firmware is provided by the OEM in an external image repository. The Main MMT triggers a new firmware update and instructs the middleware to download the specified firmware from the OEM image repository. The new firmware is transferred to the tractor/sensor. An authorised person must confirm the update process (e.g., with a physical update confirmation key). The correct firmware update for sensors could be confirmed and protected with SSL certificates.

Asset List: Field: (Tractor with Secure Gateway, Sensor with MQTT client, Router), Middleware (MQTT Broker, Device manager), Farm Management (System Configuration, Main MMT), Mission side (Mobile MMT)

The cyber-security assessment for this SuC focuses on firmware update security.

9.3. High-level cyber-security risk assessment

In this step of the cyber-security assessment, the necessary Security Level Target (SL-T) is determined. The SL-T indicates which security level is needed to operate the system in a secure way in the given environment by the determined attack parameters.

In general, IEC 62443 defines three types of security levels, which have to deal with different aspects of the security lifecycle (Source: IEC 62443-3-2):

- SL Target (SL-T) is the desired level of security for the identified SuC, usually determined by a risk assessment with the goal of identifying which security protection is needed to ensure its correct operation.
- SL Achieved (SL-A) is the actual level of security for the identified SuC, which can be measured after a system design is available. The purpose is to verify that the SL-A is identical or higher than the SL-T.
- SL Capability (SL-C) are the security levels that all components or systems can provide when properly configured and integrated without additional compensating countermeasures. SL-C express the necessary cyber security implementation to achieve the determined SL-T.

Table 13 shows an excerpt of the high-level security assessment result table. D6.2 includes a detailed description of the high-level cyber security assessment process and how to estimate the values shown in this table. The derived security level target is based on the potential impact of the security vulnerabilities of the assessed system.

Table 13: High-level security assessment results

SuC	N	Threat ID	Threat	Vulnerability	Threat	Component	Operation-State	AP-Motivation	AP-Elapsed-Time	AP-Expertise	AP-Target-Knowledge	AP-Access	AP-Equipment	AP-Likelihood	Sum	Security-Level Target	Severity-Safety	Severity-Finanz	Severity-Operation	Severity-System	Impact-Safety	Impact-Finanz	Impact-Operation	Impact-System	Sum	Impact-Level
Sensor - MW - MMT	1	SuC-B.1	Field sensor: The sensor data are read by an unauthorised person to get business status data of the farm company.	Espionage	Virus, malware	Embedded Device	Installation, Operation	2	1	2	1	2	2	3	13	SL2	1	2	2	2	2	3	4	4	20	Medium
Sensor - MW - MMT	2	SuC-B.1	Field sensor: The sensor data are read by an unauthorised person to sell the information to a data collector company.	Espionage	Virus, malware	Embedded Device	Installation, Operation	2	1	1	1	1	2	5	13	SL2	1	1	2	2	2	3	4	4	19	Medium
Sensor - MW - MMT	3	SuC-B.1	Field sensor: An unauthorised person manipulates the data of the sensor and forces wrong decision by the farm company.	Destruction	Virus, malware	Embedded Device	Installation, Operation	2	1	2	2	1	3	3	14	SL2	1	2	2	2	2	3	4	4	20	Medium
Sensor - MW - MMT	4	SuC-B.1	Field sensor: The sensor is periodically waked up by a service access to limit the life time of the sensor battery.	Destruction	Virus, malware	Embedded Device	Operation	2	1	1	1	2	2	5	14	SL2	1	2	2	2	2	3	4	4	20	Medium
Sensor - MW - MMT	5	SuC-B.1	Field sensor: Parts of the sensors are mis-used to act as a permanent data transmitter which gain a transmission channel bandwidth overload.	Mis-usage	DDOS attack	Embedded Device	Operation	2	1	2	1	1	3	5	15	SL2	1	1	2	2	2	3	4	4	19	Medium
Sensor - MW - MMT	6	SuC-B.1	Farm computer: The computers, connected to the outside (Internet) are kidnapping to form a botnet for DDOS attacks.	Mis-usage	DDOS attack	Host Device	Operation	2	2	2	2	2	2	3	15	SL2	1	1	2	2	2	3	4	4	19	Medium
Sensor - MW - MMT	7	SuC-B.1	Field sensor: Insert false join packets by emulation - Add confusing data by adding emulated false sensors		Virus, malware	Embedded Device	Operation	2	1	1	1	1	3	4	13	SL2	1	2	2	2	2	3	4	4	20	Medium
Sensor - MW - MMT	8	SuC-B.1	Field sensor: Destroy sensor - Disable proper work of the sensor by destroing the sensor	Destruction	Virus, malware	Embedded Device	Operation	2	2	1	2	2	2	3	14	SL2	1	2	2	2	2	3	4	4	20	Medium
Sensor - MW - MMT	9	SuC-B.1	Field sensor: Relocate sensor - Falsification of the geographic location by moving the sensor to an other place on the field.	Destruction	Virus, malware	Embedded Device	Operation	2	1	2	1	1	3	1	11	SL1	1	1	2	2	2	4	4	4	20	Medium
Sensor - MW - MMT	10	SuC-B.1	Field sensor: Security parameter extraction - Root keys (from which session keys are generated) are generated uniquely for each device during	Mis-usage	Virus, malware	Embedded Device	Installation, Operation	2	2	1	2	2	3	2	14	SL2	1	1	2	3	2	3	4	4	20	Medium
Sensor - MW - MMT	11	SuC-B.1	Field sensor: Device Cloning - Stealing or reuse of key material stored in the sensor	Mis-usage	Virus, malware	Embedded Device	Operation	2	1	2	2	1	2	1	11	SL1	1	2	2	2	2	3	4	4	20	Medium
Sensor - MW - MMT	12	SuC-B.1	Field sensor: Firmware replacement - Manipulate sensor firmware to generate confusing data	Mis-usage	Virus, malware	Embedded Device	Operation	2	1	2	1	1	2	1	10	SL1	1	2	2	2	2	3	4	4	20	Medium
Sensor - MW - MMT	13	SuC-B.1	Field sensor: Network Flooding Attack - Emulate field sensors to generate a flood of sensor data which overload the useabel network	Sabotage	Virus, malware	Embedded Device	Operation	2	1	2	1	2	2	4	14	SL2	1	1	2	2	2	3	4	4	19	Medium
Sensor - MW - MMT	14	SuC-B.1	Field senosr: RF Jamming Attack - The radio signal of the sensors are covering by a radio jamming signal.	Sabotage	Virus, malware	Embedded Device	Operation	2	1	1	2	1	3	6	16	SL2	1	2	2	2	2	3	4	4	20	Medium
Sensor - MW - MMT	15	SuC-B.1	Field sensor: Rogue End-Device Attack - The firmware of an existing sensor is manipulated but using the original key material.	Sabotage	Virus, malware	Embedded Device	Operation	2	1	2	1	1	2	2	11	SL1	1	2	2	2	2	3	4	4	20	Medium
Tractor - MW - MMT	1	SuC-B.2	The tractor vital data are read by an unauthorised person to derive business status of the farm company.	Espionage	Virus, malware	Embedded Device	Installation, Operation	4	2	2	2	3	3	4	20	SL2	1	2	2	2	2	3	4	4	20	Medium
Tractor - MW - MMT	2	SuC-B.2	The tractor geographical location data are read by an unauthorised person to derive position and work status of the farm company.	Espionage	Virus, malware	Embedded Device	Installation, Operation	4	2	1	2	3	2	3	17	SL2	1	1	2	2	2	3	4	4	19	Medium

