"5G for Drone-based Vertical Applications"

D2.4 – Definition of the trial controller architecture, mechanisms, and APIs

Document ID:	**D2.4**
Deliverable Title:	**Definition of the trial controller architecture, mechanisms, and APIs**
Responsible Beneficiary:	**EUR**

Topic:	**H2020-ICT-2018-2020/H2020-ICT-2018-3**
Project Title:	**Unmanned Aerial Vehicle Vertical Applications' Trials Leveraging Advanced 5G Facilities**
Project Number:	**857031**
Project Acronym:	**5G!Drones**
Project Start Date:	**June 1st, 2019**
Project Duration:	**42 Months**
Contractual Delivery Date:	**M24**
Actual Delivery Date:	**May 28th, 2021**
Dissemination Level:	**PU: Public**
Contributing Beneficiaries:	**WP2 Partners**

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 857031.

Document ID: D2.4
Version: V4
Version Date: 27/05/2021
Authors: Adlen Ksentini (EUR), Mohamed Mekki (EUR), Karim Boutiba (EUR), Piotr Dybiec (DRR), Pawel Korzec (DRR), Gokul Srinivasan (RBX), Tanel Järvet (CAF), Thomas Lutz (FRQ), Gregor Mogeritsch (FRQ), Syed Ali (UO), Stavros Kolometsos (NCSRD), Anastasios Gogos (NCSRD), George Makropoulos (NCSRD), Pantelis Sarantos (NCSRD), Harilaos Koumaras (NCSRD), Steven Roelofsen (INV), Yannick Schaffer (INV), Pawel Montowtt (INV), Dimitris Tzempelikos (MoE), Abderrahmane Abada (AU), Oussama Bekkouche (AU), Hamed Hellaoui (AU), Tarik Taleb (AU)
Security: Public

Approvals

	Name	Organization	Date
Coordinator	Jussi Haapola	UO	28/05/2021
Technical Committee	Farid BenBadis	THA	25/05/2021
Management Committee	Project Management Team, Additional Reviewers	UO, AU, THA, UMS, AIR, NCSRD, HEP, ORA	16/05/2021

Document History

Version	Contribution	Authors	Date
V0	ToC	EUR	22/11/2020
V0-1	V0-1 and updated ToC	EUR	28/03/2021
V1	First version including all contributions	All	20/04/2021
V1.1	Editor review	EUR	23/04/2021
V2	Second version	All	03/05/2021
V2.1	Internal review	All	16/05/2021
V3	Third version	All	23/05/2021
V3.1	TM review	THA, AU, NCRSD	26/05/2021
V4	Final version	All	27/05/2021

Executive Summary

The 5G!Drones project aims to trial use case scenarios of UAVs on top of 5G facilities to evaluate 5G systems to support UAV vertical industry requirements. Four representative use cases with different scenarios have been selected to run on four different 5G facilities. In order to run UAV trials: (i) the trial system should abstract the low-level details to allow the UAV verticals to describe the trial scenario, in terms of mission planning and the needed network and computing resources, by providing intuitive and user-friendly interfaces; (ii) each trial consists of running one or two Network Slices (one uRLLC and eMBB or mMTC); (iii) each 5G facility uses a different template to deploy and run a network slice.
The key element of the 5G!Drones architecture is the Trial controller, which includes several functions of the Drone Operator, such as Preparation and Validation of the flight plan, deployment of the UAV use case scenario on top of one of the available 5G facilities. A web portal is defined, which allows interacting with the Trial system easily and defines the flight plan and needed resources to run a trial. The Trial controller ensures the following functions:

