DEDICAT 6G: Dynamic coverage Extension and Distributed Intelligence for human Centric Applications with assured security, privacy and Trust: from 5G to 6G

Deliverable D2.4
Revised System Architecture

Project Details

Call	H2020-ICT-52-2020
Type of Action	RIA
Project start date	01/01/2021
Duration	36 months
GA No	101016499

Deliverable Details

Deliverable WP:	WP2
Deliverable Task:	Task T2.2 and T2.3
Deliverable Identifier:	DEDICAT6G_D2.4
Deliverable Title:	Revised System Architecture
Editor(s):	François CARREZ (UoS)
Author(s):	Editor + WP2 partners
Reviewer(s):	
Contractual Date of Delivery:	June 30th, 2022
Submission Date:	June 30th, 2022
Dissemination Level:	PU
Status:	Final
Version:	V1.0
File Name:	DEDICAT6G_D2.4_Revised System Architecture_v1.0.docx

Deliverable History

Version	Date	Modification
V1.0	30/06/2022	Final version, submitted to EC through SyGMa

Table of Content

List of Acronyms and Abbreviations

Acronym/Abbreviation	Definition
a.k.a.	Also Known As
AAA	Authentication Authorization Accounting
ACL	Access Control List
AF	Application Function
AGV	Automated Guided Vehicle
AI	Artificial Intelligence
AMC	Adaptive Modulation Coding
AMF	Access Mobility Function
AP	Access Point
API	Application Programming Interface
AR	Augmented Reality
B5G	Beyond 5G
BLE	Bluetooth Low Energy
BLEMAT	Bluetooth Low Energy Micro-location Asset Tracking
BS	Base Station
C&C	Command & Control
CE	Coverage Extension
CEaaS	Coverage Extension as a Service
CEDM	Coverage Extension Decision Making
CU	Control Unit
CV	Context View
D6G	DEDICAT 6G (used in tables only)
DA	Distributed Agents
dBm	decibel mWatt
DC	Design Constraint
DCH	Design Choice
DDoS	Distributed Denial of Service
DIKW	Data-Information-Knowledge-Wisdom
DNS	Domain Name Service
DoA	Description of Action
DoS	Denial of Service
DU	Distributed Unit
DVFS	Dynamic Voltage and Frequency Scaling
E2E	End-to-End
EC	Edge Computing
EE	Execution Environment
EN	Edge Node
eNB	e(volved) NodeB (a.k.a. E-UTRAN NodeB)

e.g.	"exempli gratia" (Latin locution)
FC	Functional Component
FCAPS	Fault/Configuration/Audit/Performance/Security
FG	Functional Group
FLOPS	Floating Operation per Second
FM	Functional Model
FREQ	Functional Requirement
FV	Functional View
GDPR	General Data Protection Regulation
gNB	(next)g(eneration)NodeB *(replaces 4G eNB)*
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GUI	Graphical User Interface
H/M/L	HIGH/MEDIUM/LOW
HMI	Human Machine Interface
HTTP	Hyper-Text Transfer Protocol
H/W	Hardware
i/f	interface
I/O	Input/Output
IAB	Integrated Access and Backhaul
ID	Identifier (mostly in datatypes) or Intelligence Distribution - depending on the context
IP	Internet Protocol
IDaaS	Intelligence Distribution as a Service
IDDM	Intelligence Distribution Decision Making
IEEE	Institute of Electrical and Electronics Engineers
IMS	IP Multimedia Sub-system
IoT	Internet of Things
IoV	Internet of Vehicle
i.e.	"id est" (Latin locution)
IV	Information View
JSON	Java-Script Object Notation
KPI	Key Performance Indicator
LDM	Local Dynamic Map
LiDAR	Light Detection And Ranging
µS	micro-Service
MA	Mobile Assets
MAC	Medium Access Control
MANO	Management Network Orchestration
MAP	Mobile Access Point
MCS	Mission Critical Service
MCV	Manned Connected Car

MCX	Mission Critical {PTT, Video, Data Services}
MEC	Mobile Edge Computing
ML	Machine Learning
MQTT	MQ Telemetry Transport
MSC	Message Sequence Chart
NAS	Non-Access Stratum
NDV	Network Deployment View
NEF	Network Exposure Function
NF	Network Function
NFREQ	Non-Functional Requirement
NFV	Network Function Virtualization
NFV-I	NFV Infrastructure
NFV-O	NFV Orchestrator
NG-RAN	Next Generation RAN
NODM	Network Operation Decision Making
NPN	Non Public Network
NR	(5G) New Radio
NSSAI	Network Slice Selection Assistance Information
NSSF	Network Slice Selection Function
NWDAF	Network Data Analytics Function
OBU	On-Board Unit
OAM	Operation, Administration and Maintenance
OPS	Operation per Second
OS	Operating System
OV	Operation View
P	Priority Level (i.e. H/M/L) used in requirement tables only
p/f	platform
PCF	Policy Control Function
PDCP	Packet Data Convergence Protocol
PDU	Protocol Data Unit
PE	Physical Entity
PEV	Physical Entity View
PF-x	Platform Functional (requirement number)-x
PFCP	Packet Forwarding Control Packet
PHY	Physical layer
PLMN	Public Land Mobile Network
PNF-x	Platform Non-Functional (requirement number)-x
PoP	Point of Presence
POV	Point of View
PPDR	Public Protection and Disaster Relief
PS	Physical System

PST	Privacy, Security & Trust
PTT	Push-To-Talk
QAM	Quadratic Amplitude Modulation
QCI	QoS Class Identifier
QPSK	Quadratic Phase Shifting keying
RAM	Random Access Memory
RAN	Radio Access Network
RAT	Radio Access Technology
RDF	Resource Description Format
Resp.	Respectively
REST	Representation State Transfer
RF	Radio Frequency
RLC	Radio Link Control
RPC	Remote Procedure Call
RRC	Radio Resource Control
RSRP	Reference Signal Received Power
RSS	RDF Site Summary
RSU	Roadside Unit
RU	Radio Unit
S/W	Software
SF-x	Scenario Functional (requirement number)-x
SIMD	Single Instruction Multiple Data
SLA	Service Level Agreement
SLAM	Simultaneous Location And Mapping
SMF	Session Mobility Function
SNF-x	Scenario Non-Functional (requirement number)-x
SNR	Signal Noise Ratio
SoTA	State of The Art
SPP	Security and Privacy Protection
SSL	Secured Socket Layer
STRIDE	Spoofing (identity), Tampering (with data), Repudiation, Information (disclosure), Denial (of service), Elevation (of privileges)
T&C	Terms & Conditions
ToC	Table of Content
TSL	Transport Layer Security
UAV	Unmanned Aerial Vehicle
UC	Use-Case
UDR	Unified Data Repository
UE	User Equipment (e.g., mobile phone)
UF-x	Unified Functional (requirement number)-x
UML	Unified Modelling Language
UNF-x	Unified Non-Functional (requirement number)-x

UPF	User Plane Function
V2X	Vehicle to x
V2V	Vehicle to Vehicle
VEC	Virtual Environment Control
VIM	Virtual Infrastructure Manager
VM	Virtual Machine
VN	Vehicular Node
VNF	Virtual Network Function
VP	Viewpoint
VRU	Vulnerable Road User
vs.	versus (Latin locution)
WMS	Warehouse Management System
w.r.t.	with respect to

List of Figures

List of Tables

Executive Summary

This revised architecture document is the second in a series of three incremental versions. It aims to give a revision and extension of the system architecture which consists of a set of architecture views and perspectives. We tackle here many aspects of the desired DEDICAT 6G architecture, especially functional and non-functional and provide an improved and more precise understanding of which functionalities the platform will provide and which qualities (in terms of performance, trust, reliability - to name just a few) this platform will feature.

Before introducing the architecture, it is worth reminding the main technical objectives of DEDICAT 6G, that are the three following pillars: 1) dynamic coverage extension of existing legacy 5G Network, 2) dynamic intelligence distribution towards (mobile) edge nodes and 3) novel solutions for trust management based on federated learning.

The architecture work follows a precise and logical methodology that encompasses various steps, starting from an extensive requirement engineering process including a few new requirements from D2.3 (included in the new Volere template), followed by a revised functional decomposition that gives a full catalog at the functionalities which must be specified and implemented in order to fulfill those project technical objectives and an updated description of the non-functional properties it must implement.

Other steps involve defining the technical perimeter of the platform (what is in and what is out), elucidating the role of the external entities (human or not) and their interactions with the DEDICAT 6G system and most importantly, defining an extensive list of system use-cases (already introduced graphically and textually in the initial version) featuring precise sequence diagrams.

The Network Deployment view includes now, in addition to the generic view provided in D2.2, a customized network deployment view for each of the four project Use Cases.

Finally, the new Information view provides an extensive list of interfaces for most of the platform components with associated data models. Those interfaces are referred to, in the system use-case sequence diagrams.

The perspective section has been extended with the Performance perspective, in order to complement the existing Privacy, Security and Trust ones.

This second version of the architecture document will be complemented by a third and final iteration (D2.5), the main goal of which, will be to be fully aligned with WP3, 4 and 5 technical results. The overall planned improvements for D2.5 are listed in the conclusion section.

1 Introduction

This document provides a second version of DEDICAT 6G architecture as a set of architecture views and perspectives. A logical pathway leads us from the requirement collection and analysis to the targeted system functional decomposition, which consists of a large set of components interacting with each other and with the existing 5G network in order to implement the main concepts introduced in DEDICAT 6G "Description of Action" document [1]. Those key concepts are 1) dynamic radio coverage extension using a variety of mobile access points and edge nodes, 2) dynamic intelligence distribution relying on those edge nodes and 3) federated learning-based trust management.

This architecture document is intended to drive the work of the other technical work packages. It gives a precise logical blueprint for DEDICAT 6G solution implementations. However, when it comes to project Use Cases realization and demonstration as planned in WP6, only a subset of the whole catalog of functional components described in this document may be eventually implemented.

The remaining of this section introduces the document structure and technical achievements and elucidates the differences between D2.2 and D2.4.

While functional components and system use-cases in functional components introduced in D2.2 were mostly text-based with some accompanying figures, this version of the architecture brings much more detail when it comes to describing the nature of interactions taking place between the components. We have updated the existing set of text-based system use-case descriptions with UML sequence diagrams and added more use-cases (especially some dealing with security) in order to expand the overall portfolio of platform behavior patterns. Those use-cases also go to the detail of the interaction, elucidating the interfaces and data structures used to organize the information within.

An outline of the purpose and outcomes of the following sections of this document follows:

- Section 2 introduces the overall methodology (after Rozanski & Woods [2]) and concepts, which are used along this document. It provides a very clear, thorough, and formal framework for elucidating the views and perspectives that ultimately define the DEDICAT 6G architecture.

  This process is initiated with an extensive requirement engineering process that starts with 1/ collecting requirements taking two different standpoints (scenario holder and platform designer) and 2/ performing a threat analysis. The result of those two steps is embodied into a set of unified functional and non-functional requirements reflecting both standpoints at once. Two different pathways can then be followed:

  - The functional pathway: from understanding a functional requirement some needed functionalities can be identified, it could also be some information-related design choices or even more, some deployment constraints and strategies. Those different inputs are used to build views up (e.g., in that case, the Functional, Information and Network Deployment views respectively);
  - The non-functional pathway: since we are in presence of the characterization of a system quality (non-functional requirement), we cannot impact the views straightaway. On the contrary, we need to elucidate first the corresponding perspective (like Privacy, Trust, Performance, Ethics, Reliability, Scalability etc.) Doing so, we delve into elaborating strategies and tactics to be followed in order to implement the desired system properties. The result of that process is a list of Design Choices that in turn will impact one, if not all, views. Such design choices can be of different nature: e.g., one or several new functional components for the Functional view that are considered as mandatory for implementing the property, or a particular component deployment strategy impacting then the Network Deployment view.

- Section 3 does not bring new content but has been considerably reduced in order to keep the overall document at a reasonable page count. However, the few new scenario requirements identified in D2.3 have been integrated into the existing Volere template;
- Section 4 is fully dedicated to the architecture views, namely (and in order) the Physical-Entity, Context, Functional, Information and Network Deployment views. It is worth noting here that the Instantiation view will be dealt with in the third iteration of this document (D2.5) as its main objective is to show the mapping of concrete implemented components onto the logical components introduced in the Functional view . In a nutshell:
    - The Physical-Entity view (Section 4.1) is about defining the actors involved in DEDICAT 6G (human or not), their roles and their expected interactions with the system. We also emphasis here the pieces of information that are exchanged with DEDICAT 6G or captured by DEDICAT 6G (using sensors for instance) pinpointing potential privacy issues. This view is really focused on that data aspect;
    - The Context view (section 4.2) is about defining the perimeter of DEDICAT 6G elucidating which devices -at large- (a.k.a. physical systems) are considered as part of DEDICAT 6G or laying outside its perimeter. This allows us in particular to identifying those physical systems that can be used to support the dynamic distribution of intelligence towards the (mobile) edge nodes;
    - The Functional view (Section 4.3) proposes a functional model which is a layered group of clusters hosting functional components sharing similar purposes and concerns. Then follows the functional decomposition that gives a comprehensive list of functional components that are meant to implement the functional requirements. Finally, the system use-cases section provides an illustration of how those functional components interact with either each other or with the legacy 5G system, for example to support *Intelligence Distribution as a Service (IDaaS)* and *Coverage Extension as a Service (CEaaS)*;
    - The Information view (Section 4.4) provides the definition of the functional component interfaces alongside their related data structures. It also provides the descriptions of several selected data flows;
    - Finally, the Network Deployment view (Section 4.5) provides the base-line generic deployment of the DEDICAT 6G (including the core part and edge/far edge parts) alongside the 5G legacy system. This section also introduces its customized versions used for the 4 project Use Cases.
- Section 5 completes the DEDICAT 6G architecture with the system perspectives. For each perspective (which represents a particular targeted quality of the system) we propose a set of activities and tactics that lead to a set of design choices, while each design choices impacts one particular view. As a result, each perspective potentially impacts all views or just a few depending on the nature of the property and implemented tactics. In this second iteration we add the Performance perspective to the existing Privacy, Security and Trust ones;
- The final Section 6 provides a conclusion and presents shortly the planned updates for the next and final document iteration. It is followed by the bibliography.

1.1 Delta to previous version

- Section 2 "Methodology" is left unchanged;
- Section 3 "DEDICAT 6G Requirement Engineering" has been drastically shortened as the Volere template synthetizes all that work. However, the new scenario requirements brought in D2.3 have been taken care of, and the unified requirements have been updated;
- Section 4.1 "Physical-Entity view" remains mostly unchanged, bar very few slight adjustments;
- Section 4.2 "Context view" has been reduced: the old UML diagrams (which were revisited in D2.3) have been replaced by a few examples of system use-cases with UML se-

quence diagrams applied to one of our project Use Cases, namely UC2 “Enhanced Experience”;

- Section 4.3 has been extensively updated:
    - The list of FCs and their descriptions has been revised (Section 4.3.2), especially the following ones which are worth reading carefully:
        - Service Orchestrator, Edge Service Orchestrator and Load Balancing FCs in the Service Operation FG (Section 4.3.2.1;
        - EC Policy Registry FC and EN Registry FC: on the one hand their descriptions in Section 4.3.2.2 have been simplified, but on the other hand their associated interfaces and data structures in the new Section 4.4.1.5.4 and Section 4.4.1.5.3 provide much more insight about the way they are meant to operate;
        - SLA Registry FC has been extended functionality-wise (adding of SLA enforcement) and renamed w.r.t. D2.2;
        - µS/FC Registry and µS/FC Repository FCs have been renamed w.r.t. D2.2 and their scope covers both Vertical µServices and platform FCs;
        - µS Discovery & Lookup FC has been discarded. This functionality is now part of µS/FC Registry FC;
        - Swarm Operation FC has been extensively worked on, description-wise in Section 4.3.2.3 and interface-wise in Section 4.4.1.3.1;
        - The other FCs within the Coverage Extension FG have been revised as well, but to a lower extend;
        - All FCs in the pre-existing Decision Making FG have been extensively revised, and two new FCs dealing with UE-{BS/MAP} association have been added;
        - Network Performance Analytic FC has been added in the Analytic FG, Section 4.3.2.5;
        - The four UE and MAP Status Agents and corresponding Awareness FCs have been added in Section 4.3.2.6;
    - The list of system use-cases (Section 4.3.4) has been extended to cover more behavior patterns and includes also in this new iteration some relating to SLA enforcement, security, privacy and trust;
    - The system use-case descriptions include UML sequence diagrams referring to the interfaces as defined in the Information view.
- The Information view in Section 4.4 is brand new. It provides the interfaces of most of the platform FCs and the data structures they are based on. It also includes a few system data flows with focus on context-awareness;
- Section 5 adds the new Performance perspective;
- Annex A: has been removed as the requirements and some additional useful related information can be accessed from the Volere template directly.

1.2 Quick access to the main D2.4 outcomes

In this section (Table 1) the reader can quickly access the main outcomes of the current architecture iteration which are summarized into a table with clickable links to the relevant sections.

Table 1: Summary of project outcomes and quick access links

Category	Outcome	Link
Requirement engineering	DEDICAT 6G threat analysis	Section 3.1
Views	**Physical-Entity view**	Section 4.1
	Inventory of physical-entities (actors and role definition / nature of interactions)	Table 9
	Inventory of information of interest	Table 10

	Context view	Section 4.2
	Perimeter of DEDICAT 6G (inventory of physical systems)	Table 11
	Inventory of FCs at the edge of DEDICAT 6G or outside (includes scenario-specific FCs)	Table 12
	Scenario UML use-cases	Section 4.2.2
	Functional view	Section 4.3
	Functional model	Figure 11, Section 4.3.1
	Functional decomposition (list of FCs per FG)	Section 4.3.2 & 4.3.3
	System use-cases	Section 4.3.4
	Information view	Section 4.4
	Context-awareness related data structures and interfaces	Section 4.4.1.1
	Decision-making related data structures and interfaces	Section 4.4.1.2
	Coverage extension related data structures and interfaces	Section 4.4.1.3
	Service operation related data structures and interfaces	Section 4.4.1.4
	Intelligence distribution related data structures and interfaces	Section 4.4.1.5
	Analytics related data structures and interfaces	Section 4.4.1.6
	Security, privacy and trust related data structures and interfaces	Section 4.4.1.7
	Network Deployment view	Section 4.5
	Baseline NDV	Section 4.5.1
	Smart Warehousing NDV	Section 4.5.2
	Enhanced Experience NDV	Section 4.5.3
	Public Safety NDV	Section 4.5.4
	Smart Highway NDV	Section 4.5.5
Perspectives	Privacy perspective	Section 5.1
	Security perspective	Section 5.2
	Trust perspective	Section 5.3
	Performance perspective	Section 5.4

2 Methodology

2.1 Some elements of Rozanski & Woods terminology

2.1.1 Views

Views are used to described non-overlapping aspects of a concrete system and defined as:

"A view is a representation of one or more structural aspects of an architecture that illustrates how the architecture addresses one or more concerns held by one or more of its stakeholders." [2]

In DEDICAT 6G we will define the following views. Some of them are generic while others are application dependent and therefore bound to our four project Use Cases. Those later ones are marked with an asterisk.

• Physical-Entity view*	(Section 2.6.1)
• Context view*	(Section 2.6.2)
• Functional view	(Section 2.6.3)
• Information view	(Section 2.6.4)
• Network Deployment view *	(Section 2.6.5)
• Instantiation view	(Section 2.6.6 but not covered in D2.4)

2.1.2 Viewpoints

In order to describe a view, architects use viewpoints which aggregates different architectural concepts like for instance data flows, sequence diagrams, data modelling… in order to describe that particular aspect of the system. The definition by the IEEE 1471 standard [6] is:

"A viewpoint is a collection of patterns, templates, and conventions for constructing one type of view. It defines the stakeholders whose concerns are reflected in the viewpoint and the guidelines, principles, and template models for constructing its views. "[7] In DEDICAT 6G we will be using various UML-based viewpoints to describe various aspects of the views, e.g.:

- *Data modelling:* specification of data models using UML object-oriented modelling;
- *Interface modelling:* definition of the various methods that the functional components need implementing for other components to use (staying at a logical level);
- *UML use-cases:* to specify the interactions taking place between the scenario actors (including pieces of equipment outside the perimeter of the DEDICAT 6G platform) and the DEDICAT 6G system it-self;
- *UML sequence diagrams:* sequence diagrams (a.k.a. message sequence charts or interaction diagrams) provide a temporal sequence of interactions occurring between actors and/or functional components (each materialized by a lifeline) during a specific behavior pattern;
- *Data flows:* identifying the flows of data between the different functional components during a specific task;
- *Textual descriptions.*

This list is not exhaustive and additional viewpoints could be used if needed.

2.1.3 Perspectives

*An **architectural perspective** is a collection of activities, tactics, and guidelines that are used to ensure that a system exhibits a particular set of related quality properties that require consideration across a number of the system's architectural views.* [7]

where a quality property is defined as:

*A **quality property** is an externally visible, non-functional property of a system such as performance, security, or scalability [7]*

Perspectives provide a more abstract description of a system, focusing on how the system behaves in opposition to what the system must do. Focusing on the qualities of the system, versus its functionalities, we can derive a number of categorized high-level objectives that need then to be analyzed (resulting into the so-called strategies and tactics) before being translated into concrete *Design Choices (DCH)*. Those categories are the architecture perspectives. The following preliminary set of perspectives is relatively common among IT systems and is by no mean to be considered as exhaustive; additional perspectives could be needed depending on the nature of the targeted IT system:

1. Trust, Security and Privacy (usually split into three separate perspectives);
2. Availability and Resilience;
3. Evolution and Interoperability;
4. Performance and Scalability.

Perspectives are further explained in Section 2.7.

2.2 Introduction to architecting process

Having discussed a few points of terminology, we can elucidate now the whole methodology (see Figure 1 below) that logically relates the different architecting tasks with each other.

In this figure, plain arrows refer to control flow while dashed arrows refer to dependency.

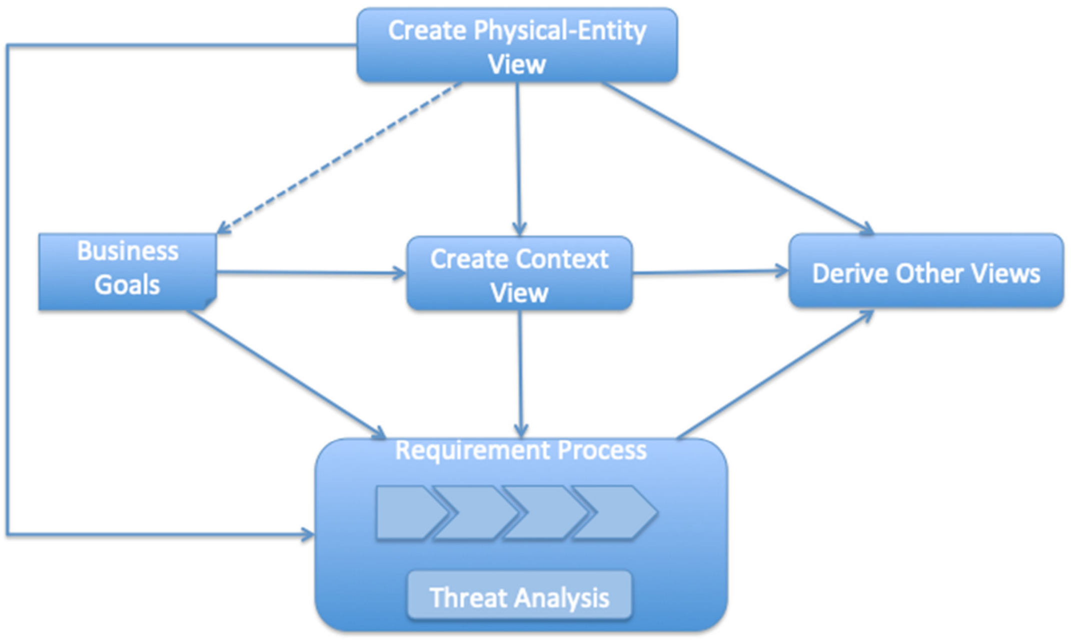


Figure 1: Simplified view of the architecting process

It is important to build first the *Physical-Entity view*[1] *(PEV)* because the exact nature of those entities (whether they are human or physical object) and what sort of information is tracked (e.g., location, pace/gait for humans and various sorts of quantities for physical objects) have an in-

[1] The PEV is described in more detail in Section 2.6.1

cidence on various aspects of the system and are fed into *Context view*[2] *(CV)* and the requirement process for further analysis.

Second, the Context view focuses on relationships between the system and its environment (consisting of people a.k.a. actors in D2.1 [3]), side (legacy) systems) which also impacts the requirement process.

Third, business goals also impact the requirements process (via NFREQ) and the CV as the relationships between people/actors and the system may depend on those goals.

Finally, the remaining arrows are straightforward, based on the CV, PEV and requirement process outputs the other views are elucidated.

It is worth noting that part of "derive other views" box is the activity of dealing with perspectives, i.e. to describe tactics and sub-sequent concrete DCHs that –as we explain in detail in Section 2.6.6 below- do impact the design of the other views (i.e. FV, IV and NDV).

2.3 Introduction to the Threat Analysis

This section describes the process, which will be used in the context of DEDICAT 6G for conducting a threat analysis.

The threat analysis can be considered as a part of the overall requirement engineering process, but was kept outside the Figure 2 below because it is conducted in parallel and more than once: 1st time to help identifying Security/Trust/Privacy requirements and a 2nd time to identify additional threats to the architecture itself (including security), after a first version has been built.

A threat analysis needs to be performed in order to come up with a list of potential threats to the system under design (some being identified during requirement collection), which are worth taking into account. In this process, we mitigate between known vulnerabilities, risks and potential impacts. This step will result in a set of security/privacy/trust-focused requirements.

During that process, one traditionally begins with a definition of the elements that have to be protected. Then, a thorough analysis of possible threats is conducted. How identified threats may actually affect elements to be protected, leads to the definition of risks. These risks have to be categorized, considering parameters such as criticality or probability of occurrence. In DEDICAT 6G the STRIDE [4] methodology is used for performing the threat analysis. Some more detail about STRIDE is available in Section 3.1.

2.4 Introduction to the Requirement engineering process

The following Figure 2 is taken from D2.1 [3] (where it was first introduced) and is a reminder of the tasks involved during the requirement engineering process, namely:

- **Requirement collection:** to collect requirements from both the project Use Cases UC1-4 and platform points of view;
- **Requirement analysis:** This activity can be split into the two following steps:
  - **Requirement consistency check:** to check and deal with potential requirement inconsistencies, especially when multiple sources are involved;
  - **Requirement rewriting, factorization and alignment:** to discard duplicates and factorize as much as possible, to align with common vocabulary;
- **Requirement mapping:** The unified functional requirements need to be mapped to the *Functional view* (FV) where one or more *Functional Groups (FG)* (e.g., Decision Making FG) and *Functional Components (FC)* (e.g., Coverage Extension Decision Making FC)

[2] The CV is described in more detail in Section 2.6.2

can be identified (see as an illustration the "green" right-most part of the VOLERE template sample in Figure 3).

This will result into the functional decomposition of the targeted system. Some of the non-functional requirements can be mapped to the Information and Deployment views. In a nutshell the requirement mapping is about:

1. mapping functional requirements (FREQs) to the Functional View performing the so-called functional decomposition;
2. mapping Technology and design constraints to design choices;
3. mapping Non-Functional Requirements (NFREQs) to other views (bar functional) and perspectives.

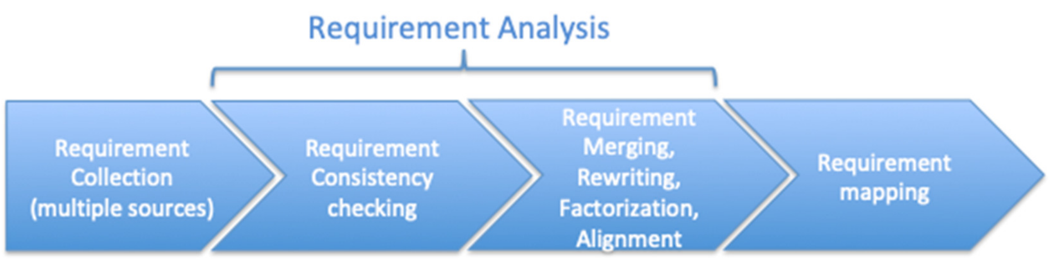

Requirement Analysis
Requirement Collection (multiple sources)
Requirement Consistency checking
Requirement Merging, Rewriting, Factorization, Alignment
Requirement mapping

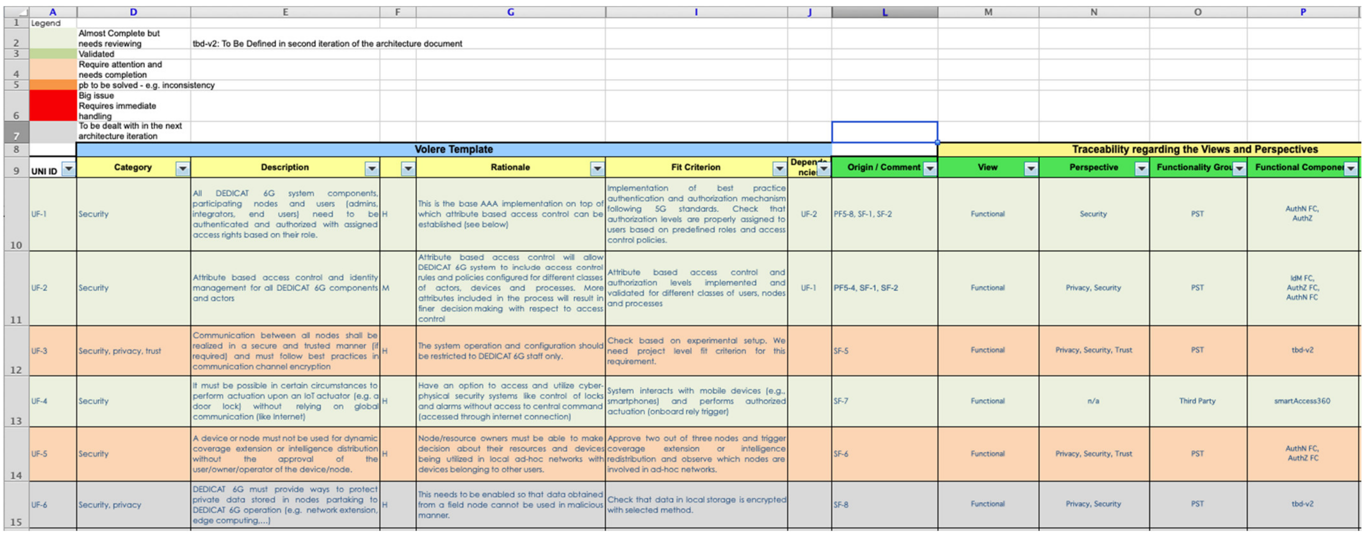
Figure 2: Requirement engineering steps

2.5 Requirement engineering supporting Tools

This additional section introduces a supporting tool used for summarizing the outcomes of the various steps described above, in the form of an Excel sheet. This is an outcome of Task 2.2, after various requirements have been unified. It is then updated during the mapping phase.

The VOLERE template used in DEDICAT 6G [5] has been adapted from the version used by the IoT-A project for collecting requirements (see Figure 3 below) which is itself an extension of the original VOLERE template [8], augmented in order to comply with the Rozanski & Woods methodology using views, viewpoints and perspectives [7].

In particular, this extension allows us to capture the essence of the view / perspective mappings as explained in Section 2.4.



Legend	
	Almost Complete but needs reviewing
🟩	Validated
🟧	Require attention and needs completion
🟧	pb to be solved - e.g. inconsistency
🟥	Big issue Requires immediate handling
▨	To be dealt with in the next architecture iteration

tbd-v2: To Be Defined in second iteration of the architecture document

	Volere Template							Traceability regarding the Views and Perspectives			
UNI ID	Category	Description		Rationale	Fit Criterion	Depen-ncie	Origin / Comment	View	Perspective	Functionality Grou	Functional Componer
UF-1	Security	All DEDICAT 6G system components, participating nodes and users (admins, integrators, end users) need to be authenticated and authorized with assigned access rights based on their role.	H	This is the base AAA implementation on top of which attribute based access control can be established (see below)	Implementation of best practice authentication and authorization mechanism following 5G standards. Check that authorization levels are properly assigned to users based on predefined roles and access control policies.	UF-2	PF5-8, SF-1, SF-2	Functional	Security	PST	AuthN FC, AuthZ
UF-2	Security	Attribute based access control and identity management for all DEDICAT 6G components and actors	M	Attribute based access control will allow DEDICAT 6G system to include access control rules and policies configured for different classes of actors, devices and processes. More attributes included in the process will result in finer decision making with respect to access control	Attribute based access control and authorization levels implemented and validated for different classes of users, nodes and processes	UF-1	PF5-4, SF-1, SF-2	Functional	Privacy, Security	PST	IdM FC, AuthZ FC, AuthN FC
UF-3	Security, privacy, trust	Communication between all nodes shall be realized in a secure and trusted manner (if required) and must follow best practices in communication channel encryption	H	The system operation and configuration should be restricted to DEDICAT 6G staff only.	Check based on experimental setup. We need project level fit criterion for this requirement.		SF-5	Functional	Privacy, Security, Trust	PST	tbd-v2
UF-4	Security	It must be possible in certain circumstances to perform actuation upon an IoT actuator (e.g. a door lock) without relying on global communication (like Internet)	H	Have an option to access and utilize cyber-physical security systems like control of locks and alarms without access to central command (accessed through internet connection)	System interacts with mobile devices (e.g., smartphones) and performs authorized actuation (onboard rely trigger)		SF-7	Functional	n/a	Third Party	smartAccess360
UF-5	Security	A device or node must not be used for dynamic coverage extension or intelligence distribution without the approval of the user/owner/operator of the device/node.	H	Node/resource owners must be able to make decision about their resources and devices being utilized in local ad-hoc networks with devices belonging to other users.	Approve two out of three nodes and trigger coverage extension or intelligence redistribution and observe which nodes are involved in ad-hoc networks.		SF-6	Functional	Privacy, Security, Trust	PST	AuthN FC, AuthZ FC
UF-6	Security, privacy	DEDICAT 6G must provide ways to protect private data stored in nodes partaking to DEDICAT 6G operation (e.g. network extension, edge computing,...)	H	This needs to be enabled so that data obtained from a field node cannot be used in malicious manner.	Check that data in local storage is encrypted with selected method.		SF-8	Functional	Privacy, Security	PST	tbd-v2

Figure 3: Glimpse at the VOLERE template

2.6 Introduction to the Architecture Views

This section introduces the six main architecture views, which will be used for the DEDICAT 6G architecture. We particularly focus on their purposes and on the viewpoints that can be used for elucidating them.

It is worth reminding that not all views are dealt with in this second iteration of the deliverable; the remaining Instantiation view will be addressed in its third and final iteration.

2.6.1 Physical-Entity View

The purposes of the Physical-Entity view are:

- To identify the *Physical-Entities (PE)* of interest for the DEDICAT 6G system. Those entities can be human or physical objects;
- To describe their properties of interest and how those PEs will be kept track of by the DEDICAT 6G system;
- In case some PEs are associated with sensors, to give detail about how the sensors relate to the object e.g., whether they are physically attached to the PEs or which physical quantities they measure;
- To give an exact list of the information captured by the DEDICAT 6G system either it is via sensors or personal devices (e.g., smart phones).

It is worth noting that having such information early, directly impacts 1/ the NFREQ, especially on the privacy aspects, and 2/ the Context view (as we will see in the next section) while providing essential early inputs to the data management deliverable D1.3 [9] and its revised version part of WP1.

As for viewpoints, we are using essentially textual description and tables.

2.6.2 Context View

According to Rozanski & Woods, the Context view ought to achieve the following tasks:

- To clearly define the perimeter of the DEDICAT 6G system, identifying external entities to the DEDICAT 6G platform (called physical systems later on);
- To define the nature and characteristics of those external entities
- To identify human actors and roles and their relation to the DEDICAT 6G system;
- To define external interfaces, please note that those interfaces are pretty much Use Case dependent, we will therefore focus on those required by UC1-UC4;
- To elucidate any external interdependencies, e.g., with any legacy systems used.

Viewpoint-wise, we will mainly use UML use-cases for elucidating actor-to-system interactions.

2.6.3 Functional View

The *Functional view (FV)* is probably the view requesting the most of work as it covers many different, but related aspects. The activities for building that view up are:

- To elaborate and describe a *Functional Model (FM)* that consists of a set of functional groups organized in layers, usually from the less to the most abstract following the well-known and widely referred to, *Data-Information-Knowledge-Wisdom (DIKW)* paradigm [10]; the FGs identified in the FM are the same FG found in the VOLERE template and used for requirement mapping;
- Based on the FM, to identify per FG the functional components that are needed to cover the FG objectives. At this stage the collected functional requirements are a main source of information for completing this activity. This results into a functional de-

composition that eventually provides the main viewpoint of the FV embodied into the functional model filled up with all relating FCs (as result of the requirement mapping on one side and perspectives elicitation and outcome on the other side);

- To provide high-level descriptions of all FCs in textual terms, focusing on their purposes and goals;
- To elucidate a set of system use-cases that aim to identify typical essential patterns resulting from 1/ either person-to-system or physical object-to-system interactions or 2/ from internal autonomous process like e.g., "decision-making leading to the migration of intelligence to the edge nodes". Those use-cases will be essential for the common understanding of what the system does, using for that purpose both textual descriptions/explanation and UML sequence diagrams. They will be a very meaningful input to the technical WPs, helping them to follow the architecture principles for the sake of architecture compliance.

As far as viewpoints are concerned, we will use the UML notation for static and dynamic inter-FG interactions, e.g., UML sequence diagrams and UML use-cases for interactions taking place across the system boundary box.

Please note that whenever human actors partake into human-to-system interactions, it is important to emphasize which particular role is endorsed by the human counterpart.

2.6.4 Information View

The *Information View (IV)* is all about data/information and interface. This view will therefore elucidate the following information-related aspects:

- To provide component interfaces and associated data models;
- To elucidate data flows occurring between system FCs and those occurring across the boundary box of the system.

The viewpoints used for the IV are 1) UML data flow diagrams, 2) UML data class/object modelling and 3) interface modelling.

2.6.5 Network Deployment View

The *Network Deployment view (NDV)* aims to describe how FCs are physically deployed in both DEDICAT 6G cloud & edge, and alongside the 5G/B5G network architecture.

This view is an essential illustration tool when 1) describing the edge-computing aspects of the DEDICAT 6G system and its dynamic nature relying on complex decision-making and 2) elucidating how the DEDICAT 6G components interact with the 5G legacy components.

The main viewpoint will be an extension of the FV main viewpoints where edge devices and external entities are explicitly pictured alongside the typical: `Access` → `Far Edge` → `Edge` → `Cloud` deployment.

Additional viewpoints are used to picture the deployment itself linked to technical Use Cases as already done in the FV.

2.6.6 Instantiation View

This last view shows how the developed software components instantiate and deploy upon the DEDICAT 6G architecture. Because there is not a one-to-one mapping between logical FCs and developed concrete components (usually several FCs are implemented in one bigger software component though the converse may happen as well) the instantiation view acts as an overlay to the functional view and shows how developed software concrete components map to the logical FCs resulting from the functional decomposition.

2.7 Introduction to the Architecture Perspectives

According to Rozanski & Woods, an architectural perspective "is a collection of activities, tactics and guidelines that are used to ensure that a system exhibits a particular set of related quality properties that require consideration across a number of the system's architectural views" [7].

In this definition, a quality property is meant to be "an externally visible, non-functional property of a system such as performance, security or scalability" [7].

As we can see, architectural perspectives are orthogonal to architectural views; therefore, any architecture or design decision pertaining to non-functional or quality requirements often spans more than one architectural view, if not all.

We identify hereafter a comprehensive list of perspectives which are relevant to DEDICAT 6G. Each perspective focuses on specific non-functional requirements or desired quality properties of the architecture. The following list is a set of perspectives (already introduced shortly in Section 2.1.3) that can be of interest for DEDICAT 6G, though probably not all will be eventually covered:

- **Evolution** (or Evolvability): is a quality of a system that has been designed in such a way it can easily be adapted to new technologies;
- **Interoperability**: ability of a system to easily interoperate with other systems at various levels like technical, syntactical, semantic and organizational;
- **Availability**: ability of a system to be fully (or partly) available when required;
- **Resilience**: ability of a system to effectively handle failure or attacks that could affect the system availability;
- **Privacy**, **Security** and **Trust**:
  - *Privacy*: ability of a system to deal with all kinds of personal data and in particular to implement reliably privacy policies about accessing, sharing that data or hiding people's identity;
  - *Security*: ability of the system to reliably control, monitor and audit who can perform what actions on what resources, to detect and recover from failures insecurity mechanisms and to resist to cyber attacks;
  - *Trust*: ability of a system to establish and enforce trusted relation between the different parties involved in a system (end-users, component, data) in such a way system operation and behaviors comply to expected ones;
- **Performance**: ability of a system to predictably perform its operations within its mandated performance requirements and profiles;
- **Scalability**: ability of the system to cope with increasing demand in computing, networking, storage resulting from increasing volume of system usage;
- **Usability**: quality that illustrate how easy a system can be used, how easy data can be apprehended by the end-users, how easy the GUI is understandable and ergonomic while maintaining efficient work.

The non-exhaustive list of qualities may be updated or adapted according to the architects' needs.

Each desired quality will be then associated with a set of activities (for instance activities associated with Trust, Security and Privacy are the collection of trust requirements, the conduction of risk and threat analysis, the definition of a trust model, etc.)

Then defining a certain number of tactics allows showing how the desired system quality can be eventually reached. Because a tactic can span more than one view, the implementation of a tactic through *Design CHoices* (DCH) can lead to more than one of those DCHs (e.g., a tactic for realizing anonymity can lead to a collection of DCHs relating to data structure for the Information view and DCHs relating to interfaces, storage and security-related functionality in the Functional view).

As shown in Figure 4 below, architecture perspectives (the grey horizontal boxes) are focusing on system properties/qualities and are orthogonal to the architecture views. As a consequence, when deciding to consider a tactic that would allow the system to reach a desired quality, results in the definition and implementation of concrete DCHs that impact more than one view, if not all.

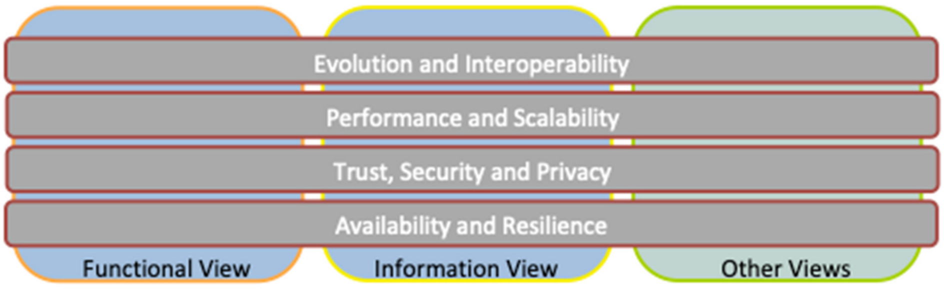

Evolution and Interoperability
Performance and Scalability
Trust, Security and Privacy
Availability and Resilience
Functional View
Information View
Other Views

Figure 4: Architecture Views and Perspective

In the Perspective section of the DEDICAT 6G (Section 5), we will describe perspectives using a table structure as shown in Table 2 below[3]:

Table 2: Perspective description

Targeted System Quality	Describes the overall objectives of the considered perspective (say Privacy), considered as a category. From the requirement analysis phase, we will come up with several objectives falling under one of the lists of perspectives (see above), each one of these objectives (say privacy related objective) will be declined into one or several tactics (say anonymisation, pseudonymisation, etc.), which can be considered as high-level DCH.
Requirement(s)	Gives the list of NFREQs concerned with that perspective.
Activities	Gives a list of activities needed when dealing with the perspective. It includes activities like: • Performing a threat analysis; • Performing system / network stress assessment; • Simulations; • Updating list of requirements; • Validating against requirement (especially for any NFREQ associated with performance thresholds), etc.
Tactics	Consists of a set of high-level abstract design choices that can be used to reach a desired property. Then each one of these tactics binds to a set of DCHs which relate to a particular view as explained earlier. Such DCHs can be: • Adopting certain algorithm; • Adopting specific deployment strategies; • Adopting a certain architecting choice; • Introducing specific FCs, etc.

As far as DCH description is concerned we will be using the following Table 3 structure:

[3] inspired from the IoT-A Architectural Reference Model [11]

Table 3: Example of Design Choice descriptions

Design Choice ID	View	FG/FC	Technical description
PRIV-01	FV	Security/ xyz FC	This component is responsible for…
PERF-01	NDV	n/a	In order to fulfill this desired quality, the deployment of….

3 DEDICAT 6G Requirement Engineering

As we mentioned in the introduction Section 2.4, the requirement engineering process is the sole purpose of Task 2.2. However, the results of the first activity of that task, namely Requirement Collection, are fed into three deliverables:

- Initially D2.1, then revised in D2.3: Project Use Cases requirements from the WP6 point of view (project Use Case work package);
- Initially D2.2: Platform requirements from WP3, 4 and 5 points of view (technical work packages).

Please note that both Use Case and platform requirements can be functional or non-functional.

It is worth reminding that the platform technical ambitions have been motivated after a thorough *State of The Art* (SoTA) analysis applied to the corresponding technical fields. The SoTA conclusions are documented in the DEDICAT 6G DoA [1] as a justification to our technical ambitions. Those platform requirements are a translation of those legitimate technical objectives.

The second activity takes all available requirements from the requirement collection phase and produces a list of *UNIfied requirements* (UNIs). The unification process consists of aligning the vocabulary (the requirements are collected by various project partners with different technical backgrounds), identifying replicates, factorizing, and rewriting. We also pay particular attention to identifying dependencies and possible conflicts, which need to be eventually dealt with.

Finally, the third requirement-related activity, namely requirement mapping map all requirements (resp. Functional, Non-Functional) to specific views and FG/FC (FREQ), or/and perspectives (NFREQ).

Before delving into those specific requirement-engineering activities, it is worth reminding a special case that applies to Privacy, Security and Trust.

Traditionally, before elucidating requirements applying to those three system qualities, one usually engages first into a Threat Analysis, which –in a nutshell- aims at:

1. Evaluating the vulnerabilities of a system constituents;
2. Elucidating potential threats and attacks towards those identified vulnerabilities;
3. Assessing the impact of such threats and
4. Assessing the resulting impact.

Conducting a threat analysis prior (or at least in parallel) to requirement collection is crucial as its conclusions drive the identification of functional and non-functional requirement relating to Privacy, Security and Trust.

3.1 Threat Analysis

A threat analysis consists of understanding and identifying the assets to be protected, as well as identifying and evaluating possible threats against those assets. To assess the security of a system, we must look at all the possible risks and warnings. The STRIDE (Spoofing identity, Tampering with data, Repudiation threats, Information disclosure, (distributed) Denial of service and Elevation of privileges) [4] model is a useful tool for threats classification.

A STRIDE analysis must be addressed in the early phase, in order to incorporate smoothly the security/privacy/trust-related features to the architecture right from the beginning.

Consequently, a threat analysis based on an attacker's perspective was conducted to derive the hazard scenario. STRIDE is based on a reference architecture for determining the overall image of a system. The purpose of threat modelling is to understand how an attacker could penetrate the system.

The following Table 4 gives some examples of threats, their definition, potential scenarios and the countermeasure that could be undertaken, in order to face the threat.

Table 4: STRIDE model with general scenarios and measures

Threat	Property Violated	Definition	Expected scenarios	Measures
Spoofing	Authentication	Impersonalizing something or someone else.	If the system user settings are not set properly, attackers might spoof them	Set the password appropriately: SSH Login with private key
Tampering	Integrity	Modifying data or code	If an illegal program has access to a cryptographic key or an encryption mechanism that holds the cryptographic key, the software replaced will misuse the real ID of device	MAC Applying a tamper-proof mechanism to the device
Repudiation	Accountability	Claiming to have not performed an action.	If the user does not have a log of the communication, it is likely to negate the fact of the operation that the user performed improperly	Acquisition and maintenance of various logs
Information disclosure	Confidentiality	Exposing information to someone not authorized to see it	The attacker exploits the encrypted key and obtains the encryption key and decryption key between nodes	Implemented Anti-malware, Secure Key Management
Denial of service	Availability	Deny or degrade Service to users	The function might be stopped if unauthorized access is performed over a network	Apply response limit
Elevation of privileges	Authorization	Gain capabilities without proper authorization	Allowing a user to run commands as admin	Restrict users who can get administrator rights

Threats are caused by attackers and result from the exploitation of various assets vulnerabilities which then become the target of such threats. The nature of a threat usually depends on the very nature of the targeted vulnerability. Using STRIDE enables analysis based on attributes such as confidentiality other than availability or integrity. Here follows a list of system elements and assets that must be protected:

- User workstations;
- Mobile devices and assets;
- Network devices (hubs, switches, routers, AP);
- Servers;
- Specialized devices;
- Software and services (OS, applications, client programs);
- Data (stored, archived, databases, data in-transit);
- Communications.

A list of possible threats targeting those general system elements and assets can be:

- Physical damage;
- Unauthorized access to data and services;
- Unauthorized disclosure of information;
- *Denial of service (DoS)*;
- Theft of data and services;
- Corruption of data and services;
- Viruses, worms, Trojan horses.

Focusing more on the networking aspects, possible network threats are for instance:

- Redirects the flow of data to attacker machine;
- Modifies data flowing over the network;
- IP and DNS spoofing;
- DNS compromise;
- Spoofing a user account;
- Spoofing a role.

The examples above are very general threats, however the DEDICAT 6G Threat Analysis (Table 5 and Table 6) shows that some of those general threats, in addition to more DEDICAT 6G-specific threats, need being considered.

Risk assessment relies very much on an appropriate preceding categorization and classification of threats. The result of the risk assessment procedure should be security requirements specification.

The Figure 5 below elucidates the process of risk assessment [4].