According to the results from the high-level cyber-security assessment, the security level is determined as **SL-T = 2**.

After the definition of the target security level SL-T the defined SuCs are split up in zones. The zones are connected via conduits. According to the IEC 62443 security standard, each zone and each conduit is assessed with reference to seven Foundational Requirements (FR) for different topics. For each security level, different strong requirements are specified for each FR. Table 14 summarizes the description and the meaning of the different FR's for the security level target SL-T 2:

Table 14: Security Level SL2 Foundational requirements description

Security Level 2 foundational requirements		
FR 1: Access Control	Identify and authenticate AACS (Agriculture Automation Control System) users by mechanisms which protect against intentional unauthorized access by entities using simple means	Passwords and user authentication
FR 2: Use Control	Restrict use of the system or assets according to specified privileges to protect against circumvention by entities using simple means.	Mapping to roles and authorization enforcement
FR 3: System Integrity	Protect the integrity of information in the system against manipulation by someone using simple means.	Session handling, cryptography, recognize changes
FR 4: Data Confidentiality	Prevent the dissemination of information to an entity actively searching for it using simple means.	Encryption
FR 5: Restricted Data Flow	Prevent the intended circumvention of zone and conduit segmentation systems by entities using simple means.	Network segmentation
FR 6: Timely Response to Events	Monitor the operation of the system and respond to incidents when they are discovered by actively collecting forensic evidence from the system	Logs
FR 7: Resource Availability	Ensure that the system operates reliably under normal and abnormal production conditions and prevents denial-of-service situations by entities using simple means.	System backup and recovery

9.4. Zones and conduits split up

IEC 62443 defines security zones as "groups of physical or logical assets that share common security requirements, which have clearly defined borders (physical or logical)". The zones are connected by so called conduits. A conduit includes necessary security measures to: a) control the access to the conduit; b) resist denial of service attacks; and, c) prevent the spreading of any type of attacks. The conduit works as a shield for the succeed zone and protects the integrity and confidentiality of communications.

Each zone implies an objective Security Level, derived from the SL-T. After a security analysis, the components of the zones and conduits must offer a Security Level Capability (SL-C). The SL-C must be equal to the SL-T. If it is less, the gap must be compensated by including additional security technologies. These adaptions of the zones and the conduits are done as long as no gap remains after the detail cyber-security risk analysis. As seen in Table 14, IEC 62443 gives hints how to compensate the gap with extending the cyber-security measures requirements. A conduit can improve, with appropriate security measures (e.g. a firewall), the SL-T of the subsequent zones, also when the SL-C of the desired zone has a lower value.

The following figures show the zone and conduits definitions for the detailed cyber security assessment for the defined SuCs.