- Preparation & Validation: This function is triggered upon the reception of a request from the Trial Owner through the Web Portal, describing the trial scenario. During this first phase the flight plan is prepared, then the function proceeds to its validation. To validate the trial and ensure safe access to the airspace by the UAVs, the Trial Controller interacts with the UTM and the facility through a dedicated interface in order to avoid collision with other flights, check authorisation, and availability of 5G resources. In case of a rejection, the Trial Owner is requested to update the flight plan following the encountered limits. This process is repeated until the flight is validated and approved or the Trial Owner decides to cancel the request.
- Deployment: After the flight plan validation and the reception of the network as well as the computing resources requirement, in the form of a Blueprint, the deployment module translates the latter to a Network Slice Template (NST), which will be used to request the creation of a network slice on top of the selected 5G facility to run the trial. As no standard exists but only recommendation for the NST, each facility has its own NST model. Therefore, the Trial Translator has to translate the Blueprint to an NST according to the selected facility model or template.
- Key Performance Indicator (KPI) measurement: A key feature of the 5G!Drones project is the capacity to measure the KPI of the trial, which is composed of UAV and 5G KPI. Thus, one of the functions assumed by the Trial controller is to measure KPI of the run trial and present them through a graphical interface to the trial owner.

To ensure the above-mentioned functions, the trial controller has been divided into several individual components: Web portal, Trial Validator, U-Space Adapter, Trial Translator, Trial Repository (also referred as repositories' module), Life Cycle Manager (LCM), Trial Enforcement and KPI Monitoring. Each component is responsible for specific tasks, which combined together cover all the trial controller functions.
This deliverable builds on the trial controller architecture first introduced in the deliverable D2.1. It further details and updates each component's role and functions. For each component, the deliverable provides: (1) a detailed description of the functions it covers, (2) its architecture; (3) the exposed interfaces to the other components; (4) workflow detailing some key functions. Furthermore, the deliverable demonstrates how all these components interact to validate and run a trial, as well as covering the case where a flight plan operation is rejected by U-Space.
It should be noted that this deliverable includes the final architecture and functions of the 5G!Drones Trial controller.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

5G	5th Generation
5G PPP	5G Infrastructure Public Private Partnership
A2C	Advantage Actor Critic
API	Application Programming Interface
CRUD	Create, Read, Update and Delete
dFPL	Drone Flight Plan
DQL	Deep Queue Learning
DQN	Deep Queue Network
DRL	Deep Reinforcement Learning
eMBB	Enhanced Mobile BroadBand
GCS	Ground Control Station
GSMA	Global System for Mobile Communications Association
GST	Generic Slice Template
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IAM	Identity and Access Manager
KPI	Key Performance Indicator
KPIC	Key Performance Indicator Component
LCM	Life Cycle Manager
ML	Machine Learning
mMTC	Massive Machine-Type Communications
NS	Network Slice
NST	Network Slice Template
RAN	Radio Access Network
RL	Reinforcement Learning
TE	Trial Enforcement
UAV	Unmanned Aerial Vehicle
uRLLC	Ultra-Reliable Low-Latency Communications
UTM	Unmanned Traffic Management

VNF	Virtual Network Function

1. INTRODUCTION

1.1. Objective of the document

The trial controller is the key element of the 5G!Drones trial system. It is in charge of executing all the necessary steps to run a UAV trial. The trial controller first allows the trial owner to specify the flight plan and the 5G resources needed to run the trial. Then, the trial controller runs all the needed verifications to validate the flight plan with U-Space and guarantees that the 5G resources are available at the 5G facility. After the validation steps, the trial controller enforces the trial as a network slice and manages its life-cycle as well as the KPI measurement.

This deliverable aims to give a detailed description of the trial controller, which will allow finalising the implementation and integration phases. Building on the initial architecture introduced in D2.1, it details the role of each module composing the trial controller, namely web portal, trial validator, U-Space adapter, trial translator, trial repository, Life Cycle Manager (LCM), trial enforcement, and KPI monitoring. For each component, the deliverable provides: (1) a detailed description of the functions it covers, (2) its architecture; (3) the exposed interfaces to the other components; (4) workflow detailing some key functions.

1.2. Structure of the document

The deliverable is divided into two main sections, in addition to a conclusion. Section 2 is recalling the trial controller architecture, the components composing the trial controller, and high-level workflows representing: trial deployment, KPI measurement job creation and execution, trial deletion, and the case of a trial rejection. The objective of the section is to show how the different components interact together to run a trial, including both accepted and rejected cases, measure KPI, and delete a trial.

Section 3, on the other hand, is dedicated to each component of the trial controller and organised into different sub-sections. The objective is to give more details on each module's functions, architectures, exposed interfaces, and workflows of the component's representative functions.