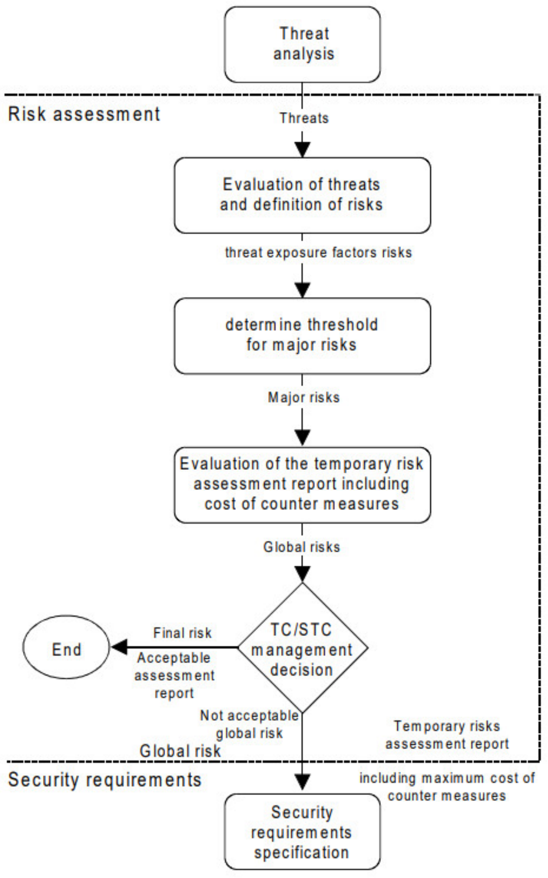

Threat analysis
Risk assessment
Threats
Evaluation of threats and definition of risks
threat exposure factors risks
determine threshold for major risks
Major risks
Evaluation of the temporary risk assessment report including cost of counter measures
Global risks
End
Final risk
Acceptable assessment report
TC/STC management decision
Not acceptable global risk
Global risk
Temporary risks assessment report
Security requirements
including maximum cost of counter measures
Security requirements specification

Figure 5: Methodology for risk assessment

Considering the nature of the DEDICAT 6G projects we have come up with the following general risk assessment (see Table 5).

Table 5: General risk definition and the assignment of evaluation values

Risks	Likelihood of occurrence	Potential impact
loss of integrity or confidentiality	High	High
data interception of signaling and user data	Low	Medium
Modifying data or code	Low	High
Hardware failure caused by cyber attack	Low	High
Data leaks	Low	Medium
loss of privacy	Medium	Medium
loss of availability of resources or service	Medium	High
Loss of trustworthiness to authorities	Low	Medium
Destruction of components	Medium	High
Installation of intentional malfunction, sabotage	Low	High

However, the Table 6 below gives a much clearer focus on the physical systems (as described in Table 11) laying at the edge of the DEDICAT 6G platform. It provides the risk definition and assessment pertaining to those physical systems.

Table 6: Risk definition and the assignment of evaluation values to the Physical Systems

Physical system name	Risks	Likelihood of occurrence	Potential impact
Edge-terminal	• Loss of availability of resources or service • Destruction of components	• Low • Medium	• Medium • Medium
AGVs	• Loss of integrity or confidentiality • Hardware failure caused by cyber attack • Loss of availability of resources or service • Modifying data or code • Loss of trustworthiness • Installation of intentional malfunction, sabotage	• Low Medium • Medium • Medium • Low • Low	• Medium High • High • Medium • Low • High
Forklift/machine	• Hardware failure caused by cyber attack	• Medium	• High
SmartAccess360 controller	• Loss of integrity or confidentiality • Hardware failure caused by cyber attack • Loss of availability of resources or service • Modifying data or code • Loss of trustworthiness	• Low • Medium • Medium • Medium • Low	• Medium • High • High • Medium • Low
Warehouse per-	• Loss of availability of resources or service	• Medium	• High

sonnel smartphone/mobile device	• Loss of integrity or confidentiality • Data leaks • Loss of privacy	• Low • Low • Medium	• Low • Medium • Medium
(B)5G Networking Equipment	• Loss of integrity or confidentiality • Hardware failure caused by cyber attack • Loss of availability of resources or service • Data leaks • Installation of intentional malfunction, sabotage	• Low Medium • Medium • Low • Low	• Medium High • High • Medium • High
Video streaming platform	• Loss of availability of resources or service • Modifying data or code • Loss of trustworthiness	• Medium • Medium • Low	• High • Medium • Low
Drones	• Loss of integrity or confidentiality • Hardware failure caused by cyber attack • Loss of availability of resources or service • Data leaks • Loss of trustworthiness • Installation of intentional malfunction, sabotage	• Low • Medium • Medium • Low • Low • Low	• Medium • High • High • Medium • Low • High
Smart phones	• Loss of availability of resources or service • Data leaks • Loss of privacy	• Medium • Low • Medium	• High • Medium • Medium
smartGlass	• Modifying data or code • Data leaks • Loss of privacy • Installation of intentional malfunction, sabotage	• Medium • Low • Medium • Low	• Medium • Medium • Medium • High
Connected Car (maybe different from UC1)	• Hardware failure caused by cyber attack • Loss of availability of resources or service	• Medium • Medium	• High • High
MCS mobile server	• Hardware failure caused by cyber attack • Loss of availability of resources or service • Modifying data or code • Data leaks	• Medium • Medium • Medium • Low	• High • High • Medium • Medium
Attendee smartphone	• Loss of integrity or confidentiality • Data leaks • Loss of privacy	• Low • Low • Medium	• Low • Medium • Medium
1st Responder smart phone	• Loss of availability of resources or service • Loss of integrity or confidentiality • Data leaks • Loss of privacy	• Medium • Low • Low • Medium	• High • Low • Medium • Medium
SmartGate	• Loss of integrity or confidentiality • Loss of trustworthiness • Installation of intentional malfunction, sabotage	• Low • Low • Low	• Low • Low • High
Smart vehicle	• Hardware failure caused by cyber attack • Loss of availability of resources or service	• Medium • Medium	• High • High

	• Modifying data or code	• Medium	• Medium
Smarter vehicle (incl. tablet-like terminal)	• Loss of availability of resources or service • Modifying data or code	• Medium • Medium	• High • Medium
Sensing Nodes	• Loss of availability of resources or service • Modifying data or code • Data leaks	• Medium • Medium • Low	• High • Medium • Medium
RSU	• Data leaks	• Low	• Medium

As the very purpose of a risk analysis is to understand which threats a system needs to mitigate, the logical next step is to provide a list of security requirements.

It is worth mentioning here that the threat analysis in this document only focuses on security threats. Privacy and Trust are first handled in the requirement collection/analysis process and then dealt with in their respective perspectives (Section 5).

Security requirements will be associated with the security features. Requirements must be implemented early in the development phase and the appropriate security features are mounted on the devices themselves.

The two following tables (respectively Table 7 and Table 8) give the list of general security requirements and those applying to the DEDICAT 6G physical systems as well. Those requirements are fully embodied into the Privacy, Security and Trust-related FREQ and NFREQ.

Table 7: List of general security requirements

Security Requirement	**Description**
Access, Authentication, and Authorization Management	Authenticate users through central AuthN/AuthZ systems, grant the minimum, sufficient access, or privileges, employ role-based access controls, access sensitive data only as necessary for job duties, encrypt authentication and authorization mechanisms
Audit logging and analysis	Enable logging for endpoints, include essential events and elements in logs, restrict log access to authorized individuals, automate alerting on logging failures
Cryptography and key management	Protect digital assets and communications, implement GDPR, user/role access to the encryption keys
Network and data security	Implement default-deny, least-privilege policies on network devices, encrypt network traffic, securely configure network infrastructure devices
Code integrity	Validates the integrity of a component/driver or system file each time it is loaded into device memory, encrypts external transmission of data, implements application logs with important event data
Data validation and sanitization	Validate on Input - ensuring that incoming data is uncompromised before it is allowed to be processed, sanitize device/storage media before transfer, ensure sanitization methods meet the Standard's requirements

Table 8: List of security requirements for physical systems

Physical system name	Security Requirement
Edge-terminal	• Access, Authentication, and Authorization Management • Network and data security • Audit logging and analysis
AGVs	• Access, Authentication, and Authorization Management • Audit logging and analysis • Network and data security • Code integrity
Forklift/machine	• Access, Authentication, and Authorization Management
SmartAccess360 controller	• Access, Authentication, and Authorization Management • Audit logging and analysis • Cryptography and key management • Network and data security • Code integrity • Data validation and sanitization
Warehouse personnel smartphone / mobile device	• Access, Authentication, and Authorization Management • Audit logging and analysis • Data validation and sanitization
(B)5G Networking Equipment	• Access, Authentication, and Authorization Management • Audit logging and analysis • Cryptography and key management • Network and data security • Code integrity
Video streaming platform	• Audit logging and analysis • Code integrity
Drones	• Access, Authentication, and Authorization Management • Audit logging and analysis • Network and data security • Code integrity
Smart phones	• Access, Authentication, and Authorization Management • Audit logging and analysis • Data validation and sanitization
SmartGlass	• Access, Authentication, and Authorization Management • Audit logging and analysis
Connected Car (maybe different from UC1)	• Access, Authentication, and Authorization Management • Audit logging and analysis
MCS mobile server	• Access, Authentication, and Authorization Management • Audit logging and analysis • Network and data security • Code integrity • Data validation and sanitization
Attendee smartphone	• Access, Authentication, and Authorization Management • Audit logging and analysis
1st Responder smart phone	• Access, Authentication, and Authorization Management • Audit logging and analysis
SmartGate	• Access, Authentication, and Authorization Management • Audit logging and analysis
Smart vehicle	• Access, Authentication, and Authorization Management • Audit logging and analysis
Smarter vehicle (incl. tablet-like terminal)	• Access, Authentication, and Authorization Management • Audit logging and analysis
Sensing Nodes	• Access, Authentication, and Authorization Management • Audit logging and analysis • Network and data security • Code integrity • Data validation and sanitization
RSU	• Audit logging and analysis

3.2 Requirement Collection

The requirement collection was indeed started in D2.1 with the collection of the Use Case-focused requirements where the Use Case holder takes the stance of a DEDICAT 6G customer deciding about 1) what are the needed functionalities (which features the targeted system must provide) and 2) overall system qualities (qualifying how the targeted system must behave).

Then another stance was taken when editing the first architecture document iteration D2.2: the platform provider's point of view. In accordance with the technical objectives we have stated in the DoA [1]. D2.2 provided then a list of functional (respectively non-functional) platform-focused requirements.

Finally, all requirements (scenario and platform) were unified and mapped to views and perspectives resulting into the so-called *Unified requirements*, which was the purpose of the Volere template Excel sheet, delivered at the same time D2.2 was.

Following the WP2 work plan, a second iteration of the scenario description and requirement collection document (D2.3 [3]) gave the opportunity to reassess the Use Cases requirements in order to consider the progress achieved in WP6. That new list of requirements is part of the deliverable D2.3 and is therefore not included here.

However, D2.4 does provide an update of the VOLERE template that takes those new requirements into account (as part of Task 2.2 recurring activities). It can be found on the project website [5].

4 DEDICAT 6G Architecture Views

4.1 Physical-Entity View

As introduced in Section 2.6.1, the main purposes of the PE View are:

- To make an inventory of external entities involved in DEDICAT 6G operation;
- To identify which information is exchanged and discuss the privacy related issues;
- To define the roles and natures of interactions taking place between the *Physical-Entities (PE)* and the DEDICAT 6G system (edge and Cloud).

We first perform the inventory of physical-entities and then address the nature of data the DEDICAT 6G either captures (e.g., via sensors) or has access to, with emphasis on potential existing data privacy issues.

4.1.1 Inventory of Physical-Entities

This first section provides an inventory of, so-called, physical-entities, which are people/objects/network equipment that are in the focus of an IT system and carry information of interest for the targeting IT system, e.g., in the process of building up a context (in the sense of context-awareness). PEs are paramount for IoT system because intrinsic characteristics of the PEs are tracked down by sensors and translated into properties of their cyberspace counter parts virtual-entities. However, PEs are also important for DEDICAT 6G as human PEs, robots, cars and legacy network equipment are interacting with the system and indeed also carry information of interest to the DEDICAT 6G system. This is the purpose of respectively Table 9 and Table 10 to identify such PEs and to identify the information of interest and their required levels of privacy.

Making an inventory of those various PEs and roles helps:

1. Understanding which entities partake into DEDICAT 6G operation and identifying the nature of exchange information;
2. Steering privacy / security / trust-related requirements and technical objectives;
3. Feeding important information into GDPR-related documents;
4. Defining the Context view (e.g., human/system interface) by elucidating the nature of interactions taking place between entities and DEDICAT 6G.

Examples of such entities we are dealing with in DEDICAT 6G range widely in nature, e.g.:

- People/individuals: e.g., tracking people position and gait;
- Crowd: e.g., determining whether the crowd is in "panic" mode;
- Network (pieces of equipment or sub-systems): if DEDICAT 6G "senses" network condition in order to feed the decision-making FC before triggering some network extension-related activities;
- Cars, Robots, AGVs etc.

So most importantly, an entity can indeed be considered as a PE in the PE view and also as a *Physical System (PS)* in the Context view when it comes to identifying entities, which are 1/ either at the edge of DEDICAT 6G or, 2/ definitely outside the DEDICAT 6G perimeter (as elucidated in the Context view in Section 4.2.1).

Table 9: Actors / Roles / Relation to the system

UC	Actor/Entity	Description	Role	Description of Interaction (short)
All	Network operator	Operator and owner of network infrastructure (core, radio access network)	Managing and operating the network	Provides policies i.e., high-level rules that should be followed in context handling by intelligence distribution or coverage extension. These also include potential rules and priorities on goals to be achieved, such as the maximization of QoS levels, and the minimization of cost factors (e.g., resource consumption)
All	Service provider	Entity offering wireless communications services over a network infrastructure that it does not own	Managing and operating the wireless communication service in coordination with the infrastructure owner	Similar to the network provider it provides policies, i.e., high-level goals and rules that need to be achieved and followed by the mechanisms for intelligence distribution and coverage extension
All	Vertical actor (a.k.a. DEDICAT 6G customer)	A third party that relies on DEDICAT 6G functionalities to improve its business operation. Examples of such services are Intelligence Distribution as a Service or Coverage Extension as a Service	The role of the Vertical operator is about potentially anything. The 4 DEDICAT 6G scenarios provide 4 realistic and relevant examples of such vertical actor, and the kind of services it relies on	A vertical can rely on DEDICAT 6G to manage the deployment of its own components using its own edge nodes or the one operated by DEDICAT 6G. It can also require Network Extension whenever it needs temporary extra capacity in order to operate its business
All	Owner of edge node (e.g., small server/car/drone etc.)	Actor engaged in a DEDICAT 6G scenario who provides H/W devices that can be used as edge nodes by DEDICAT 6G. It can be e.g., the Network Operator or a Vertical customer	Provides his/her edge devices for edge processing potentially based on some offered incentives	Registers his/her edge devices as potential edge nodes allowing a basic set up that will allow edge processing in the scope of intelligence distribution
3	Event attendee	Person attending an organized event e.g., concert	n/a	Attendees are sharing personal information including potentially heath condition in order to ease the 1st responder's action in case health condition deteriorates during the concert. They also may (subject to formal consent) participate in crowd analysis by

				feeding DEDICAT 6G system with location and gait information
3	Connected Cars	Vehicle acting as edge processing and mobile communication node in the DEDICAT 6G system. Performing Public Safety Critical Communication processes	Mobile communication node, edge processing	As an edge node it may Runs run DEDICAT 6G components for networking and decision-making. It may act as a Mobile Access Point
All	Network Equipment	5G legacy equipment provided by the Network Operator	Provides the Core and RAN 5G components that DEDICAT 6G relies on as for Intelligence Distribution and Coverage Extension	A lot of interactions take place between 5G legacy pieces of equipment and Software in both directions. This includes receiving and exploiting various sort of information coming from the 5G network, especially concerning its operation and performance. DEDICAT 6G also needs to instrument some functions supported by the legacy network like e.g., slicing. It also has to inform the 5G network when coverage extension is performed. It also deploys some of the RAN and Core components as part of Intelligence Distribution and coverage extension support
1	Warehouse worker	Person working in the smart warehouse and directly interacting with deployed DEDICAT 6G resources and other warehouse systems	Implementing warehouse processes e.g., picking, packing, checking, loading/unloading	Registers on smart warehousing mobile app on work mobile device. Signs consent form. Gets assigned with a role and access rights. Uses mobile app to organize daily tasks. Interacts with AGVs in line with a task. Receives notifications and alerts based on smart warehousing decision-making.
1	Warehouse managers	Person managing and supervising smart warehouse processes and resources. Utilized management features of the DEDICAT 6G system deployed for the smart warehouse	Overall management. Training, conformance to safety measures/standards etc. Supervision of flows and processes within the warehouse	Registers and gets corresponding role in the smart warehousing DEDICAT 6G system. Utilizes DEDICAT 6G smart warehousing management system and dashboard as well as DEDICAT 6G mobile app to configure smart

				warehouse processes (AGVs, IoT system, schedules and notifications). Utilizes the dashboard and mobile app to assign roles to workers. Issues digital keys through SmartAccess360 system. Defines safety zones through the dashboard. Sends and receives push notification
1	AGV	Automated guided vehicle acting as edge processing and mobile communication node in the DEDICAT 6G system. Directly performing smart warehousing processes (loading, offloading, inspection)	Mobile communication node, edge processing, IoT platform (sensing/actuating capabilities)	As an IoT platform it directly interacts with smart warehousing systems and workers. As an edge node it may Runs run DEDICAT 6G components for Analytics, networking and decision-making. It may collect, store and analyse data. It may act as a Mobile Access Point
1&3	IoT controller	Edge IoT controller interacting with smart sensing and actuation systems in smart warehouse setting. Can run edge processing component and participate in range extension and provide auxiliary communication paths	Fixed communication node and edge processing node	It directly interacts with smart warehousing systems and workers. Runs DEDICAT 6G components for AI, networking and decision-making. It collects, stores and analyses data
2,3	Smart Glasses	The Smart glasses will be connected to smart phone to have access to the network. Each application will be built and adapted to the Use Cases for message, photo, exchange as well as for video streaming	Provide the ORA-2 smart glasses to the use-cases along with dedicated application for each use-case	Interact with core DEDICAT 6G system for use-case and provide AR overlay
All	Smartphone/ mobile device	Mobile device (smartphone, tablet) running DEDICAT 6G application or application directly interfacing with DEDICAT 6G system. Provides the main user interface for interacting with the performed processes and deployed resources	Edge processing and the main interface towards end- users, node of ad-hoc network, relay	Interacts with core DEDICAT 6G system for use-case and provide AR overlay. It is the main interface for end-users

1	Environmental sensors	Temperature, humidity, light intensity	Measuring environmental parameters	IoT controller and AGV collect information from sensors for edge-based processing and decision-making, or to be transferred to centralized decision-making process
1	*Bluetooth low energy (BLE)* location beacon	Low energy beacon is provided to workers, installed on mobile assets (e.g., forklift) for detecting indoor location and proximity to points of interest.	Used to detect proximity or derive indoor position	Beacons work with IoT controllers in BLEMAT solution for indoor positioning
3	Victim	The victim is a person who happens to have his life at risk due to various reason like an accident or a natural disaster. She waits for being rescued by so-called, first responders or asks security staff for assistance. The victim is a person attending an event or a civilian caught in a natural disaster	People who need rescue (First Responders) or assistance (Security staff)	After software acknowledgement by the victim, data (such as name, surname, age, localization, video, picture) through victims' smartphone is shared with DEDICAT 6G platform. The data are used for injuries evaluation and precise localization of victims by DEDICAT 6G system (e.g., decision-making or mission management)
3	Event operator	After the event and on Rescue team request, the Event operator is responsible to give authorization to access to some attendee's data (Name, Gate, Place)	Authority for sharing attendee data	Collection of data (including name, surname, disability, attendee localization) shared with DEDICAT 6G system which will support the evaluation of the situation
3	Member of Security staff	Person working in the smart warehouse and directly interacting with deployed DEDICAT 6G resources and other Event place systems	Applying Security rules during an event	Security staff is able to perform requested tasks and send status through the DEDICAT 6G platform. The radio coverage is provided by either Connected Car or available AVG and provides connectivity to the Security Staff where the accident or disaster took place
3	Doctor	Doctor who coordinates the medical staff on the scene. Taking the decision for different operation to take for the care of victims. Using the DEDICAT	Manage the care's operation of victims	Doctor gets situational awareness from Mobile C&C supported by Connected Cars networking and node edge computing in order to stay connected during the Crisis

		6G to monitor the situation and communicate with the medical team. Communicate with other authorities using DEDICAT 6G system		Management process
3	Medical staff member	Applying the decision and strategy of the doctor and interact with him using DEDICAT 6G system	Working under the commandment of doctor	Based on Mobile Client installed on Smartphone, Policeman is able to perform requested tasks and send status through the DEDICAT 6G platform. The coverage is provided by the Connected Car and available AVG in order to provide connectivity to the medical staff at the incident location
3	Police Operator	The Police Operator receives call and defines the mission which is sent to Policemen. Keeps contact with the Police Team on the field. He uses a C&C system which is interconnected to DEDICAT 6G	Receives call from public and victims. Organizes the mission	Operator gets situational awareness from Mobile C&C supported by Connected Cars networking and node edge computing in order to stay connected during all the Crisis Management process
3	Police Field Officer	Receives the mission from the C&C, manage the mission on the field and its team on the field using DEDICAT 6G system. Keeps the contact with the C&C and share the evolution of the situation	Manage the Police team on the film and the mission	Field officer gets situational awareness from Mobile C&C supported by Connected Cars networking and node edge computing in order to stay connected during the Crisis Management process
3	Policeman	Executes the order shared by the Police Field Officer using the DEDICAT 6G system. Reports the ongoing situation to the Officer. Some data are captured and share automatically with the DEDICAT 6G system in order to receive information from DEDICAT 6G Intelligence	Executing orders from Police Field officer	Based on Mobile Client installed on Smartphone, Policeman is able to perform requested tasks and send status through the DEDICAT 6G platform. The coverage is provided by the Connected Car and available AVG in order to provide connectivity to the Policeman at the incident location
3	Firefighter Operator	The Firefighter Operator receives call	Receives call from public and victims.	Operator gets situational awareness from

		and defines the mission which is sent to Firefighters. Keeps contact with the Firefighters team on the field. He uses a C&C system which is interconnected to DEDICAT 6G	Organizes the mission	Mobile C&C supported by Connected Cars networking and node edge computing in order to stay connected during all the Crisis Management process
3	Firefighter Field Officer	Receives the mission from the C&C, manage the mission on the field and its team on the field using DEDICAT 6G system. Keeps the contact with the C&C and share the evolution of the situation	Manage the Firefighter team on the film and the mission	Field officer gets situational awareness from Mobile C&C supported by Connected Cars networking and node edge computing in order to stay connected during all the Crisis Management process
3	Firefighter	Executes the order shared by the Firefighter Field Officer using the DEDICAT 6G system. Reports the ongoing situation to the Officer. Some data are captured and share automatically with the DEDICAT 6G system in order to receive information from DEDICAT 6G Intelligence	Executing orders from Firefighter Field officer.	Based on Mobile Client installed on Smartphone, Firefighter is able to perform requested tasks and send status through the DEDICAT 6G platform. The coverage is provided by the Connected Car and available AVG in order Firefighter is connected to the network while he is closed to the incident
3	MCX App	Client application allowing contextual information sharing and communication between users during Mission Management Such information displays Resources position, picture or other files, voice or video communication	Main interfaces for Public Safety or Private Security users to the MCX system. Allow authentication and manage security for information sharing	The MCX Client interfaces with MCX FC which is part of DEDICAT 6G platform in order to benefit from MCX features delivered by the platform
2, 3	Drones	Drone equipped with 5G and WIFI radio interface, computing capability (depending on UC specific needs). Drones are an efficient mean of use in order to get information from the crisis as they come up on the scene	The main purpose of the drone (considering the baseline scenario) is to provide 5G coverage extension. Additional purposes can include WIFI hotspot, some advanced IA-based algorithms like sudden crowd movement detection or video-monitoring (in crisis context for exam-	The drone is integral part of the baseline DEDICAT 6G architecture. There is a wide range of interactions involving dynamic migration of intelligence to the Edge part of the Drone, interaction with the legacy 5G network to ensure the coverage

		quicker than First Responders. During the crisis management, drones can continuously deliver information on the evolution of the situation during Mission Management	ple). In the context of UC3, the drones will share their positions in order to leverage the awareness of the situation with the shared images	extension, plus additional interaction depending on the embedded scenario-related FCs (e.g., use MCX Client to connect to MCX Server which are part of DEDICAT 6G platform in order to use the resilient connection to upload video and share position
3	Site Environment ("Large Event" UC3 scenario)	During the crisis management, Sensors like video camera surveillance can deliver information on the situation during Mission Management	DEDICAT 6G will offer connectivity capability to third party devices in order to make data sharing available for First Responders	Camera surveillance device from site will connect to DEDICAT 6G platform in order to get through a gateway the video flow and make it available for sharing with MCX data and video features
3	Emergency Vehicle (Police car, fire truck, ambulance…)	Integrated device running DEDICAT 6G application or application directly interfacing with DEDICAT 6G system. Provides the user connectivity for interacting with the performed processes and deployed resources.	Edge processing and the main interface towards end-users, node of ad-hoc network, relay	As an edge node it may Runs run DEDICAT 6G components for networking and decision-making. It may act as a Mobile Access Point.
3	MCX Smartphone	The device is used on the field by rescuers and runs the MCX applications.	Devices running MCX App for users on the field	MCX Smartphones are connected to DEDICAT 6G platform in order to get 6G/B5G connectivity while Commercial or other private network can't ensure connectivity services. MCX Smartphone are connected to Connected Cars or AVG.
3	MCX Server	The MCX Server host all MCX services and support Critical Group Communications	Server running MCX services	MCX Server is part of DEDICAT 6G platform and can deliver MCX services through DEDICAT 6G Node, AVG or Connected Cars.
4	Vulnerable Road Unit (VRU)	Road users such as pedestrian or cyclists	Moving around the road using a handheld device	VRUs are connected to the network notifying their presence on the road by providing their location
4	Driver	Road user using a motorized vehicle (car)	Moving around the road guided by an On-Board Unit (OBU)	Drivers are connected to the network notifying their presence on the road by providing their location

4.1.2 Inventory of captured data

Table 10 below gives a complete inventory of the various pieces of data collected from PEs by either 1) the four DEDICAT 6G scenarios (UC number in 1st column) or 2) by the DEDICAT 6G platform (D6G in first column) for the sake of its own operation (e.g., to build up a B5G network context).

Table 10: Inventory of collected data

UC/D6G	Actor/Entity	Nature of data	Capture Mode {sensed/declared/assigned/measured)	optional {Y/N}	Privacy policy {Y/N}
2, D6G	Network Equipment	Server energy Consumption	Sensed	N	n/a
		Packet latency	Sensed	N	n/a
		Throughput	Sensed	N	n/a
		Number of connected users (to video streamer)	Sensed	N	n/a
3	Event attendee (Gait Analysis µS)	Pseudonym	Assigned	N	N
		Location	Sensed	N	Y (but data is anonymized)
1, D6G	AGV	Location	Declared / sensed	N	n/a
		System status	Declared	N	n/a
		Proximity of points of interest	Declared / Sensed	N	n/a
		Process status	Declared	N	n/a
1	IoT controller	Proximity of points of interest	Declared / Sensed	N	n/a
		System status	Declared	N	n/a
		Process status	Declared	N	n/a
1	Worker	Location	Sensed	N	Y
		Authorization level	Declared	N	Y
		Task	Declared	N	Y
1	Manager	Location	Sensed	N	Y
		Authorization level	Declared	N	Y
		Task	Declared	Y	Y
1	Warehouse environment	Environmental parameters (temperature, humidity, light intensity)	Sensed	N	n/a
		System status	Declared	N	n/a
2	Event Attendee	Location	Sensed	Y	Y
3	Field Officer	Location	Sensed	N	Y
		Mission status	Declared	N	N
		Proximity of points of interest	Declared / Sensed	N	N
		Health status	Sensed	Y	Y

3	Team member	Location	Sensed	N	Y
		Mission status	Declared	N	N
		Health status	Sensed	Y	Y
3, D6G	Drone	Location	Sensed	N	n/a
		Authorization level	Declared	N	n/a
		Video	Sensed	N	Y
3	Site environment	Environmental parameters (temperature, humidity, light intensity)	Sensed	N	n/a
		System status	Declared	N	Y
3	MCX Smartphones	Throughput	Sensed	N	n/a
		Packet latency	Sensed	N	n/a
		Connectivity	Sensed	N	n/a
3	MCX Server	Throughput	Sensed	N	n/a
		Packet latency	Sensed	N	n/a
		Connectivity	Sensed	N	n/a
3	Victim	Localization	Sensed	N	Y
		Health description	Declared	N	Y
		Picture	Declared	Y	Y
3	Event organizer	Attendees list	Declared	Y	Y
		Attendees localization	Declared	N	Y
		Site mapping	Declared	N	Y
		Security Staff Communication ID	Declared	N	N
4	VRU	Position	Sensed	N	Y (but data is anonymized)
4	Smart & Smarter Cars	Position	Sensed	N	Y (but data is anonymized)
		Speed	Sensed	N	N
		Video / Lidar	Sensed	N	N

4.2 Context View

As explained in Section 2.6.2 this view:

1. Identifies the various physical systems outside the DEDICAT 6G cloud platform and decides if they are part of DEDICAT 6G (IN) or not (OUT) depending on whether or not they sit at the edge of the DEDICAT 6G system;
2. (doing so) defines the perimeter of DEDICAT 6G platform (hence made of Edge + Cloud parts);
3. Defines external interfaces between PSs and DEDICAT 6G;
4. Defines external interfaces between human actors (as identified in the PE view) and DEDICAT 6G

In D2.2 items 3) and 4) were addressed through informal actions (in UML use-case diagrams). In this iteration of the document, we only keep the material relating to items 1) and 2).

4.2.1 Defining the perimeter of the DEDICAT 6G system

In order to help defining the perimeter of the platform, i.e., what components lie inside and outside the system, we start with an inventory of:

1. All physical systems which are not physically part of the platform;
2. All *micro-services (µS)* and FCs part of those physical systems which are either part of the platform or considered as being at its edge.

Note 1: please note that in the rest of this document µSs always refer to components belonging to a Vertical (UC holder) while FCs always refer to a component which is part of the platform architecture. Both sorts of components may be deployed at the edge as part of dynamic intelligence distribution.

The following Table 11 gives this inventory and provides additional information, whose columns are described below:

- **UC/D6G**: which UC (UC number) is at the origin of the whole table row and/or "D6G" if the PS is a *de facto* B5G or network extension element (used by scenario but considered as integral part of DEDICAT 6G system);
- **PS name**: Physical System name (device/equipment/ smart phone/robot etc.);
- **PS@edge**: PSs which are candidate for edge computing (meaning they could execute µSs and FCs dynamically migrated from the platform). Must obviously be "Y" if the list of edge µS/FC is non-empty;
- **Edge µS/FC**s: µS/FCs which will be hosted by the PS i.e., dynamically migrated from the cloud;
- **Legacy µS/FCs**: µS/FCs which are provided by the PS by default (excluding migrated components which are listed in edge µS/FC column);
- **Description/comment:** can be used to describe µSs/FCs informally if they are not individually identified yet.

Note 2: Part of this inventory (e.g., robots, smart vehicle) is also described in the PE View, however it is focusing on completely different matter, as explain in the PEV introduction.

Note 3: A PS can be listed as an Edge node (PS@edge="Y") even if the list of edge µS/FCs is empty. It simply means that no edge µS/FC has been identified by the scenarios for that particular PS. However, any business application willing to use that particular PS could instruct the DEDICAT 6G system to set-up a migration policy leading to migrating some business-specific µS to that edge PS (see the IDaaS scenario in Section 4.3.4.10).

Table 11: Inventory of physical systems as identified in the 4 Scenarios

UC / D6G	PS name	PS@ edge (Y/N)	Edge µSs/FCs	Legacy µSs/FCs	Description/ comment
D6G	Small server	Y	anything required	n/a	Depends mainly on 1) the verticals and the sort of services they would like to deploy to such servers 2) the 5G components that would have to be deployed in the context of Coverage Extension

1,3 D6G	AGVs	Y	AGV/Robot capabilities such as camera operation, remote control, Image processing, providing wireless connectivity to other nodes deployed as Virtual Network Functions (VNFs)	n/a	AGVs may be used as sensors and actuators as well as MAPs. To support this type of functionality services will be deployed on AGVs as VNFs
1	Forklift/machine	N	n/a	n/a	Mainly "talks" with SmartAccess360 controller/cloud
1, 3	SmartAccess360 controller	Y	• Indoor positioning µS • Env Sensing µS • Threat Analysis FC • Trust metric FC	• IoT controllers	Indoor Positioning FC and Env Sensing FC are vertical FC that are meant to be deployed to the edge by D6G EC capabilities. Threat Analysis FC will be deployed by D6G by default to all EC nodes as part of the PST strategy
1	Warehouse personnel smartphone/mobile device	Y	• Threat analysis FC • AR µS • D6G Consent-Form FC • Trust Metric FC	• SmartAccess 360 app	SmartAccess360 application used to turn a smart phone into a digital key for smart actuation and managing doors and access control to areas
1,2,3, D6G	(B)5G Networking Equipment	Y	• Multicast-Broadcast Service µS • Unicast-Multicast Controller µS	n/a	5G Network equipment is considered as an external layer of D6G, and further described in section 4.3.3
2	Video streaming platform	Y	• VideoTrans-Coder µS	n/a	Video transcoding from high quality to adaptive streaming
3/ D6G	Drones	Y	Services such as camera operation, remote control, Image processing, providing wireless connectivity deployed as Virtual Network Functions	n/a	Similar to AGVs, drones may be used as sensors and actuators as well as MAPs. To support this type of functionality services will be deployed on Drones as VNFs
2	Smart phones	N	n/a	• video-Streamer app	
2	Smart Glass	N	n/a	• Video&Audio Capture & Compression FC • EventGUI FC	FC that uses the Video and Audio capability of the smartGlass to capture a live event
3, D6G	Connected Car (maybe differ-	Y	• MCS µS	n/a	Delivers Mission Critical services in order to add a network cov-

	ent from UC1)				erage in case of infrastructure lost or support the coverage in case of large event and infrastructure overload or failure due to the amount of connection during the disaster
3	MCS mobile server	N	n/a	• MCS	In case of a disaster, first responders may have in their vehicles part of the MCS-cloud functionalities deployed to a mobile PC/server in order to maintain the provision of needed MCS functionalities despite the likely lack of connectivity. (not deployed by DEDICAT 6G)
3	Attendee smartphone	Y	• Crowd Analysis µS	• Gait_Analysis app • Attendee Consent Form FC	FC developed for the sake of UC3 only, reports gait to Crowd Analysis µS
			n/a	• Attendee App	App specially designed for the event attendees (organized context). includes request for personal information that are handled by the event organizers only and a first consent Form (for eventually sharing this information with event deployed medics)
			• Dedicat6G_Consent Form app	n/a	As soon as the EventApp consent form has been agreed upon D6G is requested to send a D6G-specific consent form for movement tracking. This FC is deployed to the Attendee Smartphone
3	1st Responder smart phone	N	n/a	• First ResponderApp • SmartAccess 360 mobile App	The first responder must be able to reach out to the event organizer system to access a person's health record by scanning the event bracelet (QRcode) as part of the event Context. SmartAccess360 mobile application FC can be integrated into the First Responder app to allow interaction with SmartAccess360 controllers
3	smart glass	N	n/a	• GUI_FR • Video&Audio Capture & Compression	They provide a GUI to first responders
		N	n/a	• GUI_SEC • Video&Audio Capture & Compression	They provide a GUI to SECurity staff

3	smartAcces360	Y	• AuthZ FC • AuthN FC • IdM FC	• IoT Controller • SmartActuation	This H/W-based version of smartAccess360 will be physically deployed at the gate side and run some dynamically deployed DEDICAT 6G FCs (as edge FCs). SmartActuation perform access control based on the AuthN/AuthZ/IdM FCs
3	SmartGate	N	n/a	• GateController	They provide interface to the actuable gate / turnstile
3	Event Information System	N	n/a	• Gateway service component	Deliver interfaces to connect External System to DEDICAT 6G platform in order a specific External System could continue to run its proper security services (e.g., video surveillance during a crisis management on a large event)
4	Smart vehicle	N	n/a	• Driver-Awareness	They report information about other cars and VRUs
4	Smarter vehicle (incl. tablet-like terminal)	Y	• Road Information processor µS • Video&Lidar Capture µS	• Driver-Awareness	n/a
4	Sensing Nodes	Y	• Env Sensing µS	n/a	They also send raw data from other PS if needed
4	RSU	Y	• Road Information processor µS	n/a	A RSU also features a WIFI access point, therefore providing connectivity in the surroundings
4	VRU's smart phone	N	n/a	• VRU-Awareness	Vulnerable Road User app

This next Table 12 gives additional information about the µSs/FCs introduced in the table above.

Table 12: µSs/FCs at the edge or outside the DEDICAT 6G perimeter

µS/FC name	µS /FC Edge (Y/N)	Short description
Gait Analysis app	N	Based on attendee's mobile phone 3-axis accelerometer data and compass this FC infers respectively attendee's gait (standing immobile, walking, running, etc.) and her overall direction
Crowd Analysis µS	Y	Receives individuals' gaits and generates a crowd status
Attendee App µS	Y	Provides a registration page, consent form page and the gait Analysis FC as a µService. This later one gets deployed and activated remotely if and only if the attendee agrees to the T&C of the consent form
Indoor Positioning µS	Y	Provides indoor location of entities/actors based on Bluetooth beacons and triangulation. The associated coordinate system is local to the warehouse map
Env. Sensing µS	Y	Collects and returns elements of environmental context
IoT Controller µS	N	Is a native functionality from the SmartAccess360
Augmented Reality µS	Y	Augmented reality overlay for smart glasses and smartphone applications

		(using camera) enabling navigation and context-aware notifications and points of interest to users in the context of the use-case
SmartAccess 360 mobile app	N	Part of the SmartAccess 360 solution residing on the end-user mobile devices. The app allows users to utilize granted digital access keys for doors and other smart actuation points (e.g., turn on/off lights etc.)
Multicast-Broadcast-Service µS	Y	Provides a video multicast/broadcast service for cellular networking
Unicast-Multicast Controller µS	Y	Part of the video multicast/broadcast service. Provides ability to switch dynamically from unicast mode to multicast and vice versa
Video Transcoder µS	Y	Receives high quality video streams from user devices (uplink) and transcodes the streams to adaptive representations
Threat Analysis FC	Y	FC to be instantiated at all Edge nodes. Runs ML models for cyber security threat detection and identification. Updated/trained on collected system logs. Performed in federated learning mode
Trust Metric FC	Y	FC to be installed at all edge nodes. Trustworthiness metrics are calculated for edge nodes, processes, users and data streams. Trust metric value indicates if a node can join a local network, if process output can be further used, if user can execute specific rule
AuthZ FC	Y	Native DEDICAT 6G Authorization functionality with role and attribute-based authorization and access control. Applied on users and devices
AuthN FC	Y	Native DEDICAT 6G Authentication functionality applied to end-users. Roles defined on the level of the project and for each UC
IdM FC	Y	Native DEDICAT 6G identity Management functionality used for assigning identity and roles to systems, devices and users
GateController app	N	This component offers an interface to lock/unlock a smart actuable door/turnstile
Attendee Consent Form app	Y	Displays a consent form where the given/Family names are given and where the individual agrees/denies movement tracking. Agreeing will either unlock the tracking functionality
Video&Audio Capture app	N	Captures ambient audio and video from the smartGlass
Video-Streamer app	N	Native application in the smartPhone used for video/audio streaming (when smartGlass is not available for that purpose)
MCS app	N	subset of the MCS cloud based MCS platform
DriverAwareness app	N	Reports information about other cars and VRUs
VRU-Awareness app	N	Application used to keep the VRU aware of imminent danger or road environment (nearby cars, other VRUs (i.e., pedestrians)
RoadInformation Processor µS	Y	Analyses and processes the information before pushing the result to the RSU smartPhone
Video&LiderCapture µS	Y	Captures video and Lidar streams (i.e., collects information about the car surroundings)
Video&Audio Capture & compression app	N	Native FC in the smartGlass that performs Video/Audio capture and compression in order to save bandwidth
MCS Audio µS	Y	MCS Audio FC is a functional component, which delivers audio Push-To-Talk and Full-Duplex over IP communication feature to users. The component takes into account *QoS Class Identifier (QCI)* for connectivity priority Several MCS Audio FC can be deployed to support efficiency and resiliency of voice communication
MCS Video µS	Y	MCS Video FC is a functional component, which delivers real-time video over IP communication or streaming feature to users. The component takes into account QCI for connectivity priority. Several MCS Video FC can be deployed to support efficiency and resili-

		ency of video streaming
MCS Situation µS	Y	MCS Situation FC is a functional component, which delivers operational situation feature to users. The component takes into account QCI for connectivity priority. This component transmits operational situation to PPDR or Public Safety organizations' infrastructure
MCS Location µS	Y	MCS Location FC is a functional component, which delivers location and mapping feature. This component takes QCI into account for connectivity priority. It transmits users position to PPDR or Public Safety organizations' infrastructure
MCS Registration µS	Y	MCS Registration FC is a functional component which delivers registration and authentication features

4.2.2 UML sequence diagrams Use Case example

Deliverable D2.3 has provided a quite detailed UML description of the interactions taking place in UC1-4 between the various components (edge, cloud), going one step beyond the content of this exact same section of D2.2.

However, those UML diagrams were not considering any explicit timing and did not give much formal detail about the nature of the interactions and content exchanged, meaning they did not relate to any precisely defined interface.

In order to overcome those two limitations, we provide here a few examples of sequence diagrams which aim at showing how the various *System Use-Cases (SUC)* elucidated in Section 4.3.4 can be applied to the somehow static project Use Cases UML diagrams like displayed in D2.3.

In that matter, we shall focus only on those D2.3 UML diagrams that directly refer to some of the SUCs, ignoring those which were focusing on the "business part" of the description and which are for obvious reasons not part of Section 4.3.4., as not directly system related.

We will consider here only one UC and focus on the aspects that are common to all of them (user registration, authentication and CEaaS/IDaaS request). Since CEaaS covers also intelligence distribution, we will select "Enhanced Experience" (UC2) in the following of this section as it show-cases both CE and ID aspects.

Those UML sequence diagrams focus on interactions taking place between (external) entities -either they are human actors or physical systems- and the DEDICAT 6G overall system (made of edge-based and cloud-based components). They feature a list of vertical bars (also called lifelines) representing the interacting parties (human actors, µServices and functional components) and a set of horizontal arrows that materialize interactions.

As far as interactions are concerned, plain arrows can be understood as synchronous method invocations (if we refer to object-oriented paradigm) while shallow arrows follow the asynchronous paradigm, relying mostly on the exchange of asynchronous messages (one request followed by one response) between the two communicating parties, like we would do using REST protocol. In the few following examples, we will mostly use synchronous calls when they can be answered straight away (e.g., invoking a registry method) and asynchronous calls and responses when they entail a chain of additional interactions involving more lifelines.

4.2.2.1 CEaaS example (taken from "Enhanced Experience" UC2 UML)

In this section we revisit some the UC2 UML diagrams provided in D2.3 (see Figure 6 and Figure 7 below) and translate them into UML sequence diagrams, using the FC logical interfaces described in Section 0.

As a reminder this first figure below is all about a vertical (UC2 then) requesting a *Coverage Extension as a Service (CEaaS)*. Before being able to make the service request, a Vertical operator needs being known from the platform and consequently needs also being authenticated.

The two first actions are then 1/ user registration and 2/ user authentication (login).

Then follows the CEaaS service request it-self, which is a negotiation between the user and the platform following the ContractNET[4] protocol (more detail is available in Section 4.3.4.8).

After the service terms have been agreed upon by both parties (and embodied into a *Service Level Agreement (SLA)*), several uploads which are necessary for completing the IDDM and CEDM FCs service set-ups are performed. Finally, UC2 servers (ENs) are deployed and made ready for configuration by the platform.

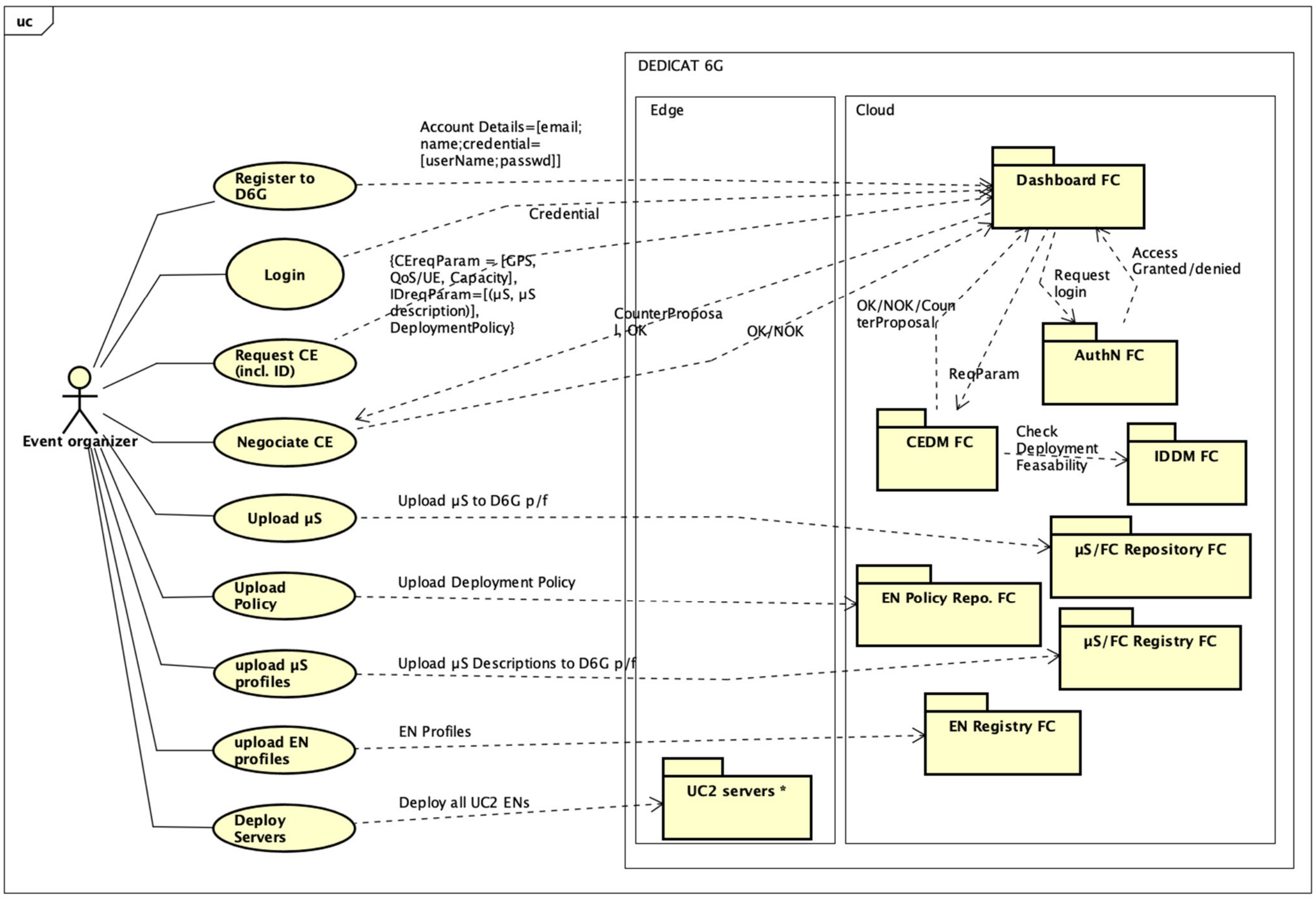


Figure 6: UC2 CEaaS set-up - as shown in D2.3.

[4] https://en.wikipedia.org/wiki/Contract_Net_Protocol

The second Figure 7 below focusses on all steps resulting from the subsequent decisions taken by the CEDM and IDDM FCs, namely network extension dimensioning, MAP and EN configuration including FCs and µSs deployment and MAP physical deployment on-site. At the end of that phase, the CEaaS is ready to operate.

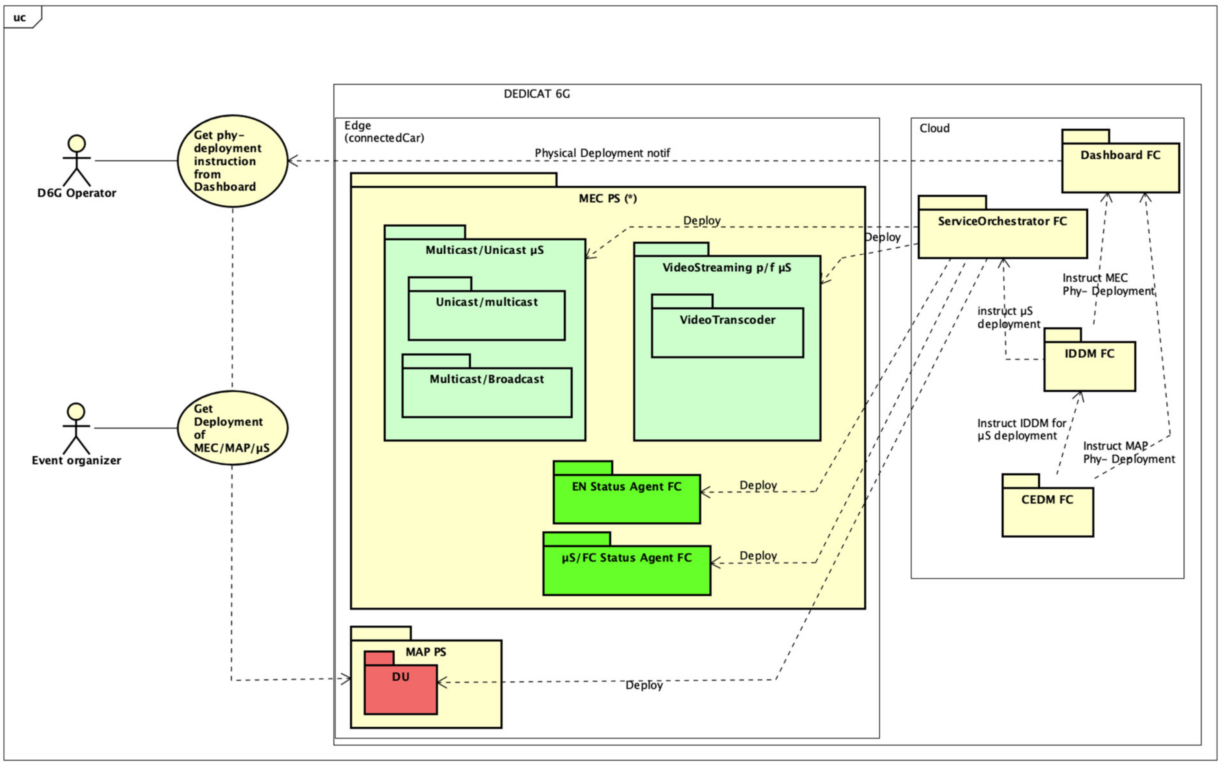


Figure 7: UC2 Deployment phase (MAPs/ENs/µSs/FCs) - as shown in D2.3.