SuC-1: UAV – Middleware - MMT

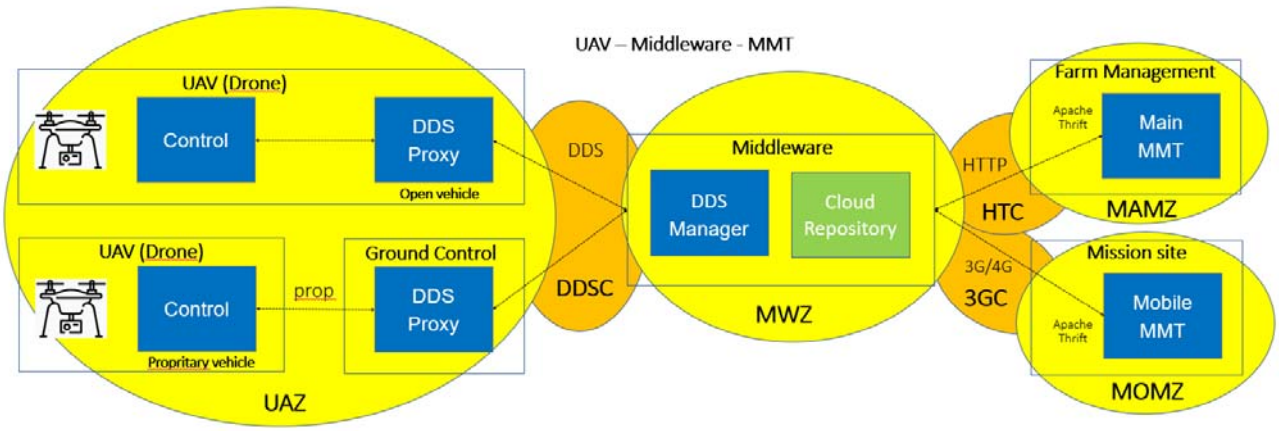

Figure 42: SuC-1: Zones and Conduits split up

Table 15: SuC-1: Zone / Conduit Overview

Zone / Conduit	Description
UAZ	UAV Zone
MWZ	Middleware Zone
MAMZ	Main MMT Zone
MOMZ	Mobile MMT Zone
DDSC	DDS Conduit
HTC	HTTP Conduit
3GC	3G Conduit

SuC-2: UGV – Middleware – MMT

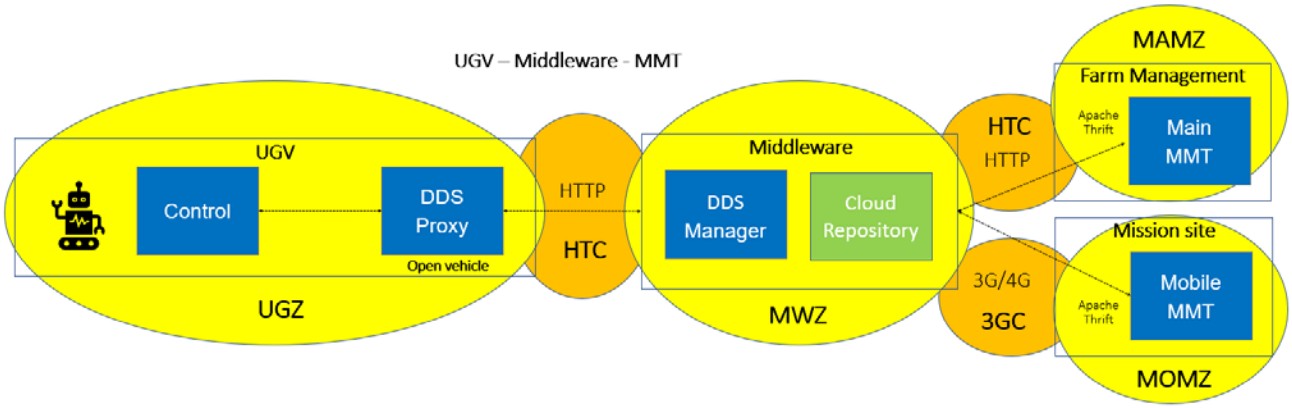

Figure 43: SuC-2: Zones and Conduits split up

Table 16: SuC-2: Zone / Conduit Overview

Zone / Conduit	Description
UGZ	UGV Zone
MWZ	Middleware Zone
MAMZ	Main MMT Zone
MOMZ	Mobile MMT Zone
HTC	HTTP Conduit
3GC	3G Conduit

SuC-3: Sensor – Middleware – MMT

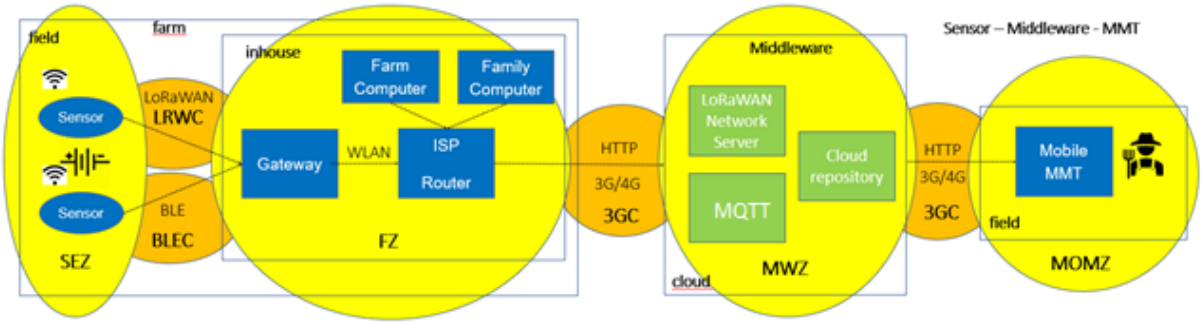

Figure 44: SuC-3: Zones and Conduits split up

Table 17: SuC-3: Zone / Conduit Overview

Zone / Conduit	Description
SEZ	Sensor Zone
FZ	Farm Zone
MWZ	Middleware Zone
MOMZ	Mobile MMT Zone
LRWC	LoRaWAN Conduit
BLEC	Bluetooth Low Energy Conduit
3GC	3G Conduit