1.3. Target audience

This project consortium deliverable is mainly addressed to the general public for obtaining a better understanding of the framework and scope of the 5G!Drones project, and particularly the role of the trial controller. It shows how a UAV trial is defined, validated with U-Space, enforced on top of a 5G facility as a network slice, and managed. Furthermore, this deliverable is also targeting the two communities in the scope of the project, namely 5G (5G PPP and beyond) and UAV.

2. THE TRIAL CONTROLER ARCHITECTURE

In this section, we first recall the trial controller architecture and components. For each component, we highlight its main role in the architecture. Then, we present high-level workflows that illustrate how the different trial controller components interact to deploy a trial, including both successful and rejected cases, manage the trial's life-cycle, collect KPI measurements, and delete a trial.

2.1. Overall architecture concept and design

As introduced in D2.1 [1] and shown in Figure 1, the trial controller is composed of:

- Web portal: It is the entry point of the Trial Controller. It is the place allowing the trial owner to define the Operation Flight Plan for Use Cases and browse the existing plans. The Web portal interacts with:
    - Trial Translator to allow the Trial owner to create the Network Slice Template (NST) that runs the trial on top of the chosen facility.

  - Trial Validator to validate Flight Plan as well as the trial on the targeted facility.
  - Trial Repository to register the new trial and create the Trial ID.

- Trial Translator: It allows the trial owner to describe the computing and network resources needed to run the trial and the UAV services to generate a NST needed by the 5G facility to create a network slice to run the trial. The Trial Translator is called by the Web portal through a HTTP redirection to redirect the Trial owner to the web interface exposed by the Trial Translator. Hence, the Trial owner fills a set of fields to create the NST automatically. After the creation of the NST, the latter is communicated to the Trial Repository to update the trial information. The Trial Translator is active only for the NST definition. In this document, the Trial Translator is called Web portal 2, while Web portal is called Web portal 1**.**
- Trial Repository: This module stores all information and details regarding the trial, trial templates, Virtual Network Functions (VNFs), UAVs, and facilities. It stores the actual status of the trial and allows other modules to retrieve information on the existing trials. The Trial Repository allows:
  - Web portal 1 to create a new entry for a Trial and create a TrialID
  - Trial Validator to update the status of the Trial, i.e., rejected/accepted
  - Trial Translator to store the NST corresponding to a Trial
  - LCM to retrieve the NST corresponding to a Trial.
- Trial Validator: It treats the Operation Flight Plan validation requests. It is active from the moment the validation is requested until the start of the implementation. It will stop if validation is failed or cancelled or the validated plan is withdrawn. In case of failures, it should give back the information about the errors to ease the corrective actions. Besides, the trial validator is in charge of checking if the targeted facility is ready to host the trial. Flight plan validation is done via the U-Space Adapter, while the trial validation at the facility is done through the Abstraction layer. Once the status (i.e., accepted or rejected) of the validation plan is known, the Trial Validator updates Trial Repository, after which Web Portal 1 and LCM can update their information about plan's status accordingly.
- Life Cycle Manager: It is a key element of the Trial Controller. It supervises the trial during its execution. When the time for the experiment approaches, the LCM triggers the network slice creation and instantiation to run the trial and starts measuring the KPIs. LCM is in touch with U-Space adapter to monitor the Flight plan's status, and if the status of the validation plan changes, it reacts accordingly. LCM triggers the network slice creation and instantiation using the Trial Enforcement, as well as the creation of the measurement jobs. Besides, LCM requests the KPI measurement data from the KPI monitoring.
- Trial Enforcement is responsible for the execution and automation of the trials as well as monitoring the trial status in 5G facility. The Trial Enforcement uses the Abstraction layer to manage the Life Cycle of the Networks Slice running the trial.
- U-Space Adapter: This module is specialised in aggregating and translating the interfaces between U-Space and Trial Controller. The Adapter will ensure that U-Space services are accessible and readable by Trial Controller and the data forwarded to U-Space is in an acceptable format.
- KPI Monitoring: It is the module that receives the KPI measurement from the 5G facility through the Abstraction layer and adapts them in order to be consumed by LCM. It also receives UAV related status information, which are reflected in telemetry data.