In order to complete the picture, we now give information in Table 13 about the UC2/DEDICAT 6G servers, alongside their capabilities and resources. We also give some run-time requirements pertaining to the deployed µServices in Table 14. Those µSs - together with some FCs – are meant to be deployed using those 4 UC2 servers and 1 additional Connected Car supplied by DEDICAT 6G. These servers consist of rack servers, which are also used in the actual 5G network set-up in order to run the core functionalities as well as routing the network infrastructure in 5GTN Oulu facility.

Table 13: List of Servers and Characteristics (UC2)

	Connected Car	Server A	Server B
Max Energy Consumption (Watts)	240 (TDP)	210 (TDP)	410 (TDP)
Number of CPU cores/sockets/threads	48/2/96	20/1/40	48/2/96
Number of GPU Cudas	-	-	4608
Max CPU RAM (Gbyte)	128	128	384
Max GPU RAM (Gbyte)	-	-	24
MAX throughput (Mbytes/s)	1000	1000	1000
Max Disk Space (Gbyte)	1000	1000	2000
Instance Number	1	1 (A#1)	3 (B#1, B#2, B#3)

Table 14 below, shows the expected performance requirements for the identified µServices. The values have been gathered from the actual machines depicted in Table 13 by actuating the µServices and monitoring their utilization with the Linux system-level tools. The VideoStreaming p/f µS is considered as the lightest operational service which can be run and placed in the ConnectedCar as an edge service. It will serve both the unicast as well as µS Multicast/Unicast video service for multicast users according to the unicast-multicast controlling policy, which is initially based on the number of video service users. The µS Multicast/Unicast video service operates also with lightweight fashion for being only responsible of running the multicast service together with 5G core and gNB.

Far most the heaviest computing units are required by the µS VideoTranscoders, which are responsible of transcoding the requested video representations as a distributed fashion orchestrated by IDDM decision block. Initially, we are relying on CPU based SW encoding/transcoding instead of GPU in order to enable better dynamic flexibility for real-time video input/output functions between µS VideoStreaming p/f and µS VideoTranscoder units. The initial experiments using the Intel Xeon rack server indicate that 4K video encoding for 40 Mbit/s utilizes approximately 25% of the CPU with the basic video parameters with fastest encoding options. These run-time requirements will be elaborated during the actual measurements within UC2.

Table 14: List of µServices and their run-time requirements

	VideoStreaming p/f µS	Multicast/Unicast µS	VideoTranscoder µS
Needed nbr of CPU cores/threads	4/8	4/8	12/24
Needed nbr of GPU CUDAs	n/a	n/a	n/a
Needed amount of CPU RAM (in Gbyte)	1	4	12
Needed amount of GPU RAM (in Gbyte)	n/a	n/a	n/a
Needed throughput (in Mbyte/s)	200	20	100
Number of Instances required	1	1	1~3 (at least) dynamically scaled up
Needed Disk Space (Gbytes)	< 0.1	< 0.1	< 0.1

We now elucidate the corresponding three sequence diagrams and will use the information from those 2 tables above, in order to populate the needed data structures for the second sequence diagram dealing with CEaaS service request.

The first sequence diagram (Figure 8) covers the two actions (yellow oval) "Register to D6G" and "login" introduced in Figure 6, and goes through that preliminary phase where a user 1/ registers to the DEDICAT 6G platform) and 2/ authenticates. When the authentication succeeds, two authentication tokens are sent back for further use, e.g., to be granted access to a resource or to interact with component.

The interface we are using in the diagram follows the AuthN/AuthZ interfaces defined in Section 4.4.1.7.

In this Figure 8, we have both registration and authentication from the user:

- User registration to the platform:
  - User's details are entered through the "register" widget and passed on to AuthN FC (arrow 1 & 2);
  - AuthN FC then requests AuthZ FC to allocate an authorization profile to the new user (arrow 3);
  - Then follows a series of confirmation messages (arrows 4-6);
- User authentication (logging in):
  - The user's credentials (username and password) are passed on to the Dashboard FC (arrow 7) which in turn invokes the AuthN FC authentication method (arrow 8);
  - AuthN requests AuthZ to create and return the authentication token and refresh token (arrows 9-12);
  - The tokens are "passed back" to the user for further use (arrows 13-14). Those tokens are actually managed by the dashboard as part of the vertical's user personal account and are not explicitly manipulated by the user.

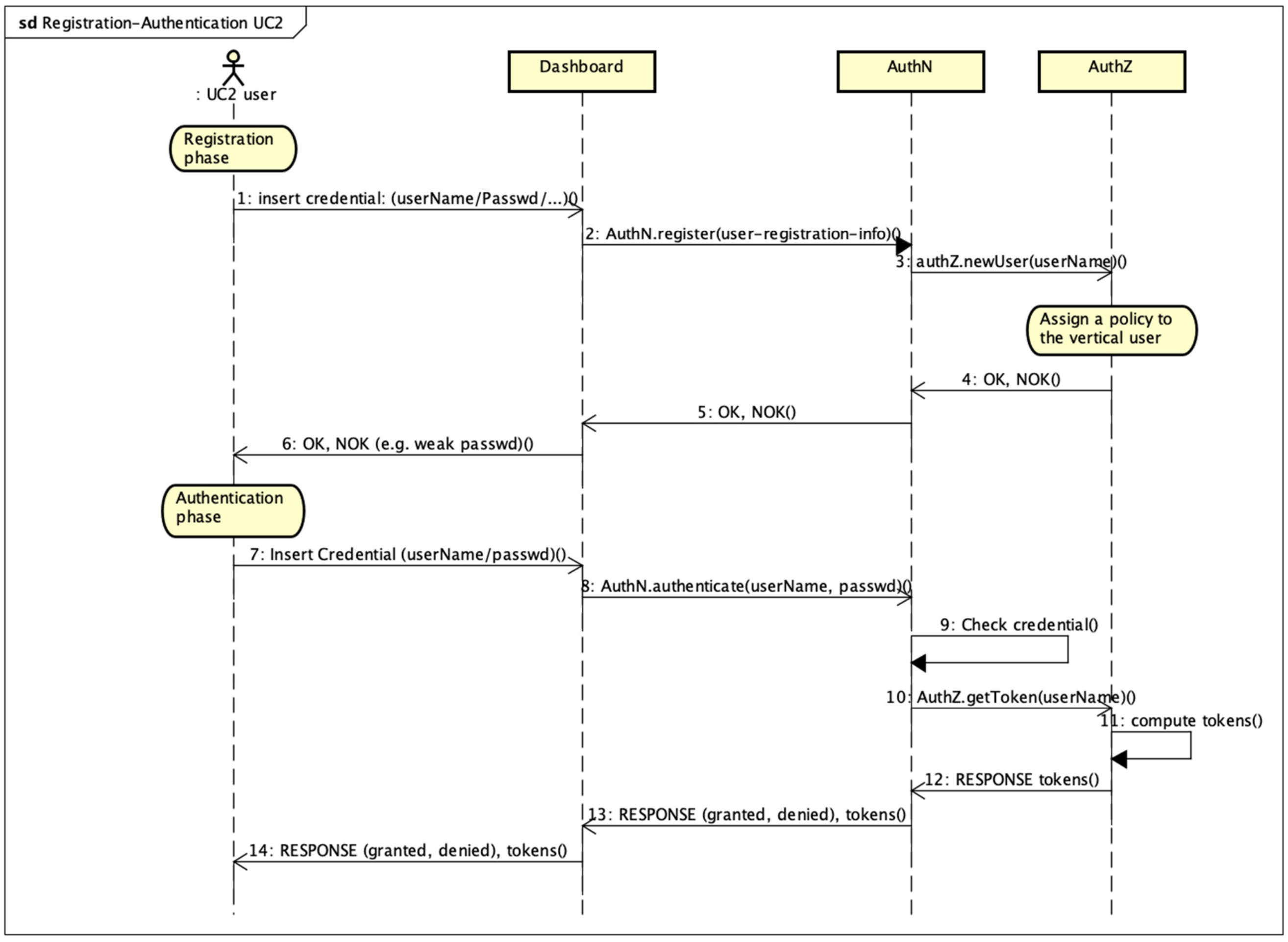

sd Registration-Authentication UC2
: UC2 user
Dashboard
AuthN
AuthZ
Registration phase
1: insert credential: (userName/Passwd/...)()
2: AuthN.register(user-registration-info)()
3: authZ.newUser(userName)()
Assign a policy to the vertical user
4: OK, NOK()
5: OK, NOK()
6: OK, NOK (e.g. weak passwd)()
Authentication phase
7: Insert Credential (userName/passwd)()
8: AuthN.authenticate(userName, passwd)()
9: Check credential()
10: AuthZ.getToken(userName)()
11: compute tokens()
12: RESPONSE tokens()
13: RESPONSE (granted, denied), tokens()
14: RESPONSE (granted, denied), tokens()

Figure 8: UC2 – a user registers and authenticates to the platform

The second sequence diagram (Figure 9) keeps on elaborating on the UML diagram, as shown in Figure 6, and provides the interactions that cover actions "Request CE" until "Negotiate CE".

It shows the CEDM FC and the IDDM FC in action. Since we are illustrating the CEaaS case, the CEDM FC leads the whole operation, but would rely on the IDDM FC for some parts of the coverage extension request that deal specifically with dynamic intelligence distribution. Those steps are:

- The customer requests the CE via the Dashboard FC web front-end, filling in all requested information concerning the ENs used, µSs to be deployed if any, characteristics of the requested extended coverage etc. (arrow 1);
- The Dashboard FC translates the information gained from previous step into a data structure and passes it on to the CEDM FC (arrow 1.1);
- The CEDM FC does some dimensioning for coverage extension and comes up with the needed MAPs - which happen to be also edge nodes. It then asks the IDDM FC to check if the vertical ENs and the MAPS can provide enough resources to execute the µSs as advertised in step 1 (first yellow bubble + arrow 1.1.1);
- The IDDM FC retrieves the characteristics of the MAPs, assuming the ones from the vertical are already known (they are part of the request) (arrow 2);
- Then the IDDM FC runs its task allocation algorithm (second yellow bubble) and finds out that the CEaaS service can be granted from the IDDM point of view (arrow 3);
- The CEDM FC confirms the Dashboard FC that the CE can be granted (arrow 4);
- Then the customer confirms the service request (arrow 5);
- Final confirmation is informed to the CEDM FC (arrow 6). The `serviceID` is now valid throughout the whole service delivery.

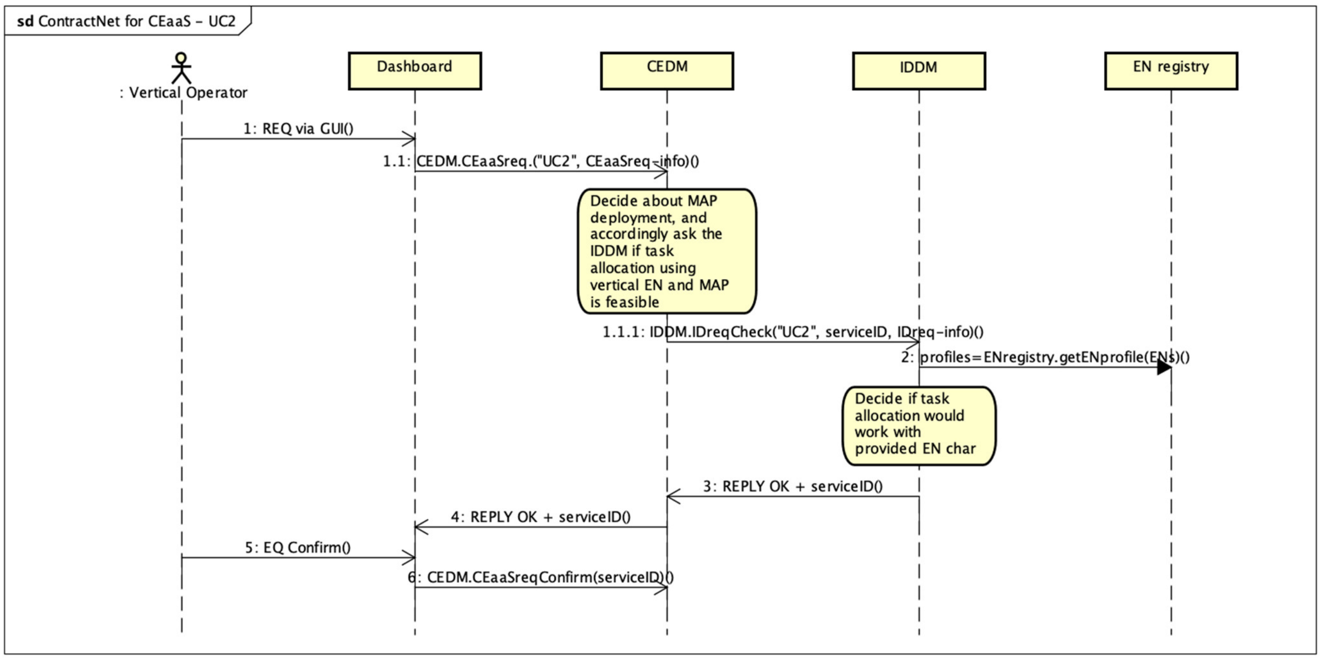

sd ContractNet for CEaaS – UC2
: Vertical Operator
Dashboard
CEDM
IDDM
EN registry
1: REQ via GUI()
1.1: CEDM.CEaaSreq.("UC2", CEaaSreq-info)()
Decide about MAP deployment, and accordingly ask the IDDM if task allocation using vertical EN and MAP is feasible
1.1.1: IDDM.IDreqCheck("UC2", serviceID, IDreq-info)()
2: profiles=ENregistry.getENprofile(ENs)()
Decide if task allocation would work with provided EN char
3: REPLY OK + serviceID()
4: REPLY OK + serviceID()
5: EQ Confirm()
6: CEDM.CEaaSreqConfirm(serviceID)()

Figure 9: UC2 - ContractNET negotiation for CEaaS

We give some examples of populated data structures as used in the figure above and as defined in the Information view in respectively, Sections 4.4.1.2 (CEaaS request), 4.4.1.5.2 (µS/FC Registry FC), 4.4.1.5.3 (EN Registry FC) and 4.4.1.5.4 (EC Policy).

`CEaaSreq-info={"UC2", "000124551", CEreq-info, IDreq-info}` - in arrow 1.1 where

```
CEreq-info= {   coverageType=terrestrial,
                terrestrialType=connectedCar
                coverageAreaRadius=100,
                location=(…,…),
                reqCapacity=300,
                reqMinThroughput=1,
                maxLatency=10,
                connectivityExtensionType-5g
                startTime:"21-june-2022",
                endTime:"22-june-2022"} and
IDreq-info=
    { [   ("serverA-class",1,["serverA#1"], - Vertical EN descriptions
                (20, 40), 0, 128, 0, 1000, 1000),
          ("serverB-class, 3, ["serverB#1","serverB#2","serverB#3"],
                (48,96),4608,384,24,1000,2000)],
      [   ("videoStreaming-class",1,(4,8),0,1,0,0.1,200), -µS descriptions
          ("multicastUnicast-class",1,(4,8),0,4,0,0.1,20),
          ("videoTranscoder-class",3,(12,24),0,12,0,0.1,100)],
      { "UC2", "CEaaS-service#1", - EC policy (preferred task→EN allocation)
          {[    ("multicastUnicast-class",[("serverB#3, 1)]),
                ("videoTranscoder-class",
                      [("serverA#1",1,EN),
                      ("serverB#1",1,EN),
                      ("serverB#2",1,EN)]]},
          {[]} – no µS dependencies
    }
```

It is worth mentioning that in the structure above, we have set deployment preferences for only 2 µSs among the 3 used in UC2 (we omitted videoStreaming µS).

Because ultimately this is the purpose of the IDDM FC to optimally place and eventually scale FCs up among available ENs, we could have put only one constraint on Multicast/Unicast µS as it has to be deployed that way for operational purpose (See UC2 Network Deployment view Section 4.5.3).

The 3rd (and final) example in Figure 10 below is slightly more complex and features several phases:

- **Uploads:** The ENs profiles, µSs/FCs profiles and images and the overall UC2 EC policy need being uploaded/created in their respective repository or registries (arrows 2-14);
- **MAP configuration**: all MAP specific components need being deployed (arrow 15-25)
- **FC deployments:** all by default FC deployment must be applied to the ENs. For the sake of space, we have condensed into 5 arrows the deployment of all needed FCs (arrows 26-39);
- **Provision of information prior to task allocation:** a new EN context is retrieved (arrows 40);
- **Task allocation and deployment:** the IDDM FC computes the best "task → EN" allocation and instruments the Service Orchestrator FC to implement the task allocation plan. Finally, it informs the CEDM FC when the deployment is completed (arrows 41-54);

- **Physical MAP deployment and CEaaS completion:** the CEDM FC request the DEDICAT 6G technical staff to perform the physical deployment of the MAP on-site and get informed when it is completed (arrows 55-60).

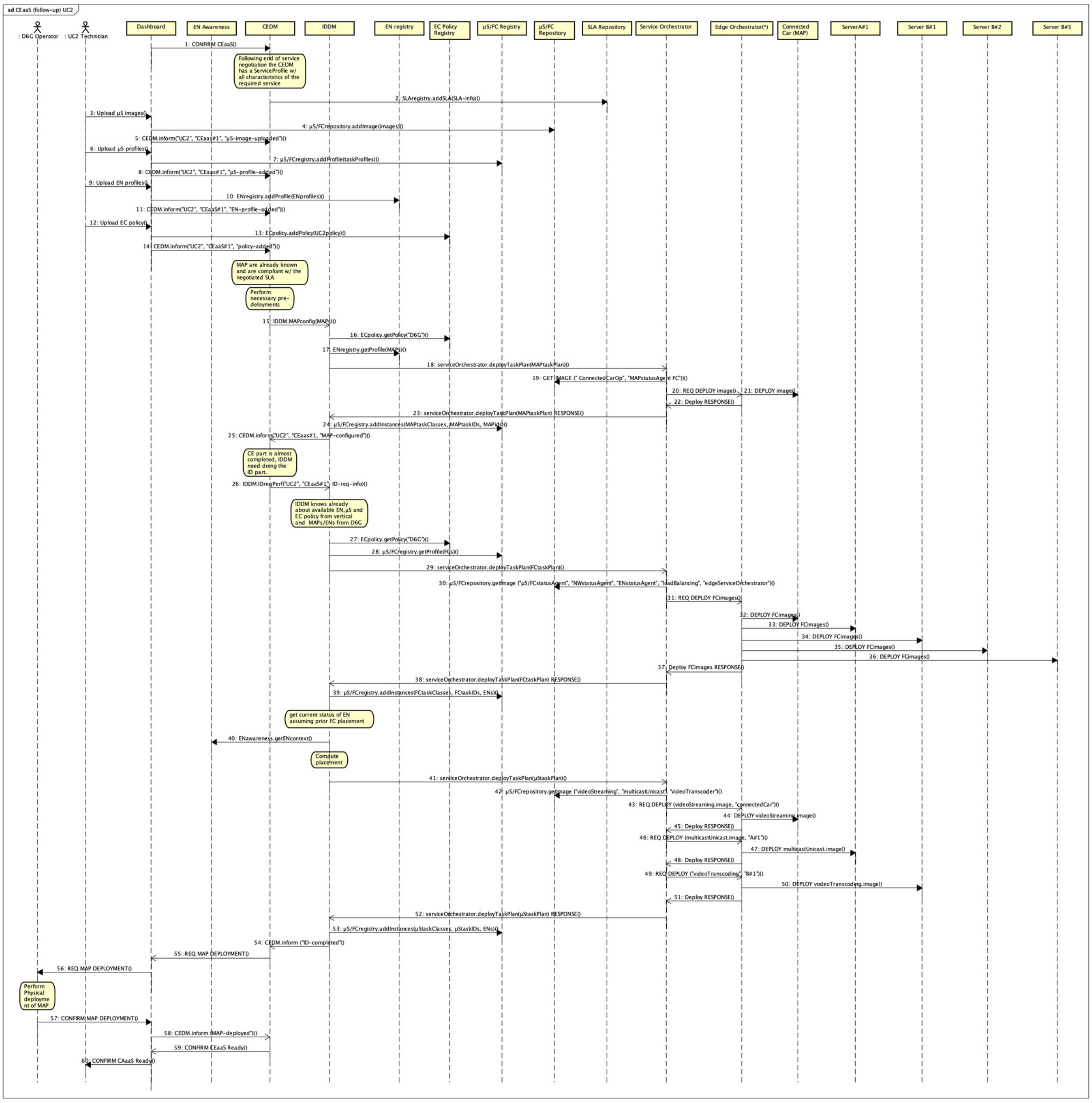

sd CEaaS (follow-up) UC2
: D6G Operator
: UC2 Technician
Dashboard
EN Awareness
CEDM
IDDM
EN registry
EC Policy Registry
µS/FC Registry
µS/FC Repository
SLA Repository
Service Orchestrator
Edge Orchestrator(*)
Connected Car (MAP)
ServerA#1
Server B#1
Server B#2
Server B#3
1: CONFIRM CEaaS()
Following end of service negotiation the CEDM has a ServiceProfile w/ all characteristics of the required service
2: SLAregistry.addSLA(SLA-info)()
3: Upload µS images()
4: µS/FCrepository.addImage(images)()
5: CEDM.inform("UC2", "CEaas#1", "µS-image-uploaded")()
6: Upload µS profiles()
7: µS/FCregistry.addProfile(taskProfiles)()
8: CEDM.inform("UC2", "CEaas#1", "µS-profile-added")()
9: Upload EN profiles()
10: ENregistry.addProfile(ENprofiles)()
11: CEDM.inform("UC2", "CEaaS#1", "EN-profile-added")()
12: Upload EC policy()
13: ECpolicy.addPolicy(UC2policy)()
14: CEDM.inform("UC2", "CEaaS#1", "policy-added")()
MAP are already known and are compliant w/ the negotiated SLA
Perform necessary pre-deloyments
15: IDDM.MAPconfig(MAPs)()
16: ECpolicy.getPolicy("D6G")()
17: ENregistry.getProfile(MAPs)()
18: serviceOrchestrator.deployTaskPlan(MAPtaskPlan)()
19: GET IMAGE (" ConnectedCarOp", "MAPstatusAgent FC"))()
20: REQ DEPLOY image()
21: DEPLOY image()
22: Deploy RESPONSE()
23: serviceOrchestrator.deployTaskPlan(MAPtaskPlan) RESPONSE()
24: µS/FCregistry.addInstances(MAPtaskClasses, MAPtaskIDs, MAPids)()
25: CEDM.inform("UC2", "CEaas#1, "MAP-configured")()
CE part is almost completed, IDDM need doing the ID part.
26: IDDM.IDreqPerf("UC2", "CEaaS#1", ID-req-info)()
IDDM knows already about available EN,µS and EC policy from vertical and MAPs/ENs from D6G
27: ECpolicy.getPolicy("D6G")()
28: µS/FCregistry.getProfile(FCs)()
29: serviceOrchestrator.deployTaskPlan(FCtaskPlan)()
30: µS/FCrepository.getImage ("µS/FCstatusAgent", "NWstatusAgent", "ENstatusAgent", "loadBalancing", "edgeServiceOrchestrator")()
31: REQ DEPLOY FCimages()
32: DEPLOY FCimages()
33: DEPLOY FCimages()
34: DEPLOY FCimages()
35: DEPLOY FCimages()
36: DEPLOY FCimages()
37: Deploy FCimages RESPONSE()
38: serviceOrchestrator.deployTaskPlan(FCtaskPlan) RESPONSE()
39: µS/FCregistry.addInstances(FCtaskClasses, FCtaskIDs, ENs)()
get current status of EN assuming prior FC placement
40: ENawareness.getENcontext()
Compute placement
41: serviceOrchestrator.deployTaskPlan(µStaskPlan)()
42: µS/FCrepository.getImage ("videoStreaming", "multicastUnicast", "videoTranscoder")()
43: REQ DEPLOY (videoStreaming.image, "connectedCar")()
44: DEPLOY videoStreaming.image()
45: Deploy RESPONSE()
46: REQ DEPLOY (multicastUnicast.image, "A#1")()
47: DEPLOY multicastUnicast.image()
48: Deploy RESPONSE()
49: REQ DEPLOY ("videoTranscoding", "B#1")()
50: DEPLOY vodeoTranscoding.image()
51: Deploy RESPONSE()
52: serviceOrchestrator.deployTaskPlan(µStaskPlan) RESPONSE()
53: µS/FCregistry.addInstances(µStaskClasses, µStaskIDs, ENs)()
54: CEDM.inform ("ID-completed")()
55: REQ MAP DEPLOYMENT()
56: REQ MAP DEPLOYMENT()
Perform Physical deployment of MAP
57: CONFIRM MAP DEPLOYMENT()
58: CEDM.inform (MAP-deployed")()
59: CONFIRM CEaaS Ready()
60: CONFIRM CAaaS Ready()

Figure 10: UC2 - CEaaS provision (following the service negotiation)

4.3 Functional View

In this section, which has been extensively updated and extended compared to its D2.2 version, we shed the light on the main functional pillars of the DEDICAT 6G architecture as outlined within the DoA. We give an updated list of the functional components that are necessary to achieve the technical objectives and to realize the vision of the DEDICAT 6G project. Those components were identified through analyzing the FREQs on the one hand and from resolving the perspectives on the other hand (via NFREQs analysis and Design Choices).

The first sub-section introduces and describes shortly those pillars (functional group), before delving into each one of them, elucidating the functional components they are made of.

Most importantly, we conclude this section with an extended set of system use-cases that illustrate typical execution patterns occurring during the platform operation with emphasis on the novel functionalities such as dynamic intelligence distribution and coverage extension.

4.3.1 Introducing the DEDICAT 6G Functional Model

This section sets the foundation of the functional decomposition by identifying the essential FGs the DEDICAT 6G (cloud and edge) platform will be built upon. It also sets the different abstraction layers, one or several FCs may belong to (see Figure 11 below).

During the requirement mapping activity taking place at the end of the requirement process, we identify which FG a given requirements fits in and start identifying relating FCs. The result of this process is the Functional Decomposition.

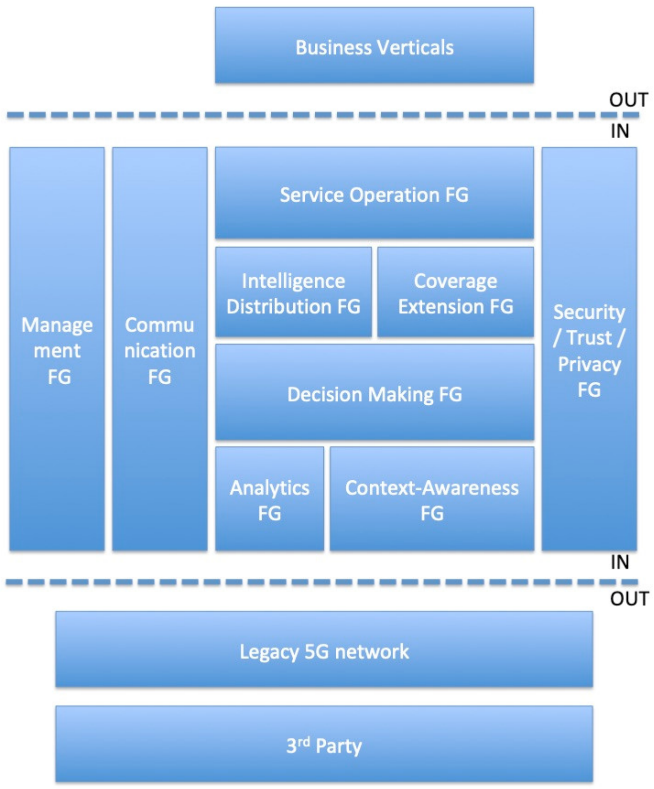


Figure 11: DEDICAT 6G Functional Model

We now explain each of the functional groups inside the perimeter of DEDICAT 6G and the nature of the FCs they are hosting:

- **Service Operation FG:** this FG deals with FCs that are responsible for implementing the decisions carried out by the decision-making FCs. This includes in particular the orchestration, edge orchestration and load balancing of network services, vertical micro-services and platform FCs;
- **Coverage Extension FG:** this FG contains FCs that are supporting the dynamic coverage extension like MAP (drones, connected cars, robots,..) operation including autonomy management, placement management, etc.
- **Intelligence Distribution FG:** this FG contains all FCs that are supporting the overall platform operation. They include registries and repositories (including look-up) for the ENs, µSs and FCs. It also includes FCs dealing with service level agreement and FC/µS deployment/migration policies. those FCs are commonly used by the components belonging to the Context-Awareness, Service Operation and Decision Making FGs;
- **Decision Making FG:** this FG hosts FCs dealing with all aspects of using available information and knowledge provided by Context-Awareness layer in order to take decision relating to different matters such as UE/MAP association, intelligence distribution and coverage extension. They are used in various contexts which include e.g., migrating components due to envisioned or observed poor network performance or network failure, deploying drones to overcome degraded radio coverage;
- **Context-Awareness FG:** this functional group contains FCs that are used to build up "contexts" that can be used to take proper decisions such as the ones introduced above. More precisely a context is a collection of pieces of information that gives the needed level of detail and characterization of a system state (or space). It is usually initially based on raw data, which is enriched using various methods (like correlation, analytics, machine learning) in order to reach the level of "knowledge" (following the well-known Data-Information-Knowledge-Wisdom classification [10]). The decision-making components can then use this knowledge to take the right decisions concerning various sorts of problems;
- **Analytics FG:** this FG contains FCs dealing with data analysis for various purposes e.g., for providing better QoE/QoS to user, or decreasing the UE processing by shifting the computation to edge nodes. The Analytics FG contains FCs such as network optimization and analytics toolbox aligned with in particular performance monitoring and decision-making needs.

In addition, we have three transversal FGs in which the FCs are expected to potentially interact with all FCs from the six previous FGs:

- **Management FG**: this functional group includes all aspects of DEDICAT 6G management (see FCAPS classification) e.g., monitoring of deployment, of network performances, or system failure, dashboards, User-to-System web interface, and all aspects of PST management (e.g., account, key management and ACL management) etc.;
- **Communication FG:** this FG contains FCs that can be used to ensure inter-FC communication like for instance message bus and HTTP/RESTful-based communication;
- **Privacy/Security/Trust (PST) FG**: this FG contains all FCs that are needed to fulfill the requirements in term of security, privacy and trust. This includes blockchain-related FCs, identity management, authentication/authorization, trust management, non-repudiation/accountability, logging, audit and more;

4.3.2 Description of DEDICAT 6G FGs and FCs

4.3.2.1 Service Operation FG

The FCs in that FG aim to implement the decisions taken by the FCs of the Decision Making FG and to translate such decisions into readable instructions or commands for external agents, e.g., the Service Orchestrator FC or the 5G core NFV Orchestrator component.

In Figure 12, we show the FCs pertaining to the Service Operation FG (in red) and the nature of interactions they engage into with the decision-making components (in green) on the one hand and the Context-Awareness FCs (in blue) on the other hand. It also outlines some flows of information which are revisited in the Information view, Section 4.4.2.

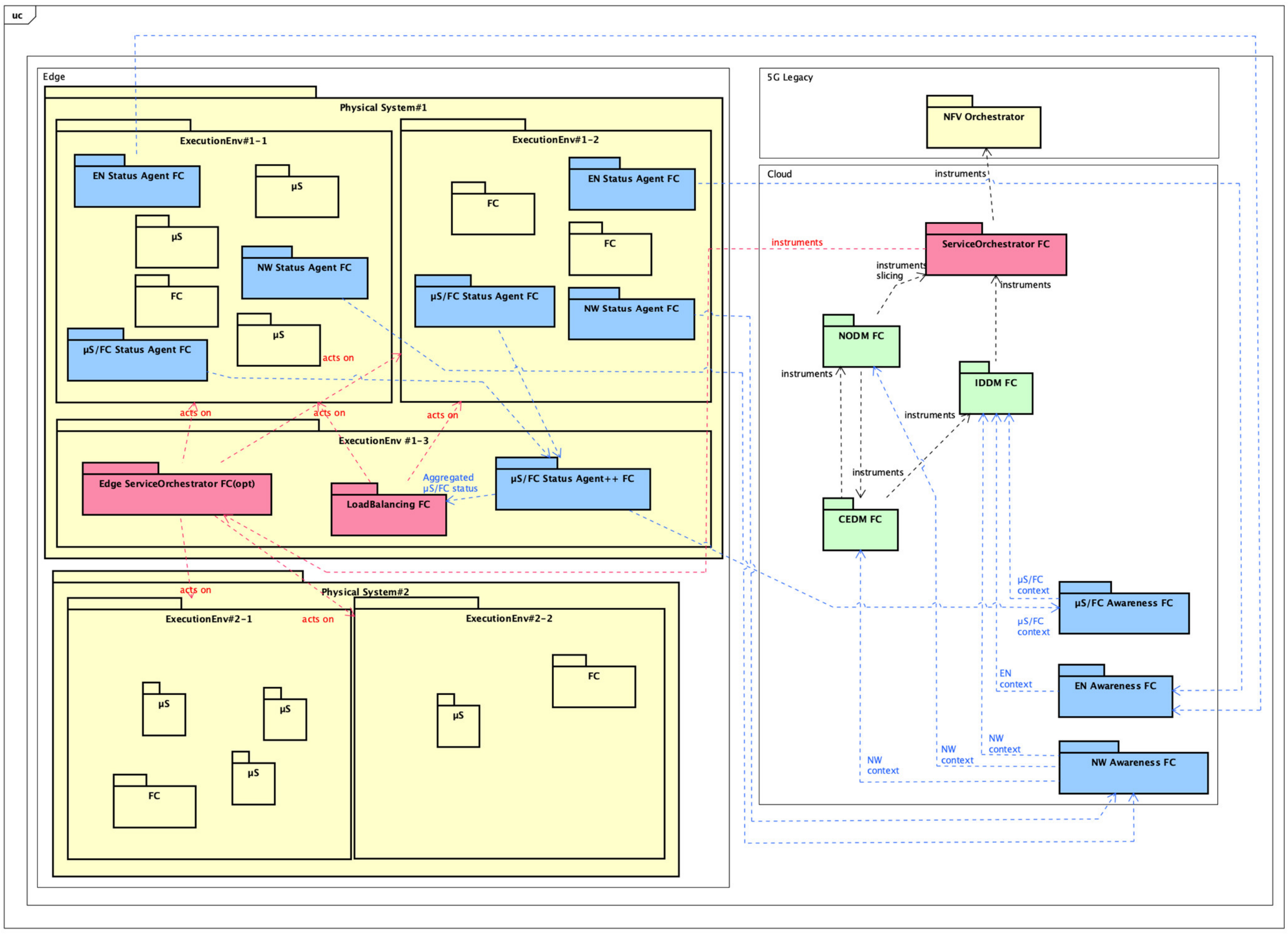


Figure 12: Service Operation FCs with their interacting FCs

The various edge- or cloud-based components that underpin the decision-making FCs by implementing their decisions follow:

- **Service Orchestrator FC:** this FC executes the outcomes of the *Intelligence Distribution Decision Making FC (IDDM FC)* which can also be instrumented by the *Coverage Extention Decision Making FC (CEDM FC)* and *Network Operation Decision Making FC (NODM FC)* as shown in the figure above. It means that the Service Orchestrator FC is a cloud component that is used a relay either towards the Edge Service Orchestrator FC (see above), which is doing the actual work at the edge or towards the NFV Orchestrator when changes in the network slices are needed. It is therefore encapsulating both interfaces. It is also devoted to interacting dynamically with the control plane of the Legacy 5G network, configuring in real-time parts of the network according to the decisions made by the DM FG components;

- **Edge Service Orchestrator FC:** the Edge (Service) Orchestrator FC is a component which is deployed in a physical system with the ability to, first accommodate and manage µS/FCs and to perform cross-border (meaning from one PS to another PS) µS/FC migration/cloning; this component will be taken off-the-shelf depending on the technology used for edge computing (e.g., Kubernates);
- **Load balancing FC:** this FC provides support for inter- and intra-node load balancing between the FCs in a single EN (but in multiple EEs). This feature is applied to service and networking components needed for the distributed nature of the micro services to be onboarded. This component receives monitoring data from the Status Agent FCs and decides when to migrate or scale µSs/FCs up (especially those which are mission critical). This component is intended to work autonomously, always trying to maintain optimal µS/FC execution in compliance with the agreed SLAs. However, it also gets instrumented by the Service Orchestrator and Edge Service Orchestrator FCs, typically when a new IDDM FC decision is issued. It is worth mentioning that, depending on the implementation details, the load balancing functionalities may be provided by the Edge Service Orchestrator FC (e.g., K8s). In a general manner, the policies for load balancing need being described within the EC Policy Repository FC and processed by the IDDM FCs in order to request the Edge Service Orchestrator FC to create the required FC instances.

4.3.2.2 Intelligence Distribution FG

This functional group stores the registry and the metadata of both the vertical applications that are uploaded to the platform, as well as of all the edge nodes that are going to be used. It serves as the back-end endpoint for the users in the DEDICAT 6G platform, since the service to be rolled-out with all the associated policies and SLAs, and also the edge nodes and their capabilities are inserted in the inner functional components of this group.

The data formats associated with edge computing policies, SLAs and edge nodes are elucidated in the Information view in Section 4.4.

- **EC Policy Registry FC:** this FC collects the information about the fine-grained policies that are applied to µSs and FCs when deploying them to the candidate ENs. Each vertical application is able to capture its own deployment constraints (sometime linked to specific physical system deployment) into such policies, before requesting a CEaaS or IDaaS, the same way DEDICAT 6G also sets up deployment constraints for its own FCs (especially the Context-Awareness FG related FCs). The interface and related data-models are described in Section 4.4.1.5.4;
- **SLA Registry FC:** is used to store the encoded SLA that results from negotiating a CEaaS or IDaaS-related bipartite contract between the DEDICAT 6G and the vertical application that wants to rely on the DEDICAT 6G to help running its business applications. In that context an SLA captures the technical and legal terms of the contract and describes the service provider duties. The SLA is an important piece of information that can be used in particular in the event of a dispute between the service consumer and service provider.

  Following the contract establishment, the DEDICAT 6G platform may engage into deploying vertical's µSs, FCs and scaling network resources up according to the EC policies and required QoS/QoE. The SLA Registry FC is then acting, at run-time, as an independent component responsible for enforcing the SLA and reporting any SLA breach to the responsible party (could be either the IDDM FC or CEDM FC depending on the nature of the SLA breach). The associated interface and associated data structures are defined in Section 4.4.1.5.5;

- **Edge Node (EN) Registry FC:** this FC is used to store and query the characteristics of an edge node or group of edge nodes, focusing on computing, networking, and energy consumption capabilities. The Information view provides an extensive description of the data models used to describe edge nodes (please see in Section 4.4.1.5.3 and 4.4.1.1.3);
- **µS/FC Repository FC:** this component is used to store images (e.g., Docker images) of the µSs and FCs that have been described using the µS/FC Registry FC (see next bullet). It is also used to retrieve a handler to a stored image that fits a specific `ClassID` (classically named service lookup). This image handler can then be passed on to the Edge Service Orchestrator FC for actual deployment in a target edge node. Its interface can be found in Section 4.4.1.5.1;
- **µS/FC Registry FC:** it provides the basic functionalities needed to upload FCs and µSs profiles (class name, list of run-time requirements, etc.) that can be passed on to the Edge Service Orchestrator FC when an instance of that class is deployed within an edge node. This registry also includes information about each created instance of each class, so updating the registry is needed after all µSs of FCs physical deployments. The interface of the µS/FC Registry FC can be found in Section 4.4.1.5.2;

4.3.2.3 Coverage Extension FG

The following components are instrumented by the CEDM FC and aim to provide support to its coverage extension decision at configuration and run-time.

- **Swarm Operation FC**: this component is deployed at the different nodes involved in coverage extension and provides a swarm of MAPs with the ability to operate autonomously in a cooperative or orchestrated way, with respect to the operation constraints set by CEDM FC decisions.

  Theoretically, autonomous operation of a swarm of MAPS could be handled in two different ways: 1/ the FC could assign master & slaves roles to the mobile entities, or 2/ each mobile entity could act as an independent agent operating in a fully distributed architecture and seeking autonomous decisions. Either way, tasks performed by the Swarm Operation FC covers:

  - Self-organizing as an ad-hoc network (including access and backhaul links);
  - Optimization of MAPs placement in order to fulfill the CEDM coverage area constraints;
  - Optimization of MAPs trajectory (path planning) in order to reach the placement fulfilling the CEDM coverage area constraints;
  - Optimization of MAP operation according to the overall requirements from the CEDM, assuming that the MAP drones/robots do not necessarily have the exact same capabilities (e.g., supported radio carrier and interfaces, autonomy, freedom of movement, self-management of resources including energy management);

  As mentioned above, the overall operation planning could follow two modes. The first mode (orchestrated) results from "negotiations" between an individual (i.e., the "master") and the rest of the MAP swarm individuals (i.e., the "slaves"). This option has a significant caveat: when the master drone needs going back to the docking station to recharge batteries, either 1/ the swarm would be left without instruction until the master drone comes back to operation or 2/ the "master" role would have to be allocated to one of the remaining flying drones.

The second mode (cooperative), which we are adopting in DEDICAT 6G, is more flexible and all drones are identical and follow the same behavior. The swarm behavior results from the cooperation between multiple independent agents distributed in mobile entities and exchanging MAP Status Agent FC and MAP Awareness FC information as global observations (e.g., current position, previous decisions, local performance metrics).

The swarm of MAPs may be contacted anytime by the CEDM FC for the sake of changing essential mission parameters or simply to stop/resume/end-up the mission (which result into terminating communication and performing a global return to assigned docking stations); The Swarm Operation FC interface can be found in Section 4.4.1.3.1.

- **UAV Operation FC:** this component is used to control the operation of a single drone or fleet of drones where each drone is controlled individually. It includes setting-up position in space, taking off/landing towards a specific position, setting up orientation and tilt, and receiving status reports on ongoing operations;
- **AGV Operation FC:** this component provides the basic interface to instruct the robot to perform its atomic actions (its basic capabilities). This includes positioning, raising arm, grasping objects, identifying objects (QR-code/RFID based, video recognition), releasing objects etc. The catalog of atomic actions a robot can perform obviously depends on its intrinsic abilities, and therefore bound to a maker/model. Based on those atomic actions it is there possible to build up libraries of compound actions able to answer the particular needs of a vertical business (please refer to the "Smart Warehouse" UC1 Use Case in D2.3 for some concrete examples). This component also reports information about the AGV status. More information about AGV Operation FC can be found in Section 4.4.1.3.3;
- **ConnectedCar Operation FC:** this component supports similar features than AGV for the deployment of mission critical services at the edge with needed specific power supply for connectivity resiliency. Regarding the size and the autonomy needed, AGV features must be delivered by car for Public Safety Use Case. Operationally, connected cars could be part of Mobile C&C truck or specific cars deployed on strategic places in order to extend the coverage. These strategic places are supported by the analysis conducted by drones. The ConnectedCar Operation FC interface can be found in Section 4.4.1.3.2.

4.3.2.4 Decision Making FG

There are different purposes to decision-making in the DEDICAT 6G projects and radically different technologies can be used. Examples are fault recovery, explicit coverage extension (on-demand CEaaS), congestion handling, etc.

Before going into the detail of their descriptions, we give an outlook of the three main cloud-based DM FCs -which are the IDDM FC, CEDM FC and NODM FC (as introduced in Section 4.3.2.1 above). We also show the components they are interacting with, at run-time, either for issuing decision or for implementing decisions.

In the Figure 13 below those three FCs (in Green), take decisions based on both contexts brought by the Context-Awareness FCs (in Blue and described in Section 4.3.2.6) and additional information stored in repositories and registries (in Orange and described in Section

4.3.2.1). Those decisions are then implemented by the DM-supporting FCs (in Yellow and described in Section 4.3.2.2 for the IDDM FC part and in Section 4.3.2.3 for the CEDM FC part).

However, as we have outlined earlier, the IDDM and CEDM FCs can also be triggered by verticals for the sake of provisioning CE- and ID-related services as well. Those services are respectively named: *Intelligence Distribution as a Service (IDaaS)* and *Coverage Extension as a Service (CEaaS)*. In that later case, those two components do not react to incoming contexts but answer to explicit service requests from verticals, that ultimately aim at operation support to their businesses. Typically, the DEDICAT 6G Use Cases aim to demonstrate those aspects by requesting either a CEaaS (see UC2 example in Section 4.2.2.1) or IDaaS depending on their actual needs.

The share of duties between the NODM, CEDM and IDDM FCs makes sure there is no possible overlap and resulting interferences in taken decisions. In this particular context, interference means that potential conflicting decisions would be issued by two different DM components at the same time (e.g., CEDM and NODM for network extension issues or CEDM and IDDM for intelligence distribution issues).

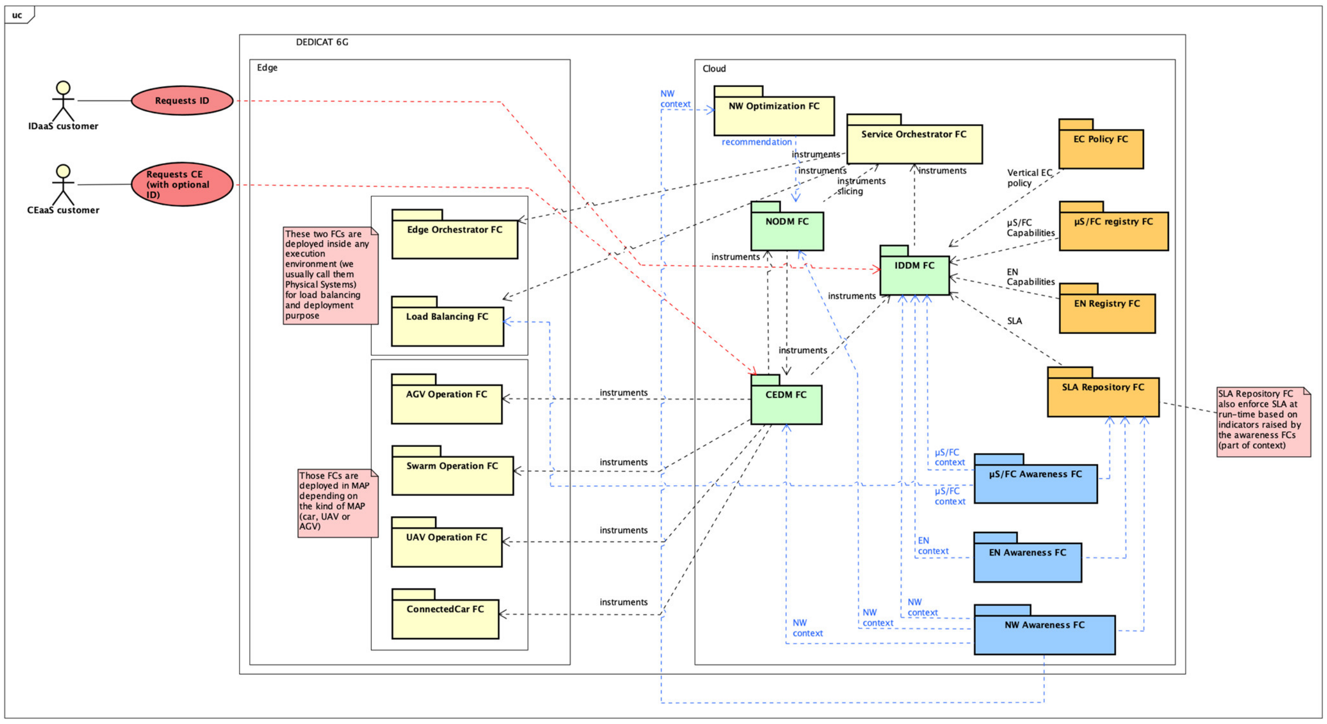


Figure 13: Decision-making FCs and their contributor FCs

- **NODM FC:** this component generates the decisions about the network provisioning and maintenance, keeping the optimal operation of the network based on data it receives from the various Context-Awareness FCs and from optimization recommendations issues by the NW Optimization FC (see Analytics FG). Besides, it is responsible for handling network degradation resulting from faulty equipment or network saturation. It may trigger actions from the CEDM FC in case network extension is needed to palliate the defect of network equipment's or to expand the capabilities of the network in a certain area.