SuC-4: Tractor – Middleware – MMT

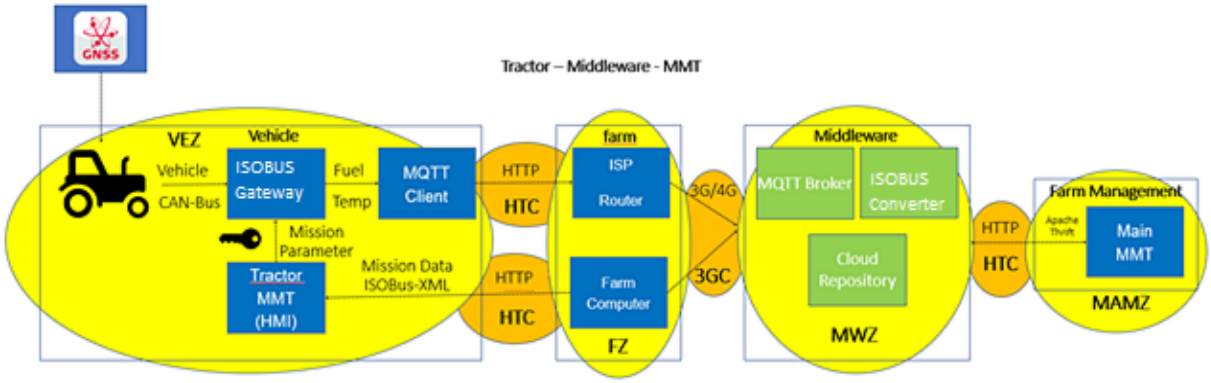

Figure 45: SuC-4: Zones and Conduits split up

Table 18: SuC-4: Zone / Conduit Overview

Zone / Conduit	Description
VEZ	Vehicle Zone
FZ	Farm Zone
MWZ	Middleware Zone
MAMZ	Main MMT Zone
HTC	HTTP Conduit
3GC	3G Conduit

SuC-5: Firmware updates – Middleware – Tractor / Sensor

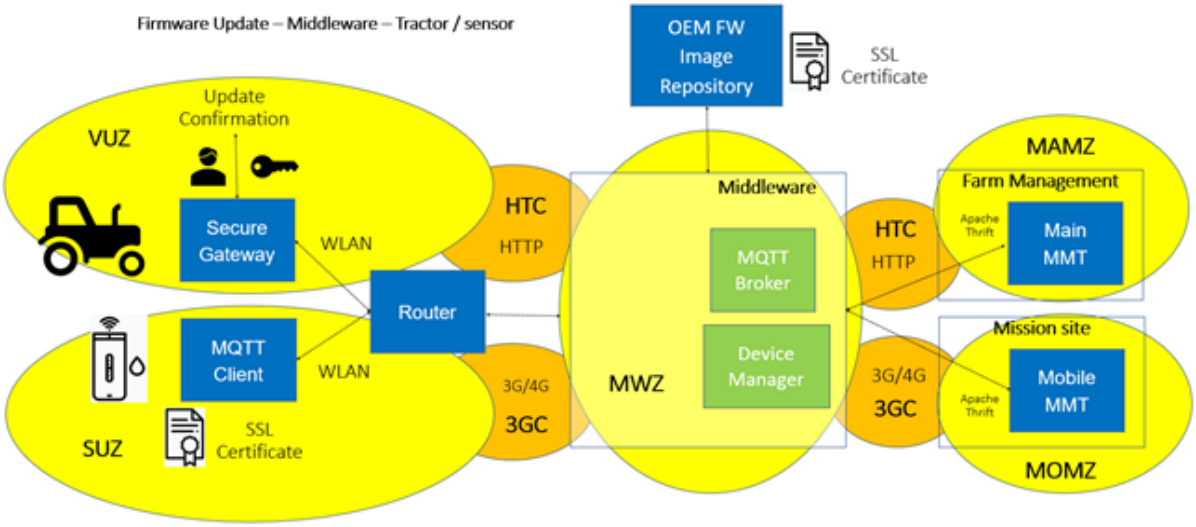

Figure 46: SuC-5: Zones and Conduits split up

Table 19: SuC-5: Zone / Conduit Overview

Zone / Conduit	Description
VUZ	Vehicle Update Zone
SUZ	Sensor Update Zone
MWZ	Middleware Zone
MAMZ	Main MMT Zone
MOMZ	Mobile MMT Zone
HTC	HTTP Conduit
3GC	3G Conduit

9.5. Cyber-security requirements and recommendations

This section describes the cyber-security requirements obtained from the cyber-security risk assessment done in relation to the defined zones and conduits (which are summarized in Table 20).

Table 20: Legend of zones and conduits

Zones	UAZ	UAV Zone
	UGZ	UGV Zone
	VEZ	Vehicle Zone
	SEZ	Sensor Zone
	FZ	Farm Zone
	MWZ	Middleware Zone
	MAMZ	Main MMT Zone
	MOMZ	Mobile MMT Zone
Conduits	LRWC	LoRaWAN Conduit
	BLEC	Bluetooth Conduit
	MQC	MQTT Conduit
	HTC	HTTP Conduit
	3GZ	3G Conduit

For a demonstrator setup the following cyber-security requirement are **recommended**. For a final product development these requirements are **mandatory** to be implemented to fit to the security level 2 (SL2) requirements.