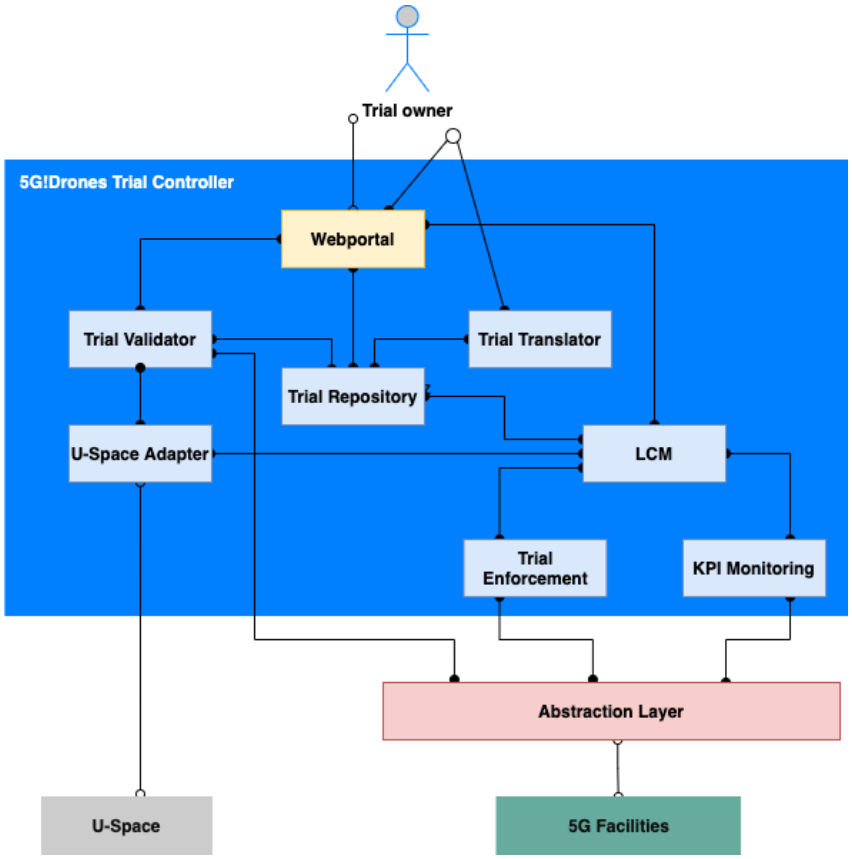

Figure 1. Trial Controller architecture.

All the interaction with the different facilities passes through the Abstraction layer defined in WP3. As a reminder, the Abstraction layer exposes a common API to manage the Life Cycle of a network slice (i.e. instantiation, runtime and deletion) running a trial and the KPI measurement. The Abstraction layer translates the different APIs to calls to facility-specific APIs. Hence, ensuring a common functioning of the trial controller whatever the targeted facility.

2.2. Trial Call flows

2.2.1. Trial creation and deployment

Figure 2 illustrates the workflow to create and deploy a trial. The first step corresponds to the flight plan definition done by the Trial owner using Web portal. In step 2, Web portal registrates the new trial at the Trial Repository, which will reply with a Trial ID that uniquely identifies a trial. Then, the trial owner is redirected to the Trial Translator or Web portal 2 to create the NST. Since each facility uses a different NST, then four Web portal 2 exist. According to the selected facility to deploy a trial, Weportal 1 redirects the trial owner to the appropriate Web portal 2 (step 3). An HTTP redirection is used, including the Trial ID as a parameter. Thereafter, Web portal 1 starts the trial validation process (step 4) by sending a request to the Trial Validator, including the Flight Plan as well as the facility name to run the trial. Web portal 2 creates the NST using the trial owner's information and stores the NST in the Trial Repository using the Trial ID as discriminator (step 3'). Trial validation is performed as two independent parallel tasks: validation of prepared operational flight plan at U-Space (4') and validation of the availability of required resources at the facility (4''). Based on the answers obtained in steps 4' and 4'', the Trial Validator updates the trial's status (step 5) at the Trial Repository.
After the trial status update and if it is validated, the trial owner initiates the LCM, through Webporal 1 (step 6), to start the implementation of the trial when the time comes. When the time comes to run the

trial, LCM fetches the NST of the trial from the Trial Repository (step 7) using the Trial ID, requests the creation and the instantiation of the network slice to run the trial at the targeted facility using the Trial Enforcement API (step 8). In its turn, the Trial Enforcement modules use the Abstraction layer API to create and instantiate the network slice (step 9). The Trial Enforcement returns the Slice ID to the LCM for life cycle procedures and creating the KPI measurement.