  The NODM FC processes the recommendations and creates the concrete actions to perform in the network infrastructure to fulfill the vertical application topology, requirements and declared SLAs. These actions describe the network elements that are involved in the operation and the configuration that needs to be applied on them. Finally, they are forwarded to the Service Orchestrator FC for applying the recommendations in the 5G infrastructure via the NFV Orchestrator (not shown in the picture). The NODM FC relies on recommendations and performance indicators from the Analytics and Context-Awareness FGs to guarantee the network availability and QoS;

- **CEDM FC:** the key aim of this component is to produce the optimal configuration of the radio network of the *Mobile Access Points (MAP)* entities and the paths (trajectories) that need to be followed by the MAP entities, in order to offer adequate QoS levels, in terms of service availability, performance and reliability. This optimization also includes the most appropriate allocation of nodes to MAPs as well as selection of nearby docking/charging stations for drone and robot MAPs to ensure connectivity of the appropriate QoS to mobile nodes. This component may also be utilized for 1) complementing the actions of NODM FC and 2) implementing necessary network extension actions needed to fulfill a CEaaS request from a vertical for instance. This component may rely on the IDDM FC as part of dynamic distribution-related aspects of coverage extension. The CEDM FC interface is described in Section 4.4.1.2;

  IDDM FC: this component is responsible for deciding the optimal placement of intelligence (i.e., FCs and µSs) in terms of data and computation but also throughput to a set of candidate ENs supported by MAPs for the networking aspects. In order to support the task→EN optimal assignment algorithms (where tasks can be either µSs or FCs), each physical system, FC and µS is characterized by a profile which describes either their maximum capabilities and available resources (EN) or their required resources (µS and FCs) which is the exact purpose of the two EN Registry FC (for ENs) and µS/FC Registry FC (for both µSs and FCs) described earlier.

  Additionally, the IDDM FC is responsible for handling any incoming IDaaS request from a vertical. Answering such a request also leads ultimately to allocating µSs and FCs to the available PSs. The parameters expected to be passed by the Vertical on to the IDDM FC along with the IDaaS request are elucidated in Section 4.4.1.2. A corresponding system use-case example (UML sequence diagram) is also described in Section 4.3.4.10. The IDDM FC interface is available in Section 4.4.1.2.

Finally, the following two new components are introduced in this document iteration to provide support to the optimization of the UE-{MAP, BS} association process. The way they interact with each other and with some other FCs is described in detail in the system use-case section, and more particularly in Sections 4.3.4.6 and 4.3.4.7, using UML sequence diagrams.

- **UE-BS Selection DM FC:** this component is responsible for selecting the best sub-set of gBS and/or aBS with which to further establish an association. How frequently this process takes place depends on the change in contextual information like new users coming, users moving leading to changes in channel qualities, etc. The selection depends on several criteria such as the ground users' mobility, the co-channel interference, the users traffic request and the BS load balancing;
- **UE-BS Negotiation FC:** after a subset of candidate BSs for UE association (only from the UE point of view) has been elected, negotiation take place between the UE and individuals BS in the subset in order to select the subset of BSs which agree to engage into an association from the UE (From the BS point of view).

4.3.2.5 Analytics FG

This FG contains the FCs that are monitored and analyzed by the DEDICAT 6G platform. This FG is closely related to Decision Making and Context-Awareness FGs where the monitored analytics can be used for directing the system performance towards the predetermined criteria e.g., for decreasing the network load, optimizing network KPIs, or altering the device specific characteristics, such as power consumption. The functional entities can provide the results for evaluating the system performance against the baseline criteria as well selecting optimized network parameters dynamically depending on its state.

- **Network Optimization FC:** ensuring the desired network coverage and selection of suitable computation server for a set of user applications have interrelated optimization targets. An important common factor is to meet the end-to-end delay targets with minimal cost that may involve various parameters such as number of required stations and their energy consumption. While optimization may involve only either network-related parameters, such bandwidth allocation and user association, the most sophisticated approach is to also include computation-related parameters, such as the CPU rate, of target servers in the network. The optimization framework must be aware of conditions of heterogeneous users that may have different QoS targets and different location-dependent channel characteristics.

  A promising way to maximize the coverage is to dynamically optimize network topology while ensuring adequate frequency resources and interference mitigation. The DEDICAT 6G platform enables sophisticated means for configuring the network topology according to the optimization goals set in the DM FG. A suitable optimization cycle must be carefully selected, especially if the coverage optimization is done iteratively which means that the optimal configuration is achieved after a certain period of time and iterations may consume additional resources. The key asset in dynamic network configuration is the MAP, which can be a ground or aerial based component, as described in Section 4.3.4. Moreover, the limitations (e.g., feasible trajectories, computing capability, and operation times without recharging) of specific MAPs must be considered before reacting to the optimization process. The selected centralization degree of the whole optimization process inherently affects the required interaction between the different involved FCs;

- **Platform Performance analytics FC:** this component is responsible for assessing the way the DEDICAT 6G platform performs based on pre-defined KPIs which are assessed regularly relying in particular on the analytics FCs.

  the following KPIs are identified that can be used for network performance optimization, prediction aligned with decision-making, edge computing and context-awareness:

  - Overall performance;
  - System utilization rate;
  - CPU utilization;
  - Programmability, flexibility, scalability, availability and usability;
  - Power consumption;
  - Quality of service including latency and throughput;
  - Security level;
  - Cost;
  - Number of users.
- **Analytics toolbox FC:** provides a collection of general-purpose Machine Learning algorithms that can be used by other FCs from this FG or also other components from the other FGs like for instance decision-making-related FCs or Security FCs like Auditing.
- **Network Performance Analytics FC:** this new component is used by NW Awareness FC in the context of the MAP placement optimization, in order to infer the global reward of the network. It collects information of each agent (e.g., UE or MAP) and evaluates a network performance metric (e.g., percentage of UEs with satisfied QoS, average

network sum rate, cost of the current deployment). The interface of the FC is defined in Section 4.4.1.6.1;

4.3.2.6 Context-Awareness FG

The FCs in that group are responsible for building various kinds of contexts depending on the specific needs of the FCs from the Decision Making FG. It mainly relies on information it can collect from physical systems and/or knowledge it can gain from using the analytics FCs. Such contexts are part of the main inputs for decision-making FCs and includes information about edge nodes, µSs/FCs, MAPs, UEs and network.

In the following list of FCs, we often refer to Status Agent FCs and Awareness FCs for a given purpose (e.g., µS/FC, edge nodes, etc.). There is a difference about the nature of data they do report: status agents are collecting raw data and provide information about the collected raw data (information being enriched data structures with data and meta-data) while Awareness FCs are aggregating and transforming incoming information into knowledge that can be used to take decision. As a matter of fact, information and knowledge can be very different in nature as they are used in different abstraction layers.

Before going into the detail of Status Agent FCs and corresponding Awareness FCs, it is worth mentioning that while the former ones (many edge-based instances) report by default information to the later one (one sole cloud-based instance) we have a special case for the EN Status Agent FC that also report to the µS/FC Awareness FC and vice-versa the µS/FC Status Agent that also reports to the EN Awareness FC. This allows to correlate information from the two flows of information in order to enrich µS/FC contexts on one side and EN contexts on the other side. Typically doing so, we are able to provide information about energy consumption per µS/FC (please see also Status Agent and Awareness FCs -related data models in Section 4.4.1.1 for more detail). The interfaces of all Status Agent and Awareness FCs can be found in Section 4.4.1.1.

The accuracy and relevance of a decision is dependent on the validity and freshness of the contexts it is based on. In order to guaranty both decision qualities, it is paramount to ensure that the raw data sampling rate is appropriately chosen and implemented at the awareness agents sides, in line with the one used for contexts.

The FCs in this group include the following:

- **µS/FC Status Agent FC:** they provide information about the status of µSs (lifecycle, resource consumption etc.) to the EN Awareness FC and Load Balancing FC. It also reports µS/FCs status to the µS/FC Awareness FC for µSs and FCs context aggregation. This µS/FC context is then used by the IDDM FC as one of the two main sources of information for task/PS allocation (the other one being the EN Awareness FC). As we already mentioned above, µS/FC information also flows to the local Load Balancing FC, the action of which spans all EEs in its own EN;
- **µS/FC Awareness FC:** it gathers and correlates information from both µS/FC Status Agents and EN Status Agents and provides the IDDM FC with enriched contexts. Such contexts include information about µSs and FCs such as: where they are deployed (EN/EE), their consumed CPU/GPU power in % w.r.t. the EN characteristics, consumed energy power, consumed EN RAM & Disk, etc. The µS/FC Status Agent and Awareness FCs related interfaces and data structures are described respectively in Sections 4.4.1.1.1 and 4.4.1.1.2;
- **EN Status Agent FC:** the EN status Agent FCs collect the necessary runtime information available in a concrete edge node to help creating an overall picture of the computational status of all edge nodes available (EN resource consumption). Those agents are deployed through the network and report regularly their statuses to EN Awareness

FC. Such status includes CPU/GPU available computation time (expressed in %), available memory/storage, bandwidth, etc. in addition to an overall severity status. Some basic management capabilities can be held in upper FC instances, like the sampling rate, also, the way status is reported is part of the agent configuration at deployment time, but can also be updated dynamically;

- **EN Awareness FC:** the Edge Node Awareness FC compiles an overall Edge Node context for decision-making FCs to use. EN Status Agent and Awareness FCs related interface and data structures are described in Sections 4.4.1.1.3 and 4.4.1.1.4;
- **NW Status Agent FC**: the Network Status Agent FCs main aim is to periodically report the networking statuses of the network nodes they supervise, to the NW Awareness FC so that it can build up an overall network context;
- **NW Awareness FC:** the Network Awareness FC receives and compiles information of the edge nodes networking statuses received from all NW Status agents in order to build an overall context for the decision-making FCs (especially the NODM and CEDM FCs); NW Status Agent and Awareness FCs related interface and data structures are described respectively in Sections 4.4.1.1.9 and 4.4.1.1.10;
- **UE Status Agent FC**: it is responsible for assessing the strength of *Reference Signal Received Power (RSRP)* from surrounding gBS & MAP and collecting the needed information from the UE side in order to feed the UE-BS Association DM FC;
- **UE Awareness FC**: it aggregates the different UE contextual information for the CEDM FC to use. UE Status Agent and Awareness FCs related interface and data structures are respectively described in Sections 4.4.1.1.5 and 4.4.1.1.6;
- **MAP Status Agent FC:** those components maintain a status about their local MAP including deployment information; the computing-related aspects of a MAP (as a MAP is also an edge node) are covered by the EN Status Agent FC, meaning that both EN- and MAP- related Status Agent FCs need being deployed by default in each MAP, as part of the "D6G" EC Policy;
- **MAP Awareness FC:** it compiles an overall MAP status to relevant decision-making FCs. Information about MAP status, location, on/off, battery level, status of UE/MAP association, etc. MAP Status Agent and Awareness FCs related interface and data structures are respectively described in Sections 4.4.1.1.7 and 4.4.1.1.8;

4.3.2.7 Management FG

- **Dashboard FC:** this front-end component provides graphical display of analytics or row information, like for instance performance indicators about how the DEDICAT 6G does perform. It also provides access (via a web-based interface) to all the configuration and monitoring functions that allow the proper operation of the platform, including FCs from the back-end below, but also configuration functions which are introduced in other FGs:
    - Management of Edge Nodes: declaring capabilities of an edge node;
    - Declaring and uploading µServices;
    - Setting-up policies (e.g., deployment policies);
    - Negotiating SLA etc.
- **Management FC**: this *back-end* component is the home of all management functions (Fault / Configuration / Accounting / Performance / Security) such as:
    - **Accounting FC:** this sub-FC keeps track of the usage of resources by verticals in order to feed ticketing FC with the proper information for the sake of billing;
    - **Ticketing FC:** this counterpart of the accounting sub-FC, is responsible for billing a vertical application based on its resource consumption and associated service fees;

- **PST Management FC**: this FC relates to the management and configuration of the FCs belonging to the PST Functional Group (whose main focus is enforcement). It can alternatively act as a front-end to the PST management functions hold at the PST FG side (depending on the case).

4.3.2.8 Communication FG

This FG provides support to inter-FC communication - including secured communication - either the FCs are located in the DEDICAT 6G cloud platform, at the edge side of DEDICAT 6G or part of an external physical system legacy FCs (therefore outside the perimeter of DEDICAT 6G). Two different paradigms (and therefore communication channels) are envisioned as follows:

1. **Publish/subscribe:** this paradigm allows a source party to reach various potential sinks by publishing messages on a queue that several sources can subscribe to. Such sources are typically, the awareness and status related agents and other DEDICAT 6G FCs, but also 5G legacy components, which raise alarms/fault/performance related events. Several sinks can then consume that information based on subscription and filtering capabilities (e.g., according to a particular topic, scope, etc.);
2. **Asynchronous one-to-one:** one single source communicates asynchronously with one single sink (e.g., a DEDICAT 6G FC with a 5G Legacy component with the use of REST API).

We therefore propose as a first approach the following FCs:

- **Pub/Sub Client FC:** The client part of the message queue allows to access the queue for the sake of publishing messages and consuming messages according to the client role;
- **Pub/Sub Server FC:** the server part of the queue is responsible for managing topic subscriptions, sustaining the flow and distribution of messages issued by Publisher clients to the Subscriber (and therefore message consumer) clients. It is also responsible for connecting with the Logging FC for the purpose of event logging. Those servers are used by Awareness FCs and there will be one server (queue) for each of them. There is an additional instance of that server which is used by the Data Market Place for the sake of collecting data from the Vertical client in order to build time series databases that can be used by those clients for data visualization;
- **AsyncComm Client FC:** this component stands at the side of a FC that is willing to send a unicast POST/GET/PUT/DELETE message to another FC (in opposition to the message queue which is multi-cast);
- **AsyncComm Server FC:** Likewise, this component stands at the side of a component which is willing to answer a POST/GET/PUT/DELETE message sent by another FC, as the sole recipient of that message.

4.3.2.9 Privacy, Security & Trust FG

We define in this section the FCs that are meant to cover all needed functionalities necessary to ensure Trust, Security and Privacy within the DEDICAT 6G eco-system:

- **AuthN FC:** authentication functionality applied to end-users and µSs/FCs. Roles defined on the level of the project and for each UC;
- **AuthZ FC**: authorization functionality with role and attribute-based authorization and access control. Applied on users, devices and µSs/FCs;
- **IdM FC:** identity Management functionality used for assigning identity and roles to systems, devices and users;

- **Threat Analysis FC**: this FC performs threat detection, identification and classification and is executed either in centralized or in edge processing nodes. Threat analysis is based on ML models trained and updated on collected system logs. The ML models are performed in federated learning mode with global model running in DEDICAT 6G cloud and local models running on edge computing nodes. Federated learning approach allows for globally defined ML models to be distributed closer to the data sources, perform updates/re-training on the collected data and report tuned parameters to the global model for further performance improvements. All threats will be categorized based on severity/impact and decision-making systems will include threat category in their processes;
- **Trust Metrics FC:** this FC needs to be installed at all edge nodes as well as on the central/cloud resources. Trustworthiness metrics are calculated for edge nodes, processes, users and data streams. Trust metric value indicates if a node can join a local network, if process output can be further used, if a user can execute specific rule. Trust metrics are implemented as ML models whose outputs are written on private permissioned blockchain through dedicated smart contracts. This way all stakeholders in DEDICAT 6G instance have access to immutable record of trust metrics calculated for all actors, resources and processes. Decision-making processes of the DEDICAT 6G system must consult calculated trust metrics before proceeding with decision implementation or execution. The following trust metrics are envisioned for the DEDICAT 6G system:
    - Device-based metrics - capturing a device state and feature set;
    - Connection-based metrics - define the connection types which are established in DEDICAT 6G systems;
    - Behavior-based metrics - capturing the user's and device's behavior within the observed network;
    - Context-based metrics - expected operations of a node in a known context. e.g., in case of failures;
    - Composite metrics computed based on the weighted calculation of different security, reliability, safety and privacy metrics at the device, connection, behavior, and application levels.
- **Distributed Ledger FC:** this FC is a basis for trusted data transactions between different stakeholders and their services/devices/actors. It is based on private permissioned blockchain (Hyperledger Fabric) and a set of smart contracts for facilitating read/write/report operations and implementing relationships between actors and processes. Trust metrics are calculated based on information stored in distributed ledger and calculation results are stored in the ledger. Multiple Hyperledger Fabric channels can be implemented to facilitate deployment requirements and integration rules dictated by the DEDICAT 6G system and its use-cases.
- **Logging FC**: this FC enables collection and secure storage of all types of logs across the DEDICAT 6G system (with main focus on security) and implements blockchain technology to ensure logs consistency and trustworthiness.
  Its main functions are:
    - Collection of logs (transactions, audit of sessions, reports);
    - Storage of trusted logs in a secure manner;
    - Providing access to logs for authorized clients;
- **Audit FC:** this FC provides surveys and reports based on algorithms exploiting the data logs collected and stored by the Logging FC, e.g., security threats like intrusion detection resulting from complex correlations and deductions exploiting the data logs and that cannot be achieved in real time.

4.3.3 Description of other layers

After getting into the detail of the FCs that populate the different functional groups of the DEDICAT 6G architecture, we elucidate the FCs, which seat outside its perimeter.

4.3.3.1 Legacy 5G Network FG

We give here, a catalogue of 5G components, which interact directly with the DEDICAT 6G FCs and we also explain the nature of the intended interactions (e.g., deployed, providing information or receiving information).

CORE components

In this section are introduced some relevant highlights and components of the 5G system [12] [13], and how the DEDICAT 6G platform is interfacing with the *5G Core* (5GC) components. In Figure 14 below are depicted the main components of the 5G system [14]. It is worth noting that:

- According to the SDN principle, basic for 5G/B5G networks, Control and User Plane are separated in order to allow Control and User Plane systems to scale up independently. In Figure 14 below, the components devoted to the user plane are showed in red and the ones placed in the control plane are in blue;
- In order to fully support scalability and resilience of the *Network Functions (NF)*, there is a separation between computation and storage;
- The NF capabilities are exposed through well-defined RESTful APIS, following a service-based architecture;
- Support for non-3GPP and fixed network access, providing mobile network convergence;
- Network slicing to provide end-to-end logical network separation for automate the offering of different levels of QoS virtual set-up.

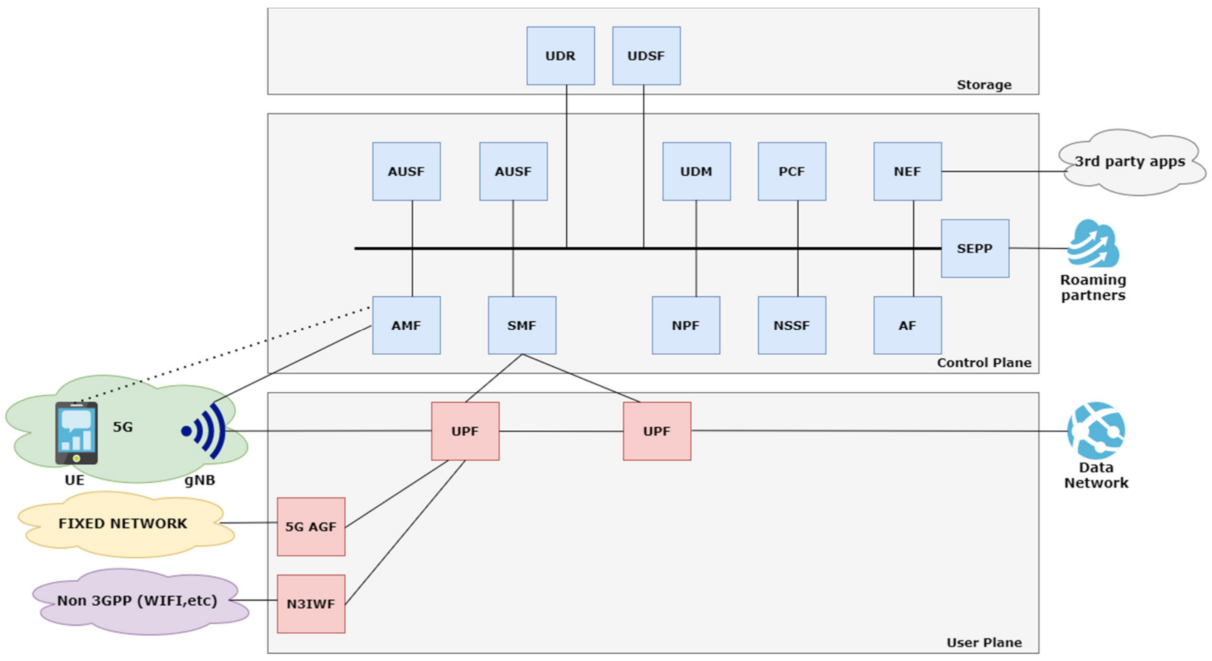


Figure 14: 5G Core components

The most relevant components for the DEDICAT 6G architecture are:

- *Access Mobility Function (AMF)*:

  - Control which *User Equipment* (UE) can access the 5GC network and to exchange traffic with DNs;
  - Manages the mobility of UEs when they roam from one *gNodeB* (gNB) to another for session continuity;
- *Session Management Function (SMF)*:
  - Keeps the trace of related PDU sessions and QoS flows;
  - Receives from PCF rules and enforce them in UPF, gNB and UE for QoS flows management;
- *Network Slice Selection Function (NSSF)*:
  - Support with the selection of the Network Slice instances that will serve a particular UE. Besides, the NSSF will dispose the Allowed *Network Slice Selection Assistance Information* (NSSAI) that is assigned to the device;
  - Reallocation of AMF in case it is not able to support all network slice instances for a given UE;
- *Policy Control Function (PCF)*:
  - Hosts network policies to create PCC rules to be forwarded to the SMF;
- *User Plane Function (UPF)*:
  - Forwards traffic between the RAN and DNs;
  - Enforces QoS of UEs based on the SMF templates through the N4 *Packet Forwarding Control Packet (PFCP)* interface;
- *Network Exposure Function (NEF)*: Supports external exposure of capabilities of network functions. External exposure can be categorized as:
  - Monitoring capability;
  - Provisioning capability;
  - Policy/Charging capability;
  - Network status;
  - Reporting capability;
  - Analytics reporting capability.

  DEDICAT 6G Network Awareness FC subscribes to certain events thus detects them in the platform. The events and metrics are available in the 3GPP 23.502 [15], some examples of the events tracked by the Network Awareness FC are:
  - Loss of Connectivity (AMF- Network detects that the UE is no longer reachable for either signaling or user plane communication);
  - UE reachability (UDM- Detected when the UE transitions to CM-CONNECTED state or when the UE will become reachable for paging);
  - Communication failure (AMF-RAN or NAS level failure is detected based on connection release and it identifies RAN/NAS release code);
  - PDU Session Status (SMF-PDU session established or released);
  - Number of UEs present in a geographical area (AMF- It indicates the number of UEs that are in the geographic area described by the AF);
- *Network Data Analytics Function (NWDAF)*: It performs data analysis, collecting data from NEF, NFs, *Operations, Administration and Maintenance (OAM)* or the *Unified Data Repository (UDR)* and provides the analytical result to the AF, the 5GC NFs and the OAM [16] [17]. NWDAF and NFs cooperate to build and supply consistent and efficient policies, analytics output results, and finally decision-making in the *Public Land Mobile Network (PLMN)*. In can be delivered in distributed architectures providing analytics at different points of presence: at the edge in real-time and a central function for analytics, which need central aggregation (e.g., service experience), etc. NWDAF collects data from different sources of the 5GC and delivers analytics services using a request

or subscription model. DEDICAT 6G network optimizer and network prediction leverages this component to place analytics at 5GC level;
- *Application Function* (AF): It is a logic representation of a broad set of capabilities intended to support the 5G Core system. The AF can interact with the 5GC in the following ways:
    - Assisting in traffic routing;
    - Accessing the NEF;
    - Interacting in policy management;
    - Enabling *IP Multimedia Subsystem (IMS)* interaction with the 5GC.

  Moreover, depending on the operator's deployment, AFs can be considered as trusted or untrusted. Only in case they act as trusted, AFs can interact directly with the relevant NFs. Otherwise, their scope is limited to the NEF, acting as a gateway for the rest of the NFs. In addition, the entity of the AF is used in the 5GC to be the entry point for other standardized subsystems, such as the ETSI MEC [18]. Thus, the DEDICAT 6G platform, or part of it, will be recognized as AF in the 5GC system. This is the way DEDICAT 6G can interact with the 5GC system to configure the 5G network environment when needed. It should be noted that this potential functionality may not be implemented during the life of the DEDICAT 6G project due to the limitations of the project itself coupled with the specific approaches and objectives to be achieved by the DEDICAT 6G Use Cases.

Radio Access Network

The different *Radio Access Network (RAN)* functions and architectures are discussed in [19],[20] and [21]. The RAN functions between the radio antenna site and central locations are shown in Figure 15(a) below.

The RAN components which are the most relevant to the DEDICAT 6G architecture are:

- The *Radio Frequency* (RF) signal processing is responsible for the D/A conversion and the RF front-end;
- The physical *(PHY)* layer is responsible for coding and modulation;
- *The Medium Access Control (MAC)* layer is responsible for buffering, multiplexing and de-multiplexing segments, including real time scheduling decisions about which segments are transmitted and when;
- The *Radio Link Control (RLC)* layer is responsible for segmentation and reassembly, including reliably transmitting/receiving segments by implementing an automatic repeat request;
- The *Packet Data Convergence Protocol (PDCP)* layer is responsible for (de)compressing IP headers, ciphering and integrity protection and making a forwarding decision (i.e., whether to send the packet down to UE or forward it to another base station for handover or link aggregation;
- *Radio Resource Control (RRC)* layer is responsible for configuring the coarse grain and policy related aspect. The RRC runs in the control plane and does not process packets on the user plane.

In [20], the *Third Generation Partnership Project (3GPP)* defined a *Next Generation RAN (NG-RAN)* architecture where 5G NR base station (a.k.a. gNB) functionality is split between two logical units:

- The *Central Unit (CU)* which is responsible for non-real time, higher L2 and L3. The CU runs the RRC and PDCP layers. The split architecture enables a 5G network to utilize different distribution of protocol stacks between CU and DUs depending on midhaul availability and network design. It is a logical node that includes the gNB functions like transfer of user data, mobility control, RAN sharing, positioning, session management etc., with the exception of functions that are allocated exclusively to the DU. The CU controls the operation of several DUs over the midhaul interface. In the 3GPP model, the CU is connected to the 5G core (5GC) via the NG interface and the CU is connected to the DU via the F1 interface, as shown below in Figure 15(b);

- The *Distributed Unit (DU)* which is responsible for real time L1 and L2 scheduling functions. DU sits close to the *Radio Unit* (RU) and runs the RLC, MAC, and parts of the PHY layer. This logical node includes a subset of the eNB/gNB functions, depending on the functional split option. Its operation is controlled by the CU.

The 3GPP studied several different functional splits between the CU and DU in [20]. Functions that need real-time processing are grouped within the DU, while those not requiring real-time are grouped within the CU.

In a separate study [21], the *ITU Telecommunication Standardization Sector (ITU-T)* adopted a slightly different transport network architecture for 5G that is comprised of three logical elements: CU, DU and RU, as shown in Figure 15(c). In this model, the mid and lower layer functions are divided between the DU and RU.

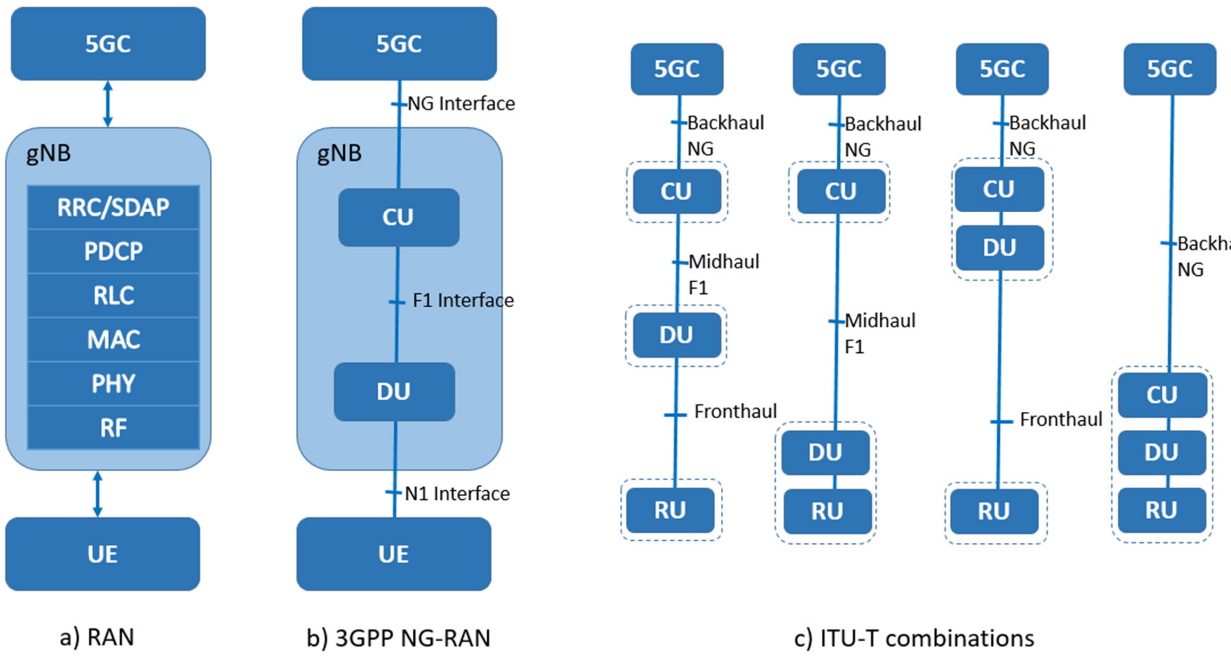


Figure 15: Radio access network functions (a), 3GPP NG-RAN Architecture (b), possible CU, DU, RU combinations (c)

The architecture of *Integrated Access and Backhaul (IAB)* networks [22] represents a fundamental evolution in 5G networks. Two types of links are supported in IAB networks:

- An access link is a link between an access UE and an IAB node or IAB donor;
- A backhaul link is a link between an IAB parent node and IAB child node:
  - o The IAB parent node is responsible for scheduling the downlink/uplink traffic for both access and backhaul links;
  - o The IAB child node at the end of the transmission chain is responsible for scheduling the downlink/uplink traffic between itself and the UEs.

In order to avoid the expensive installation of fiber for the backhaul, DEDICAT 6G can provide with IAB a better alternative for connectivity extension and cell densification by connecting new MAPs wirelessly to backbone networks (see Figure 16 below) and sharing the spectrum for access and backhaul links. If temporary coverage or capacity needs to be added in a particular area, IAB nodes can be opportunistically deployed/activated to deliver services.

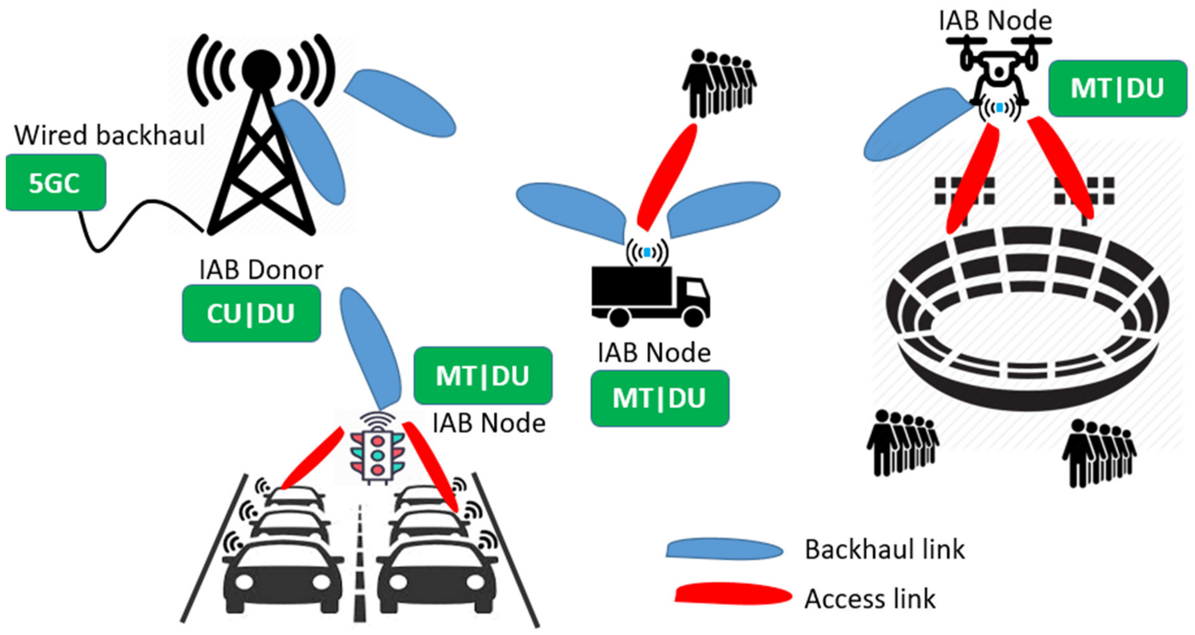


Figure 16: Usage of IAB in connection with MAPs

4.3.3.2 Business Verticals FG

This FG hosts the business applications that are using the services offered by DEDICAT 6G.

4.3.3.3 3rd Party FG

This FG hosts all 3rd party products that are supporting the applications ran by the verticals when using the DEDICAT 6G services (like CEaaS and IDaaS). We do not give here any detail about those 3rd party software components as those FCs are not part of the DEDICAT 6G architecture, but Table 12 gives a good overview of the FCs identified so far and pertaining to the four DEDICAT 6G Use Cases.

4.3.4 System use-cases

System use-cases are meant to describe interactions taking place between the logical FCs within specific context (or execution patterns) and will act as a guidance for the implementation activities taking place in WP3, 4, 5 and 6.

In this second and improved version of the system use-cases, we describe textually what are the steps involved in every SUC and illustrate the interactions taking place between the FCs partaking into those SUC steps using UML sequence diagrams. Each of the sequence diagram relates to the interfaces and data-models defined in the Information view.

4.3.4.1 Initial deployment at T_0

This first SUC elucidates the sequence of actions that take place at t_0. It covers the discovery of the network topology and the decisions taken by the DEDICAT 6G platform to undergo the initial FC deployment (likely to be mostly awareness-related agents and aggregators). Since there is theoretically no pre-deployed edge nodes yet, the main target of the initial deployment is the B5G network instrumented by the DEDICAT 6G platform. The objective is then to start building up the deployment that provides continuous flow of information that can trigger decision-making. This means that all FCs meant to be running in the cloud need being started as well (e.g., the Decision Making FCs, Service Operation FCs, cloud-based context-awareness components, etc.)

Again, at that stage it is assumed that no edge node has been already deployed.

However, mobile edge nodes (MAPs) such as connected cars, drones and robots can be pre-declared and pre-configured (see EN Registry FC) so that if it happens coverage extension is required based on context-awareness data analysis and selection criteria, the needed edge nodes can be selected and deployed straightaway, without losing time going into the configuration phase first.

As the main idea is to have those edge nodes deployment–ready, a certain number of FCs can be pre-deployed inside the edge nodes in order to save time. Those pre-deployed FCs comprise e.g.:

- Agents targeting 5G legacy components needed for handling the coverage extension scenarios (see Sections 4.3.4.4); Those agents gather in particular alarms coming from radio and network 5G equipment that can be used to trigger CE;
- NW Awareness FC and related Status Agents that provide information about the communication capabilities and statuses;
- EN Awareness FC and related Status Agent FCs that give information about the edge node at run-time;
- Specific networking related components, including 5G legacy ones (e.g., the DU)

This pre-configuration is initiated by a network operator using the front-end GUI from the Management FG (as for EN characteristics) while the pre-deployment of FCs is achieved automatically based on the "D6G"EC deployment policy.

We give below an example of initial deployment for one of the DEDICAT 6G scenario namely the UC2 scenario "Enhanced Experience".

The main focus in this second UC is performing live video streaming from the event site relying only on the underlying mobile network. The initial deployment at T_0 will be introduced with 5G network followed with the baseline measurements for enabling easier comparison against DEDICAT 6G solution(s). The objective is to start building up the deployment where different edge nodes can improve the network performance according to the certain level (i.e., energy, latency, throughput, load distribution) with the help of triggered analytics, context-awareness and decision-making policies. In later iterations dynamic intelligence with coverage extension will be added for enhancing the network flexibility and adaptation. Most part of the processing is planned to be executed in the edge nodes following the FCs with the context introduced earlier with the UC2:

- 5G legacy network components enriched with DEDICAT 6G optimizations;
- Application environment;
- Network analytics for providing necessary real-time status information to edge nodes.

Various example of interaction diagrams pertaining to configuration can be found in the following diagrams, e.g., the two sequence diagrams elucidating the CEaaS and IDaaS cases (resp. Figure 23 and Figure 24).

4.3.4.2 Managing Edge Nodes

When a new edge node is created and advertised to the platform via the EN Registry FC, it brings computing capability (and eventually additional radio coverage depending on the exact nature of the node) to the DEDICAT 6G platform that can be used in different contexts for the sake of dynamic intelligence distribution. The exact characteristics of the new ENs are to be stored in the EN Registry FC so that it can be discovered by the DM components and used whenever they do fit the requested criteria. More detail about the type of information stored in the EC Registry FC can be found in the Section 4.3.2.2

After being properly registered, the edge node needs being installed a certain number of FCs - mainly awareness FCs- which are needed by the platform in order to be able to take proper decisions at run time (after the edge node has been actually activated). Such examples are described already in the Section 4.3.4.1 above.

Additional aspects of EN management can be found in some of the detailed sequence diagrams shown in this Section 4.3.4, e.g. in IDaaS and CEaaS where the declaration of EN to the platform and their configuration are described.

4.3.4.3 Coverage Extension – Reacting to degradation of service

This first CE scenario is triggered by an observed degradation of 5G service in a certain area meaning that DEDICAT 6G uses its ability to monitor and enforce QoS in real-time using various contexts like NW, UE, EN contexts.

This interaction diagram below (Figure 17) assumes that the MAPs are not already deployed so it covers the phases where:

1. the problem is discovered based on received contexts (arrows 1 - 6);
2. a decision is taken by the CEDM (3rd bubble);
3. Optional interaction with a DEDICAT 6G operator takes place in order to get a formal notification to pursue (arrows 7 - 10);
4. Pre-configuration of needed FCs into the UAV, involving the IDDM (arrows 11-21) is preformed; in particular, arrow 13 is needed in order to know the characteristics of the MAPs selected by the CEDM and which MAP specific FCs they need. In our case the MAP is a UAV so the Swarm Operation FC deployment is mandatory alongside the MAP Status Agent FC;
5. The UAV are physically deployed on-site (arrows 22 - 25);
6. The UAVs are configured for autonomous operation, using the SWARM Operation FC (arrow 26);
7. The CEDM instrument the UAVs to start operation (arrow 27).

This scenario could ensure a smooth and non-disrupted transition between pre-existing legacy 5G set-up and DEDICAT 6G-enhanced B5G set-up (opposite to the next scenario where service disruption occurs) if the whole process starts when Degraded (only) status are received from the contexts.

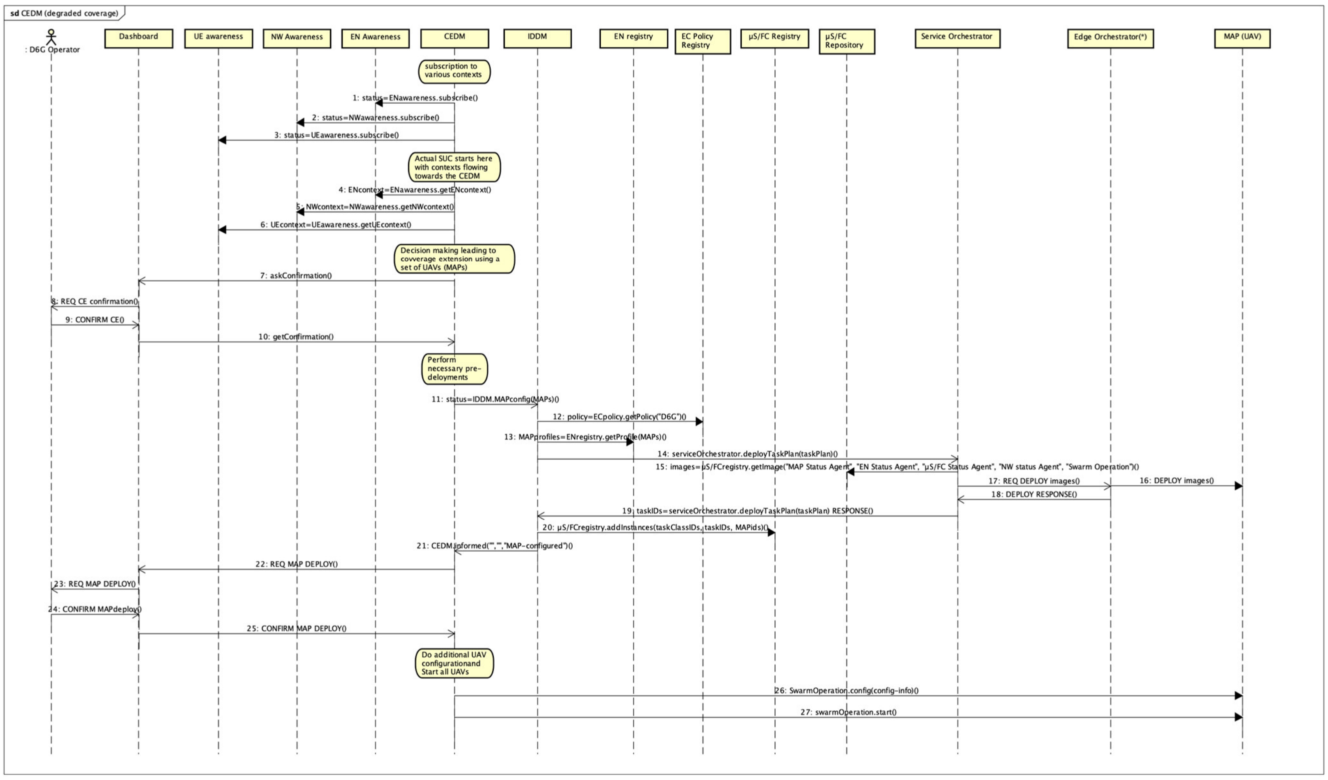

sd CEDM (degraded coverage)
: D6G Operator
Dashboard
UE awareness
NW Awareness
EN Awareness
CEDM
IDDM
EN registry
EC Policy Registry
µS/FC Registry
µS/FC Repository
Service Orchestrator
Edge Orchestrator(*)
MAP (UAV)
subscription to various contexts
1: status=ENawareness.subscribe()
2: status=NWawareness.subscribe()
3: status=UEawareness.subscribe()
Actual SUC starts here with contexts flowing towards the CEDM
4: ENcontext=ENawareness.getENcontext()
5: NWcontext=NWawareness.getNWcontext()
6: UEcontext=UEawareness.getUEcontext()
Decision making leading to covverage extension using a set of UAVs (MAPs)
7: askConfirmation()
8: REQ CE confirmation()
9: CONFIRM CE()
10: getConfirmation()
Perform necessary pre-deloyments
11: status=IDDM.MAPconfig(MAPs)()
12: policy=ECpolicy.getPolicy("D6G")()
13: MAPprofiles=ENregistry.getProfile(MAPs)()
14: serviceOrchestrator.deployTaskPlan(taskPlan)()
15: images=µS/FCregistry.getImage("MAP Status Agent", "EN Status Agent", "µS/FC Status Agent", "NW status Agent", "Swarm Operation")()
17: REQ DEPLOY images()
16: DEPLOY images()
18: DEPLOY RESPONSE()
19: taskIDs=serviceOrchestrator.deployTaskPlan(taskPlan) RESPONSE()
20: µS/FCregistry.addInstances(taskClassIDs, taskIDs, MAPids)()
21: CEDM.informed("","","MAP-configured")()
22: REQ MAP DEPLOY()
23: REQ MAP DEPLOY()
24: CONFIRM MAPdeploy()
25: CONFIRM MAP DEPLOY()
Do additional UAV configurationand Start all UAVs
26: SwarmOperation.config(config-info)()
27: swarmOperation.start()

Figure 17: CEDM reacting to a perceived degradation of service

4.3.4.4 Coverage Extension – Reacting to a radio network failure

This second CE-related scenario is triggered by the 5G legacy network and assumes that NW Status Agents have been deployed and interfaced with 5G legacy components in order to feed the NW Awareness FC with 5G Core or RAN information (such as performance indicators, faulty equipment alarms etc.).

Network contexts are subscribed by default by the NODM FC that in turn ought to decide – should a problematic context occur - if solving the technical issue must involve Coverage Extension or not. The NODM FC then instrument the CEDM FC to deal with that NW context.

During the next step the CEDM will -- depending on the nature of the received context and its parameter -- plan and initiate the deployment of Mobile Access Points with 5G capability, based on the pool of available MAPs and their characteristics. Additional MAPs might be needed in case the existing ones cannot fulfill the constraints of that particular case (e.g., no drone has been already physically deployed in the targeted area).

We do not provide a sequence diagram as it would be very similar to the previous one (Figure 17).

4.3.4.5 Coverage Extension – MAP Placement optimization

The MAP placement optimization is a fully distributed algorithm where MAPs collaborate with each other, implementing the same algorithm as shown in Figure 18. This algorithm loops between steps 3 and 8 as explained below in the note. Simulations have shown that it converges relatively quickly towards an optimal spatial distribution.

Once the configuration and start requests from the CEDM FC have been received (arrows 1&2), the Swarm Operation FC collects information from several FCs:

- NW Awareness FC, e.g., information about the total network utility function (arrow 3);
- MAP Awareness FC, e.g., information about the MAP positions and the previous decisions taken by the MAP (arrow 5);
- UE Awareness FC e.g., RSS, AoA between the MAP and the UE, UE status and UE QoS (arrow 6);
- MAP status Agent e.g., information about MAP capabilities such as energy, cost or position (arrow 4).

Once the information is collected, the MAP can take locally the decision about its placement and update its own position (arrow 7). This FC is in charge of defining the best trajectory to follow until the destination or the next position and to avoid MAP collisions. Upon validation of the request, MAP can move following the defined trajectory or to the next position. This position change updates the UE/MAP/Network awareness FCs. The NW Awareness FC can calculate the new total network utility function using the NW Performance Analytics FC and broadcast its results.

Please note: arrow 9 is actually the same than arrow 3 as the process loops from arrow 3 to arrow 8. It is just meant to "close" the loop explicitly.

This new MAP placement can imply a new UE-{MAP/BS} selection. Thus, a request is sent to UE-BS Selection DM FC.

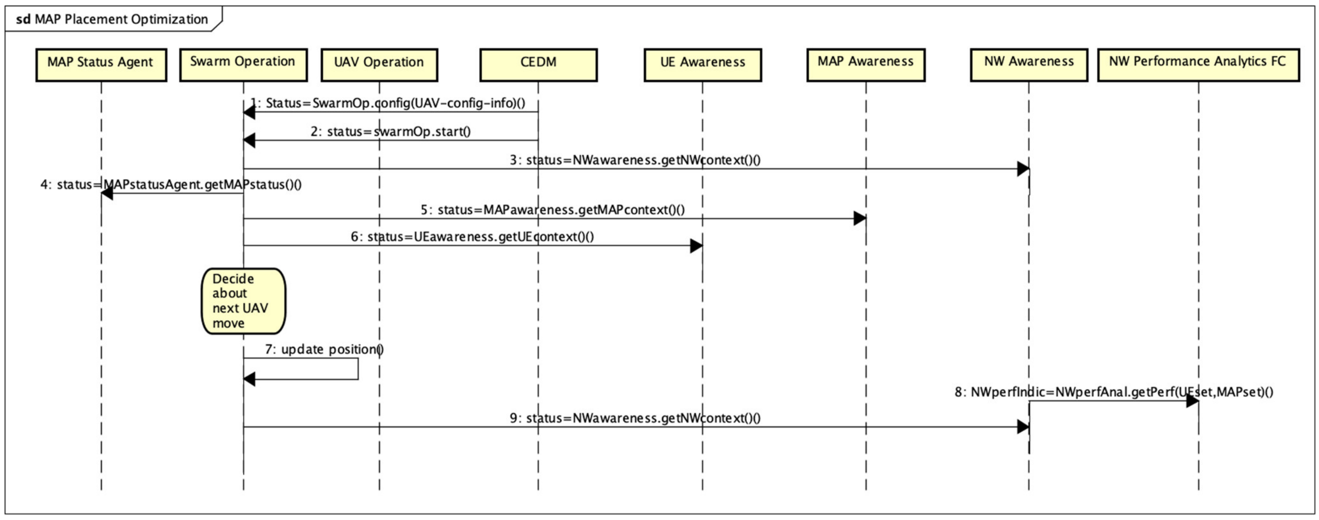

sd MAP Placement Optimization
MAP Status Agent
Swarm Operation
UAV Operation
CEDM
UE Awareness
MAP Awareness
NW Awareness
NW Performance Analytics FC
1: Status=SwarmOp.config(UAV-config-info)()
2: status=swarmOp.start()
3: status=NWawareness.getNWcontext()()
4: status=MAPstatusAgent.getMAPstatus()()
5: status=MAPawareness.getMAPcontext()()
6: status=UEawareness.getUEcontext()()
Decide about next UAV move
7: update position()
8: NWperfIndic=NWperfAnal.getPerf(UEset,MAPset)()
9: status=NWawareness.getNWcontext()()

Figure 18: Sequence diagram for MAP placement optimization

4.3.4.6 Coverage Extension - Setting up an optimized UE/{BS,MAP} association

This first UE-{BS/MAP} association-related SUC focuses on the mechanisms that take place between a UE and surrounding fixed BSs and MAPs in order to find the better association between the two entities. It elucidates in particular the DM process and the information that is used for that purpose. This set-up activity is meant to be continuously initiated by the UE and cooperatively performed between the UE and BSs / MAPs.

The selection process results of a tri-partite negotiation between the UE (the initiator) and the BSs and MAPs.

Prior to the negotiation, the UE-BS Selection DM FC at the UE side, starts to collect information from several FCs:

- NW Awareness FC, e.g., information about the total network utility function, the RSS, and network contexts like supported protocols, channels, and IDs;
- MAP Awareness FC e.g., information about the RSS, AoA for all the MAPs
- UE awareness FC e.g., information about other users such as their positions and their last decision;
- UE status Agent FC, e.g., information about the requested service level;
- Network Performance Analytics FC, e.g., the current perceived network performance.

Once all this data is collected, the UE-BS Selection DM FC seeks to find the best possible association(s). When UEs are already associated with BS/MAPs, this DM FC decides whether it keeps the association to the existing MAP/BS or handovers to another one. For UEs capable of multi-connectivity, multiple eligible BSs and/or MAPs can be decided in order to engage into an association process. This information passed by the DM on to the UE-BS Negotiation FC includes: a sub-set of eligible BSs and MAPs IDs and a network context.