The cyber-security requirement identified are the following:

- **CY-SEC-1 (FR1-AC)**: The system shall provide functions to uniquely identify and authenticate all users, restricting access for unauthorised people.

Vulnerability	A system interface without user identification and authentification allows unstricted access from anyone and anywhere
Countermeasure	Use at least a single factor authentification mechanism with username and password
AFarCloud	FMS access should require a user username/password. MQTT broker should use a username/password protection mechanism
Belongs to the zones	UAZ ● UGZ ● VEZ ● SEZ ○ FZ ● MWZ ● MAMZ ● MOMZ ● LRWC ○ BLEC ○ MQC ○ DDSC ○ HTC ○ 3GZ

- **CY-SEC-2 (FR 1-AC)**: The user credentials shall be restricted in format and length

Vulnerability	A simple structured password can be guessed by an attacker by trying
Countermeasure	Using a password setup check: a minimal length (8), a mix of letters, numbers and special signs
AFarCloud	Password length should be 8, including lowercase & uppercase alphabetic characteres, numbers and symbols
Belongs to the zones	UAZ ● UGZ ● VEZ ● SEZ ○ FZ ● MWZ ● MAMZ ● MOMZ ● LRWC ○ BLEC ○ MQC ○ DDSC ○ HTC ○ 3GZ ○

- **CY-SEC-3 (FR1-AC)**: The number of failed login attempts in a time period (e.g., 24 h) shall be limited to 3 tries.

Vulnerability	The attacker guess the password with a bruce force attack
Countermeasure	Using a login attemp statistic counter
AFarCloud	After a number of wrong authetification tries the system should prohibit further inputs for a longer time
Belongs to the zones	UAZ ● UGZ ● VEZ ● SEZ ○ FZ ● MWZ ● MAMZ ● MOMZ ● LRWC ○ BLEC ○ MQC ○ DDSC ○ HTC ○ 3GZ ○

- **CY-SEC-4 (FR1-AC)**: The user identities and credentials shall be managed by a credential management system and be stored encrypted.

Vulnerability	Unencrypted password can be catched by a sniffer tool
Countermeasure	Storing credentials encrypted. Using a credential management system (password server)
AFarCloud	Credentials should be stored encrypted

Belongs to the zones	UAZ	UGZ	VEZ	SEZ	FZ	MWZ	MAMZ	MOMZ	LRWC	BLEC	MQC	DDSC	HTC	3GZ
	●	●	●	○	●	●	●	●	○	○	○	○	○	○

- **CY-SEC-5 (FR2-UC)**: Missions shall be defined for authorised users only.

Vulnerability	Wrongly defined missions may affect the performance, reliability and safety of the system													
Countermeasure	Using a user access table to control the access permission to define missions													
AFarCloud	The MMT should control the access permission of the users to define missions													
Belongs to the zones	UAZ	UGZ	VEZ	SEZ	FZ	MWZ	MAMZ	MOMZ	LRWC	BLEC	MQC	DDSC	HTC	3GZ
	●	●	●	●	●	●	●	●	●	●	●	●	●	●

- **CY-SEC-6 (FR2-UC)**: Actions which can result in serious impacts shall be supported with a dual approval confirmation.

Vulnerability	A user action with serious impact, without a dual approval confirmation, can lead to serious damage in case of of a fault													
Countermeasure	To perform an action with serious impact, a multiple-step confirmation can help to increase the security and safety													
AFarCloud	A firmware update on a vehicle should follow a two step process: 1) an update should only be triggered by authorised personal (username/password); 2) prior to perform the upgrade on a vehicle, an authorized user should aprove the operation													
Belongs to the zones	UAZ	UGZ	VEZ	SEZ	FZ	MWZ	MAMZ	MOMZ	LRWC	BLEC	MQC	DDSC	HTC	3GZ
	●	●	●	●	○	○	○	○	○	○	○	○	○	○

- **CY-SEC-7 (FR2-UC)**: Safety must be ensured at all times while operating autonomous vehicles.

Vulnerability	A hacker could take control over or shut-down an autonomous vehicle. A malfunction of a vehicle may lead to unsafe situations.													
Countermeasure	The manual mode of operation must be a priority over the autonomous mode of operation													
AFarCloud	The manual operation mode of UAVs/UGVs must be a priority over the autonomous operation mode. Besides, UAVs/UGVs should operate with a command plausibility check mechanism to detect fault or manipulated commands. In case of a malfunction, UAVs/UGVs should enter a safe operation state in a full autonomous manner and the UAV/UGV should perform a full autonomous emergency landing/stop													
Belongs to the zones	UAZ	UGZ	VEZ	SEZ	FZ	MWZ	MAMZ	MOMZ	LRWC	BLEC	MQC	DDSC	HTC	3GZ
	●	●	●	○	○	○	○	○	○	○	○	○	○	○

- **CY-SEC-8 (FR2-UC):** The system shall provide the capability to terminate open sessions automatically after a configurable time period of inactivity.