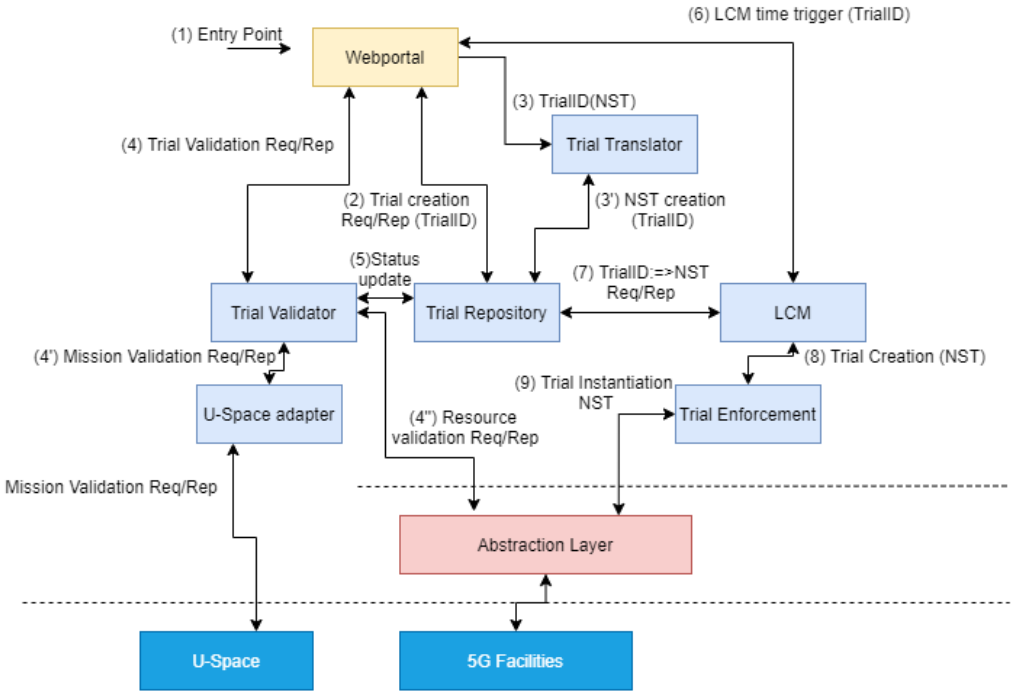

Figure 2. Trial creation and deployment workflow.

Going into more details, Figure 3 demonstrates the setup of 5G services between LCM, Trial Enforcement and 5G facility. As a result, the slice described by NST along with all required VNFs should be deployed.

As shown on the diagram and mentioned earlier, the process of slice deployment is triggered on time basis. Process is initiated and then managed by LCM module. At first LCM fetches the trial configuration information from the Trial Repository (based on unique TrialID). In the next step, it requests Trial Enforcement module to perform slice instantiation. Based on the NST stored in the Trial Repository, Trial Enforcement requests the facility (through the abstraction layer) to create the slice. After successful creation, returned SliceId is stored by LCM along with updated trial status to the Trial Repository. SliceId is used then to create measurement job. In the next steps onboarding and instantiations of VNF applications is performed. Two options are described in the diagram: onboarding and instantiation of cloud based VNF applications and onboarding and instantiation of edge based VNF applications. Depending on the trial scenario only one or two of mentioned options will be executed. Each of those options consists of two steps: onboarding and instantiation. Onboarding is a process of downloading the VNF image to the target system. In current implementation of the Trial Controller images will be fetched from dedicated library (not directly from the Trial Repository, which keeps only reference to the file). After successful onboarding, VNF application is instantiated. Each step is confirmed by status update of the trial to the Trial Repository. Last, optional sequence refers to automated testing of instantiated resources.