This SUC related also to the one described in Section 4.3.4.7 about MAP deployment, where close monitoring of existing association can result into modifying the deployed MAPs configurations or performing additional MAP deployment.

The following message chart (Figure 19) shows the UE-association part. The first half part (arrows 1-7) is about acquiring the relevant environment contexts, which are about corresponding entities (The info of UE/MAPs for BSs and the info of BSs/MAPs for UEs). The latter part is the actual operation of UE association. UE-BS Selection DM FC will find a set of BSs/MAPs for the UE association based on UE context and network context information. Then, it passes the info of selected BSs/MAPs to UE-BS Negotiation FC. UEs first propose their associations, then BSs and MAPs will check their capacity, current traffic load and existence of other UEs who want to also join. Then, they decide whether they accept a certain user or not. Their decision will be fed back to the UEs. The process of proposal-accept/reject can be occurred between multiple UEs and multiple BSs/MAPs at the same time. Then, UE-BS Negotiation FC will initiate the negotiation process with BSs and MAPs to decide the connection configuration to decide the connection configuration.

At the end of the procedure, the UE updates its perceived rate and broadcasts it, in particular to the network status agent FC calculating the total network utility function, hence validating the global network performance of UE-BS association procedure.

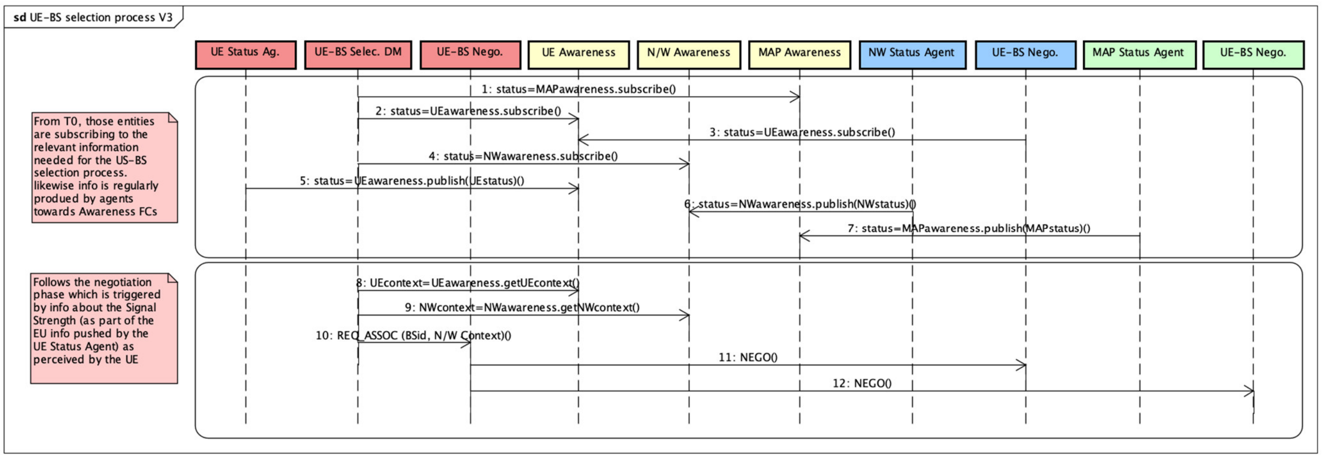

sd UE-BS selection process V3
UE Status Ag.
UE-BS Selec. DM
UE-BS Nego.
UE Awareness
N/W Awareness
MAP Awareness
NW Status Agent
UE-BS Nego.
MAP Status Agent
UE-BS Nego.
From T0, those entities are subscribing to the relevant information needed for the US-BS selection process. likewise info is regularly produed by agents towards Awareness FCs
1: status=MAPawareness.subscribe()
2: status=UEawareness.subscribe()
3: status=UEawareness.subscribe()
4: status=NWawareness.subscribe()
5: status=UEawareness.publish(UEstatus)()
6: status=NWawareness.publish(NWstatus)()
7: status=MAPawareness.publish(MAPstatus)()
Follows the negotiation phase which is triggered by info about the Signal Strength (as part of the EU info pushed by the UE Status Agent) as perceived by the UE
8: UEcontext=UEawareness.getUEcontext()
9: NWcontext=NWawareness.getNWcontext()
10: REQ_ASSOC (BSid, N/W Context)()
11: NEGO()
12: NEGO()

Figure 19: Sequence diagram for UE/{BS,MAP} association

4.3.4.7 Coverage Extension – Real-time Performance Monitoring and Triggering Coverage Extension

Compared to the case described above, optimizing the association between UE and fixed-BS and deployed MAP is achieved in real-time in order to get always as good and reliable as possible connection from the UE point of view.

Figure 20 shows the process of the optimization of coverage extension. The assumptions made before the MSC below executes are:

- UEs, BSs/MAPs are broadcasting their information, thus UEs or BSs/MAPs can get the context information of corresponding entities;
- MAPs are already deployed with instructions from the BS;
- UE-Association is carried out by negotiating between UE and BS/MAP.

It also relates to the following SUC (4.3.4.12, 4.3.4.10, 4.3.3.9).

As shown in the first half part of the chart, the required context information is gathered by UEs, BSs, and MAPs (arrows 1-7). Then, the continuous monitoring of network performance is carried out by Perf. Analytics FC. Specifically, with context information from UE, network and MAP, the FC calculates the current network performance and UE's QoS/QoE levels (arrows 8-10). The calculated performance indicator is reported to NODM (arrow 11) and NODM assesses the need of changes in the network configurations. NODM also acquires the MAP context including the battery level (arrow 12). In case NODM finds the necessity of network configuration changes, it will notify CEDM (arrow 13) and CEDM could consider two options: 1) to decide upon new network configuration settings including redeployment of MAPs, (in)activation of a certain MAPs in a centralized manner and 2) to trigger self-configuration by MAPs.

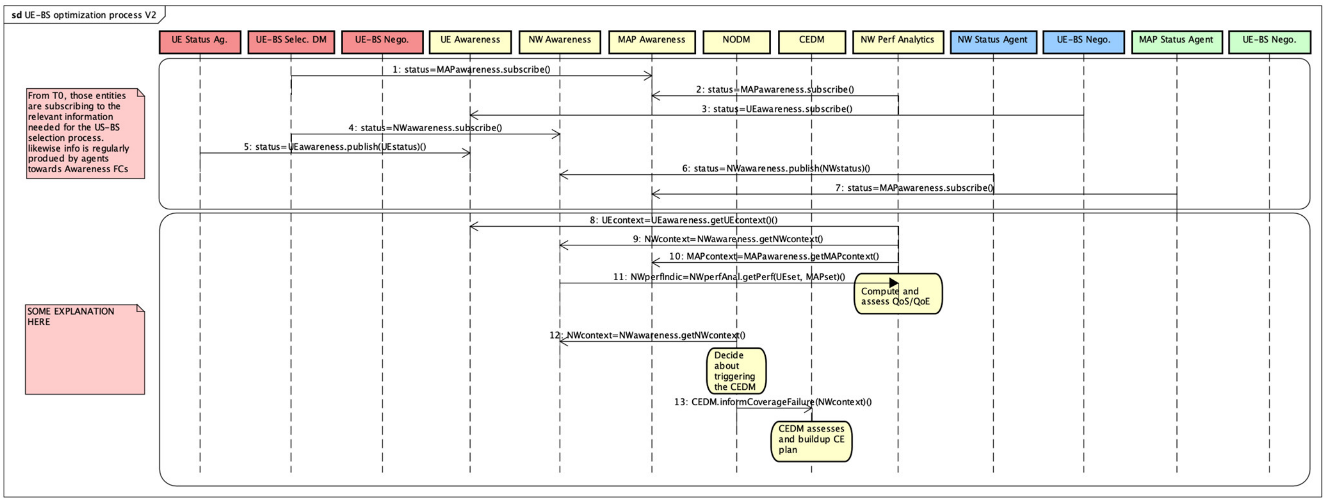

sd UE-BS optimization process V2
UE Status Ag.
UE-BS Selec. DM
UE-BS Nego.
UE Awareness
NW Awareness
MAP Awareness
NODM
CEDM
NW Perf Analytics
NW Status Agent
UE-BS Nego.
MAP Status Agent
UE-BS Nego.
From T0, those entities are subscribing to the relevant information needed for the US-BS selection process. likewise info is regularly produed by agents towards Awareness FCs
1: status=MAPawareness.subscribe()
2: status=MAPawareness.subscribe()
3: status=UEawareness.subscribe()
4: status=NWawareness.subscribe()
5: status=UEawareness.publish(UEstatus)()
6: status=NWawareness.publish(NWstatus)()
7: status=MAPawareness.subscribe()
8: UEcontext=UEawareness.getUEcontext()()
9: NWcontext=NWawareness.getNWcontext()
10: MAPcontext=MAPawareness.getMAPcontext()
11: NWperfIndic=NWperfAnal.getPerf(UEset, MAPset)()
Compute and assess QoS/QoE
SOME EXPLANATION HERE
12: NWcontext=NWawareness.getNWcontext()
Decide about triggering the CEDM
13: CEDM.informCoverageFailure(NWcontext)()
CEDM assesses and buildup CE plan

Figure 20: Sequence diagram for Real-time Performance Monitoring and Triggering Coverage Extension

Thus, the joint MAP placement and UE-{MAP/BS} association can be launched following two options: request from the operator or self-optimization.

4.3.4.8 Requesting an IDaaS or a CEaaS service

The two Figure 21 and Figure 22 aim at elucidating the ContractNET protocol as used in the context of CEaaS and IDaaS service requests.

This first phase initiates the process of provisioning an IDaaS or CEaaS service. It involves a technical person acting on behalf of a vertical business manager. We assume this person who is responsible for requesting the service has already registered to the platform and is currently authenticated (meaning he has received – as part of the authentication process- an authentication token.

This service negotiation phase uses a simple protocol -namely the ContractNet protocol – which in the most general case, allows a party -say A- to contact a set of parties -say B_i- in order to be supplied a service fulfilling a set of constraints. In our case, there is only one potential service supplier B: the DEDICAT 6G platform.

The protocol features a limited number of message exchanges:

1. Party A provides a set of service parameter to B; typically, part of the parameter are constraints about the required QoS, plus a set of parameters that are pertaining to the very nature of the service, which we will elucidate later on in the Information view;
2. Party B could either accept to provide the service (and then the protocol would terminate with an OK response being sent back to A) or would come back to A with a counter proposal, which typically would request a lower QoS or scaled up computing resources;
3. At this stage, A has the possibility of a/ agreeing upon the terms of the counterproposal (and send an OK message to B) or b/ denying it and terminating the protocol with no service provision been consequently secured;

The next two Figure 21 and Figure 22 then provide two UML sequence diagrams illustrating that protocol applied respectively to CEaaS and IDaaS. The difference between requesting an CEaaS or IDaaS will only be captured by the information passed in the 1st message exchange a.k.a. the service parameter.

Since the service is requested by the vertical via the Dashboard FC, which remains its main entry point to the platform, the initiating party A (as referred to above) in the following interaction diagram is the Dashboard FC and for a IDaaS request the contacted party (B) is IDaaS FC (respectively CEaaS FC for a CEaaS request).

The arrows between the "Vertical Operator" actor and the Dashboard FC lifelines aims to illustrate the use of the dashboard GUI. In the first figure:

- A user accesses the CEaaS service request widget from the dashboard and enters all information as described in the Section 4.4.1.2. then the formatted service request is sent to the CEDM FC via its interface (arrows 1&1.1);
- The CEDM FC, after analyzing the request parameters, comes up with a tentative list of MAPs (vertical ENs and µS are already known from the request) and requests the IDDM FC to check if a deployment is possible assuming the individual characteristics and constraints of all ENs (incl. MAPs), FCs and µSs involved (arrow 1.1.1);
- The IDDM FC gathers needed additional information and performs its task→EN allocation algorithm, and finally, either validates the initial request or proposes an update of that request (often due to a EN scalability issue) which is called counterproposal (or CP) – arrows 2-4;
- Finally, the user has the possibility to accept either the initial or upgraded requests or to abort/deny those. (arrows 5-12).

The three different outcomes have been modeled in three different UML fragments.

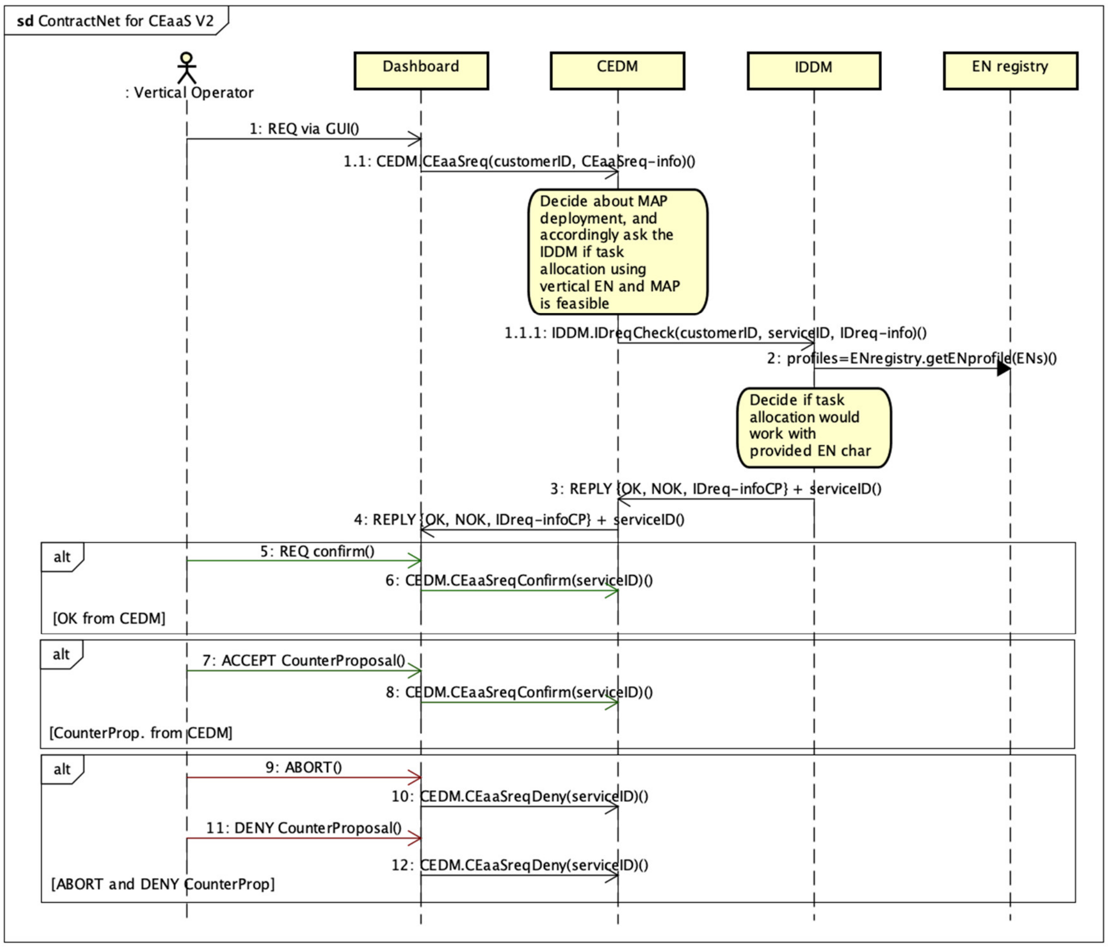

sd ContractNet for CEaaS V2
: Vertical Operator
Dashboard
CEDM
IDDM
EN registry
1: REQ via GUI()
1.1: CEDM.CEaaSreq(customerID, CEaaSreq-info)()
Decide about MAP deployment, and accordingly ask the IDDM if task allocation using vertical EN and MAP is feasible
1.1.1: IDDM.IDreqCheck(customerID, serviceID, IDreq-info)()
2: profiles=ENregistry.getENprofile(ENs)()
Decide if task allocation would work with provided EN char
3: REPLY {OK, NOK, IDreq-infoCP} + serviceID()
4: REPLY {OK, NOK, IDreq-infoCP} + serviceID()
alt
5: REQ confirm()
6: CEDM.CEaaSreqConfirm(serviceID)()
[OK from CEDM]
alt
7: ACCEPT CounterProposal()
8: CEDM.CEaaSreqConfirm(serviceID)()
[CounterProp. from CEDM]
alt
9: ABORT()
10: CEDM.CEaaSreqDeny(serviceID)()
11: DENY CounterProposal()
12: CEDM.CEaaSreqDeny(serviceID)()
[ABORT and DENY CounterProp]

Figure 21: A Vertical requests an CEaaS

As far as IDaaS service provision is concerned, we have a similar process, at the difference that since the CEDM is not involved, we do not deal with any MAP when checking feasibility out:

- A user accesses the IDaaS service request widget from the dashboard and enters all information as described in the Section 4.4.1.2. Then the formatted service request is sent to the IDDM FC via its interface (arrows 1&2);
- The IDDM FC gathers needed additional information and performs its task/EN allocation algorithm, and finally either validates the initial request or proposes an update of that request (often due to a EN scalability issue) which is called counterproposal - or CP (arrows 3&4);
- Finally, the user has the possibility to accept either the initial or upgraded requests or to abort/deny those (arrows 5-12).

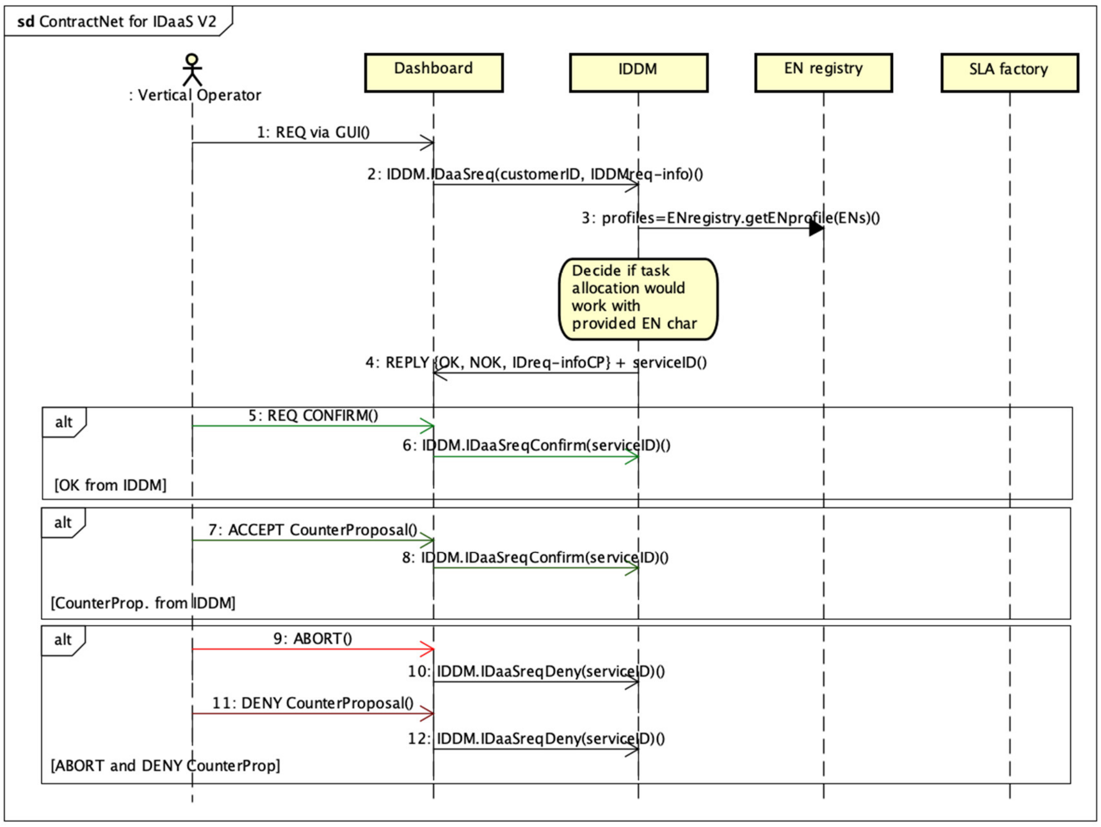

sd ContractNet for IDaaS V2
: Vertical Operator
Dashboard
IDDM
EN registry
SLA factory
1: REQ via GUI()
2: IDDM.IDaaSreq(customerID, IDDMreq-info)()
3: profiles=ENregistry.getENprofile(ENs)()
Decide if task allocation would work with provided EN char
4: REPLY {OK, NOK, IDreq-infoCP} + serviceID()
alt
5: REQ CONFIRM()
6: IDDM.IDaaSreqConfirm(serviceID)()
[OK from IDDM]
alt
7: ACCEPT CounterProposal()
8: IDDM.IDaaSreqConfirm(serviceID)()
[CounterProp. from IDDM]
alt
9: ABORT()
10: IDDM.IDaaSreqDeny(serviceID)()
11: DENY CounterProposal()
12: IDDM.IDaaSreqDeny(serviceID)()
[ABORT and DENY CounterProp]

Figure 22: A Vertical request an IDaaS

Let us emphasize that those two diagrams are about requesting a service only. The next phases for IDaaS and CEaaS consists of setting the service up and undertaking all necessary steps to its actual implementation.

4.3.4.9 CEaaS (Coverage Extension as a Service)

The DEDICAT 6G system provides coverage extension as a service to vertical business applications, meaning that a third-party application would request explicitly an extension of the radio coverage with its own specific QoS requirements in order to be able to run the application while fulfilling its QoS constraints.

This section starts with the CEaaS terms already agreed upon and known (see previous sections), so the CEDM FC already knows about the MAPs it selected to cover the CE part and the IDDM already knows too about the µSs requirements and ENs provided by the vertical. The steps involved in provisioning the CEaaS service are:

1. Uploads of all EN and µS profiles to the associated registries, upload of all µS images to the repository, upload of the vertical deployment policy;
2. The CEDM FC instruments the IDDM FC to configure the MAPs (only FC relating to the AP function, e.g. MAP Status Agent and Swarm Operation FCs)
3. The CEDM FC instruments the IDDM FC to deploy all remaining FCs to all ENs (including the MAPs) such as µS/FC Status Agent and EN Status Agent FCs;
4. The IDDM FC then calculates the best µS → EN assignment and instruments the Service Orchestrator FC to implement the resulting task plan;
5. Finally, the CEDM FC organizes the physical deployment of the MAPs, which ends the service set-up.

Figure 23 below shows a sequence diagram that elucidates the very detail of the steps involved in setting up the CEaaS after the parameters have been agreed upon (that step is described in former Section 4.3.4.8).

The following steps are considered:

- **Uploads:** EN profiles, µS/FC profile and images and EC Policy need being uploaded/created in their respective repository or registries (arrows 2 to 14);
- **MAP configuration:** all by-default FCs deployment must be applied to the MAPs. The "D6G" EC policy FC is then retrieved before the relevant FCs are deployed to the MAPs as a MAP: MAP Status Agents and Swarm Operation FCs (arrows 15-28);
- **Provision of FC profile prior to FC deployment:** At this stage the IDDM FC needs to retrieve the "D6G" policy and the profiles of all subsequent FCs to be deployed to support the decision-making, i.e. all Agents FCs, Awareness FCs, load Balancing and Edge Service Orchestrator FCs which then need being deployed towards the ENs (including MAPS) either they belong to the Vertical or platform (arrow 29-31);
- **FC deployment to ENs:** FCs are deployed, and the profiles of the new service instances are created, attached to their class profiles (arrows 32-39);
- **Task allocation and deployment:** The IDDM FC needs refreshing its knowledge about EN current status following the previous step and then it computes the best µS → EN allocation and instruments the Service Orchestrator FC to implement the task allocation plan. Again, all profiles of newly created µS instances must be create in the µS/FC Registry FC. Finally, it informs the CEDM FC when the deployment is completed (arrows 40-51);
- **Physical MAP deployment and CEaaS completion:** the CEDM FC request the DEDICAT 6G tech staff to perform the physical deployment of MAPs on-site and get informed when it is completed (arrows 52 to 57).

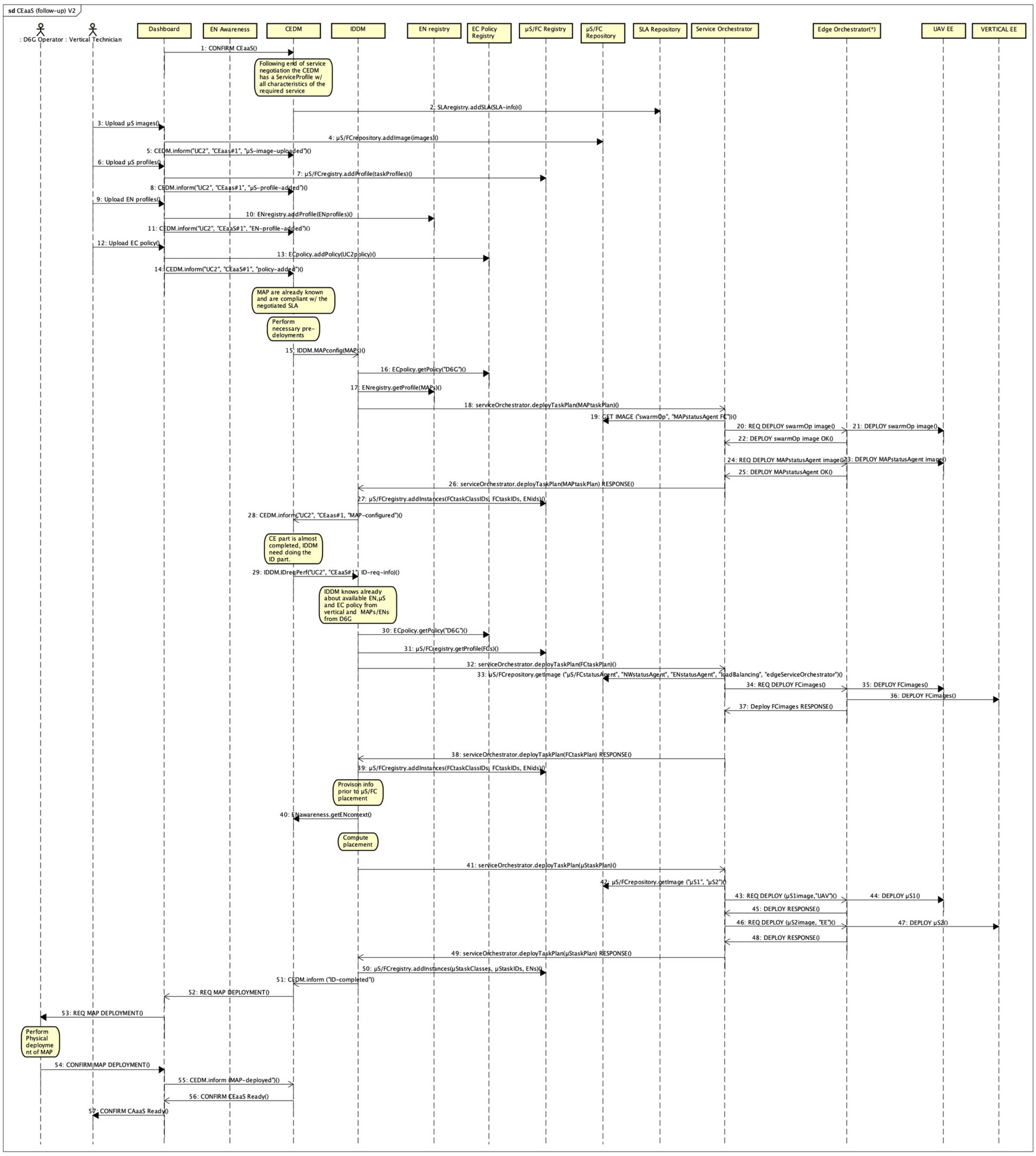

sd CEaaS (follow-up) V2
: D6G Operator
: Vertical Technician
Dashboard
EN Awareness
CEDM
IDDM
EN registry
EC Policy Registry
µS/FC Registry
µS/FC Repository
SLA Repository
Service Orchestrator
Edge Orchestrator(*)
UAV EE
VERTICAL EE
1: CONFIRM CEaaS()
Following end of service negotiation the CEDM has a ServiceProfile w/ all characteristics of the required service
2: SLAregistry.addSLA(SLA-info)()
3: Upload µS images()
4: µS/FCrepository.addImage(images)()
5: CEDM.inform("UC2", "CEaas#1", "µS-image-uploaded")()
6: Upload µS profiles()
7: µS/FCregistry.addProfile(taskProfiles)()
8: CEDM.inform("UC2", "CEaas#1", "µS-profile-added")()
9: Upload EN profiles()
10: ENregistry.addProfile(ENprofiles)()
11: CEDM.inform("UC2", "CEaaS#1", "EN-profile-added")()
12: Upload EC policy()
13: ECpolicy.addPolicy(UC2policy)()
14: CEDM.inform("UC2", "CEaaS#1", "policy-added")()
MAP are already known and are compliant w/ the negotiated SLA
Perform necessary pre-deloyments
15: IDDM.MAPconfig(MAPs)()
16: ECpolicy.getPolicy("D6G")()
17: ENregistry.getProfile(MAPs)()
18: serviceOrchestrator.deployTaskPlan(MAPtaskPlan)()
19: GET IMAGE ("swarmOp", "MAPstatusAgent FC"))()
20: REQ DEPLOY swarmOp image()
21: DEPLOY swarmOp image()
22: DEPLOY swarmOp image OK()
24: REQ DEPLOY MAPstatusAgent image()
23: DEPLOY MAPstatusAgent image()
25: DEPLOY MAPstatusAgent OK()
26: serviceOrchestrator.deployTaskPlan(MAPtaskPlan) RESPONSE()
27: µS/FCregistry.addInstances(FCtaskClassIDs, FCtaskIDs, ENids)()
28: CEDM.inform("UC2", "CEaas#1, "MAP-configured")()
CE part is almost completed, IDDM need doing the ID part.
29: IDDM.IDreqPerf("UC2", "CEaaS#1", ID-req-info)()
IDDM knows already about available EN,µS and EC policy from vertical and MAPs/ENs from D6G
30: ECpolicy.getPolicy("D6G")()
31: µS/FCregistry.getProfile(FCs)()
32: serviceOrchestrator.deployTaskPlan(FCtaskPlan)()
33: µS/FCrepository.getImage ("µS/FCstatusAgent", "NWstatusAgent", "ENstatusAgent", "loadBalancing", "edgeServiceOrchestrator")()
34: REQ DEPLOY FCimages()
35: DEPLOY FCimages()
36: DEPLOY FCimages()
37: Deploy FCimages RESPONSE()
38: serviceOrchestrator.deployTaskPlan(FCtaskPlan) RESPONSE()
39: µS/FCregistry.addInstances(FCtaskClassIDs, FCtaskIDs, ENids)()
Provison info prior to µS/FC placement
40: ENawareness.getENcontext()
Compute placement
41: serviceOrchestrator.deployTaskPlan(µStaskPlan)()
42: µS/FCrepository.getImage ("µS1", "µS2")()
43: REQ DEPLOY (µS1image,"UAV")()
44: DEPLOY µS1()
45: DEPLOY RESPONSE()
46: REQ DEPLOY (µS2image, "EE")()
47: DEPLOY µS2()
48: DEPLOY RESPONSE()
49: serviceOrchestrator.deployTaskPlan(µStaskPlan) RESPONSE()
50: µS/FCregistry.addInstances(µStaskClasses, µStaskIDs, ENs)()
51: CEDM.inform ("ID-completed")()
52: REQ MAP DEPLOYMENT()
53: REQ MAP DEPLOYMENT()
Perform Physical deployme nt of MAP
54: CONFIRM MAP DEPLOYMENT()
55: CEDM.inform (MAP-deployed")()
56: CONFIRM CEaaS Ready()
57: CONFIRM CAaaS Ready()

Figure 23: CEaaS UML Interaction Diagram

4.3.4.10 IDaaS (Intelligence Distribution as a Service)

This section starts with the IDaaS service request terms being already agreed upon by both the Vertical and platform, and known (see previous sections), so the IDDM FC knows already about the µSs requirements and ENs provided by the vertical. The steps involved in provisioning the service are:

1. Uploads of all EN and μS profiles to the associated registries, upload of all μS images to the repository, upload of the vertical deployment policy;
2. The IDDM FC deploys all mandatory FCs to all ENs such as μS/FC Status Agent and EN Status Agent FCs;
3. The IDDM FC then calculates the best μS → EN assignment and instruments the Service Orchestrator implement the resulting task plan which ends the service set-up.

Please note that this scenario does not involve Coverage Extension.
The UML diagram below (Figure 24) shows a "typical" interaction diagram for setting-up and configuring the service. The initial part consisting of negotiating the terms and characteristics of the required service is elucidated in the Figure 22 above.
In this figure we only remind of the latest step of the negotiation phase where the Dashboard confirms the IDaaS that the vertical user agrees upon the service terms (either relating to the initial request or to the IDDM counter proposal).
One can notice on the right-most side of the sequence diagram an EE lifeline. It refers to actual target of the Edge Service Orchestrator FC: an *Execution Environment (EE)* hosted by a Physical System. It means that while the IDDM FC assigns globally μSs/FCs to an edge node – as part of the task→EN allocation algorithm, the Edge Service Orchestrator FC takes the relay, in order to assign the μSs/FCs to the available EEs within that edge node.
In this example we don't make explicit any edge node deployment. On the contrary, it is assumed already deployed and up and running, so that needed FCs and μSs can be successfully deployed into it.
We have already illustrated a MAP/EN deployment scenario in Section 4.3.4.9 (CEaaS Interaction diagrams) and shall go deeper into those MAP/EN deployment matters in Section 4.3.4.12.

The sequence diagram below is made of various steps as described hereafter:

- **Uploads:** EN profiles, μS/FC profiles and images and EC Policy need being uploaded/created in their respective repositories or registries (arrows 2-14). This is what normally follows the IDaaS request and its negotiation process;
- **EN configuration:** all by-default FCs deployment (all needed Agents, Load Balancing, Edge Service Orchestrator FCs) must then be applied to the vertical edge nodes. The "D6G" EC policy is therefore retrieved before the relevant FCs are deployed to the ENs (arrows 15-23);
- **Provision of information prior to task allocation:** the first deployment had impact on the EN loads so the new EN context is needed before going on with μS → EN allocation (arrow 24);
- **Task allocation and deployment:** the IDDM FC computes the best μS → EN allocation and instruments the Service Orchestrator FC to implement the resulting task allocation plan. Finally, it informs the Dashboard FC when the deployment is completed (arrows 25-34);

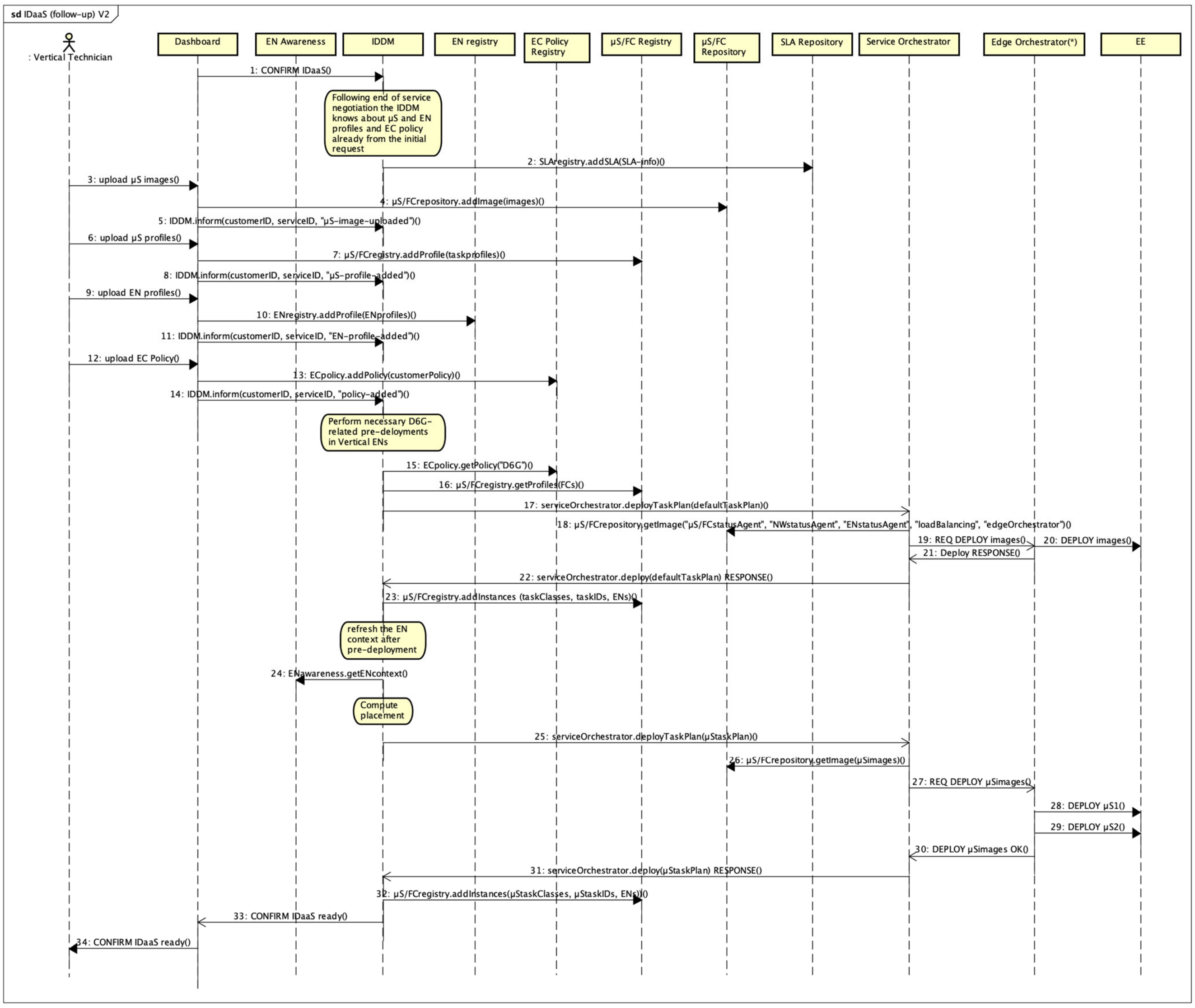

sd IDaaS (follow-up) V2
: Vertical Technician
Dashboard
EN Awareness
IDDM
EN registry
EC Policy Registry
μS/FC Registry
μS/FC Repository
SLA Repository
Service Orchestrator
Edge Orchestrator(*)
EE
1: CONFIRM IDaaS()
Following end of service negotiation the IDDM knows about μS and EN profiles and EC policy already from the initial request
2: SLAregistry.addSLA(SLA-info)()
3: upload μS images()
4: μS/FCrepository.addImage(images)()
5: IDDM.inform(customerID, serviceID, "μS-image-uploaded")()
6: upload μS profiles()
7: μS/FCregistry.addProfile(taskprofiles)()
8: IDDM.inform(customerID, serviceID, "μS-profile-added")()
9: upload EN profiles()
10: ENregistry.addProfile(ENprofiles)()
11: IDDM.inform(customerID, serviceID, "EN-profile-added")()
12: upload EC Policy()
13: ECpolicy.addPolicy(customerPolicy)()
14: IDDM.inform(customerID, serviceID, "policy-added")()
Perform necessary D6G-related pre-deloyments in Vertical ENs
15: ECpolicy.getPolicy("D6G")()
16: μS/FCregistry.getProfiles(FCs)()
17: serviceOrchestrator.deployTaskPlan(defaultTaskPlan)()
18: μS/FCrepository.getImage("μS/FCstatusAgent", "NWstatusAgent", "ENstatusAgent", "loadBalancing", "edgeOrchestrator")()
19: REQ DEPLOY images()
20: DEPLOY images()
21: Deploy RESPONSE()
22: serviceOrchestrator.deploy(defaultTaskPlan) RESPONSE()
23: μS/FCregistry.addInstances (taskClasses, taskIDs, ENs)()
refresh the EN context after pre-deployment
24: ENawareness.getENcontext()
Compute placement
25: serviceOrchestrator.deployTaskPlan(μStaskPlan)()
26: μS/FCrepository.getImage(μSimages)()
27: REQ DEPLOY μSimages()
28: DEPLOY μS1()
29: DEPLOY μS2()
30: DEPLOY μSimages OK()
31: serviceOrchestrator.deploy(μStaskPlan) RESPONSE()
32: μS/FCregistry.addInstances(μStaskClasses, μStaskIDs, ENs)()
33: CONFIRM IDaaS ready()
34: CONFIRM IDaaS ready()

Figure 24: IDaaS UML Sequence Diagram

4.3.4.11 SLA enforcement

We have seen earlier that the SLA Registry FC is responsible for storing/retrieving and deleting SLAs. However, its most important role is SLA enforcement during the execution of CEaaS or IDaaS request, the formal and legal terms of which are captured within a service level agreement.

The following UML sequence diagram (Figure 25) illustrates the system use-case where the SLA Registry FC monitors specific contexts relating to the CEaaS `serviceID`, detects breaches and notifies the CEDM FC so that it can take corrective actions. Contract breaches being serious matter, the SLA Registry FC relies on the Logging FC for logging both incidents and corrective actions, after they have been verified.

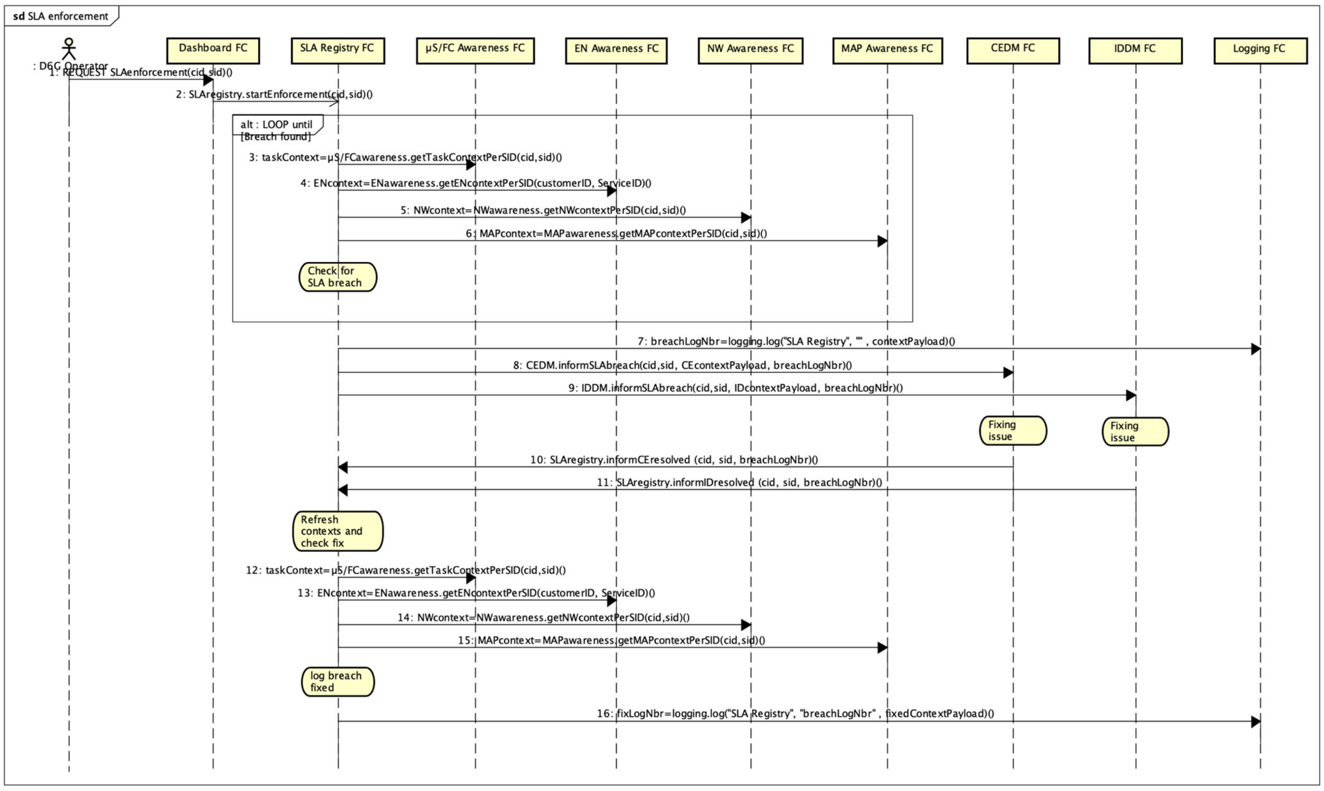

sd SLA enforcement
: D6G Operator
Dashboard FC
SLA Registry FC
µS/FC Awareness FC
EN Awareness FC
NW Awareness FC
MAP Awareness FC
CEDM FC
IDDM FC
Logging FC
1: REQUEST SLAenforcement(cid,sid)()
2: SLAregistry.startEnforcement(cid,sid)()
alt : LOOP until [Breach found]
3: taskContext=µS/FCawareness.getTaskContextPerSID(cid,sid)()
4: ENcontext=ENawareness.getENcontextPerSID(customerID, ServiceID)()
5: NWcontext=NWawareness.getNWcontextPerSID(cid,sid)()
6: MAPcontext=MAPawareness.getMAPcontextPerSID(cid,sid)()
Check for SLA breach
7: breachLogNbr=logging.log("SLA Registry", "" , contextPayload)()
8: CEDM.informSLAbreach(cid,sid, CEcontextPayload, breachLogNbr)()
9: IDDM.informSLAbreach(cid,sid, IDcontextPayload, breachLogNbr)()
Fixing issue
Fixing issue
10: SLAregistry.informCEresolved (cid, sid, breachLogNbr)()
11: SLAregistry.informIDresolved (cid, sid, breachLogNbr)()
Refresh contexts and check fix
12: taskContext=µS/FCawareness.getTaskContextPerSID(cid,sid)()
13: ENcontext=ENawareness.getENcontextPerSID(customerID, ServiceID)()
14: NWcontext=NWawareness.getNWcontextPerSID(cid,sid)()
15: MAPcontext=MAPawareness.getMAPcontextPerSID(cid,sid)()
log breach fixed
16: fixLogNbr=logging.log("SLA Registry", "breachLogNbr" , fixedContextPayload)()

Figure 25: SLA enforcement UML sequence diagram

In the sequence diagram shown in Figure 25 above, the following set of actions take place:

- At first the SLA Registry FC is instructed by an operator (arrow 1) to enforce a newly agreed SLA for a CEaaS service provision; the `customer ID` and `serviceID` are passed on to the SLA registry FC at enforcement invocation (arrow 2);
- The most important part of enforcement is then 1/ to monitor the parts of the different contexts that relates to the `CustomerID` and `serviceID` (arrows 3-6) and 2/ to detect SLA breaches (following yellow bubble). This activity takes place in a `Loop..Until` fragment;
- When an SLA breach is detected, the involved contexts are logged by the Logging FC and a `logID` is returned to the SLA registry FC (arrow 7);
- Then depending on the nature of the faulty contexts, the CEDM FC and/or IDDM FC will be notified of the SLA breach(es) and passed on the contexts they need fixing. Typically, MAP and NW contexts are passed on to the CEDM FC and µS/FC and EN contexts are passed on to the IDDM FC – arrows 8&9;
- At this stage we don't get into the detail of which actions the CEDM and IDDM FCs are engaging into in order to fix the problem(s), as many scenarios have been illustrated in this section, we just assume they were able to fix them (two following yellow bubbles on the CEDM and IDDM lifelines);
- The IDDM FC and/or CEDM FC then notifie(s) the SLA Registry FC that the SLA breach logged as `logID` has been fixed (arrows 10&11);
- The final steps involve checking the relevant new contexts (arrows 12-15) and logging them (as we assume here the issues were actually properly fixed), referring to the original `logID` (arrows 16);

Additional actions (which we don't describe here) could be 1/ to notify the client about a occurring SLA breach, 2/ to notify the client that a breach has been fixed (in a certain amount of time) 3/ has not been fixed after an agreed certain amount of time (as an additional SLA term).

4.3.4.12 MAP/MEC physical deployment and operation

In DEDICAT 6G we consider four classes of MAPs/MECs as part of our baseline architecture:

- **UAV:** they are Unmanned Aerial Vehicles. While the project is focusing on the use of drones as far as aerial vehicles are concerned, the use of airships could be considered as a very interesting alternative (deployment would be probably longer but they would enjoy eventually much greater energy autonomy and tolerate much heavier payloads);
- **AGV:** They are Autonomous Guided Vehicles (at large) and DEDICAT 6G includes the use of Robots as the baseline solution (smart vehicles are also introduced in UC4);
- **Manned Connected Vehicles** (MCV), a.k.a. Connected cars/vans: They can be easily deployed on demand and enjoy a long autonomy;
- **Small servers:** they are reasonably small in size and weight and can be easily deployed in order to support scenarios relying on Intelligence Distribution, like for instance Critical Mission management in the context of natural disaster (e.g., UC3 Scenario).

UAV, AGV and MCV are used for the purpose of coverage extension and Intelligence distribution, while small servers are used to support Intelligence Distribution only.

The use of MAP results from either 5G network malfunctions (bad performance, faulty devices, lack of capacity vs. demand) or service request for enhanced coverage or intelligence distribution (a.k.a. respectively CEaaS and IDaaS).

Therefore, the deployment of MAPs cannot be planned in advance and performed without involving human operators such as: physically moving the MAP to the suitable location and making sure they are ready for communicating with the Cloud components. Also drones (and robots as well to a lesser extend though) cannot fulfill entirely their allocated missions without docking regularly (every 20~25min for drones depending on their payload weight) in order to recharge/change batteries. These steps need human intervention as well.

After being deployed on suitable location near the operation field, the unmanned MAPs are self-organizing (as a swarm) and physically deploy in order to cover optimally a geographic area instructed by the decision-making.