Vulnerability	Open connections, which are not correctly closed, can be misused by unauthorised people
Countermeasure	Implementation of a session monitoring system, which closes automatically open sessions with no activity after a configurable time period
AFarCloud	FMS applications should implement a session management system
Belongs to the zones	UAZ ● UGZ ● VEZ ● SEZ ● FZ ● MWZ ● MAMZ ● MOMZ ● LRWC ○ BLEC ○ MQC ○ DDSC ○ HTC ○ 3GZ ○

- **CY-SEC-9 (FR3-SI):** The system shall protect the integrity of data.

Vulnerability	System integrity can be compromised by manipulated data, which can lead to unexpected results and interfere with the correct functioning of the system
Countermeasure	Adding additional verification information to the raw data to enable integrity checks before and after data transmission
AFarCloud	Data format correctness should be checked by AFarCloud components before and after transmission. For special cases which imply safety concerns, special measures should be considered: mission files (checksum checking)
Belongs to the zones	UAZ ● UGZ ● VEZ ● SEZ ● FZ ● MWZ ● MAMZ ● MOMZ ● LRWC ○ BLEC ○ MQC ○ DDSC ○ HTC ○ 3GZ ○

- **CY-SEC-10 (FR4-DC)**: The confidentiality of information in data repositories shall be ensured to prevent unauthorized disclosure.

Vulnerability	Data confidentility may be compromised if it can be read by an unauthorised person
Countermeasure	Encrypting sensitive data in repositories
AFarCloud	Sensitive data in repositories should be encrypted (e.g., passwords)
Belongs to the zones	UAZ ● UGZ ● VEZ ● SEZ ○ FZ ● MWZ ● MAMZ ● MOMZ ● LRWC ○ BLEC ○ MQC ○ DDSC ○ HTC ○ 3GZ ○

- **CY-SEC-11 (FR4-DC)**: The confidentiality of information on communication channels shall be ensured to prevent unauthorized disclosure

Vulnerability	Unprotected data on communication links, read by unauthorised people can reduce the data confidentility by uncontrolable distribution
Countermeasure	Encrypting confidential data when sending it over the communication network

AFarCloud	Section 7 in deliverable D2.4 “Communication Protocols” describes how all communication protocols used in AFarCloud could be secured. Besides, MQTT messages (sensors, actuators and tractors) should be encrypted using the TLS protocol, although in some cases, the decrease in battery lifetime of the devices should prevent its usage. DDS messages (UAVs/UGVs), being part of a VPN, are encrypted by default.

Belongs to the zones	UAZ	UGZ	VEZ	SEZ	FZ	MWZ	MAMZ	MOMZ	LRWC	BLEC	MQC	DDSC	HTC	3GZ
	○	○	○	○	○	○	○	○	●	●	●	●	●	●

- **CY-SEC-12 (FR5-RDF)**: The system shall be divided in zones with different security criticalities. The interconnection between zones with different criticalities shall be protected with data flow restriction security measures

Vulnerability	An unprotected access interface can cause an attack propagation through the whole system
Countermeasure	Using data routers with firewalls and data diodes (directional data transfer) isolates zones with different criticality
AFarCloud	Architecture is divided in three zones: farm management (local deployment), middleware (cloud deployment) and field zone. Network isolation should be ensured by separating each of the zones in a different network, where no direct access from one zone to the other is guaranteed.

Belongs to the zones	UAZ	UGZ	VEZ	SEZ	FZ	MWZ	MAMZ	MOMZ	LRWC	BLEC	MQC	DDSC	HTC	3GZ
	●	●	●	●	●	●	●	●	●	●	●	●	●	●

- **CY-SEC-13 (FR6-TRE)**: A monitoring system shall observe continously the system activity to detect attacks and identify unauthorized use of services. The system monitoring traces shall be used to analyse attack paths after an attack detection.

Vulnerability	An attack analysis is made difficult or impossible if there are no periodic system status data available
Countermeasure	Recording of all system accesses and the requested services in a tabular list with a time stamp
AFarCloud	Means for the continuous tracking of all access events to the system, should be implemented to identify and analyze system operation anomalies.

Belongs to the zones	UAZ	UGZ	VEZ	SEZ	FZ	MWZ	MAMZ	MOMZ	LRWC	BLEC	MQC	DDSC	HTC	3GZ
	●	●	●	●	●	●	●	●	●	●	●	●	●	●

- **CY-SEC-14 (FR7-RA)**: The system software shall be backed up accordingly with up-to-date data to provide a recovery from a system software failure or after a misconfiguration or after a software change due to malicious malware.

Vulnerability	A faulty control software, caused by system software failure, incorrect configuration or damage by malware will reduce the correct system operation
Countermeasure	Installing a backup solution
AFarCloud	The software update process should be monitored to perform an error-free firmware update. An intermediate buffer for data backups prior to firmware updates should be deployed. It should be ensured that only after a correct software update the new software version can become active.
Belongs to the zones	UAZ ● UGZ ● VEZ ● SEZ ● FZ ● MWZ ● MAMZ ● MOMZ ● LRWC ● BLEC ● MQC ● DDSC ● HTC ● 3GZ ●

Annex 1. Requirements identified in D2.2

This annex contains the list of requirements identified in deliverable D2.2.