In a nutshell, the system use-case for MAP deployment relies on the following steps, assuming the decision-making FCs have already decided about a deployment plan. Please refer to System UCs Sections 4.3.4.4, 4.3.4.9 and 4.3.4.10 for more detail about what actually happens before a MAP/MEC deployment):

1 Notifying a human operator about what needs to be deployed (nature of the MAP/MEC and number) and where;
2 When on site, make sure the MAP/MEC are ready for communication with the cloud (meaning e.g., their embedded gNB can talk with the closest IAB-donor) and double-check needed FCs and interfaces are available;
3 Depending on the outcome of step 2, possibly engage into MEC management in order to create entries into Edge Node registry etc. (See Managing Edge Node System Use-case above), Some –by default- or needed FCs are then automatically uploaded;
4 Possibly some additional FCs are uploaded to the MAP/MEC (in addition to FCs which are deployed in the MAP/MEC by default;

5 At this stage, the MAPs – instructed by the cloud or by the Swarm Operation FC- initiate physical deployment (to designated place or to perimeter to be covered);
6 When a MAP battery is shy of being fully depleted, the MAP returns automatically to its assigned docking station, then its battery is replaced for a fully charged one by a human operator (notification through the GUI may be used beforehand) and is then re-deploying to its initial position.

We do not develop here a new specific sequence diagram. Some examples of involving DEDICAT 6G technical staff in the context of dynamic coverage extension provisioning, can be found in the CEaaS example Section 4.3.4.9 and in the UC2 example Section 4.2.2.1.

4.3.4.13 A μS / FC registers to the platform

All platform services are meant to be used in the private network behind the firewall and load balancer to route API to different FC. Integration of new services depends on the service nature and API to be exposed to the outer world. So, registering a new service is as simple as deploying it and updating the load balancer (LB - add to the List of Acronyms and Abbreviations), and configuring it to communicate with other services.

Communication with other services can be over TCP, HTTP, or RPC protocols. TCP and HTTP communication usually implies configuring services for using TLS. Mutual TLS is also possible for enhanced security. Since RPC usually uses HTTP, HTTP2, or IP as an underlying protocol, the same TLS can be used for RPC for increased security. The other approach is to deploy a new FC on-premise and use it to communicate with the platform the same way any other client does - using API (HTTPS, TLS, or RPC) exposed through LB and firewall.

If a new service is deployed in the internal network (see Figure 26), securing communication using TLS is recommended, but not required because TLS termination and firewall filtering are done on the load-balancing level before requests even reach the service. If the service is running on-premise, the new service acts like an external client and TLS is mandatory. In order to utilize platform access control, the new service needs to protect its API endpoints by calling the internal AuthZ FC RPC API. The diagram shows a new microservice plugged into the platform.

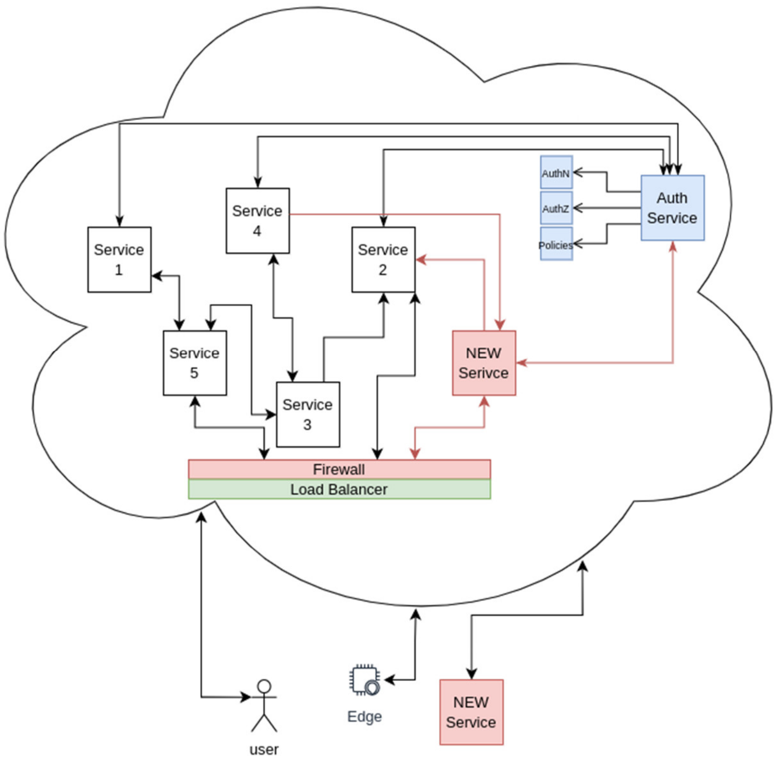


Figure 26: Authorizing a new service

4.3.4.14 A user registers to the platform

There are two ways of registering a new user:

1. Admin creates a new user profile;
2. User registers to the platform.

Both cases are similar from the flow standpoint and include sending user credentials to the AuthN FC. The only difference is that in the first case, the request is authorized against an administrator, and in the second case - it is not as there is no way to authorize a request.

The credentials consist of username and password since that is the most common way of user authentication. Also, the platform can easily be extended to support authorization protocols such as OAuth2[5] and its extension OpenID Connect[6].

The registration REST API flow is shown in the sequence diagram below (Figure 27).

[5] https://oauth.net/2/

[6] https://openid.net/connect/

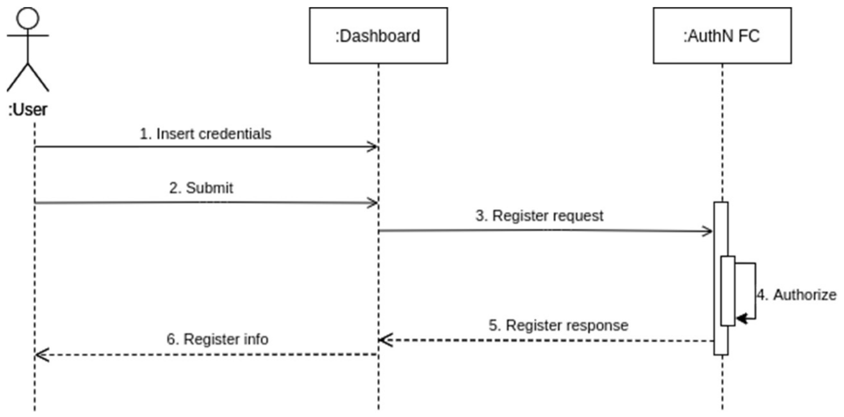


Figure 27: Sequence diagram for user registration

Register response can be an error or a success. Error response includes an error message and appropriate status code. For example, in the case of an invalid credentials format, the status code is `400 – Bad Request`. Invalid credentials format can be, for example, an empty password or password of invalid length and format or unique username violation. In case of successful registration, a new user is created with the provided username and password. The password is stored in the database in the form irreversible hash. Salt - a randomly generated string - is added to the password before hashing to prevent users with the same password to have the same hash in the database. A more precise sequence diagram that uses the actual interface as defined in Section 4.4.1.7 Is show in the Context view UC2 example, in Section 4.2.2.1.

4.3.4.15 A user (or µS/FC) authenticate to the platform

The platform is using token-based authentication (see Figure 28 below). The main benefits of token-based authentication are simplicity and its stateless nature. In the reference implementation, *Json Web Tokens (JWT)*[7] are used. The main benefit of JWT is that they are self-contained, and all the necessary access control data is encoded within the token. There are two types of tokens: Access token and Refresh token. Access token consists of encoded claims such as time of the issuing, validity period, client identifier (username, but can also be an arbitrary identifier in the case client is an application), and client role. Refresh token contains information used to generate a new access token. Access tokens are short-lived and are not stored in the database, unlike refresh tokens. Access token life is usually measured in minutes, and they cannot be revoked. Stealing both access and refresh tokens is unlikely. However, if the access token is compromised in some way, the user needs to be disabled to prevent anyone from acting on behalf of the user utilizing the stolen token.

Refresh tokens can be revoked which disables them to be used for issuing a new access token. Once used, the refresh token is archived and replaced with a new one. Each refresh token can be used only once. The diagram below shows a login flow. In the case of a successful login, the response could consist of e.g., a JSON payload containing both access and refresh tokens.

µSs and FCs must be provided with tokens that can be used for authentication and authorization matter. This can be done in different ways which are left outside the scope of this document as it is a design choice.

[7] https://jwt.io/

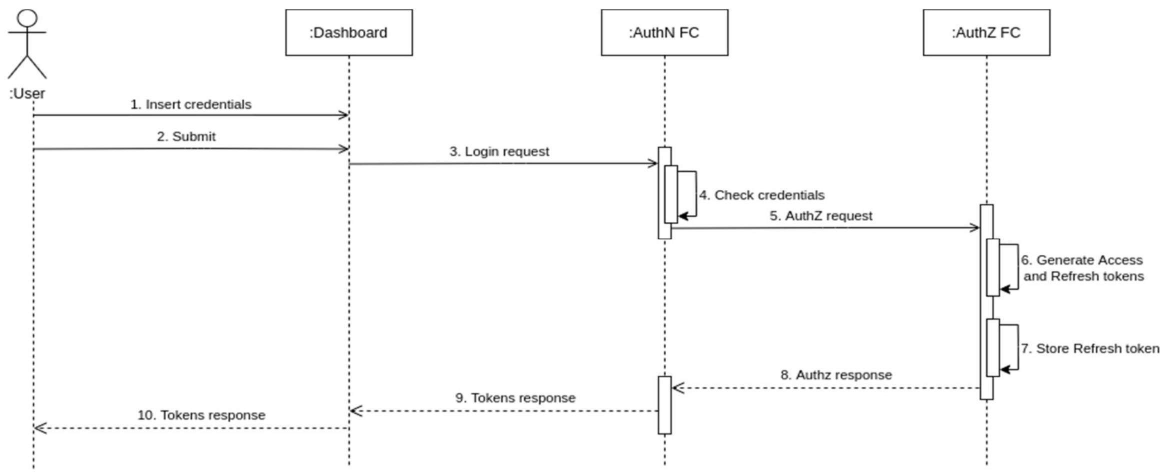


Figure 28: Sequence diagram for Authentication

4.3.4.16 An entity authorizes access to its resource

Each user-initiated request to any of the services needs to be authorized (Figure 29), except registration and login for obtaining access and refresh token requests. In REST API access token is passed in the Authorization HTTP header. AuthN FC is the central authentication and AuthZ is the main authorization component. The AuthZ FC provides the RPC API to be used by any new service that registers to the platform. That way, the new service does not have to implement client authorization but simply passes the client token alongside the desired action (such as cerate, read, view, list, etc.) to the AuthZ service for authorization and receives the response which is as simple as "access granted" and "access denied". An example of the authorized flow is shown in the diagram below. Please note that communication between Resource DC and AUTH FC is RPC, rather than REST API.

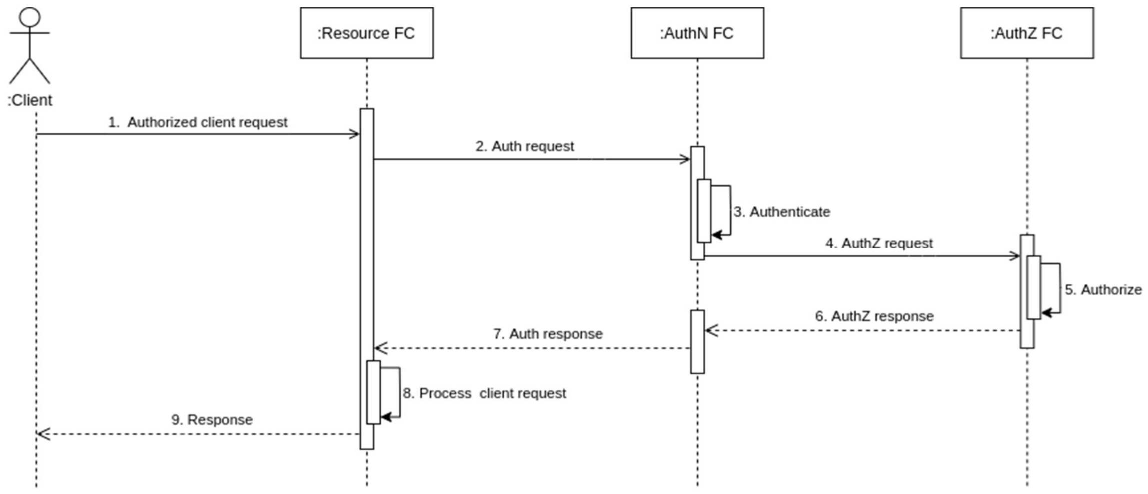


Figure 29: Accessing a request with authorization from the target

4.3.4.17 EDGE μS/FC publishes to a message queue

This use case elucidates the way the communication is handled by DEDICAT 6G in order to provide secured and confidential exchange of information between two parties involved in a "larger" system use-case. This system use-case and the three following ones are going through different options depending on 1/ the chosen communication paradigm and 2/ how the two communicating parties are deployed w.r.t. each other any interactions occurring in DEDICAT 6G will be following one of those four cases.

This first case shows how PST is involved in communication between two parties, say A and B- using publish/subscribe and sub-divides into two sub-SUC as the communications is asynchronous.

4.3.4.18 EDGE or Cloud µS/FC subscribe to a data thread and collects messages from a message queue

Initially, platform security is modeled against the request-response concept where each request holds a token for authorization. For a simple pooling model, the platform handles subscriptions the same way it handles requests.

However, the pub-sub model works a little differently depending on the protocol. The commonly used protocol in IoT use-cases (usually used by edge devices) is MQTT which requires an MQTT message broker deployed as a microservice. Only the subscribe request will be authorized in the same way as any other request. All the other messages, delivered by the broker to clients are left up to the broker and protocol specification and are not authorized by the platform itself. A broker can possibly be configured to authorize those requests using provided RPC API, but it’s up to the broker and its configuration.

4.3.4.19 EDGE or CLOUD µS/FC publishes to a CLOUD FC

Publishing can be more easily translated to the request-response model since publishing is just an extension of a request. Depending on the protocol, an access token can be different, but conceptually it works the same. For example, for MQTT Publish message has username and password fields, and an access token can be put in either of them. CoAP or some similar constrained protocols do not have headers or special metadata, so the token needs to be encoded elsewhere.

In the initial system architecture, there is no special authorization process for cloud-to-cloud communication, that is - clients pass tokens around for authentication and authorization and the same is applied to services - they act as clients to other services. If cloud FCs are deployed in the private network, it is not necessary to exchange even tokens, as token-based authorization is heavily user-focused.

4.3.4.20 CLOUD FC subscribes to a data thread and collects messages from a message queue

Works the same as EDGE FC if FC is deployed outside of the private network. If it’s deployed in the same cloud, it can be more flexible as it’s the internal network - explained in “A µS/FC registers to the platform” section above.

4.3.4.21 A µS/FC interacts with another µS/FC using Asynchronous Communication

This is really up to the internal components. In the initial implementation, components communicate using RPC - synchronously due to the nature of internal communication that’s mostly related to authorization. However, adding a message bus and using it is as simple as adding any other new FC. Of course, microservices need to be configured and the internal network needs to enable communication between components or between components and the message bus (if the bus is used for asynchronous communication).

4.3.4.22 The Data Market Place builds up a time-series database from Vertical data

One of the possible use-cases addresses using the platform for building up a vertical solution for collecting data and securing access for later visualization and processing. This can be achieved by providing an internal message broker as a communication channel between different FC. At least one FC needs to be used as a remote service for data collection - an edge device can publish authorized messages to the entry point FC. Please note that a message broker is not a requirement and a single microservice can process and push messages directly to the database. However, for the sake of scalability and to make the entire system less coupled and more distributed, a message broker will be used paired with multiple FCs to process messages forming a processing pipeline.

Entry point FC would process data and publish it to the internal message broker, where other FC can subscribe and consume messages, process them, and, if needed, push them back to the internal message broker. At least one FC will be in charge of picking up the processed messages and deciding how to store them. Finally, at least one FC will be in charge of controlling the access to the stored messages. The FC that takes care of accessing data will be running what we call DProxy - FC that can be deployed in front of data sources to handle secure access to it.

Please note that decoupling FC to microservices does not have to be mapped 1:1. For example, one microservice can be in charge of receiving, processing, and pushing data to the message broker and another for writing to the database and reading from it. The options are limitless and depend purely on the specifics of the use case.

4.4 Information View

The purpose of the Information view is three-fold:

- Providing clear and precise interface to the FC introduced in the Functional view. They follow the object-oriented paradigm without sticking to a particular programming language syntax;
- Providing the needed supporting data models;
- Elucidate some typical data flows.

4.4.1 Interfaces and Data Models

This section is structured following the Functional model, meaning structured per Functional Group and then per Functional Component belonging to that group.

4.4.1.1 Context-Awareness FG-related interfaces and data models

This section provides the general data structure for the various Status Agents and Awareness FCs part of the Awareness FG and elucidates their interfaces.

4.4.1.1.1. µS/FC Status Agent FC

The µS/FC Status Agent collects information about the execution of a µS or FC within a given EE part of a PS (edge node). In the following data structures, a µS or FC is designated as a `task`.

IMPORTANT NOTE: We consider here that 1/ the µS/FC agents know about the execution requirements of each of the µS/FC in their scope (within their PS/EN), 2/ they know about the resource allocation for each *Execution Environment (EE) implemented by the EN* and 3/ they are able to flag a µS/FC according to their execution statuses (please see status classification at the beginning of Section 4.4.1.1.2 below).

The associated logical data model follows:

- `Task-status-type:`
  - `metadata:`
    - `taskID` : refers to the µS or FC instance unique identifier
    - `timestamp: TIMESTAMPtype`
    - `severityFlag: {Optimal, Degrading, Critical}`
    - `taskClass: string`
    - `ownership:{"D6G"|(customerID} -` is `"D6G"` for all FCs and a `customerID` for all µSs
    - `serviceIS: string`
    - `ENid: string` - ID of the EN where the task is executed
    - `EEid: string` - ID of the EE where the task is executed
  - `data:`
    - `sharedCPUpwr: float` - % of EE allocated CPU (Core/Threads)
    - `sharedGPUpwr: float` - % of EE allocated GPU (CUDA number)
    - `sharedRAM: float` - % of EE RAM (in Gbyte)
    - `sharedGRAM: float` - % of EE GRAM (in Gbyte)
    - `sharedDISK: float` - % of EE allocated disk space (in Gbyte)
    - `sharedBandwidth: float` - % of EE allocated bandwidth (in Mbit/sec)
    - `reqCPUpwr: (int,int)` - nbr of Cores/Threads needed

- `reqGPUpwr: int` - nbr of CUDA needed
- `reqRAM: int` - amount of RAM needed (in Gbyte)
- `reqRAM: int` - amount of GRAM needed (in Gbyte)
- `reqDISK: int` - amount of Disk Space needed (in Gbyte)
- `reqBandwidth: float` - amount of bandwidth needed (in Mbit/sec)
- `usedCPUpwr: (int,int)` - nbr of Cores/Threads used
- `usedGPUpwr: int` - nbr of CUDA used
- `usedRAM : int`- amount of RAM used (in Gbyte)
- `usedRAM: int`- amount of GRAM used (in Gbyte)
- `usedDISK: int` - amount of disk space used (in Gbyte)
- `usedBandwidth: float` - amount of bandwidth used (in Mbit/sec)

In this structure, we report for each task 1/ its share of EE-allocated amount of CPU, GPU, disk space and RAM (in %) which is a useful information for the Load Balancing and Edge Service Orchestrator FCs (which works at the EE level) and also 2/ the amount of used CPU, GPU etc. (in absolute value). That sort of information is necessary to enforce SLAs on one side and to take ID decisions on the other side, whenever the computing resources allocated to a task is below its run-time requirement and do not allow for good-enough performances.

The µS/FC Status Agent FC (`µS/FCagent` service instance) has a single interface that consists of two methods (underlined):

- `tasks-status-info:[Task-status-type]=` **`µS/FCagent.`**<u>`getTaskStatus`</u> `(taskIDs: [string])`. It can be used to do a direct request about the instant status of 1/ all tasks supervised by the `µS/FCagent` instance or 2/ a subset of it if `taskIDs` is not left empty. This component is typically used by the Load Balancing FC;
- `status:string=`**`µS/FCagent.`**<u>`setServerREF`</u> `(µS/FCawarenessREFs: [ServiceHandler])`. This method is used to configure the µS/FC Status Agent FC so that it knows which µS/FC Awareness FCs `(µS/FCawarenessREFs)` it must connect to before publishing `taskStatus` payloads. There could be more than one depending on the chosen deployment strategy.

4.4.1.1.2. µS/FC Awareness FC

The µS/FC Awareness FC builds up contexts that can be used by the IDDM FC for its decision-making process.

It is paramount for the IDDM to easily single out FCs and µSs according to their execution statuses. Those are classified as follows:

- `Optimal`: the µS or FC is being executed in compliance to its requisites as 1/ advertised in the µS/FC Registry FC and 2/ supplied to the Load Balancing FC at deployment time (as instructed by the IDDM FC);
- `Degrading`: the µS or FC is still running above the resource requirement but the resources it uses at run-time gets very close to its limit. It is worth nothing that if a Warning status is issued it is likely the case the Load Balancing FC could not fix the issue beforehand, meaning that more drastic measures need being taken, e.g., migration, scaling the EE up, etc.
- `Critical`: Priority and immediate actions need being undertaken by the IDDM FC as the µS or FC execution requirements are no longer met.

Threshold associated with `Critical` and `Degrading` need being considered carefully when designing Status Agents and Awareness FCs. The associated logical data model follows

- `Task-context-info`
    - `metadata:`
        - `timestamp: TIMESTAMPtype` - for the whole context
    - `data:`
        - `taskList:[task]`- for each of the reported µS or FC in the array. They are ordered in the array in descending order of the `severityFlag`, starting with `Critical`, then `Degrading` and finishing with `Optimal`)
            - `metadata:`
                - `taskID: string`
                - `timestamp: TIMESTAMPtype` - corresponding to the original individual report
                - `severityFlag: {Optimal, Degrading, Critical}`
                - `taskClass: string`
                - `ownership: {"D6G"|customerID:string}`
                - `serviceID: string`
                - `ENid: string` - ID of the EN where the task is executed
                - `EEid: string` - ID of the EE where the task is executed
                - `reqCPUpwr: (int,int)` - nbr of Core/Thread needed
                - `reqGPUpwr: int` - nbr of CUDA needed
                - `reqRAM: int` - amount of RAM needed (in Gbyte)
                - `reqGRAM: int` - amount of GRAM needed (in Gbyte)
                - `reqDISK: int` - amount of disk space needed (in Gbyte)
                - `reqBandwidth: float` - amount of bandwidth needed (in Mbit/sec)
            - `data:`
                - `usedPwr: float` - amount of energy used (in Watt)
                - `sharedCPUpwr: float` - % of EE allocated CPU (Core/Thread)
                - `sharedGPUpwr: float` - % of EE allocated GPU (CUDA number)
                - `sharedRAM: float` - % of EE RAM (in Gbyte)
                - `sharedGRAM: float` - % of EE GRAM (in Gbyte)
                - `sharedDISK: float` - % of EE allocated disk space (in Gbyte)
                - `sharedBandwidth: float` - amount of bandwidth needed (in Mbit/sec)
                - `usedCPUpwr: (int,int)` - nbr of Core/Thread used
                - `usedCUDAnbr: int` - nbr of CUDA used
                - `usedRAM: int` - amount of RAM used (in Gbyte)
                - `usedGRAM: int` - amount of GRAM used (in Gbyte)
                - `usedDISK: int` - amount of disk space used (in Gbyte)
                - `usedBandwidth: float` - amount of bandwidth needed (in Mbit/sec)

The interface for µS/FC-related FCs is described now following both Publish/Subscribe paradigm and direct method calls. Of course, a realistic implementation will most likely use two message queues within the µS/FC Awareness FC as what gets into the Awareness FC is made of single `task-status-type` data while what gets out is made of `task-context-type` data. However, for the sake of simplicity, we use here direct method calls with a limited number of methods.

This same schema will be applied to all other Status Agents and Awareness FCs used in the architecture.

This public interface is provided by the µS/FC Awareness FC (`µS/FCawareness` service instance) and consists of eight methods (underlined):

1. `duration:int=`**`µS/FCawareness`**`.`<u>`connect`</u>`(taskAgentREF:ServiceHandler)`: the connect methods plays to roles 1/ to advertise to the µS/FC Awareness FC its producers, and 2/ return those producers the sampling rate (`duration`) they should use when publishing `task-info` payloads;
2. `status:string=`**`µS/FCawareness`**`.`<u>`disconnect`</u>`(taskAgentREF:taskHandler)`: the counter-part of `connect`;
3. `status:string=`**`µS/FCawareness`**`.`<u>`subscribe`</u>`()`: used by any client which wishes to receive µS/FC contexts (e.g. the IDDM FC);
4. `status:string=`**`µS/FCawareness`**`.`<u>`publish`</u>`(taskStatus:task-status-type)`: used by any µS/FC Status Agent FC that wishes to report a µS/FC status (`taskStatus`) to the µS/FC Awareness FC;
5. `taskContext:Task-context-type=`**`µS/FCawareness`**`.`<u>`getTaskContext`</u>`()`: returns the most recent µS/FC context available from the µS/FC Awareness FC;
6. `taskContext:Task-context-type=`**`µS/FCawareness`**`.`<u>`getTaskContextPerSID`</u>`(customerID:string, serviceID:string)`: returns the most recent µS/FC context available from the µS/FC Awareness FC relating to `serviceID` and `customerID`;
7. `taskContext:Task-context-type=`**`µS/FCawareness`**`.`<u>`getTaskContextPerENid`</u>`(ENids:[string])`: returns the most recent µS/FC context available from the µS/FC Awareness FC relating to `ENids`;
8. `status:string=`**`µS/FCawareness`**`.`<u>`setSamplingRate`</u>`(duration:int)`: determines how often a context will be published by the µS/FC Awareness FC. This duration (set in second) is sent as an answer everytime a µS/FC Status Agent FC client connects to the µS/FC Awareness FC.

Ex: `samplingRate=`**`µS/FCawareness`**`.connect(this)` where `this` (the `task` serviceHandler) is the reference to the calling object `taskAgent`.

IMPORTANT NOTE: here, we make the choice of implementing `getContext()` as a method call, as explained earlier in this section. However, an implementation of this µS/FC Awareness FC will rely on callback messages as the Publish/Subscribe paradigm relies on asynchronous communication. It means in particular, that a µS/FC Awareness FC client does not need to request a µS/FC context explicitly. On the contrary, it would receive those contexts as they become available (e.g. every couple of minutes, depending of the chosen sampling rate). This note holds for all following Awareness FCs.

4.4.1.1.3. EN Status Agent FC

The EN Status Agent FCs report to the EN Awareness FC information about the status of their edge nodes at runtime. In addition to usual metrics, the EN Status Agent FC is able to flag an EN according to the % of resource use w.r.t. the maximum available resources, including energy power consumption.

- `EN-status-type:`
    - `ENid: string`
    - `metadata:`
        - `timestamp: TIMESTAMPtype`
        - `severityFlag: {Optimal, Degrading, Critical}`
        - `ownership: {"D6G"|customerID:string}`

- serviceID: string
- fieldDeployed: {TRUE, FALSE}
- coreConfigured: {TRUE, FALSE} – TRUE meaning all D6G by-default deployment have been performed
- verticalConfigured: {TRUE, FALSE}
- standby: {TRUE, FALSE}
- location - 3D GPS location (altitude may be ignored if not relevant)

o data:

- maxPwr: float - maximum available power
- instantPwr: float - instantaneous total power consumption (in Watt)
- maxCPUpwr: (int, int) - maximum available CPU power (Core/Thread)
- maxGPUpwr: int - maximum available GPU power (CUDA number)
- maxRAM: int - maximum available RAM (in Gbyte)
- maxGRAM:int - maximum available RAM (in Gbyte)
- maxDisk:int - maximum available disk space (in Gbyte)
- maxBandwidth:float - maximum bandwidth (in Mbit/sec)
- usedCPUpwr:float - % of used CPU power (Core/Thread)
- usedGPUpwr: float - % of used GPU Power (CUDA number)
- usedRAM: float - % of used RAM (in Gbyte)
- usedGRAM - % of used GRAM (in Gbyte)
- usedDISK: float - % of used disk space
- usedBandwidth:float - used bandwidth (in Mbit/sec)
- EEs : [EE] - for each of EE (organised as an array of t-uples)
  - severityFlag: {Optimal, Degrading, Critical}
  - allocCPUpwr: (int, int) - number of allocated (Core/Thread)
  - allocGPUpwr: int - allocated (CUDA number)
  - allocRAM: int - amount of allocated RAM (in Gbyte)
  - allocDISK: int - amount of allocated disk space (in Gbyte)
  - allocBandwidth: float - allocated bandwidth (in Mbit/sec)
  - usedCPUpwr: float - total % used CPU power (Core/Thread)
  - usedGPUpwr: float - total % used GPU power (CUDA number)
  - usedRAM: float - total % used RAM (in Gbyte)
  - usedDISK: float - total % used disk space (in Gbyte)
  - usedBandwidth: float - total % used bandwidth (in Mbit/sec)

The EN Status Agent FC (ENagent service instance) has a single interface that consists of two methods (underlined):

- EN-status-info:EN-status-type=**ENagent.**getENstatus(ENid:string). It can be used to do a direct request about the instant status of the edge node (whether it is or not a MAP) supervised by the ENagent service instance;
- status:string=**ENagent.**setServerREF (ENawarenessREFs: [ServiceHandler]). This method is used to configure the EN Status Agent FC (ENagent) so that it knows which EN Awareness FCs (could be more than one depending on deployment strategy) it must connect to before publishing ENstatus payloads.

4.4.1.1.4. EN Awareness FC

The EN Awareness FC organizes the information in such a way problematic EN can be found first in the data structure. Because a problematic EN will imply considering all EN of the same ownership for problem contingency we then group together all ENs having the same ownership and within this group we also organize per `ServiceID` in case of a vertical ownership (or vertical-allocated by DEDICAT 6G).

- `EN-context-type`
  - `timestamp: TIMESTAMPtype` – when the context is issued
    - `[EN]` - for each `EN` in the array: organised by ascending flag levels BUT then grouped by `ownership` then `serviceID` (in case of a vertical)
      - `ENid: string`
      - `metadata:`
        - `timestamp: TIMESTAMPtype`
        - `severityFlag: {Optimal, Degrading, Critical}`
        - `ownership: {"D6G"|customerID:string}`
        - `serviceID: string`
        - `fieldDeployed: {TRUE, FALSE}`
        - `coreConfigured: {TRUE, FALSE}` - TRUE meaning that the (by-default) deployment of all needed FCs have been performed
        - `verticalConfigured: {TRUE, FALSE}`
        - `standby: {TRUE, FALSE}`
        - `location: GPStype` - 3D GPS loc
      - `data:`
        - `maxCPUpwr: (int,int)` - total available CPU power (Core/Thread)
        - `maxGPUpwr: int` - total available GPU power (CUDA number)
        - `maxRAM: int` - total amount of available RAM (in Gbyte)
        - `maxGRAM: int` - total amount of available GRAM (in Gbyte)
        - `maxDISK: int` - total available Disk space (in Gbyte)
        - `maxBandwidth: float` - maximum bandwidth (in Mbit/sec)
        - `usedPwr: float` - % used power (w.r.t. maximum available power)
        - `usedCPUpw`r: `float` - % used CPU power (Core/Threads)
        - `usedGPUpwr: float` - % remaining GPU Power (CUDA number)
        - `usedRAM: int` - total amount of used RAM (in Gbyte)
        - `usedGRAM: int` - total amount of used GRAM (in Gbyte)
        - `usedBandwidth: float` - used bandwidth (in Mbit/sec)
        - `usedDISK: float` - % remaining Disk space (in Gbyte)
        - `[EE]`: for each `EE` in the array:
          - `severityFlag: {Optimal, Degrading, Critical}`
          - `usedPwr:float` - % of contributed Power consumption
          - `allocCPUpwr: (int,int)` - allocated CPU power (Core/Thread)

- `allocGPUpwr: int` - allocated GPU power (CUDA nbr)
- `allocRAM: int` - allocated RAM (in Gbyte)
- `allocGRAM: int` - allocated GRAM (in Gbyte)
- `allocDISK: int`- allocated disk Space (in Gbyte)
- `allocBandwidth: float` - allocated bandwidth (in Mbit/sec)
- `usedCPUpwr: float` - total % used CPU power (in Watts)
- `usedGPUpwr: float` - total % used GPU power (CUDA nbr)
- `usedRAM: float` - total % used RAM (in Gbyte)
- `usedGRAM: float` - total % used GRAM (in Gbyte)
- `usedDISK: float` - total % used disk Space (in Gbyte)
- `usedBandwidth: float` - total % used bandwidth (in Mbit/sec)

This public interface (underlined) is provided by the EN Awareness FC (`ENawareness` service instance) and consists of eight methods:

1. `duration:int=`**`ENawareness`**`.connect(ENagentREF:ServiceHandler)`: the connect methods plays to roles 1/ to advertise to the EN Awareness FC its `ENstatus` producers, and 2/ return those producers the sampling rate (`duration`) they should use when publishing `ENstatus` payloads;
2. `status:string=`**`ENawareness`**`.disconnect(ENagentREF:ServiceHandler)`: the counter-part of `connect`;
3. `status:string=`**`ENawareness`**`.subscribe()`: used by any client which wishes to receive EN contexts (e.g. the IDDM FC);
4. `status:string=`**`ENawareness`**`.publish(ENstatus:EN-status-type)`: used by any EN Status Agent FC that wishes to report an `ENstatus` to the EN Awareness FC;
5. `ENcontext:EN-context-type=`**`ENawareness`**`.getENcontext()`: returns the most recent EN context available from the EN Awareness FC;
6. `ENcontext:EN-context-type=`**`ENawareness`**`.getENcontextPerSID (customer-ID:string, serviceID:string)`: returns the most recent EN contexts available from the EN Awareness FC relating to `serviceID` and `customerID`;
7. `ENcontext:EN-context-type=`**`ENawareness`**`.getENcontextPerENid (ENids:[string])`: returns the most recent EN contexts available from the EN Awareness FC relating to `ENids`;
8. `status:string=`**`ENawareness`**`.setSamplingRate(duration:int)`: determines how often a context will be published by the EN Awareness FC. This `duration` (set in seconds) is sent as an answer at the time a EN Status Agent FC client connects to the EN Awareness FC.

4.4.1.1.5. UE Status Agent FC

The EU Status Agent FC captures data that the UEs calculate or collect, when communicating with their surrounding BSs and MAPs. The associated logical data model follows:

- `UE-info-type`

- metadata:
  - UEid: string
  - timestamp: TIMESTAMPtype
- data:
  - severityFlag: {Optimal, Degraded, Critical} - relates to the ability of the UE to find a proper association
  - MAPSigStrength: [(MAPid|BSid), signalStrength, AoA] - signalstrength per MAPid or BS (in dBm) and AoA per MAPid (in Radian)
  - Location: GPStype - location of the UE (calculated and optional):
  - connectivityExtensionType: {5g, 4g, wifi,…} - supported connectivity
  - channels: [(centralFreq, width)] - supported channels (both values in Hz) as an array
  - reqDataRate: float - required QoS as data rate (in Mbit/sec)
  - currentDataRate: float - current QoS as data rate (in Mbit/sec)
  - UEstatus: {idle, connected, inactive}
  - prevDecisions:[[BSid|MAPid]] - previous BS-UE association decisions

The UE Status Agent FC (UEagent service instance) has a single interface that consists of two methods (underlined):

- UEstatus:UE-status-type=**UEagent.**getUEstatus()can be used to do a direct request about the instant status of an UE supervised by the UEagent service instance;
- status:string=**UEagent.**setServerREF (UEawarenessREFs: [ServiceHandler])This method is used to configure the UE Status Agent FC (UEagent) so that it knows which UE Awareness FCs (could be more than one depending on deployment strategy) it must connect to before publishing UEstatus payloads.

4.4.1.1.6. UE Awareness FC

We start with defining the UE context data type:

- UE-context-type:
  - timestamp: TIMESTAMPtype - timestamp of the whole context
  - UEs: [UE-status-info:UE-status-type] – UEs are an array of UE-status-info organised by descending severityFlag

This public interface is provided by the UE Awareness FC and consists of six methods:

1. duration:int=**UEawareness.**connect(UEagentREF:ServiceHandler): the connect methods plays to roles 1/ to advertise to the UE Awareness FC about its actual UE-info producers, and 2/ return those producers the sampling rate (duration) they should use when publishing UEstatus payloads;
2. status:string=**UEawareness.**disconnect(UEagentREF:ServicekHandler): the counter-part of connect;
3. status:string=**UEawareness.**subscribe(): used by any client which wishes to receive UE contexts (e.g. the UE-BS SelectionDM FC);
4. status:string=**UEawareness.**publish(UEstatus:UE-status-type): used by any UE Status Agent FC that wishes to report an UE status to the UE Awareness FC;

5. `UEcontext:UE-context-type=`**`UEawareness`**`.`<u>`getUEcontext`</u>`():` returns the most recent `UEcontext` available from the UE Awareness FC;
6. `Status:string=`**`UEawareness`**`.`<u>`setSamplingRate`</u>`(duration:int):` determines how often a context will be published by the UE Awareness FC. This duration (set in seconds) is sent as an answer every time a UE Status Agent FC client connects to the UE Awareness FC.

4.4.1.1.7. MAP Status Agent FC

- `MAP-status-type`
  - o `metadata:`
    - `timestamp: TIMESTAMPtype`
    - `MAPid: string` - needed to get access to the status of the MAP as an Edge Node
    - `ENid: string` - if relevant, typically bigger drones like octopods carrying more computing capability
    - `connectivityExtensionType: {5g, 4g, mmw, wifi,…}`
    - `location: GPStype` - 3D or 2D GPS coordinates
    - `MAPtype: {UAV, AGV, MGV,…}`
    - `ownership: {"D6G"|customerID:string}`
    - `serviceID: string`
    - `fieldDeployed: {TRUE, FALSE}`
    - `coreConfigured: {TRUE, FALSE}` - all D6G by-default deployment have been performed
    - `standby: {TRUE, FALSE}`
    - `MAPMobilityType: {no_mobility, uncontrollable, partly_controllable, fully_controllable}`
    - `MAPMobilityContraint:`
      - `maxSpeed: float –` (in meter/sec)
      - `maxAcceleration: float –` (in meter/sec^2)
      - `minAltitude: float`
      - `maxAltitude: float`
      - `noFlyZone: {TRUE, FALSE}`
  - o `data:`
    - `severityFlag: {Optimal, Degrading, Critical}`
    - `batteryLevel: int` - in % of full charge
    - `maximumFlyTime: int` - depends on the type of battery used (in minute)
    - `mobilityStatus: {flyingToLoc, docking, hovering}`
    - `commStatus: {TRUE, FALSE}`
    - `maxThroughput: float`
    - `currentTrafficLoad: float` - % of maximum throughput
    - `currentDeploymentCost: float`
    - `MAPlocation: GPStype`
    - Previous MAP operation decisions…

The MAP Status Agent FC (`MAPagent` service instance) has a single interface that consists of two methods (underlined):

- `MAPstatus:MAP-status-type=`**`MAPagent`**`.`<u>`getMAPstatus`</u>`()` can be used to do a direct request about the instant status of the MAP supervised by the `MAPagent` service instance;

- `Status:string=`**`MAPagent`**`.`<u>`setServerREF`</u> `(MAPawarenessREFs: [ServiceHandler])` This method is used to configure the MAP Status Agent FC (`MAPagent`) so that it knows which MAP Awareness FCs (could be more than one depending on deployment strategy) it must connect to before publishing `MAPstatus` payloads.

4.4.1.1.8. MAP Awareness FC

We start with defining the needed data type:

- `MAP-context-type`
    - o `timestamp: DATEtype` – timestamp of the whole context
    - o `maps: [map:MAP-status-type]`

This following public interface is provided by the MAP Awareness FC (`MAPawareness` service instance) and consists of seven methods:

1. `duration:int=`**`MAPawareness`**`.`<u>`connect`</u>`(MAPagentREF:ServiceHandler)`: the connect methods plays to roles 1/ to advertise to the MAP Awareness FC about its actual `MAPstatus` producers, and 2/ return those producers the sampling rate (`duration`) they should use when publishing `MAPstatus` payloads;
2. `status:string=`**`MAPawareness`**`.`<u>`disconnect`</u>`(MAPagentREF:ServiceHandler)`: the counter-part of `connect`;
3. `status:string=`**`MAPawareness`**`.`<u>`subscribe`</u>`()`: used by any client which wishes to receive MAP contexts (e.g. the Swarm Operation FC);
4. `status:string=`**`MAPawareness`**`.`<u>`publish`</u>`(MAPstatus:MAP-status-type)`: used by any MAP Status Agent FC that wishes to report a `MAPstatus` to the MAP Awareness FC;
5. `MAPcontext:MAP-context-type:` **`MAPawareness`**`.`<u>`getMAPcontext`</u>`()`: returns the most recent `MAPcontext` available from the MAP Awareness FC;
6. `MAPcontext:MAP-context-type:` **`MAPawareness`**`.`<u>`getMAPcontextPerSID`</u> `(customerID: string, serviceID:string)`: returns the most recent `MAPcontext` available from the MAP Awareness FC relating to `serviceID` and `customerID`;
7. `Status:string=`**`MAPawareness`**`.`<u>`setSamplingRate`</u>`(duration:int)`: determines how often a context will be published by the MAP Awareness FC. This duration (set in seconds) is sent as an answer every time a MAP Status Agent FC client connects to the MAP Awareness FC.

4.4.1.1.9. NW Status Agent FC

The NW Status Agent FC collects all the necessary information of a given network node, e.g. MAP, gNodeB, BS, and to be sent to the NW Awareness FC to create NW contexts. The corresponding data model associate is as follows:

- `NW-status-type`
    - o `metadata:`
        - ▪ `nodeType: {UE, MAP, BS, 5G routers, …}`
        - ▪ `nodeID: {BSid, ENid, EEid…}`
        - ▪ `ownership: {"D6G"|customerID:string}`
        - ▪ `serviceID: string`
        - ▪ `boardID: string` – [optional]
        - ▪ `timestamp: TIMESTAMPtype`

      - `currentLocation: GPStype`
  - `data:`
      - `nodeStatus: {Optimal, Degraded, Critical}`
      - `activeLinksNbr: int`
      - `current Traffic load: float` - resource utilization in %
      - `maxThroughput: float`
      - `connectivityType` - including channels
      - `links: [link]`– for each `link` in the array:
          - `linkID: string`
          - `linkType: {Radio, IP, Optical}`
          - `currentLinkDataRate: float` - (in Mbit/sec)
          - `totalLinkCapacity: float`
          - `availLinkCapacity: float` - resource utilization in %
          - `usedLinkCapacity: float` - resource utilization in %
          - `linkStatus: {Optimal, Degraded, Critical}`
          - `linkPropagationTime: float` – (in msec)
          - `bidirectional: {TRUE, FALSE}`
          - `originNodeID: string`
          - `destinationNodeID: string`
          - `domainID: string`
          - `channels:[channel]`- for each `channel` in the array:
              - `channelID: string`
              - `channelStatus: {Optimal, Degraded, Critical}`
              - `channelQuality: float`
              - `channelFreq: float (in kHz)`
              - `channelPort: integer`
              - `channelModulation: {QPSK, QAM,…}`
              - `currentChannelDataRate: float` - (in Mbit/sec)
      - `NWperfMetric: float` - e.g. percentage of UEs with satisfied QoS, average network sum rate, cost of the current deployment, computed and received from Network Performance Analytics FC

The NW Status Agent FC (`NWagent` service instance) has a single interface that consists of two methods (underlined):

- `NWstatus:NW-status-type=`**`NWagent.`**`getNWstatus()`. This method can be used to do a direct request about the instant NW status;
- `Status:string=`**`NWagent.`**`setServerREF (NWawarenessREFs: [ServiceHandler])`. This method is used to configure the NW Status Agent FC (`NWagent`) so that it knows which NW Awareness FCs (could be more than one depending on deployment strategy) it must connect to before publishing `NWstatus` payloads.

4.4.1.1.10. NW Awareness FC

The NW Awareness FC builds up contexts that can be used by the NODM FC for its decision-making process.

It is paramount that the NODM maps the states of the network entities in order to build the network context. A network entity status is classified as follows:

- **Optimal:** the network entity (topology/node/link/channel/route) is up and fully available, running as expected;

- **Degrading:** the network entity is up but partially available with some failure present. The optimal network route is not available, but another sub-optimal route can be found in the network context to maintain the network service up and running;
- **Critical:** Total failure, network service cannot be served through the network entity.

Thresholds associated with `critical` and `degrading` need being considered carefully when designing Status Agent and Awareness FCs.

The associated logical data model for this FC is defined as follows:

- `NW-context-type`:
  - `metadata`
    - `timestamp: TIMESTAMPtype`
  - `data:`
    - `topologies: [topology]` - for each `topology` in the array:
      - `topologyID: string`
      - `topologyStatus: {Optimal, Degraded, Critical}`
      - `networkType: {Radio, IP, Optical}`
      - `domainType: {Cloud, Edge, FarEdge, Access}`
      - `domainID: string`
      - `NWslices: [NWslice]` - where each `slice` in the array
        - `NWsliceID: string`
        - `targetQoS: float`
        - `currentQoS: float`
        - `targetLatency: float`
        - `currentLatency: float`
        - `NWsliceStatus: {Optimal, Degraded, Critical}`
      - `nodes: [node]- for each node` in the array:
        - `nodeID: string`
        - `nodeStatus: {Optimal, Degraded, Critical}`
        - `ownership: {"D6G"|customerID:string}`
        - `serviceID: string`
      - `routes: [route]` - for each `route` in the array:
        - `routeID: string`
        - `routeStatus: {Optimal, Degraded, Critical}`
        - `NWsliceID: string`
        - `links:[link]-` where for each `link` in the array
          - `linkID: string`
          - `linkStatus: {Optimal, Degraded, Critical}`
          - `linkPropagationTime: float` - (in msec)
          - `currentLinkDataRate: float` - (Mbit/s)

This public interface is provided by the NW Awareness FC and consists of 7 methods (underlined):

1. `duration:int=`**`NWawareness`**`.`<u>`connect`</u>`(NWagentREF:ServiceHandler)`: the connect methods plays to roles 1/ to advertise to the NW Awareness FC instance (`NWawareness`) about its actual `NWstatus` producers, and 2/ return those producers the sampling rate (`duration`) they should use when publishing `NWstatus` payloads;
2. `status:string=`**`NWawareness`**`.`<u>`disconnect`</u>`(NWagentREF:ServiceHandler)`: the counter-part of `connect`;
3. `status:string=`**`NWawareness`**`.`<u>`subscribe`</u>`()`: used by any client which wishes to receive NW contexts (e.g. the Swarm Operation FC);

4. `status:string=`**`NWawareness`**`.publish(NWstatus:NW-status-type)`: used by any NW Status Agent FC that wishes to report an NW status to the NW Awareness FC;
5. `NWcontext:NW-context-type=`**`NWawareness`**`.getNWcontext()`: returns the most recent NW context available from the NW Awareness FC;
6. `NWcontext:Network-context-type=`**`µS/FCawareness`**`.getNWcontextPerSID (customerID:string, serviceID:string)`: returns the most recent NW context available from the NW Awareness FC relating to `serviceID` and `customerID`;
7. `status=`**`NWawareness`**`.setSamplingRate(duration:int)`: determines how often a context will be published by the NW Awareness FC. This duration (set in seconds) is sent as an answer every time a NW Status Agent FC client connects to the NW Awareness FC.

4.4.1.2 Decision Making FG-related interface and data models

This section focuses on elucidating the decision data structures that result from the IDDM and CEDM decision processes.

The structure and content of those decisions must capture everything needed by the "decision-making" executing parties in order to carry out the decisions issued by the DM components. As a reminder:

- **IDDM decision:** Service Orchestrator FC, Edge Service Orchestrator FC and Load Balancing FC are responsible for implementing the IDDM FC decisions (in that specific order);
- **CEDM decision:** UAV Operation FC, Swarm Operation FC, AGV Operation FC and ConnectedCar Operation FC are implementing the decisions issued by the CEDM FC (depending on the kind of MAPs the Coverage Extension will be using).

Let us recall first what the purpose of the IDDM FC is. Based on 1/ contextual information that describes best the current state of the edge nodes and 2/ the inherent characteristics of, on the one hand, targeted edge nodes and, on the other hand, the tasks (FCs or µS) that need being executed on those target edge nodes- the IDDM FC decides about the best allocation of tasks to edge nodes.

It is worth noting here, that the allocation granularity is at the Edge Node level. Considering that an edge node is potentially made of several distinct execution environments (with a Load Balancing FC managing them), it is important to make some additional information available to the Load Balancing FC (via the Edge Service Orchestrator FCs) in order to ensure that the task arrangement (assigning tasks among the Execution Environments within the selected edge node) can be made by the Edge Service Orchestrator FC based on the actual needs of each µS or FC.

The result of an IDDM FC decision is therefore a mapping from the set of tasks to be executed at the edge and the set of targeted edge nodes. Additional information however needs also to be passed on to the Service Orchestrator FC which is ultimately responsible for deploying the actual µS and FC instances to the edge nodes.

Instances to be assigned can be

- Existing: the existing task (µS or FC) have a unique ID. Two cases can be distinguished:
  - Migration: the existing task instance is moved from an EE_{source} to EE_{target};
  - Cloning a new instance of the existing instance is deployed to a designated EE_{target} (it could be the same EE than for the existing task).
- Non-existing (new, especially when proceeding to the initial deployment following an IDaaS request): new instances of designated tasks need being deployed in an EE.

Now we provide some data structures that will be used when defining the CEDM FC and IDDM FC interfaces.

We therefore start with the data structures relating to the IDaaS and CEaaS requests:

- `CEaaSreq-type:`
    - o `customerID: string`
    - o `customerReqID: string`
    - o `CEreq-info: CEreq-type`
    - o `IDreq-info: IDreq-type`
- `IDaaSreq-type:`
    - o `customerID: string`
    - o `customerReqID: string`
    - o `IDreq-info: IDreq-type`

  where `IDreq-info` and `CEreq-info` are defined hereafter.

We can notice that the `CEaaSreq-type` includes `CEreq-info` but intelligence distribution related information too as well, by default (`IDreq-info`). Indeed, Coverage Extension always involves the deployment of some DEDICAT 6G FCs towards the deployed MAPs even if there are no specific additional needs for ID, in term of µS deployment from the vertical. Those components include most of Status Agents and Awareness FCs, supporting FCs (in case some DEDICAT 6G FCs need being migrated between MAPs) and of course the CE-supporting FCs as well. The deployment of those components follows strictly the DEDICAT 6G deployment policy and happens to be quite rigid. The policy dictates very clearly where which FC has to be deployed. In that very context, the task→PS allocation algorithm is not used, as shown in the related sequence diagrams.

- `CEreq-type`:
    - o `coverageType: {aerial, terrestrial}`
    - o `terrestrialType: {AGV, Robots, ConnectedCar,…}`
    - o `aerialType: {UAV, Drones, AirShips, Statics,…}`
    - o `coverageAreaRadius: int` - (in meter): defines the overall surface to be covered
    - o `location:GPStype` - (of MAP deployment)
    - o `dockingStationLocation: GPStype` - (optional)
    - o `reqCapacity : int`- forecasted number of UEs to be covered
    - o `reqMinThoughput: float` – (in Mbit/sec) per UE
    - o `maxLatency: float`
    - o `maxPowerConsumption: float` - (in Watt)
    - o `connectivityExtensionType: {5g, mmw, 4g, wifi,…}`
    - o `starTime: DATEtype`
    - o `endTime: DATEtype`
- `IDreq-type`: this data structure captures all information needed when requested a IDaaS. This information is used in order to check feasibility out and eventually grant the requested intelligence distribution service to a requesting party (a vertical)
    - o `[targetENinfo]`: provides all information about vertical-provided edge nodes (their types and characteristics, number of instances and IDs) that the IDDM can used for task allocation. As agreed, it will be the Edge Service Orchestrator FC duty to allocate the tasks to the existing EEs within the edge node based on the IDDM decision. We therefore do not need to provide here the list of EEs within one EN as they can be created dynamically at runtime. Each `targetENinfo` in the array `[targetENinfo]` consists of the following EN characteristics which are needed by the task allocation algorithm:

        - `ENclass:string` - EN class name
        - `instanceNbr: int` - nbr of deployed instances of the class for that specific IDaaS request;
        - `ENids: [string]` - the corresponding array of ENids instances in that ENclass (they therefore share the same following characteristics);
        - `maxCPUpower: (int/int)` - Available Core/Thread numbers;
        - `maxGPUpower: int` - nbr of available CUDA cores;
        - `maxRAM: int` - amount of CPU RAM (in GByte);
        - `maxGRAM: int` - amount of GPU RAM (in GByte);
        - `maxDISK: float` - amount of available storage space (in GByte)
        - `maxBandwidth: float` - maximum available bandwidth (in Mbyte/sec)
    - `[targetTaskInfo]`: provides all information about the µSs which need being deployed by the IDDM FC. Each `targetTaskInfo` in the array `[targetTaskInfo]` consists of the following EN characteristics which are needed by the task allocation algorithm:
        - `taskClass: string` – the µS class name
        - `instanceNbr: int` – the number of instances required
        - `reqCPUpwr: (int, int)` - nbr of Cores/Threads needed
        - `reqGPUpwr: int` - nbr of CUDA needed
        - `reqRAM: int` - amount of RAM needed (in Gbyte)
        - `reqRAM: int` - amount of GRAM needed (in Gbyte)
        - `reqDISK: int` - amount of Disk Space needed (in Gbyte)
        - `reqBandwidth: float` - amount of bandwidth needed (in Mbit/sec)
    - `policy-info: Policy-type` – desired deployment policy (defined in Section 4.4.1.5.4)

Here, we go in the detail of the result/outcome of a decision carried out by either the IDDM or CEDM FCs, how it is structured and what sort of information it conveys to the FCs that are responsible for implementing such decisions.

- `CEDM-outcome-type:`
    - `timestamp: TIMESTAMPtype` - when the outcome was released
    - `customerID: string`
    - `customerRequestID: string`
    - `outcome:[(MAPclass, instancesNbr, stdByNbr, deploymentLoc, dockingStationLoc)]` - where:
        - `MAPclass:string` - is a DEDICAT 6G supported MAP whose characteristics are stored by the EN Registry FC
        - `instanceNbr:int` - is the number of MAP instances of the `MAPclass` to be deployed
        - `stdByNbr:int` - is the number of instances among `InstanceNbr` which need being kept in stand-by mode (waiting for activation from the customer)
        - `deploymentLoc:GPStype` - is the location where the instances are physically deployed
        - `dockingStationLoc:GPStype` - represents where the MAP instances can recharge batteries, if applicable. [optional]
- `IDDM-outcome-type:`
    - `timestamp: TIMESTAMPtype`
    - `customerID: string`
    - `customerRequestID: string`

    - `[(ENid:string, [(taskClass:string, instanceNbr:int)])]` - where:
        - `ENid:` is the edge node unique identifier (which can be a MAP)
        - `taskClass:`represents either a µService or Functional Component Class
        - `instanceNbr:` gives the number of instances to be deployed within the physical system.

The `CEDM-outcome-type` and `IDDM-outcome-type` data is used when instrumenting their respective CE- and ID-supporting FCs via their public interfaces. They are not meant to be communicated to those FCs directly.

The CEDM and IDDM FCs do propose a few public interfaces to other FCs and also do require the public interfaces provided by the various Awareness FCs.

This public interface is used either during IDaaS and CEaaS service request, or when instrumentation is required between two DM components, e.g. when the NODM FC instruments the CEDM FC or when the CEDM FC instruments the IDDM FC, as follows:

Those two methods (underlined) are provided respectively by the CEaaS and IDaaS for the Dashboard FC (which is the web front-end for vertical service requests) to use:

- `status:string=`**`CEDM.`**`CEaaSreq(customerID: string, CEaaSreq-info: CE-aaSreq-type)` where the request status returned is `{("ok",serviceID:int), "NOK", (serviceID:int, IDreq-infoCP: IDreq-type)}`. In the first case the service is granted "as-requested" and a `serviceID` is returned to the customer "for-the-record". In the second case the service request is declined. In the third case a counter proposal is sent back to the requester alongside a `serviceID`. This counter proposal which is a modified version of the initial request (some parts being scaled up, like e.g. number of EN instances) can be accepted (in that case the `serviceID` is passed on) or rejected by the customer; both cases terminating the service request phase;
- `status:string=`**`IDDM.`**`IDaaSreq(customerID: string, IDaaSreq-info: IDaaSreq-type)`. This method follows the same logic than the previous one.

This method (underlined) is implemented by the CEDM FC for the NODM FC to use:

- `status:string=`**`CEDM.`**`informCoverageFailure ([context])` where a `context` can be of either sort of context type (NWcontext, UEcontext,...) relevant to the observed radio coverage issue (either `critical` or `degraded`) observed by the NODM FC.

These three methods (underlined) are implemented by the IDDM FC for the CEDM FC to use:

- `status:string=`**`IDDM.`**`IDreqCheck (customerID:string, serviceID: string, IDreq-info: IDreq-type)` allows the CEDM to request the IDDM to check on an `IDreq-info` to essentially validate that the ENs a propertly dimentioned to that the requested µS (and the needed FCs involved in all ID action) can be executed in compliance with their run-time requirements. Status is either `"OK"`, `"NOK"` or `IDreq-infoCP` which is a *Counter Proposal (CP)* which is a modified `IRreq-info` (typically playing with the EN instance numbers);
- `status:string=`**`IDDM.`**`MAPconfig(maps:[mapID])` this method is called by the CEDM when it instruments the IDDM to configure the MAP. By configuration, we mean here the deployment into the MAPs `[mapID]`of all necessary FCs as defined in the DEDICAT 6G EC policy;
- `status:string=`**`IDDM.`**`IDreqPerf(serviceID: string, customerID: string, IDreq-info: IDreq-type)`instruments the IDDM to perform task allocation for `ID-`

`req-info`, including requesting the Service Orchestrator FC to implement the task allocation plan.

Another example is when the SLA Registry FC enforces SLA against ID or CE-related service execution and discovers an SLA breach. In that case, it needs 1/ to inform the IDDM FC or CEDM FC about which (`serviceId, customerID`) fails and 2/ to push the contexts where the SLA breach occurs.

As far as SLA enforcement is concerned, it is important to entrust it in the hand of a separate and independent components instead of assuming that the CEDM FC or IDDM FC could be in charge. Doing this way ensures that there is a clear separation between service provision on one side and service SLA enforcement on the other side.

Those two SLA enforcement-related methods (underlined) are implemented by respectively the IDDM FC and the CEDM FC in order to be informed of an SLA breach by the SLA registry FC:

- `status:string=`**`IDDM`**`.informSLAbreach(serviceID: string, customerID: string, contexts: [context], logID: string)` where a `context` in an array containing either sort of contexts relevant to the reported breach and where `logID` refers to the log number received from the Logging FC component when when the SLA Registry FC logged the SLA breach;
- `status:string=`**`CEDM`**`.informSLAbreach(serviceID: string, customerID: string, contexts:[context], logID: string)` where a `context` an array containing either sorts of contexts relevant to the reported breach and where `logID` refers to the log number received from the Logging FC component when the SLA Registry FC logged the SLA breach.

It is of the utmost importance for the CEDM and IDDM FCs to log back the results of their actions (alongside the original SLA breach `logID`) when attempting to correct an SLA breach, so that both the identified issues and corrections can be logged.

These four methods (underlined) are offered by the CEDM FC and IDDM FC for the Dashboard FC to use. That way the Dashboard FC can be aware of the completion of the various steps necessary to complete a CEaaS or IDaaS service request, and be informed about service request confirmation or rejection by the customer:

- `status=`**`CEDM`**`.CEaaSreqConfirm(serviceID: string)`
- `status=`**`CEDM`**`.CEaaSreqDeny(serviceID: string)`
- `status=`**`CEDM`**`.inform (status:service-status-type)`
- `status=`**`IDDM`**`.IDaaSreqConfirm(serviceID: string)`
- `status=`**`IDDM`**`.IDaaSreqDeny(serviceID: string)`
- `status=`**`IDDM`**`.inform (status:service-status-type)`

where `service-status-type` is defined as follows:

```
•   service-status-type:
       o  customerID: string
       o  serviceID: string
       o  status:{"µS-image-uploaded", "µS-profile-added",
                 "policy-added", "EN-profile-added",
                 "MAP-configured", "ID-completed",
                 "MAP-deployed"}
```

4.4.1.3 Coverage Extension FG-related interface and data models

4.4.1.3.1. Swarm Operation FC

In this section, we go through the data models and interface provided by the Swarm Operation FC to the CEDM FC. We remind that the CEDM FC carries out four different phases, 1/ taking a CE decision, 2/ informing the technical crew about needed physical deployment (this is done via the dashboard), 3/ configuring the MAPs as soon as their deployment has been completed and service delivery start time has been reached, 4/ initiating the service itself.

It is worth noting that some parts of the configuration non-directly relating to the MAP operation have been carried out beforehand, including registering the MAP and its capabilities within the EN Registry FC and configuring how to access Awareness FC contexts, to name just a few.

Before going in the detail of the Swarm Operation FC interface, we elucidate a data structure which is used during the MAP configuration step, after the MAPs have been physically deployed.

`UAV-config-type:`

- `MAPid: string`
- `ownership: {"D6G"|customerID:string}`
- `serviceID: string` – if allocated to `customerID` otherwise `""`
- `swarmID: string` - an ID that identifies the MAP swarm as a whole
- `membersID: [string]` - the IDs of MAPs that are member of the swarm;
- `location: GPStype` - the location of the deployment as advertised in the CEaaS initial request which is to be considered as the center of the coverage extension perimeter;
- `radius: int` - used to calculate the perimeter;
- `connectivityConfig: (connectivityExtensionType, config)`

where `connectivityExtensionType` is `{5g, mmw, 4g, wifi,…}` and where `config` is a technical configuration data structure that relates to the `connectivityExtensionType`.

The following three methods (underlined) are used by the CEDM FC in order to operate a swarm of UAVs. Because the Swarm Operation FC (`swarmOp`) supports autonomous UAV behaviour, most of the flying aspects are left to the initiative of the individual UAV based on information they gather from other UAVs via the MAP Awareness FC.

- `status=`**`swarmOp`**`.config(UAV-config-info:UAV-config-type)` is used to set-up the UAVs before the coverage extension service starts;
- `status=`**`swarmOp`**`.start()` is used to start the service at the time defined by the initial CEaaS service request. Has to be invoked for all `swarpOp` class deployed in the UAV part of the swarm;
- `status=`**`swarmOp`**`.stop()` is used to terminate the service at the time defined by the initial CEaaS service request. Has to be invoked for all `swarpOp` class deployed in the UAV part of the swarm;

A final important remark is that the Swarm Operation FC can also be deployed within a non-autonomous MAP (or manned-AP), such as a connected car. In that case it won't be able to regularly move and adjust position like an UAV would do, but the UAVs will be able to manage/optimize their positions taking into account the coverage supplied by the manned-AP.

4.4.1.3.2. ConnectedCar Operation FC

ConnectedCars are manned and do not need the same degree of sophistication that UAV requires (despite they can be member of a UAV swarm as we have emphasized above).

However, in order to avoid having humans involved in the technical set-up and operation of the communication capabilities of the connected car (meaning here the car as MAP) we implement similar interface to those we described in the UAV-related section above.

Before going in the detail of the ConnectedCar Operation FC interface, we elucidate a data structure which is used during the MAP configuration step, after it has been physically deployed.

`car-config-info:`

- `MAPid: string`
- `connectivityConfig: (connectivityExtensionType, config)`

where `connectivityExtensionType` is `{5g, mmw, 4g, wifi, SAT, …}` and where `config` is a technical configuration data structure that relates to the `connectivityExtensionType`.

The public interface of the ConnectedCar Operation FC (`connectedCarOp`) is made of the three following methods (underlined). They are used by the CEDM FC in order to configure and operate remotely the AP capability of a connected car.

- `status=`**`connectedCarOp`**`.`<u>`config`</u>`(car-config-info)` is used to set-up the connected car (its MAP) before the coverage extension service starts.
- `status=`**`connectedCarOp`**`.`<u>`start`</u>`()` is used to start the service at the time defined by the initial CEaaS service request.
- `status=`**`connectedCarOp`**`.`<u>`stop`</u>`()` is used to terminate the service at the time defined by the initial CEaaS service request.

4.4.1.3.3. AGV Operation FC

The AGV Operation FC provides an interface that allows to access and pilot the basic actions an AGV (e.g., a robot) can perform. In order to extend the AGV basic actions and allow complex behaviors, the so-called “AGV capabilities” are introduced (please refer to UC1 “Smart Warehousing” and UML diagrams for more detail). They are µServices that are deployed within the AGV (which is therefore also an edge node) and that provide high-end features exploiting those atomic actions provided by the AGV Operation FC.

We won’t provide an exhaustive AGV Operation FC interface as it highly depends on the AGV type. However, we give now a few examples of such primitive behaviors an AGV such as a robot, ought to provide, without going into the detail of the parameters needing to be passed on:

- Identifying an object/trolley by sensing a beacon;
- Identifying an object/trolley by reading a QR-code (camera);
- Identifying an object/trolley by recognizing a shape from a list of pre-determine shapes (camera + ML);
- Facing a direction;
- Facing an on-board camera in a specific direction;
- Streaming a video feed from an on-board camera;
- Grasping an object;
- Lifting a trolley;
- Dropping an object;
- Determining own location;

- Moving to a location;
- Stopping.

Implicit AGV behavior also includes obstacle/collision avoidance by default. So, when roaming from point A to point B an AGV will have to determine a path between A and B that avoids all potential obstacles in its vicinity.

We will provide more detail about the interface in the next release of the document as some technical aspects are still being discussed in WP6.

4.4.1.4 Service Operation FG-related interface and data models

In this section we consider only the Service Orchestrator FC as the two others, namely Edge Service Orchestrator and Load Balancing FCs will be very much depending on the chosen technologies. The Service Orchestrator FC duty is then to make the interface between the invoking FC, i.e., the IDDM FC and the heavily tech-dependent Edge Orchestrator and Load Balancing FC one the one hand and the NFV orchestrator legacy 5G component on the other hand.

It has also to report the outcomes of those two components so that their actions can be reflected in the relevant registries, e.g., the EN Registry FC and µS/FC Registry FC. This includes in particular, creating new µS or FC instances (with their IDs), updating their deployment locations (in case they were moved by the Load Balancing FC), etc.

The main method provided by the Service Orchestrator FC allows to apply a new task allocation plan as issued by the IDDM FC as part of its `IDDM-outcome-type`. It is worth noting that when the task allocation plan is the update of an existing allocation plan the task allocation plan passed on to the Service Orchestrator only concerns the delta between 1/ what is already deployed as part of the initial plan and, 2/ what is impacted by the new deployment plan. This typically happens when 1/ the SLA detects that one or several FCs or µSs are not executed as required (insufficient resource allocation by the EE e.g.) 2/ the IDDM itself- based on EN and µS/FC contexts – decides to re-calculate a task deployment plan.

4.4.1.5 Intelligence Distribution FG-related interface and data models

After dealing with some CE-supporting FCs we elucidate the interface provided to IDDM FC by the Service Orchestrator FC. We won't explain the interface of Load Balancing FC and Edge Orchestrator FC because they are very much depending on the technology used and vary a lot between one technology and another.

The Service Orchestrator FC is an intermediate component that receives a deployment request from the IDDM FC which covers a whole µS and FC deployment. It will be the responsibility of the Edge Service Orchestrator FC to adjust the actual exiting deployment according to the targeted global deployment. It means that, as a result of this adjustment, some already existing µSs or FCs might be terminated, duplicated or migrated, while some new µS and FC instances would be created.

4.4.1.5.1. µS/FC Repository FC

As we have explained already in the Functional view section and more precisely in Section 4.3.2.2, this component is mostly a database storing container/VM images, and which can be queried in order to retrieve the `image` corresponding to a specific `classID` for deployment by the Edge Service Orchestrator FC.

Its interface is relatively straightfoward. It consists of three methods (underlined), which do not need specific data structure:

- `status:string=`**`µS/FCrepository`**`.upload(classID: string, image: Image-type)` is used to upload a docker image and associate it with a µS or FC `classID`.
- `Status:string=`**`µS/FCrepository`**`.discard(classID: string)` is used to discard an image associated to `classID`
- `imageHandler:ServiceREF=`**`µS/FCrepository`**`.getImage(classID: string)` is used to retrieve an image corresponding to `classID`

Some usage examples of this interface can be found in the CEaaS and IDaaS sequence diagrams in respectively Figure 23 and Figure 24.

4.4.1.5.2. µS/FC Registry FC

Unlike the previous interface, the µS/FC Registry FC requires the definition of a task class profile (as task is used to refer to either µS or FC). A profile includes mostly requirements about the computing and communication resources a task requires in order to perform with optimal performance. Each instance of the class shares the same resource requirements.

`Task-class-profile-type:`

- `metadata:`
  - `classID: string` - class of the µS or FC
  - `ownership: {"D6G"|customerID:string}` - whom it belongs to
- `data:`
  - `reqCPUpwr: (int,int)` - required number of Cores/Threads
  - `reqGPUpwr: int` - required number of CUDA
  - `reqRAM: int` - required amount of RAM (in Gbyte)
  - `reqGRAM: int` - required amount of GRAM (in Gbyte)
  - `reqDISK: int` - required amount of disk space (in Gbyte)
  - `reqBandwidth: float` - required amount of bandwidth (in Mbyte/sec)
  - `maxPwr: float` - maximum energy consumption (in Watt)
  - `instances = [instance]` - where each `instance` in the array consists of the following:
    - `instanceID: string` - ID of the deployed instance of class `classID`
    - `serviceID: string` – which `serviceID` is the instance allocated to (if any)
    - `taskREF: ServiceHandler`
    - `ENid: string` - in which edge node the instance is deployed
    - `EEid: string` - in which EE the instance is deployed

Following this data structure definition, the interface of µS/FC Registry must allow for adding, updating and removing FC and µS class descriptions, and querying the description that corresponds to a `classID`. In addition to those basic methods, we also need to create a new instance or to discard/update an existing one.

We have therefore the eight following methods (underlined):

- `task-class-profile: Task-class-profile-type =` **`µS/FCregistry`**`.getProfile (classID: string)` is used to retrieve the description of the task class and instances according to a `classID;`
- `taskREF:ServiceHandler=`**`µS/FCregistry`**`.getHandler(taskID:string;`

- `status:string=`**`µS/FCregistry`**`.addProfiles(classIDs, [task-class-profile])` is used to create a set of new task classes (with no existing instances then) with ID `classIDs` and description `[task-class-profile];`
- `status:string =` **`µS/FCregistry`**`.updateProfiles(classIDs: [string], task-class-profiles: [Task-class-profile-type])` is used to do a bulk update of classes `classIDs;`
- `status:string=`**`µS/FCregistry`**`.discardProfiles(classIDs: [string])` is used to discard a set of classes `[classID];`
- `status:string=`**`µS/FCregistry`**`.addInstances(classIDs: [string], instanceIDs: [string], ENids: [string])` is used to add a new set of class instances `instanceIDs` to the `classIDs;`
- `status:string=`**`µS/FCregistry`**`.updateInstances (classIDs: [string], instanceIDs: [string], ENids: [string])` is used to update a set of class instances `instanceIDs` from the `classIDs` profiles with new deployment `ENids;`
- `status:string=`**`µS/FCregistry`**`.discardInstances (classIDs: [string], instanceIDs: [string])` is used to discard a set of task class instances `instanceIDs` from the `classIDs` profiles.

4.4.1.5.3. EN Registry FC

Similarly to the previous component, we go first through the data model used to characterize an edge node and then define the interface. Like for µS/FC class profile, edge node profiles relate to a typical type of edge node. So, when considering using one or several different class(es) of edge node it is important to make clear how many instances of the same kind are deployed. Therefore, the edge node class profiles include also the list of actual deployed instances.

Typically for an EN belonging to the platform, it is also important to know where they are deployed and to which `customerID/serviceID` they are allocated to.

`EN-class-profile-type:`

- `metadata:`
  - `classID: string` - class of the edge node
- `data:`
  - `maxPwr: float` - maximum available peak power (in Watt)
  - `maxCPUpwr: (int,int)` - maximum available CPU power (Core/Thread)
  - `maxGPUpwr: int` - maximum available GPU power (CUDA number)
  - `maxRAM: int` - maximum available RAM in Gbytes
  - `maxDISK: int` - maximum available disk space (in Gbyte)
  - `maxBandwidth: float` - maximum bandwidth (in Mbyte/sec)
  - `instances: [instance]` - where each class `instance` in the array is:
    - `ENid: string`
    - `ownership: {"D6G"|customerID:string}`
    - `servicedID:string`
    - `fieldDeployed: {TRUE, FALSE}`
    - `coreConfigured: {TRUE, FALSE}` - all D6G by-default deployment have been performed
    - `verticalConfigured: {TRUE, FALSE}`
    - `standby: {TRUE, FALSE}`

- `location: GPStype` - 3D GPS location (altitude may be ignored if not relevant)

Following this data structure definition, the interface of EN Registry must allow for adding, updating and removing EN class profiles (including their instances), and querying the profile that corresponds to a `classID`. In addition to those basic methods, we also need to add a new instance or to discard/update an existing one.

We have therefore the six following methods (underlined):

- `EN-profiles: [EN-class-profile-type] = ENregistry.getENprofile(classID: [string])` is used to retrieve the profiles of the EN classes and instances according to an array of `[classID];`
- `status=ENregistry.addProfile(classIDs: [string], EN-class-profiles: [EN-class-profile-type])` is used to create one or several new EN class(es) and its instances with ID `classID` and profile `EN-class-profile;`
- `status=ENregistry.discardProfile(classIDs: [string])` is used to discard one or several EN class(es) `classID` profiles;
- `status=ENregistry.addInstance(classID: string, instanceID: string, ENid: string)` is used to add a new class instance `instanceID` to the `classID;`
- `status=ENregistry.updateInstance (classID: string, instanceID: string, ENid: string)` is used to update a class instance `instanceID` from the `classID` profile with a new EN deployment `ENid` of that class;
- `status=ENregistry.discardInstance (classID: string, instanceID: string)` is used to discard an EN instance `instanceID` from the `classID` profile.

4.4.1.5.4. EC Policy Registry FC

The purpose of an *Edge Computing (EC)* Policy is to provide deployment constraints or preferences to either 1/ a vertical, for the deployment of its µSs or 2/ the DEDICAT 6G platform itself for the deployment of the FCs. This policy is also used to declare dependencies between FCs and between µSs. One single vertical can be the originator of several EC Policies, namely one per CEaaS/IDaaS subscribed service. It can also have no policy attached to a service request, meaning it would entirely rely on the IDDM FC for the deployment/distribution of its µSs among available edge nodes.

A single FC or µS can be deployed in different ways:

- one per **EE**: this is typically the case for µS/FC Status Agent FC, which when deployed towards an EN, must in fact be deployed in each of the EEs, resulting in several instances of the same service being eventually deployed;
- one per **EN**: this is typically the case of the NW Status Agent FC;
- one for a group of **ENs**: this applies to the Edge Service Orchestrator whose actions can span one or several edge nodes;
- **CLOUD** deployment: This option is particularly useful when the deployment of the platform components is initiated by the platform management plane.

This taxonomy introduces the following data structure:

```
Deployment-strategy-type: {EE, EN, ENS, CLOUD}
```

Then a deployment consists of the following structure:

```
Deployment-type: [(taskClass:string,[(
                       targetEN: string,
                       instanceNbr: int,
                       strategy: Deployment-strategy-type
                       )]
```

In order to deal with service dependencies, we need to introduce the following data structures:

`Dependency-type: [Dependency-instance-type]` – one per observed dependency

`Dependency-deploymentType: {EE, EN}` - where EE (reps. EN) means same EE (same EN) than the depending entity.

```
Dependency-instance-type:  [sourceTaskClass: string,[(
                                 instanceNbr: string,
                                 targetTaskClass: string,
                                 depDeployType: Dependency-deployment-type
                                 )]]
```

Where `sourceTaskClass` is the ID of the task class (depending entity) that needs the different instances `instanceNbr` of the `targetTaskClass` to be deployed along <u>one `sourceTaskClass` instance</u> following the `Dependency-deployment-type`.

Let's take as an example the scenario of a robot that needs being deployed some capabilities. If we consider one `Cap#2 µS` than needs always 1 `Cap#3 µS` and `2 Cap#7 µSs` instances to be deployed in the same EN than the `Cap#2 µS` instance, we would have:

`[“Cap#2”,[(“Cap#3”,1,EN),(“Cap#7”,2,EN)]]`

However, if one of the two `Cap#7` instances needs being in the same EE than `Cap#3 µS`, we would have:

`[“Cap#2”,[(“Cap#3”,1,EN),(“Cap#7”,1,EN),(“Cap#7”,1,EE)]]`

The overall EC policy type becomes then:

`Policy-type:`

- `policyOwner: string` – can be `“D6G”` or any `customerID`
- `serviceID: string` – applies only if the `policyOwner` is `customerID`, meaning a vertical
- `deployments: Deployment-type`
- `dependencies: Dependency-type`

The interface of the EC Policy Registry FC consists of the following three methods (underlined):

- `status=`**`ECpolicyRegistry`**`.`<u>`addPolicy`</u>`(owner: string, serviceID: string, policy: Policy-type)`
- `status=`**`ECpolicyRegistry`**`.`<u>`discardPolicy`</u>`(owner: string, serviceID: string)`
- `status=`**`ECpolicyRegistry`**`.`<u>`updatePolicy`</u>`(owner: string, serviceID: string, policy: Policy-type)`

4.4.1.5.5. SLA Registry FC

The SLA data model contains all information that characterizes the service agreement between a customer (a vertical) and the DEDICAT 6G platform. In order to make sure that a

CEaaS or IDaaS service is supplied to the customer according to its formal service agreement, the SLA needs being enforced. As a result, the decision-making FCs (IDDM FC and CEDM FC) will have to take corrective actions, should an SLA breach occur.

The SLA registry FC stores two different kinds of information, some relating to Coverage Extension and some to Intelligence Distribution. Not-surprisingly, some of the SLA-related information is part of the CEaaS and IDaaS service requests as described in Section 4.4.1.2.

Part of SLA enforcement includes also checking out that the MAPs which are deployed for coverage extension purpose, correspond to the categories of MAPs (including their number of instances) being mentioned when establishing the CEaaS contract.

`SLA-type:`

- `metadata`
  - `customerID: string`
  - `serviceID: string`
  - `startTime: DATEtype`
  - `stopTime: DATEtype`
- `data:`
  - `SLA-CE-type:`
    - `reqCapacity: int`
    - `reqMinThoughput: float`
    - `maxLatency: float`
    - `maxPowerConsumption: float`
    - `MAPs: [MAP: MAP-deployment-type] –` the list of MAP classes and their number of instances that DEDICAT 6G pledges to deploy in order to fulfill the QoS parameters defined above
  - `SLA-ID-type:`
    - `tasks: [taskID: Task-class-profile-type] –` list of µSs whose resource consumption at run-time need being monitored against their requirements. Their characteristics can be accessed from the µS/FC Registry FC. They include run-time resources, power consumption, networking-related requirements
    - `ENs: [EN: EN-class-profile-type]` – list of ENs and their characteristics, that are being provided by the Vertical to host their µSs after IDDM allocation

where `Task-class-profile-type` and `Task-class-profile-type` are defined respectively in the µS/FC Registry FC and EN Registry FC previous Sections 4.4.1.5.2 and 4.4.1.5.3 respectively, and where `MAP-deployment-type` is defined as follows:

`MAP-deployment-type:`

- `MAPclass: string`
- `MAPtype: {UAV, AGV, MGV,…} –` as respectively drones, robots and connected car/van
- `MAPinstances: [(MAPid: string, ENid: string)]` – the list of instances in that class to be deployed by DEDICAT 6G
- `connectivityExtensionType: {5g, 4g, mmw, wifi,…}`
- `maxThroughput: float`
- `MAPmobilityType: {no_mobility, uncontrollable, partly_controllable, fully_controllable}`
- `MAPmobilityContraint:`
  - `maxSpeed: float –` (in meter/sec)

- `maxAcceleration: float` - (in meter/sec^2)
- `minAltitude: float`
- `maxAltitude: float`
- `maximumFlyTime: int` - depends on the type of battery used (in minute)

Assuming SLA purpose is all about enforcing the supply of a service -IDaaS and CEaaS- against a customer's requirements, `SLA-type` may be consisting of `SLA-CE-type` or `SLA-ID-type` or both. Indeed, requesting a CEaaS always involves some ID parts but in some cases, they could be only concerned with platform FCs (FCs, DU), i.e., when the customer does not expect the platform to provide any intelligent distribution of µSs (meaning a pure radio coverage extension is requested and nothing else).

The SLA Registry FC provides interface for adding, updating and deleting an SLA corresponding to a `serviceID` (which is a globally unique ID). It also allows to retrieve the SLA associated with such a `serviceID`.

Two more methods are used to start and stop enforcing the SLA corresponding to a `serviceID` and two additional methods in order to notify the SLA Registry FC that SLA breaches have been resolved (to be used by the IDDM and CEDM FC).

The interface therefore consists of eight methods (underlined) as follows:

- `status=`**`SLAregistry`**`.addSLA(SLA-info: SLA-type)` allows to upload an `SLA-info: SLA-type` to the SLA Registry FC;
- `status=`**`SLAregistry`**`.updateSLA(serviceID: string, SLA-info: SLA-type)` allows to update an `SLA-info: SLA-type` corresponding to an existing `serviceID` to the SLA Registry FC;
- `status=SLAregistry.discardSLA(serviceID: string)` allows to discard an `SLA-info` corresponding to an existing `serviceID;`
- `SLA-info=`**`SLAregistry`**`.retrieveSLA(serviceID: string)` allows to retrieve an `SLA-info` corresponding to an existing `serviceID;`
- `status=`**`SLAregistry`**`.startEnforcement(customerID: string, serviceID: string)` used to start enforcement of a specific ID or CE (or both) attached to a `serviceID;`
- `status=`**`SLAregistry`**`.stopEnforcement(customerID:string, serviceID:string)` used to stop enforcement of a specific ID or CE (or both) attached to a `serviceID;`
- `status=`**`SLAregistry`**`.informIDresolved (customerID: string, serviceID: string, logID: string)` used to inform that a computing related breach has been resolved
- `status=`**`SLAregistry`**`.informCEresolved (customerID: string, serviceID: string, logID: string)` used to inform that a coverage extension related breach has been resolved

4.4.1.6 Analytics FG-related interfaces and data models

4.4.1.6.1. Network Performance Analytics FC

This first analytic component is principally used by the NW Awareness FC in order to report network performance indicators that can be used when optimizing the spatial placement of MAPs (e.g., a fleet of drones) as illustrated in Section 4.3.4.5.

This component subscribes to UE and MAP contexts and upon request from the NW Awareness FC releases a performance indicator.

The interface (underlined) consists of one single method defined as follows:

- `NWperfIndic=NWperfAnal.getPerf (UEset: [string], MAPset: [string])`

Where `UEset` and `MAPset` are respectively the set of UEs potentially able to have an association with MAP(s) inside `MAPset` (receiving signal from), and the actual set of MAPs/APs forming the swarm.

4.4.1.7 Security Privacy Trust-related interfaces and data models

4.4.1.7.1. AuthN FC

The UML described in D2.3 refers to the following information:

- `User-registration-type:`
  - `username: string`
  - `credential: string`
  - `role: enum`
  - `email address: EmailADDRtype`
  - `affiliation: {string|customerID}`

The username needs to be unique, and the password needs to meet the minimal complexity required. If the username is taken, the conflict code will be returned. If data is malformed, a bad request code will be returned with the error messages. Success returns created entity code and the location of a newly created user. Two data structures are used for AuthN FC, one relating to the user registration and one to the authentication process:

- `AuthN-payload-type:`
  - `username: string`
  - `credential: string`
- `AuthN-response-type:`
  - `status: {"denied", "granted"} –` if “granted” the two following fied do contain tockens. They are left empty otherwise
  - `authToken: string`
  - `refreshToken: string`

the interface of AuthN FC consists of two methods (underlined):

- `status=AuthN.register(user-registration-info: user-registration-type)`
- `authN-response:AuthN-response-type=AuthN.authenticate(username: string, passwd: string)`

4.4.1.7.2. AuthZ FC

The authorization data models include:

- `AuthZ-request-type:`
  - `subject: string` – e.g. customer id
  - `object: string` – e.g. a resource URL
  - `action: {read, write, list, add, remove,…}`

- `AuthZ-fetch-policy:`
  - `client_id: string` - a client identifier

- `AuthZ-policies-type:`
  - `policies: [policy-object]`

- `policy-object:`

- `id: string` – e.g. policy unique identifier
    - `version: string` - e.g. policy version
    - `name: string` – an arbitrary name
    - `owner: string` - an if of the owner
    - `rules: [object]` - where each object in the list is defined as:
        - `action: {read, write, list, add, remove,…}`
        - `type: string` - resource type e.g. "users", "assets"
        - `condition: key-value map`
- `AuthZ-assign-policy:`
    - `client_id: string` - a client identifier
    - `policy_id: string` - a policy identifier
- `AuthZ-response-type: {granted, denied}`
- `AuthZ-policies-reponse: {error}`
- `Authentication-token-type:`
    - `authN-tocken: string`
    - `authN-refresh-token: string`

For authorization requests, results are "permission granted" or "permission forbidden"; or in an error that has the same effect as forbidden action.

The subject is usually a result of authentication - the ID of the client who performs the action. The object is usually an ID of the resource against which action is executed. Action is an enumeration representing an activity such as "read", "write", "list", "add", and "remove", etc. In the case of the "list" action, the object field can be empty.

Fetching the policy requires a client ID to which policies are assigned. The policy object consists of a unique ID, name, owner id (the client who created it), version number, and a list of rules. Rules are used for the evaluation of the given action against a given object type to check if the subject matches the condition. The condition describes which subject attributes (identified by keys) need to match which values (values of the map). For example, the condition {"role": "admin"} means that only the client with an attribute "role" with a value "admin" will be able to pass the condition. Policies can be assigned to and unassigned from the client.

The AuthZ interface (`authZ`) consists of the following seven methods (underlined):

- `response:AuthZ-response-type=`**`authZ`**`.`<u>`authorize`</u>`(payload: AuthZ-request-type)`
- `response:AuthZ-policies-type=`**`authZ`**`.`<u>`fetch`</u>`(payload: AuthZ-fetch-policy)`
- `response:AuthZ-response-type=`**`authZ`**`.`<u>`update`</u>`(payload: AuthZ-policies-type)`
- `response:AuthZ-policies-response=`**`authZ`**`.`<u>`assign`</u>`(payload: AuthZ-assign-policy)`
- `response:AuthZ-policies-response=`**`authZ`**`.`<u>`unasign`</u>`(payload: AuthZ-assign-policy)`
- `response:Authentication-type=`**`authZ`**`.`<u>`getToken`</u>`(userID: string)`
- `response:AuthZ-policies-response=`**`authZ`**`.`<u>`newUser`</u>`(userId: string)`

4.4.1.7.3. Logging FC

The Logging FC purpose is to log important interactions for the sake of non-repudiation and auditing. This component connects to a blockchain and therefore interactions are stored in a ledger in a completely secure and controlled way.

The typical example we want to log is the contract between a vertical and the platform which is encoded as an SLA. This is a critical feature to provide in case of a dispute occurring between the two parties whenever a Vertical would claim that it did not receive the service quality it paid for.

Equally important in the event of dispute, is the ability for the IDDM FC and CEDM FC to log the result of their actions following an SLA breach.

We provide here a simple interface that can be used by any FC or µS willing to log a data structure (e.g., a JSON `payload`, which we considered encoded as a `string` in this interface).

The interface of the Logging FC consists then of one single method (underlined):

- `logNbr:string=`**`logging`**`.log(originator:string,relatedLogNbrs:[string], payload:string)`

4.4.1.7.4. Requesting a data-tracking database

Additional data which should be added when requesting IDaaS or CEaaS. This information should include:

- The name of database (can be just the CustomerID+RequestID which are part already of the ID-req-info and CE-req-info);
- The ID to be tracked down by the DMP, that relates to the data to be read from the special queue (see the new FC in the Communication FG) and stored in the database. Typically, this ID is the "topic" when using the MQTT technology.

When the IDaaS or CEaaS request is granted, part of the response sent by the relevant FC should include a handler to the database (typically an URL with the `<name of DB>` as a suffix.

4.4.2 System Data flows (cloud/edge)

4.4.2.1 Status Agent FC → Awareness FC flows

Status Agent FCs are deployed to the Edge in the various EEs (hence within physical systems which are also Edge Nodes (as categorized in Table 11) as not all PSs are meant to be used by DEDICAT 6G for the sake of Intelligence Distribution. However, the Awareness FCs are deployed in the cloud.

Figure 30 elucidates on the one hand the flow of data from the µS/FC Status Agent and EN Status Agent FCs towards their respective Awareness FCs and Load Balancing FCs and on the other hand, the flow of data flowing from the Awareness FCs to the DM components.

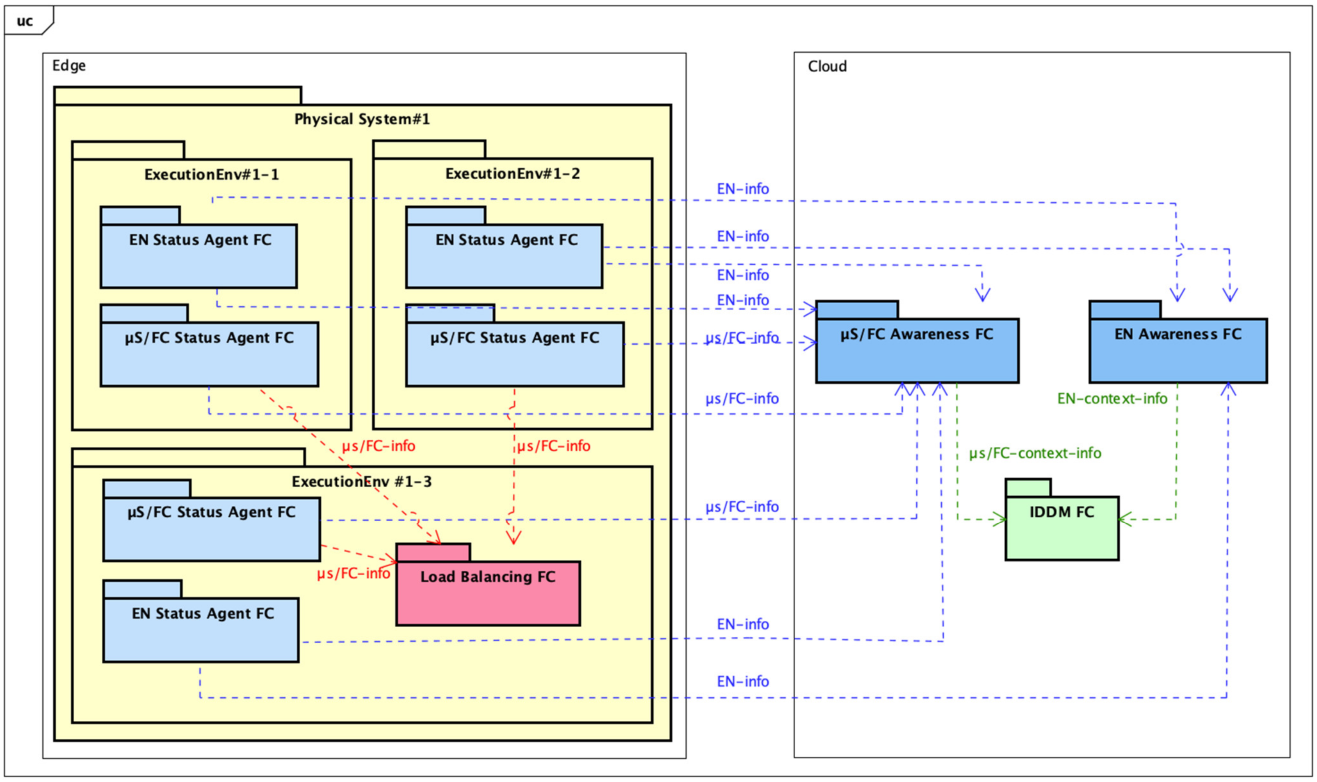


Figure 30: Dataflows between µS/FC- and EN-related FCs and xyDM FCs

The other dataflows pertaining to the UE, MAP, NW Status Agents FC and Awareness FCs are more conventional and look similar, as shown in Figure 31 below.

It's important to note that if by default the IDDM is the recipient of µS/FC and EN Awareness FCs contexts, the NODM Is supervising all network aspects. It's part of the NODM FC duties to comprehend the receive contexts and decide if they ultimately are CEDM matters (for Coverage Extension) or IDDM matters (for intelligence distribution).

This separation of duties between the three main DM components is meant to avoid interferences in term of decisions. However, the contexts are meant to be publicly available as the communication mechanisms between 1/ Status Agent FCs and Awareness FCs and 2/ the Awareness FCs and xyDM FCs is based on message queues. Therefore, any xyDM FC subscribing to the proper queues/topics will definitely access the contexts.

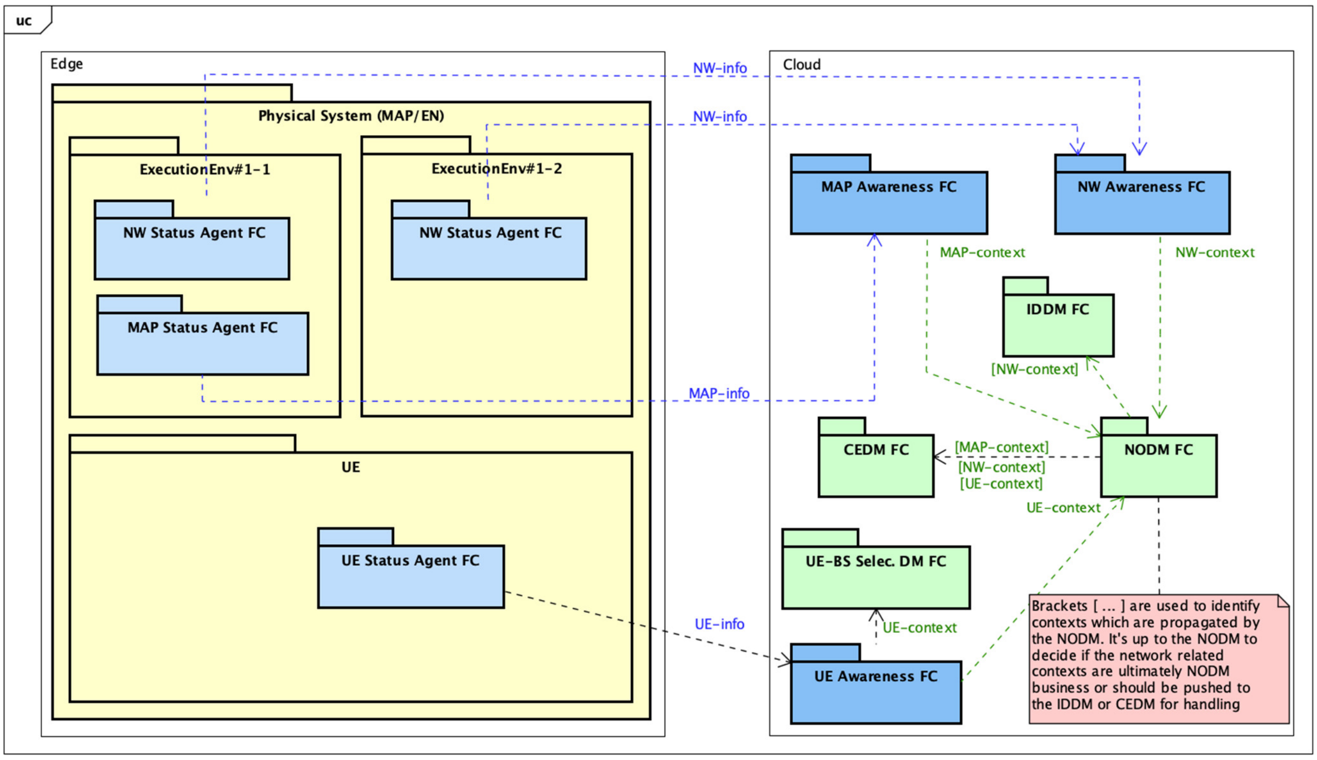


Figure 31: Other Status Agent/Awareness/xyDM dataflows

4.5 Network Deployment View

The *Network Deployment view (NDV)* leverages the Functional view providing a great focus on static and dynamic aspects of intelligence distribution and coverage extension. While the FV gives a high-level logical description of the FCs and their interactions in various contexts, the NDV will elucidate the positioning of the DEDICAT 6G platform in the broader context of an existing 5G legacy network. Therefore, the DEDICAT 6G NDV also includes the 5G network architecture for the following reasons:

1. Many interactions (sometimes bi-directional) have to take place between the DEDICAT 6G and the supporting legacy 5G;
2. Some of the 5G Core or RAN components do have to be deployed towards the far edge inside DEDICAT 6G-specific mobile access points and/or Mobile edge Computing nodes, depending on the considered scenarios;
3. DEDICAT 6G extends the capabilities of the existing 5G network therefore it seems natural to show both of them in a single view, complementing then a rather IT-flavored Functional View where the deployment issues and 5G intrinsic architecture were left aside.

In this second iteration of the Network Deployment View, we provide a revision of the existing baseline (or “generic”) NDV and also four news NDVs that leverage the generic view in the context of the four project Use Cases.

4.5.1 Generic Network Deployment View

In this first iteration of the NDV (see Figure 32 below), we put the light on providing a generic overview of a potential network deployment highlighting the interactions between the DEDICAT 6G platform and a legacy 5G network infrastructure when applying the DEDICAT 6G solution, specially focused on dynamic coverage extension and/or intelligence distribution/migration operations.