General requirements (GEN):

- 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.
- GEN-2: updated see section 2.2.
- 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.
- GEN-4: The system **shall** use a common data format for the data exchanged between software components.
- 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.
- GEN-6: All relevant entities, vehicles and nodes in the system **shall** use Epoch date/time format.
- 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).
- GEN-8: The system **shall** provide farmers with means to access information coming from other farms and external sources (i.e., meteorological data)
- GEN-9: Ground and airborne vehicles **shall** be used in at least one of the holistic scenarios.
- GEN-10: The system **should** allow the registration of new sensors, actuators and vehicles in AFarCloud before the start of a mission.
- GEN-11: The system **should** support several distinct user roles

Vehicle requirements (VEH):

- 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).

- 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.
- VEH-3: GVs **should** transmit at least the following information to the middleware: position, speed and inner status (fuel/battery left).
- VEH-4: UAVs **should** detect, react to, and report internal faults and error states, at least safety critical.
- VEH-5: GVs **should** detect, react to and report internal faults and error states, at least safety critical.
- VEH-6: Data regarding UAV and satellite monitoring **should** be collected and stored in AFarCloud data repositories.
- VEH-7: GVs **shall** have machine interfaces (e.g., displays) showing information from the AFarCloud system to the operator.
- VEH-8: Tractors **shall** have a power supply (12V) for connecting the machine interface.
- VEH-9: UAVs **should** have a safe state for each operational situation whenever possible. See SEC-18.
- VEH-10: GVs **shall** have a safe state for each operational situation.
- VEH-11: UAVs **shall** switch to safe state whenever possible if a flight safety critical incident is detected.
- VEH-12: GVs **shall** switch to safe state if at least one contributing device of a mission reports a safety critical incident.

HMI requirements (HMI):

- HMI-1: The operator **shall** be given a warning if an abnormal situation occurs. Comment: Could be an alarm and/or a visual indication.
- HMI-2: The operator **shall** see ground and airborne vehicles locations in a map.
- HMI-3: Manual mode **shall** overrule automatic mode.
- HMI-4: The data gathered by the system **shall** be displayed to the operator in a user-friendly way.
- HMI-5: Cow geofencing: the operator **should** receive a warning when a cow gets close to previously established borders.
- 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.

- HMI-7: The operator **should** receive an alert if a cow is being excluded by the rest.
- 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.
- 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.
- 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).
- 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.
- HMI-12: The system **should** define user groups that can access different HMI solutions (or technologies).
- HMI-13: The system **shall** provide a user interface to configure which data is sent from a vehicle to the cloud.
- HMI-14: Sensor data **shall** be visualized in form of dynamic charts.
- HMI-15: User **should** be allowed to define alerts based on observed values or their timeline.
- 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.).
- HMI-17: The user interface **shall** be compatible with the system architecture of the vehicle (e.g., CAN, Ethernet, etc).

Communication requirements (COM):

- COM-1: There **shall** be (at least) a local network to support systems' communications in all demonstrators.
- 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).
- COM-3: Communication between Wireless Sensor Networks (WSN) nodes **should** be possible through multi-hop communications.
- COM-4: The in-vehicle gateway in the tractor **should** provide communication between the cloud and the onboard ISOBUS and CAN bus.

- 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.
- 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.
- COM-7: Communication between battery powered WSN nodes **shall** be realized by energy efficient technologies (with low battery consumption).
- COM-8: Communication link between ruminal probes and concentrator/relay **should** be based on technology that is not hindered by body tissues.
- COM-9: Software updates **should** consider vehicle status (e.g., if an update can be applied without influencing ongoing vehicle operation).
- COM-10: Software updates of UAV **should** not be attempted while the UAV is in use.
- 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
- 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)
- 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).
- 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.
- COM-15: The AFarCloud communication architecture **shall** provide different levels of data reliability (e.g., TCP or UDP transmissions)
- 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)
- 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.
- 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.

- COM-19: The AFarCloud communication architecture **should** allow the creation of remote disconnected networks.
- COM-20: Low delay future communications (like 5G) **should** be kept in mind, at least between the UAVs and the local network servers
- COM-21: Remote data collection **shall** be supported by the deployment of one or more WSN.
- 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).
- 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).

Distributed intelligence requirements (INT):

- INT-1: The system **shall** improve the profitability of crops production in the farm.
- INT-1-1: The system **shall** determine the status of the plant through the monitoring of the soil and climate conditions.
- INT-1-2: The system **shall** be able to determine when to water the crops.
- INT-1-3: The system **shall** be able to predict diseases on crops.
- INT-1-4: The system **shall** be able to detect presence of weeds.
- INT-1-5: The system **should** be able to monitor macro and micronutrient levels.
- INT-1-6: The system **shall** be able to determine the best time to harvest within a reasonable timeframe.
- INT-1-7: The system **shall** help to reduce the usage of pesticides in the crops by 30%.
- INT-2: The system **shall** improve the profitability of livestock and milk production in the farm.
- INT-2-1: The system **shall** help improve animals' welfare w.r.t. standards measures.
- INT-2-2: The system **shall** monitor the location of cows.
- INT-2-3: The system **shall** monitor the activity of cows (steps, and other movements).
- INT-2-4: The system **shall** monitor eating and rumination periods, and shall estimate daily intakes, grazing habits and watering habits.
- INT-2-5: The system **shall** allow calving detection of animals.
- INT-2-6: The system **shall** allow in heat detection of animals.