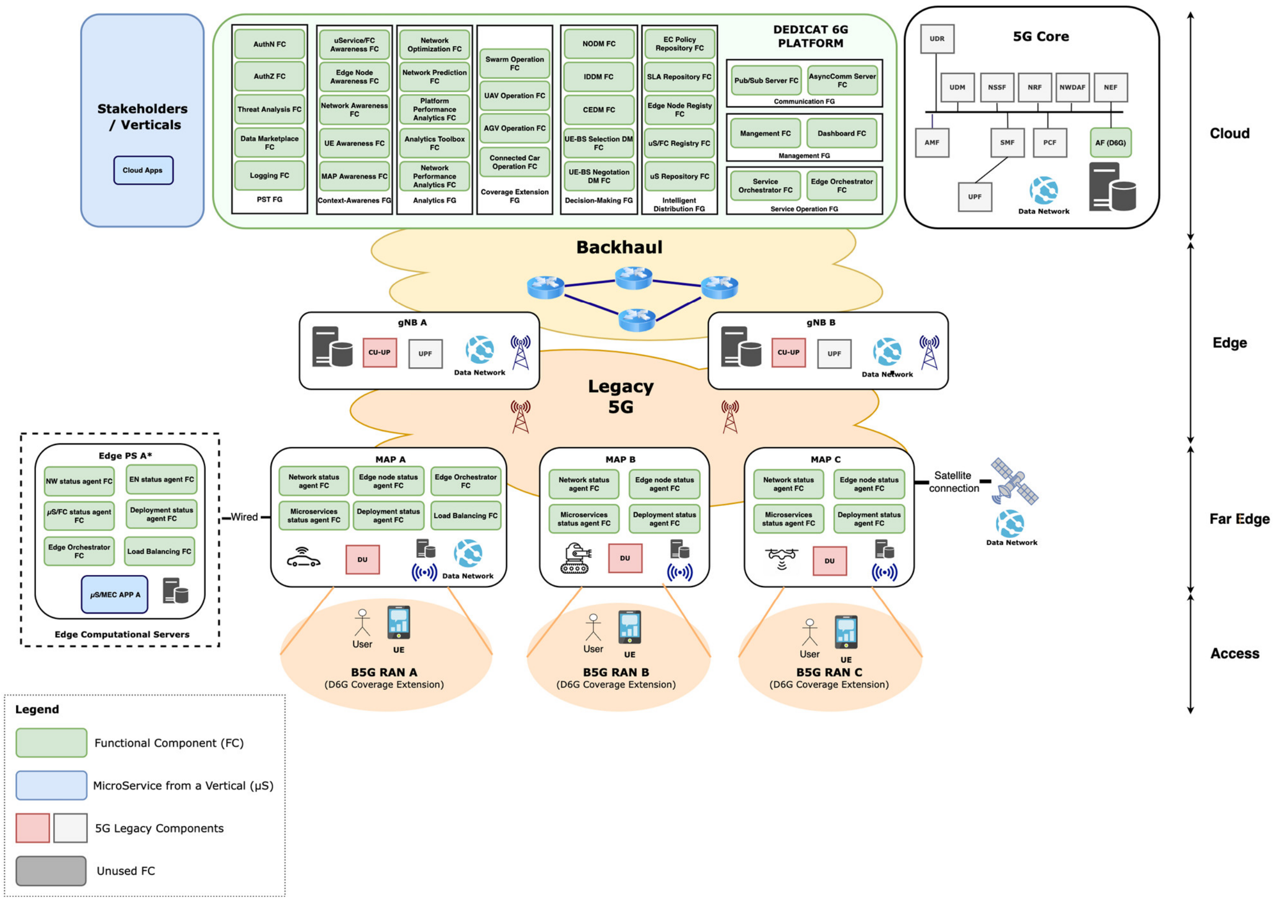


Figure 32: Generic simplified DEDICAT 6G network deployment schema

Figure 32 above shows a diagram of the simplified and baseline deployment where the DEDICAT 6G platform supports a 5G/B5G ecosystem in accommodating a vertical service. Just for the sake of simplicity and in order to provide a clear picture of the environment, we assume a hierarchical network scenario organized in four main domains:

- **Access:** the closest part to the final users (UEs) of the network where network access is provided. In Figure 32 above this domain is represented by the 5G orange cloud icon on the left. It encompasses all the typical pre-deployed technologies of the 5G legacy infrastructure to enable the access of user's traffic (e.g., fixed wireline/wireless or mobile RAN);
- **Far Edge:** ordered by proximity to the user, the next network domain is the *Far Edge*, where nodes are close enough to guarantee minimum latency. Nodes belonging to this domain have very limited computational resources, but still enough to host tiny or small virtual pieces of software aimed at providing high performance with the lowest latency. In our scenario, the far edge nodes will have mobility capabilities, e.g., robots and drones, with embedded equipment meant to transmit data and be part of or interact with the RAN system. In addition, nodes can be used as MEC hosts to deploy vertical µServices (MEC apps) in small VMs or containers. Finally, their computational resources can be utilized by the DEDICAT 6G platform for deploying some agents that could perform specific tasks of some key FCs;
- **Edge:** the *Edge* domain is composed of a set of nodes and links still close to the end-users but with larger resources compared to the far edge entities. From a hierarchical point of view, it is the link between the far edge domains and the core cloud. Moreo-

ver, from our deployment view, edge nodes can host, not only the same functionalities and agents as far edge nodes - if required, but also other FCs agents or instances needed to manage and control some of the functionalities available in the far edge nodes, e.g., swarm operation or connected car operation FCs. In addition, depending on the vertical requirements (embodies in SLAs) and the nature of its implementation, in this domain we can find the first potential entry point to the data network or private network facilities, i.e., using 3GPP terminology, 5G *Non-Public Networks* (5G-NPN). If so, the 5GC system architecture will require instantiating the UPF to enable *End-to-End* (E2E) 5G connectivity;

- **Core Cloud:** the last domain in this network hierarchy is the *Core Cloud*, a set of remote servers, mainly devoted, in this context, to first, accommodate the control plane functions of the 5GC (e.g., NEF, AMF, SMF, PCF or UDR, among others). Typically, the core cloud comprises the connection to the core/backbone networks, that can provide access to the data network or the Internet. In those cases, the 5GC system demands to place here the UPF.

It is noteworthy that the DEDICAT 6G platform will be remotely located in the cloud, and able to reach out to all nodes and UEs within its deployment. This position is essential to provide a global picture of the scenario from above to support the whole system. At all nodes involved in the establishment of the vertical service from the far edge to the cloud core, the DEDICAT 6G platform will instantiate specific agents to monitor the network status and retrieve the necessary metrics required by its corresponding FCs.

As complementary assumptions of the NDV we consider that all the nodes from the far edge to the core cloud will be part of the *Network Function Virtualization Infrastructure (NFVI)*, thus facilitating the establishment of network slices and network services instantiation. Regarding the ETSI NFV [23] architecture, the NFVI is orchestrated by the *NFV Orchestrator* (NFV-O), which is placed at the control plane part of the network. It is responsible to perform NFV *Management and Network Orchestration* (MANO) functions, basic in a 5G-based scenario. The DEDICAT 6G platform will assist the NFV-O on the instantiation of network services and network slices, according to the outcomes coming from the NODM FC and executed by the Orchestration FC.

In order to complement the description of the NDV, here we expose an exemplary generic use-case where a vertical requests the deployment of its service. To simplify the use-case and maximize the understanding, we will assume that the vertical requirements can be satisfied by establishing only one network slice.

As depicted in Figure 32, we assume in this example that a UE has moved from a coverage-covered area to another one. This change is monitored by the system and notified to the DEDICAT 6G platform. At this point, just for the sake of simplicity, we could assume two procedures or a mix of them that can be triggered to handle it: to extend or move the current coverage or migrate the needed intelligence:

- **Coverage extension**: the new UE location and other relevant information, extracted by the Context-Awareness FG, is used as input by the Coverage Extension DM FC that produces the correct configuration and path of the far edge mobile node (drone, AGV or robot). Next, the corresponding FC of the Coverage Extension FG (Swarm Operation FC or Connected Car Operation FC) will react according to the output of the CEDM FC. Finally, the far edge node moves to the new coordinates, changing the coverage area and re-enabling the UE connectivity. If proceeding, the network slice must be updated and reconfigured accordingly.

- **Migrate Intelligence**: when coverage extension is not possible to be used as a solution or the UE enters the coverage area of another node, migrating intelligence may be a feasible option to reconnect UE connectivity. If so, the FCs in the Context-Awareness FG (thanks to the agents deployed at the nodes and other information sources as NEF) will provide the necessary information to the Intelligence Distribution DM FC to calculate the optimal location of the intelligence. As can be seen in Figure 32, the migration of DEDICAT 6G and vertical µServices (MEC apps) from the original node to a new one. This also has some impact on traffic steering and, in turn, on the establishment of the network slice. Then, with the help of the Orchestration FC and the Network Optimization FC, a new path is established in the network that traverses and the new DU, CU and UPF, if applicable, to enable E2E connectivity in the network slice, always fulfilling the SLA and vertical requirements.

Please note that the description and level of detail of the NDV can be much more extensive, however, for the purpose of this document, it is out of its scope. Further information will be provided in later documents and, specially, in WP6 outcomes.

4.5.2 "Smart warehousing" (UC1) Network Deployment View

The Figure 33 below provides a depiction of the UC1 "Smart Warehousing" Network Deployment view emphasizing both networking and computing aspects.

UC1 is implemented in a warehouse which has full broadband fixed and WIFI networking capability; therefore, UC1 does not rely on DEDICAT 6G for dynamic Coverage extension. However, it relies on our DEDICAT 6G platform for dynamic Intelligence Distribution. Consequently, it can be seen in the top part of the figure, that within the decision-making FG only IDDM FC is used.

The IDDM is used at first for instrumenting the initial deployment of all FCs and µSs (according to respectively the DEDICAT6G and UC1 deployment policies) following the IDaaS service request from the Vertical to the DEDICAT 6G platform.

Then at run-time, it will instrument the Edge Service Orchestrator FC (via the Service Orchestrator FC) for µSs and FCs migration or just for creating new instances of existing µSs and FCs. It is typically the case for the AGV capabilities which we can see deployed in AGV#1 and AGV#2 at the bottom of the figure. The IDDM FC decision at run-time is based on contexts that report for example degraded services (AGV capabilities performance below expected QoS) or overloaded servers/edge nodes (SmartAccess360 e.g.).

Communication-wise, AGVs and other edge nodes are 5G, WIFI and Bluetooth enabled. 5G is primarily used to communicate with the DEDICAT 6G entities while WIFI / BlueTooth are used for other purposes including proximity detection and turnstile access control (BLE beacons).

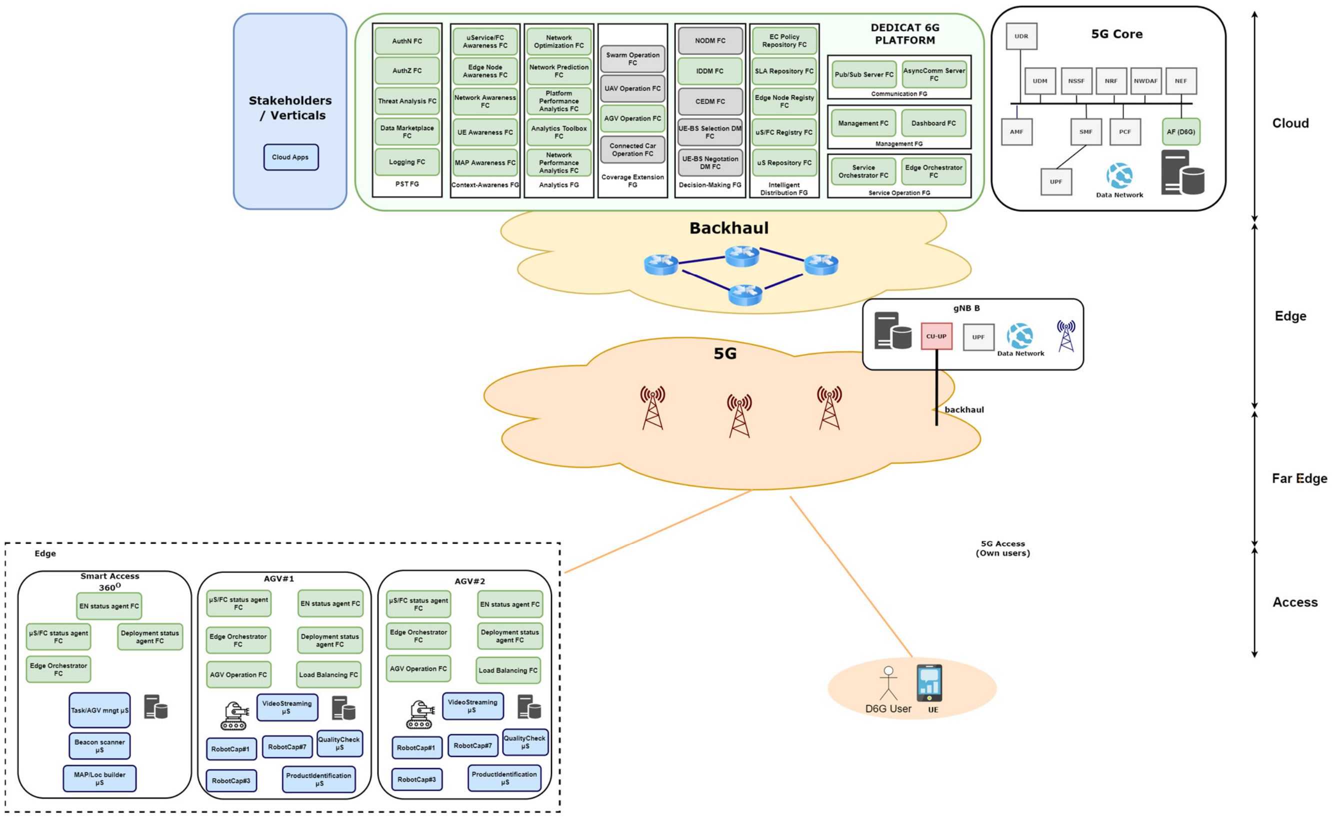


Figure 33: UC1 Deployment View

4.5.3 "Enhanced Experience" (UC2) Network Deployment View

The following Figure 34 shows the UC2 Network Deployment view, as a customization of the generic Deployment View elucidated in Section 4.5.1.

At the top of that figure we can see the list of all of the FCs for UC2 deployed in the cloud (in green) and in the lower part of the figure (Far Edge) we can also notice some FCs (Edge Orchestrator and Load Balancing FCs in addition to a plethora of agents FCs) which are deployed to the UC2 servers (denoted ConnectedCar, A#1, B#1, B#2, B#3) at the time the CEaaS service request is set-up by the DEDICAT 6G platform. The allocation shown here is just an example as it is the duty of the IDDM FC to assign and instrument the FCs and µSs deployment to the Edge servers based on servers' characteristics and µS/FCs run-time requirements.

UC2 identifies multiple servers which have different characteristics depending on their dedicated tasks to be performed. Server A is connected directly to the MAP (a ConnectedCar supplied by DEDICAT 6G as result of CEaaS service request)) for acting as the video service platform for serving the UEs a.k.a. video clients and for receiving the content stream from the dedicated clients (e.g., from Smart Glasses users). As the Server A interconnects with the MAP utilizing the Edge Service Orchestrator and Load Balancing FCs, the video can be directed to other Edge PS servers for performing the video transcoding identified as the heaviest computational work to be done in this use case. These servers can be connected directly to the MAP via wired network, or they can be located further on the edge of other gNB, but still wire-connected.

The 5G core part with gNBs includes also EPC core functionalities using the LTE anchor base stations, which is typical in 5G NSA infrastructure and needed for executing the mobile multicast (FeMBMS) feature in implementation point of view beyond Release-14. The multicast server (Server B#3) needs to interconnect with the core part as well as for the e/gNBs. The content feed for the multicast stream originates from the VideoStreaming p/f.

The MAP (which acts like a IAB Node) is connected to the gNB (its IAB donor) respectively through their respective DUs. The gNB A connects to the gNB B (via their respective DU and CU) for backhauling.

It is important to mention that the IAB Donor has the ability to serve its own additional UEs (not DEDICAT 6G related then), in addition to providing a backhaul link to the MAP.

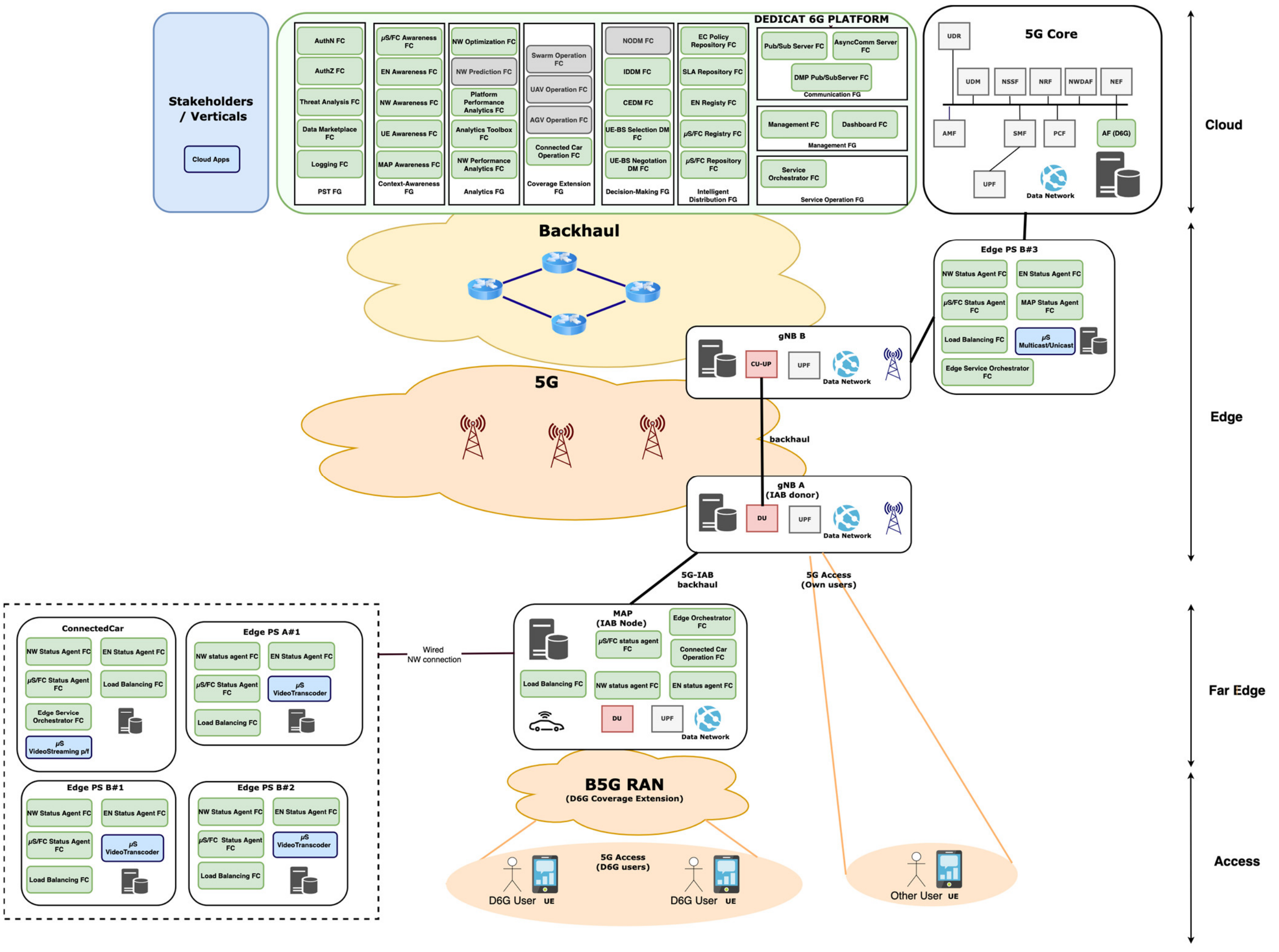


Figure 34: UC2 Deployment View

4.5.4 "Public Safety" (UC3) Network Deployment View

Figure 35 and Figure 36 show the Network Deployment view of UC3 "Public Safety" for both contexts:

- Connectivity loss after a natural disaster;
- Connectivity limitation or failure during a large event.

As a customization of the generic Deployment View elucidated in Section 4.5.1.

The figure is divided in 4 planes:

- Cloud;

- Edge;
- Far Edge;
- Access.

At the Cloud plane is a list of FCs deployed in the DEDICAT 6G cloud in which the one available to deploy *Mission Critical Services (MCS)*. At the Far Edge plane there are the FCs which support the Access plane with the main FCs for Edge Service Orchestration and Load Balancing FCs. Based on the deployment strategy and the collection of data from Agent, the DEDICAT 6G platform deploys the MCS FCs to the Far Edge plane to support local MCS communications.

The IDDM FC assigns and instruments the FCs and µSs deployment to the Edge servers based on servers' characteristics and µS/FCs run-time requirements. Those servers are connected to the MAP (a ConnectedCar) using wired network.

The MCS FCs are deployed at the MAP and accessible through the Access plane by users to be able to communicate during crisis management. Based on Notification Agent FC, the orchestration will be able to deliver to Edge or Far Edge the MCS FCs needed to support the communication features for users (audio or video…).

4.5.4.1 Description of deployment view for context #1

In the context #1 (Figure 35 below), PPDR users are facing a failure in connectivity and need to rely on DEDICAT 6G platform for their critical communications. When a failure of the normal connectivity is detected, the system applies mechanisms to make Mission Critical Services available at the Edge and allows deployment at Far Edge plane. This distribution is applied based on configuration operated by the operator at the PPDR organization.

On the field (at the Far Edge plane), PPDR users will be able to communicate with the connectivity distributed by the MAP and access to the Edge servers to use specific applications during crisis management.

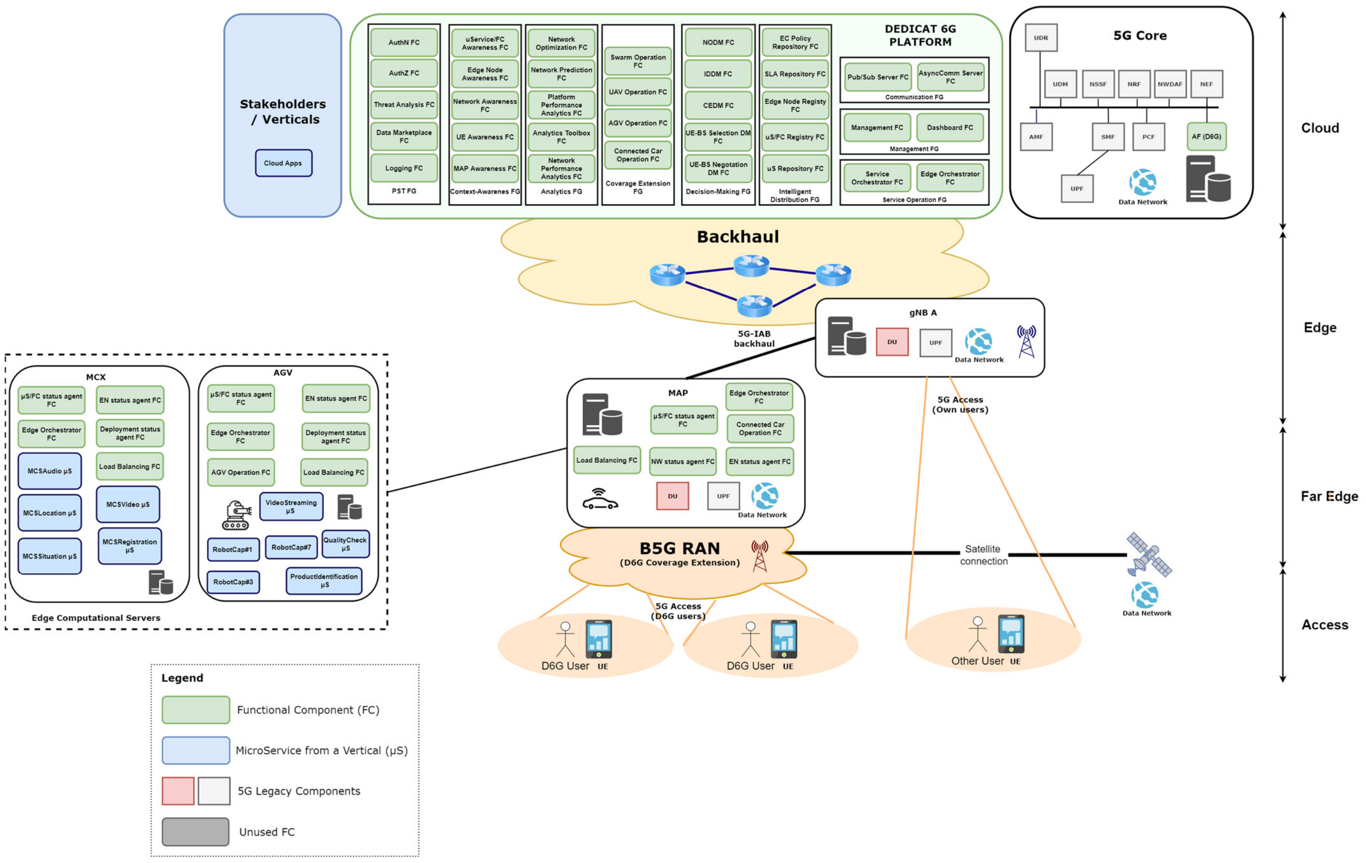


Figure 35: UC3 Context #1 Deployment View

4.5.4.2 Description of deployment view for context #2

In the context #2 (Figure 36), the DEDICAT 6G platform deploy several MAPs to support the lack in connectivity resources for Private Security users and a dedicated MAP for First Responders in case of connectivity failure during a crisis. Mission Critical Services are available at the Edge and allows deployment at Far Edge plane. On the field (at the Far Edge plane), First Responders and Private Security users will be able to communicate with the connectivity distributed by the MAP and access to the Edge servers to use specific applications during crisis management.

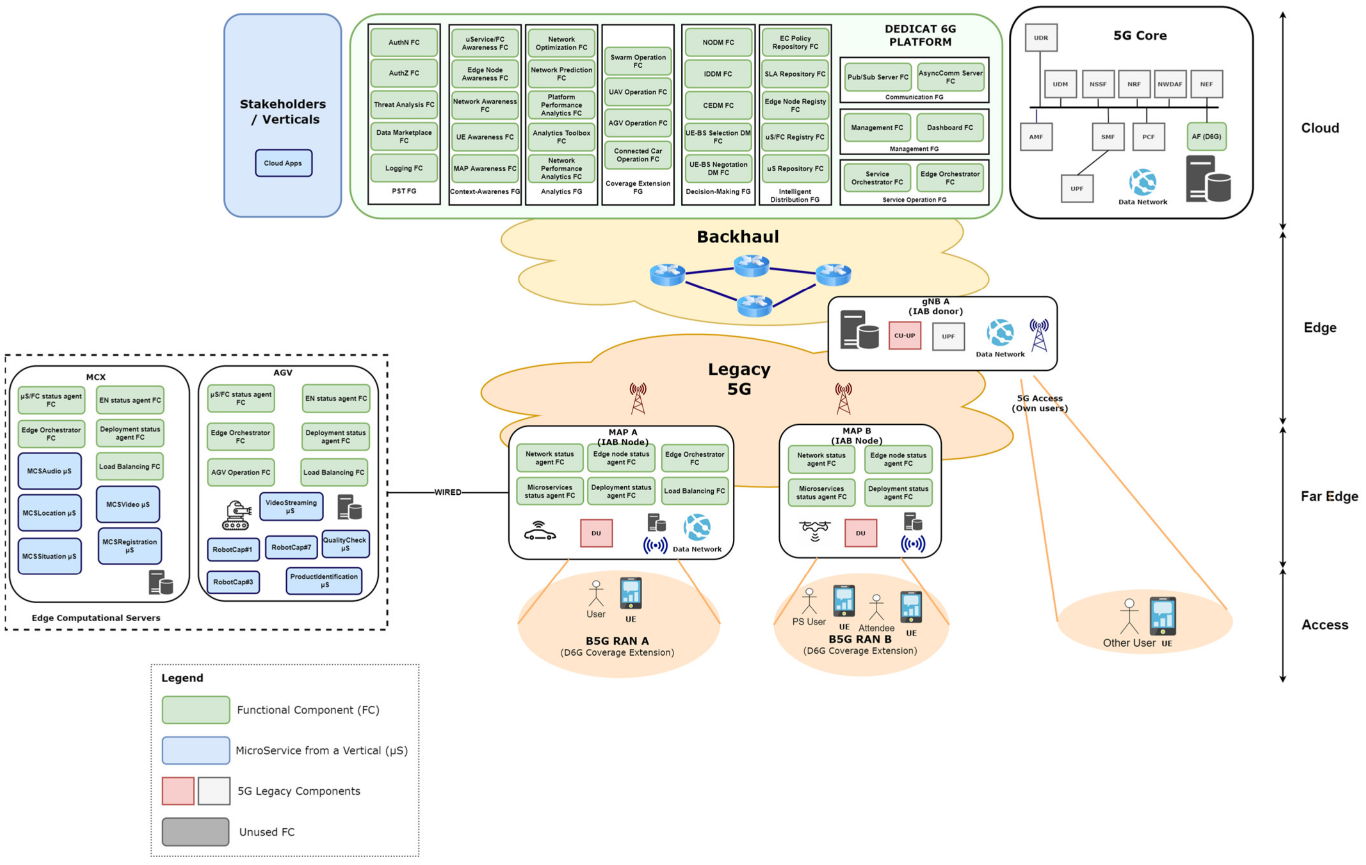


Figure 36: UC3 Context #2 Deployment View

4.5.5 "Smart Highway" (UC4) Network Deployment View

In this section, we present the Network Deployment view (see Figure 37 below) for the Smart Highway use case. The network deployment is represented by the merge of the general DEDICAT 6G network architecture with the actual network available for the UC4. The UC4 network is composed by three layers: the Far Edge, the Edge, and a Cloud Infrastructure. Furthermore, the transport network is composed by fiber link between the Cloud and the Edge, and a wireless link between the Far Edge infrastructure and the backhaul network. This wireless link is composed by ITS-G5 and 5G mobile network.

The Far Edge is represented in the figure by the combination of the OBU Roof Unit, the OBU Car Unit and the MAP. The Edge is composed of the RSU and Sensing Nodes, and the Cloud is composed of the DEDICAT 6G platform and another cluster for the deployment of Vehicular Applications. Each layer of the UC4 network architecture will support the following FC described in deliverable D2.3. The Far Edge and the Edge domains will support monitoring and Intelligence Distribution functions such as the EN Status Agent FC, the μS/FC Status Agent FC, and the Load Balancing FC. Furthermore, both domains will be able to host the vehicular application (V2X Application μS). The Far Edge domain (the OBU) will host as well, the NW Optimization FC and the NW Status Agent FC. The Edge domain will host the Sensing Node μS. In the Cloud domain, some centralized components will be placed. The IDDM FC for example, that is responsible for the decision-making of the intelligence distribution will be hosted in this layer. Other FCs that will be hosted in the Cloud domain also includes the μS/FC Awareness FC, Dashboard FC, EN Awareness FC, Service Orchestrator FC, NODM FC and the CEDM FC. Moreover, the Local Dynamic Map application will also be hosted in a different cluster than the DEDICAT 6G platform.

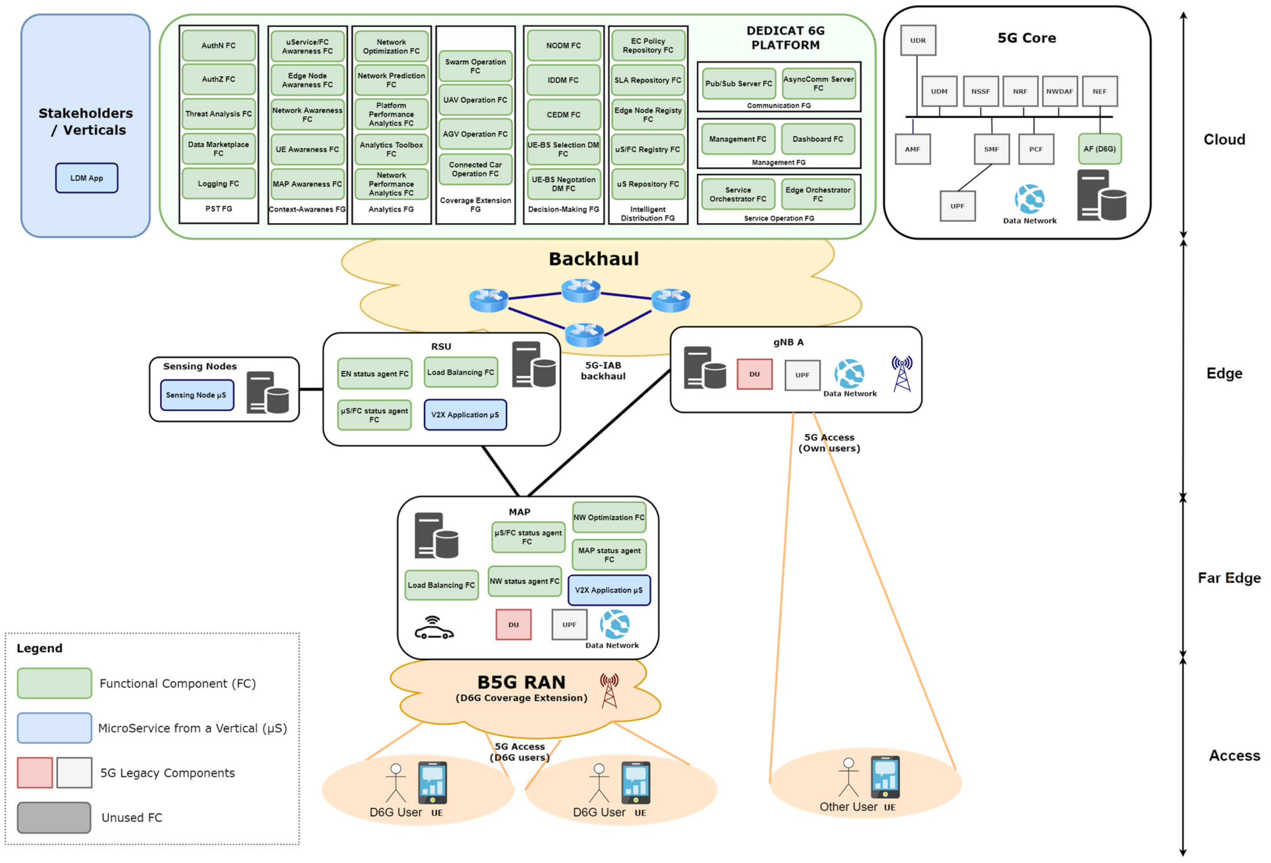


Figure 37: UC4 Deployment View

5 DEDICAT 6G Architecture Perspectives

Architecture perspectives are used in the NFREQ pathway as explained in the overall architecture Methodology (see in Section 2).

They aim to analyse the different NFREQs, which by essence are not focusing on functionalities but rather on system properties, by elucidating activities and tactics that can help reaching such system properties. The result of this analysis is the identification of an additional set of needed functional components that aim to complement the set of FCs resulting from the FREQ analysis. In this section we address the three Privacy, Security and Trust perspectives, followed by the Performance perspective.

5.1 Privacy Perspective

The project will implement mechanisms and tools for ensuring privacy in the scope of applications supported over B5G/6G networks. Such mechanisms and tools will include functionality for controlling data flow in terms of privacy and confidentiality.

Threat detection mechanisms will be based on federated learning boosting privacy protection by moving ML model training process to the source of data instead of transferring data to a centralized entity. Implement comprehensive policies, procedures and protocols for handling personal data. The following Table 15 summarizes how the privacy perspective is dealt with.

Table 15: Privacy perspective survey

Targeted System Quality	Full (100%) protection of obtained personal data information Private data cannot be used in malicious manner
Requirement(s)	• Event logging and auditing • Data anonymisation; • Usability; • Privacy by design. FREQ-3, FREQ-6 to FREQ-11, NFREQ-1 to NFREQ-3, NFREQ-51 to NFREQ-59
Activities	• Setup logging procedure; • Generate logs; • Check log completeness; • Check personal data at source; • Check data when stored in database; • Hide system information from the user; • Perform Audit.
Tactics	• Logging functionality: the logs will allow investigating a system malfunction. It will be possible to know which data/functionality has been abused. Controlling the quality of log data by analyzing and adding missing information to the logs; • Audit functionality: This functionality is responsible for analysing the logs in an offline manner in order to create security and management surveys and reports (which could include threats like intrusion detection attempts or platform malfunction); • Implementing algorithm for creating and storing permanent records of events that can be reviewed and checked • Plan and carry out thorough testing of the integration algorithms, analyzing their performance and the results produced; • Implement comprehensive policies, procedures and protocols for handling personal data.

Applying the tactics above, results in the following list of Design Choices (Table 16 below)

Table 16: Design choices for the Privacy perspective

Design Choice ID	View	FG/FC	Technical description
PRIV-01	Functional	Security FG / {Logging FC, Audit FC}	Introducing system logging functionality
PRIV-02	Functional	Security FG / {Logging FC, Audit FC}	Implementing algorithm for checking log completeness
PRIV-03	Functional	Security FG / Data marketplace FC	Implementing system algorithm for comparison data
PRIV-04	Functional	Security FG / AuthZ FC	Adopting policies and protocols for personal data

More detail about this privacy perspective will be given in the next iteration of this deliverables (D2.4) based on results obtained from WP5 work (and D5.1 in particular).

5.2 Security Perspective

Security is composed of confidentiality, integrity or absence of unauthorized system alterations, and availability for authorized actions only and protecting against eavesdropping and replay attacks.

The following Table 17 summarizes how the security perspective is deal with.

Table 17: Security perspective survey

Targeted System Quality	All data inside the system or its part will be protected against cyber attacks. Full (100%) protection of unauthorized internal and external accesses.
Requirement(s)	• Data encryption; • Auditing; • Data protection; • Minimal performance and scalability; • Compliance; • Zero trust security framework. FREQ-1 to FREQ-20, FREQ-22, NFREQ-71 to 74
Activities	• Perform a threat assessment; • Protection of intelligence distribution and local resources; • Verification activity; • Confirming system compliance with security standards and policies; • Un-trust all devices, networks, and users.
Tactics	• Introducing security requirements; • Implementing decision-making security algorithms; • Introducing intelligent auditing mechanism; • Adopt mechanism required to authorize and authenticate system components and users; • Implement algorithm for data confidentiality;

	• Zero trust mandates that information security pros treat all network traffic as untrusted. This way, network is more efficient, more compliant and more cost effective; • Event logging and perform audit

The project will implement security and data protection framework based on existing industry standards and will specify and implement federated learning mechanisms for training ML models capable of detecting and classifying security threats and data protection risks

Applying the tactics above, results in the following list of Design Choices (Table 18 below).

Table 18: Design choices for the Security perspective

Design Choice ID	View	FG/FC	Technical description
SEC-01	Functional	Security FG / Edge Node Awareness FC	Implementing decision-making security algorithms
SEC-02	Functional	Security FG / AuthN, AuthZ FC	Authorize and authenticate system components and users
SEC-03	Functional	Security FG/ Threat Analysis FC	Implement algorithm for data confidentiality
SEC-04	Functional	Security FG/ AuthN FC	Enable Zero trust security approach
SEC-05	Functional	Security FG/{Logging FC, Audit FC}	Event logging and offline audit

5.3 Trust Perspective

Trust management platform for DEDICAT 6G dynamic networking and computational distributed systems would be responsible to ensure the integrity of highly dynamic and distributed communication and computation systems.

Platform will be based on private permissioned blockchain like Hyperledger Fabric. The trust management platform will facilitate identity management for users, devices and services.

Trust architecture as a security framework in 5G/6G networks is a solution to address security requirements in a network with untrusted infrastructure, devices and personals. Every access request is individually authorized and monitored during the access period for compliance with security policy rules.

The following Table 19 gives summarizes how the Trust perspective is dealt with.

Table 19: Trust perspective survey

Targeted System Quality	Full (100%) trusted communication between parties, devices and sub-systems
Requirement(s)	• Trust assurance; • Trusted relation distribution; • Identify trust goals; • Trust requirements validation. FREQ-3, FREQ-9 to FREQ-11, FREQ-21 to FREQ-26, NFREQ-81
Activities	• Calculate trust metrics; • Private permissioned blockchain; • Trustworthiness policies.
Tactics	• Utilizing private permissioned blockchain to specify and implement trust management platform based on blockchain technologies and to specify and implement trust metrics and levels of trustworthiness for DEDICAT 6G networks; • To implement security and privacy protection compliance auditing and certification procedures – certificates written to private blockchain through smart contracts; • Validate security, data protection and implement trust KPIs throughout project pilots.

Applying the tactics above, results in the following list of Design Choices (Table 20 below)

Table 20: Design choices for the Trust perspective

Design Choice ID	View	FG/FC	Technical description
TR-01	Functional	PST FG / Distributed Ledger FC	Implement trust management platform based on blockchain
TR-02	Functional	PST FG / Distributed Ledger FC	Implementing smart contracts
TR-03	Functional	PST FG/{Trust Metrics FC, Threat Analysis FC}	Continuously validate security controls with breach and attack simulation and checking readiness of data protection security protocols

5.4 Performance Perspective

This fourth perspective is about system performance. We have identified five different tactics that can help to improve system performance, following five different angles (see the tactics below).

Table 21 below provides a general survey about the Performace perspective objectives, the FREQs it covers and most importantly the set of activities and tactics we followed in the DEDICAT 6G project as far as performance increase is concerned.

Table 21: Performance Perspective survey

Targeted System Quality	The ability of the system to predictably execute within its mandated performance profile and to handle increased processing volumes should this happen
Requirement(s)	FREQ-3, FREQ-6 to FREQ-11, NFREQ-1 to NFREQ-3, NFREQ-51 to NFREQ-59
Activities	• Decrease latency and improve capacity • Improve energy efficiency • Improve computing effectiveness and flexibility • Capture performance indicators • Monitor performances and enforce SLA • Keep decision-making and SLA enforcement roles separated • Perform simulations
Tactics	• Extending radio coverage with UAV, AGV, connectedCars…) • Optimizing UE-BS association • Balancing intelligence among Edge Nodes and within Edge Nodes across multiple execution environments • Performance monitoring • SLA enforcement

5.4.1 Optimizing UE-{MAP/BS} association

After MAPs are deployed to cover a crowded group of users, users need to be associated with BS/MAPs. Users would require different level of quality of services for various services. Since the association will be leaded by each UE, UE will focus on satisfaction of its QoS level. From MAPs, considering its limited battery power, it can also consider the energy efficient perspective. Considering a given number of users and allowable radio resources including MAPs and limited power, the proposed UE association approach intends to maximize the QoS satisfaction level by providing appropriate communication connectivity for each UE.

Table 22: Design choices for UE-{MAP/BS} association optimization

Design Choice ID	View	FG/FC	Technical description
PERF-01	Functional	UE Status Agent FC	It collects the info about the UE's satisfaction considering required QoS
PERF-02	Functional	UE Awareness FC	It identifies UE's existence and collects the info about the required QoS level and supportable communication capability
PERF-03	Functional	UE-{BS/MAP} Association DM FC	It identifies a subset of possible BSs/MAPs by using the context of BSs and MAPs
PERF-04	Functional	UE-{BS/MAP} Association Negotiation FC	It decides the subchannel and power for communication link setup
PERF-05	Information	n/a	UE status and context-related data structures

5.4.2 Radio Coverage Extension

This Table 23 below introduces the DCHs relating to radio coverage extension. They introduce the various FCs that support this feature, starting with all FCs responsible for building up appropriate contexts (all Status Agent and Awareness FCs), followed by the CEDM FC that takes decisions based on those contexts. Finally, we have the FCs (all Operation FC) that are ultimately implementing those decisions.

Table 23: Design choices for the radio coverage extension

Design Choice ID	View	FG/FC	Technical description
PERF-10	Functional	UE Status Agent FC	It collects the perceived rate from Performance Analytics FC of each UE, calculate the total network utility function and broadcast it to Swarm Operation DM FC of each MAP
PERF-11	Functional	UE Awareness FC	It measures the RSS, AoA, status and QoS for all users and sends the UE info to the Swarm Operation FC of each MAP
PERF-12	Functional	MAP Status Agent FC	It evaluates the MAP capabilities (energy, cost, position) and sends it to the Swarm Operation FC
PERF-13	Functional	MAP Awareness FC	It collects MAP information (position and previous decisions of MAP) and sends it to the Swarm Operation FC
PERF-14	Functional	Swarm Operation FC	It collects the request of CEDM or automatic generate it in a self-* mode. It collects the total network utility function from Network Status Agent FC, the RSS, AoA, QoS from all UEs (UE Awareness FC), the MAP information (position and previous decision of MAP) from MAP deployment Awareness FC, the MAP capabilities from MAP Status agent. It identifies the next destination or next move and request it to the UAV Operation FC and waits for the confirmation.
PERF-15	Functional	UAV Operation FC	It receives the next destination or next move from Swarm Operation FC, defines the best trajectory to follow until the destination or the next position and avoids collisions with other MAPs
PERF-16	Functional	AGV Operation FC	It provides radio communication management features on top of a catalog of basic capabilities
PERF-17	Functional	ConnectedCar Operation FC	It provides mainly radio configuration and operation management functionalities
PERF-18	Functional	CEDM FC	The operator can request via the CEDM FC the coverage extension and the deployment of a MAP. This request is forwarded to the Swarm Operation FC

5.4.3 Balancing Intelligence

In the event of fallen nodes or congestion on systems, deployed services will have to migrate intelligently to keep the platform online and ensure a sufficient QoS. Functions will be orchestrated and balanced at the edge nodes and from a cloud perspective, thanks to the IDDM FG that will provide intelligent directives on how and where to act. Table 24 below provides the needed FCs that support intelligence balancing.

Table 24: Design choices for intelligence balancing

Design Choice ID	View	FG/FC	Technical description
PERF-20	Functional	Service Orchestrator FC	It executes the outcomes of the DM FG to deploy FCs and network components.
PERF-21	Functional	Edge Service Orchestrator FC	It has the ability to migrate/clone µS/FC between different Physical Systems.
PERF-22	Functional	Load Balancing FC	It supports for inter and intra-node load balancing between the FCs.
PERF-23	Functional	IDDM FC	It decides the optimal placement of intelligence in terms of data and computation as µServices.

5.4.4 Performance monitoring and SLA enforcement

Context-awareness and SLA enforcement are important bricks to the establishment of overall good system performance. The first one provides all components described in the previous sections with the essential information about the overall system health while SLA enforcement is an essential mechanism that can help to maintain the system performance as it should be (especially with regards to the QoS agreed during a IDaaS or CEaaS service agreement between a vertical and the platform). The following Table 25 provides the list of FCs involved in this activity.

Table 25: Design choices for the performance monitoring and SLA enforcement

Design Choice ID	View	FG/FC	Technical description
PERF-30	Functional	all Status Agent FCs	They capture µS/FC, EN, MAP, UE and NW statuses and report to their respective Awareness FCs
PERF-31	Functional	all Awareness FCs	It provides µS/FC, EN, MAP, UE and NW contexts for system performance monitoring
PERF-32	Functional	SLA Registry FC	It stores the terms of a service agreement between a customer (vertical) and the service provider (the platform) for the delivery of CEaaS and IDaaS. It also enforces those terms using performance monitoring. Finally, it reports to the IDDM and CEDM FCs when a SLA breach occurs
PERF-33	Functional	IDDM and CEDM FC	It is responsible for maintaining the performance through performance monitoring using the instruments described in earlier section of the Performance perspective

6 Conclusions

This document provided a second iteration of the DEDICAT 6G architecture.

The main achievements compared to the previous version D2.2 are:

- An updated set of UNIfied requirements (that takes into account D2.3 new scenario requirements) and new VOLERE template;
- A requirement cross-check has been achieved (visible in Volere template – right hand side);
- The Context view content has been partly replaced with an example of interaction diagrams intended to implement the two first UC2 UML diagrams of D2.3. It aimed at illustrating how generic system use-cases can be reused and instantiated in a particular Vertical context;
- The Functional view has been revised (new FCs and updated FC descriptions);
- An extended list of system use-cases covering the three project pillars with interaction diagrams;
- A new Information view that precisely defines most of the FC interfaces with the needed data models, plus some data flow diagrams. The vast majority of the interaction diagrams refers to those interfaces;
- A revised baseline Network Deployment view and four additional UCs Network Deployment views;
- A new additional Performance perspective.

This updated version of the DEDICAT 6G architecture provides then the reader with a very precise view of the functionalities making the platform, a non-less precise definition of the interfaces they provide, and a wide illustration of how those components work with each other, either being used in a particular UC context or used to elucidate a selected platform behavior (system use-case).

The final iteration of the architecture document will focus on the following aspects:

- Completing and revising the interface according to WP3, 4 and 5 results;
- Revising the Functional View, in case missing components are identified;
- More generally: improving the content and level of detail everywhere possible and aligning all sections with the latest outcomes from WP3, 4, 5 and 6;
- Providing the Instantiation View that offers a mapping of actual implemented concrete components onto the logical FCs as described in this document;
- Completing the Perspective section if needed;
- Cross-checking the architecture with the VOLERE template in order to identify eventual gaps or inconsistencies;

References

[1] DEDICAT 6G consortium, "DEDICAT 6G Description of Action".

[2] N. Rozanski and E. Woods, Rozanski, Nick and Woods, Eoin. "Software Systems Architecture – Working with Stakeholders Using Viewpoints and Perspectives", Addison Wesley, 2011.

[3] DEDICAT 6G Deliverable D2.3 "Revised Scenario Description and Requirement Collection"

[4] Hernan, S.; Lambert, S.; Shostack, A.; & Ostwald, T. "Uncover Security Design Flaws Using the STRIDE Approach". MSDN Magazine. November 2006.

[5] DEDICAT 6G list of UNIfied requirements (as a VOLERE template) https://dedicat6g.eu/wp-content/uploads/2021/09/Dedicat6G_UNI_Requirements_v1.xlsx

[6] IEEE Architecture Working Group, "IEEE Std 1471-2000, Recommended practice for architectural description of software-intensive systems", 2000.

[7] Rozanski, Nick and Woods, Eoin, "Applying Viewpoints and Views to Software Architecture", http://www.viewpoints-and-perspectives.info/vpandp/wp-content/themes/secondedition/doc/VPandV_WhitePaper.pdf (Last accessed 28/09/21)

[8] Atlantic Systems Guild Ltd., "Volere Requirements Resources", http://www.volere.co.uk/ (Last accessed 28/09/21)

[9] DEDICAT 6G Deliverable D1.3 "Data Management Plan".

[10] DIKY Pyramid, Wikipedia page https://en.wikipedia.org/wiki/DIKW_pyramid (Last accessed 28/09/21)

[11] "IoT-A Deliverable D1.5 – Final Architectural Reference Model for the IoT v3.0", Carrez, F. editor https://www.dropbox.com/s/8u9yfbmxbwskw2w/D1.5%20%202013.07.15%20VERYFINAL.pdf?dl=0 (Last accessed 28/09/21)

[12] 3GPP TS 23.501 V1.0.0, "System Architecture for the 5G System; Stage 2 (Release 15)", June 2017.

[13] 3GPP TS 23.791 V1.0.0, "Study of enablers for Network Automation for 5G; (Release 16)", December 2018.

[14] https://derekcheung.medium.com/5g-core-pdu-session-and-qos-part-1-a12852e1b342 (Last accessed 28/09/21)

[15] 3GPP, Specification 23.502 "Procedures for the 5G System (5GS)", Release 15, 3GPP, 2017.

[16] 3GPP TS 23.791 V1.0.0, "Architecture enhancements for 5G System (5GS) to support network data analytics services; (Release 16)", June 2019.

[17] 3GPP TS 23.791 V1.0.0, " 5G System; Network Data Analytics Services; Stage 3 (Release 15)", June 2018.

[18] ETSI MEC, "Multi-access Edge Computing (MEC) 5G Integration, V2.1.1", ETSI, 2020

[19] Innovation in 5G backhaul technologies, June 2020, [online] Available: https://www.5gamericas.org/wp-content/uploads/2020/06/Innovations-in-5G-Backhaul-Technologies-WP-PDF.pdf (Last accessed 28/09/21)

[20] 3GPP, “TR 38.801, Technical Specifications Group Radio Access Network: Study on new radio access technology: Radio access architecture and interfaces,” 3GPP, 2017.

[21] ITU-T, “Technical Report GSTR-TN5G - Transport network support of IMT-2020/5G,” ITU-T, 2018.

[22] 3GPP, “Study on New Radio Access Technology: Radio Access Architectures and Interfaces. 3GPP Technical Specification,” 3GPP, 2018.

[23] [ETSI-NFV] ETSI, “NFV Release 4 Definition”, ETSI, 2021.