- INT-2-7: The system **shall** estimate reproduction rates of animals.
- INT-3: In WSN, the system **should** notify the coordinator node when another node leaves the system.
- INT-4: In WSN, the system **shall** provide node heartbeat mechanism.
- INT-5: The system **shall** be designed for efficient distributed mining of large-scale amounts of data. Data exchange should be small or compressed.
- INT-6: The system **shall** be designed to guarantee interoperability between communicating entities in the platform (e.g., using a well-defined API).
- 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).
- 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).
- INT-9: Knowledge extraction algorithms **shall** use pre-processed data, and/or raw data, stored in AFarCloud repositories.
- INT-10: Knowledge extraction algorithms **shall** provide output to AFarCloud repositories, i.e., DBs and/or ontology.
- 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.
- INT-12: The system **shall** anonymize sensitive data before sharing externally.
- 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).
- INT-14: Cooperative data maps used for navigation and awareness **should** be shared and used by the multiple UAVs.

Sensor and WSN requirements (SEN):

- SEN-1: The quality control software based on computer vision in vineyards **should** require the weather parameters (air temperature and vapour pressure deficit).
- SEN-2: updated see section 2.2.

- SEN-3: The system **shall** gather soil information from the WSN including soil moisture, temperature, electrical conductivity and various depth.
- SEN-4: The system **should** provide a specification about the registration of computing nodes in the distributed system.
- SEN-5: The system **should** allow to pre-process on the sensor level.
- SEN-6: The system **should** monitor the conditions of an individual cow in the herd.
- SEN-7: The system **should** backup/cache measured data if connection is temporarily lost.
- SEN-8: updated see section 2.2.
- SEN-9: The system **shall** exchange sensor data by standardized interfaces and protocols.
- SEN-10: The system **shall** provide sensor data in format suitable for integration with systems for herd management.
- SEN-11: The system **should** register metadata for sensors (accuracy, range, calibration, location, maintenance, validation…).

Safety and security requirements (SEC):

- SEC-1: The system architecture **shall** have a security by design principles approach.
- SEC-2: Novel components for GV **shall** be developed along ISO25119.
- SEC-3: The in-vehicle gateway for the tractor **shall** provide mechanisms for authentication to prevent identity spoofing.
- 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.
- 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).
- SEC-6: End-to-end data consistency **shall** be ensured.
- SEC-7: A security risk assessment **shall** be performed for security relevant system components, to define the necessary security.
- SEC-8: The communication between the different actors **shall** be secure by adequate security measures to prevent data spying and data manipulation.
- SEC-9: Only authenticated users **shall** have access to the user interfaces and the system components.
- SEC-10: The control system **shall** provide the capability to identify and authenticate all human users.

- 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.
- SEC-12: The integrity of the control system **should** be ensured to prevent unauthorized manipulation.
- SEC-13: The confidentiality of information on communication channels and in data repositories **should** be ensured to prevent unauthorized disclosure.
- 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.
- 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.
- SEC-16: Software updates **should** be performed periodically when provided from the control system supplier.
- 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.
- SEC-18: Safe states and the shutdown paths for all autonomous vehicles **shall** be defined for all states of a mission.
- 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.
- 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.
- SEC-21: Security mechanism towards Man-In-the-Middle and/or replay attacks **should** be implemented.
- SEC-22: Security mechanisms for end-to-end communications **shall** be implemented, if possible.

Cloud services requirements (CLOUD):

- CLOUD-1: AFarCloud users **shall** be able to provision resources without any interaction with the service provider's staff.

- 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).
- CLOUD-3: AFarCloud platform **shall** support resource pooling, multi-tenancy and dynamic assignment of computing resources based on customer demand
- CLOUD-4: AFarCloud platform **shall** support rapid elasticity so that resources can be quickly provisioned and released, sometimes automatically, based on demand
- CLOUD-5: AFarCloud platform **should** support metering of services compatible with usage and charging models
- CLOUD-6: Updated Heterogeneous Systems Support - Cloud management **shall** leverage the latest hardware, virtualization and software solutions with high-availability and resilience.

Cloud Resources Monitoring requirements (CLDMON):

- 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.
- 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.
- CLDMON-3: The AFarCloud cloud resource monitoring component **shall** monitor the response time of the contracted cloud resources, initially taking also into consideration the 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).
- 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.

- 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.
- CLDMON-7: The component Cloud Resource Monitoring **should** include a User Interface where the following data can be included by the user: endpoint of the cloud service contracted (e.g., IP), availability as per the SLA, maximum response time (in ms), maximum performance. Furthermore, the UI shall also provide a dashboard where the monitored values can be easily seen by the operator of the application.

Development Tool requirements (DEV):

- DEV-1: The development of dependable autonomous systems **shall** be assisted by a design flow management tool.
- DEV-2: The implementations **shall** be validated by proper validation methods.
- DEV-3: The collection of safety evidences **shall** be assisted by AVLs FSM-Tool.
- DEV-4: A set of simulated tests **should** be run prior to any software upgrade.
- DEV-5: The LoRa network **should** be implemented and validated for multi-robot scenarios.
- DEV-6: The multispectral images and passive sensors dataset **should** contain data from fields others than the demonstrators.
- DEV-7: The communication times of passive sensors **should** be properly evaluated.
- DEV-8: The design flow management tool **shall** support the security risk analysis process – according IEC62443
- DEV-9: The design flow management tool **shall** support security level implementation process for the defined security zones and conduits - according IEC62443
- 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.