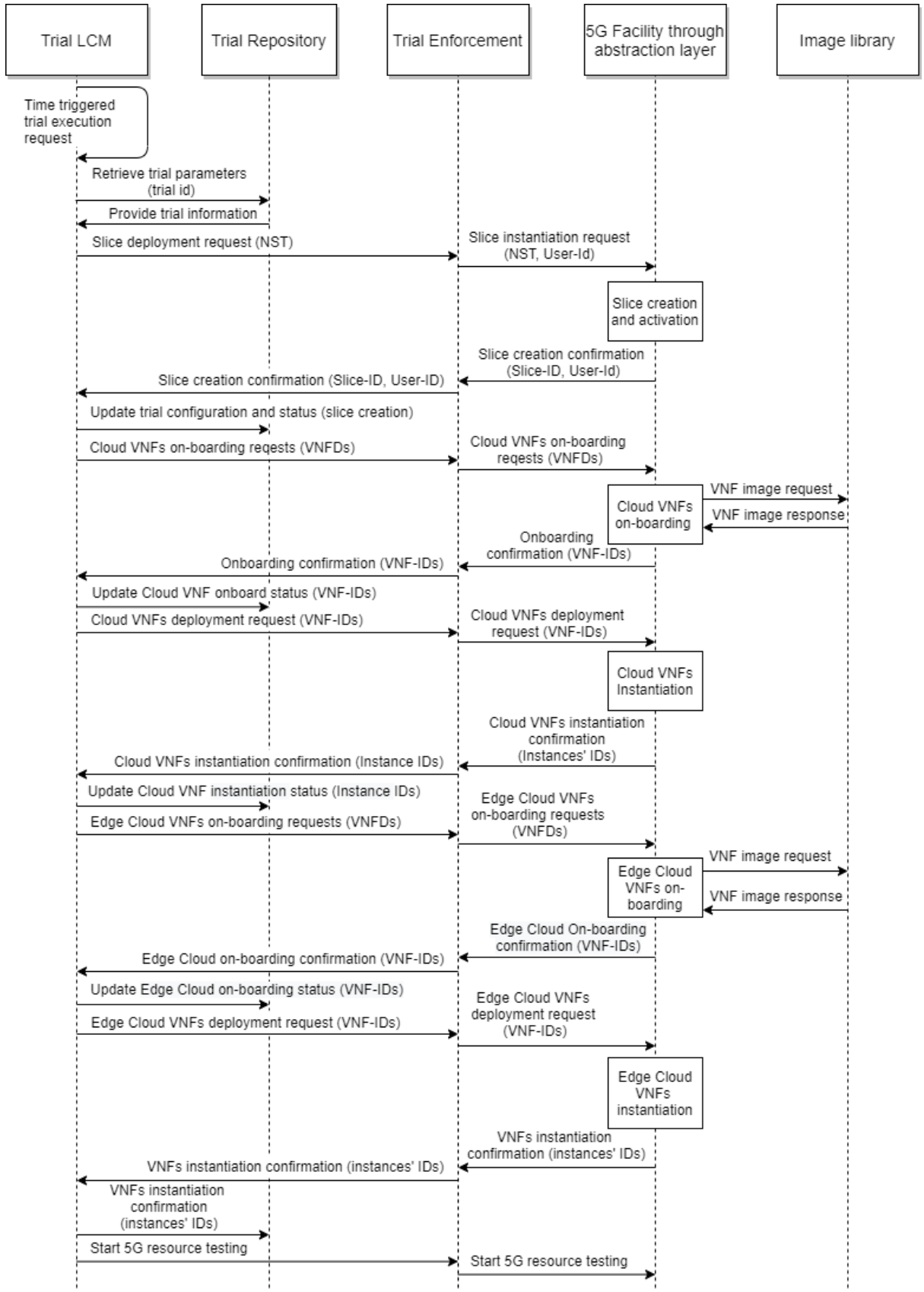

Figure 3. Setup of 5G services for a trial.

From now on, the trial can be activated. Process of the trial activation is shown in Figure 4.

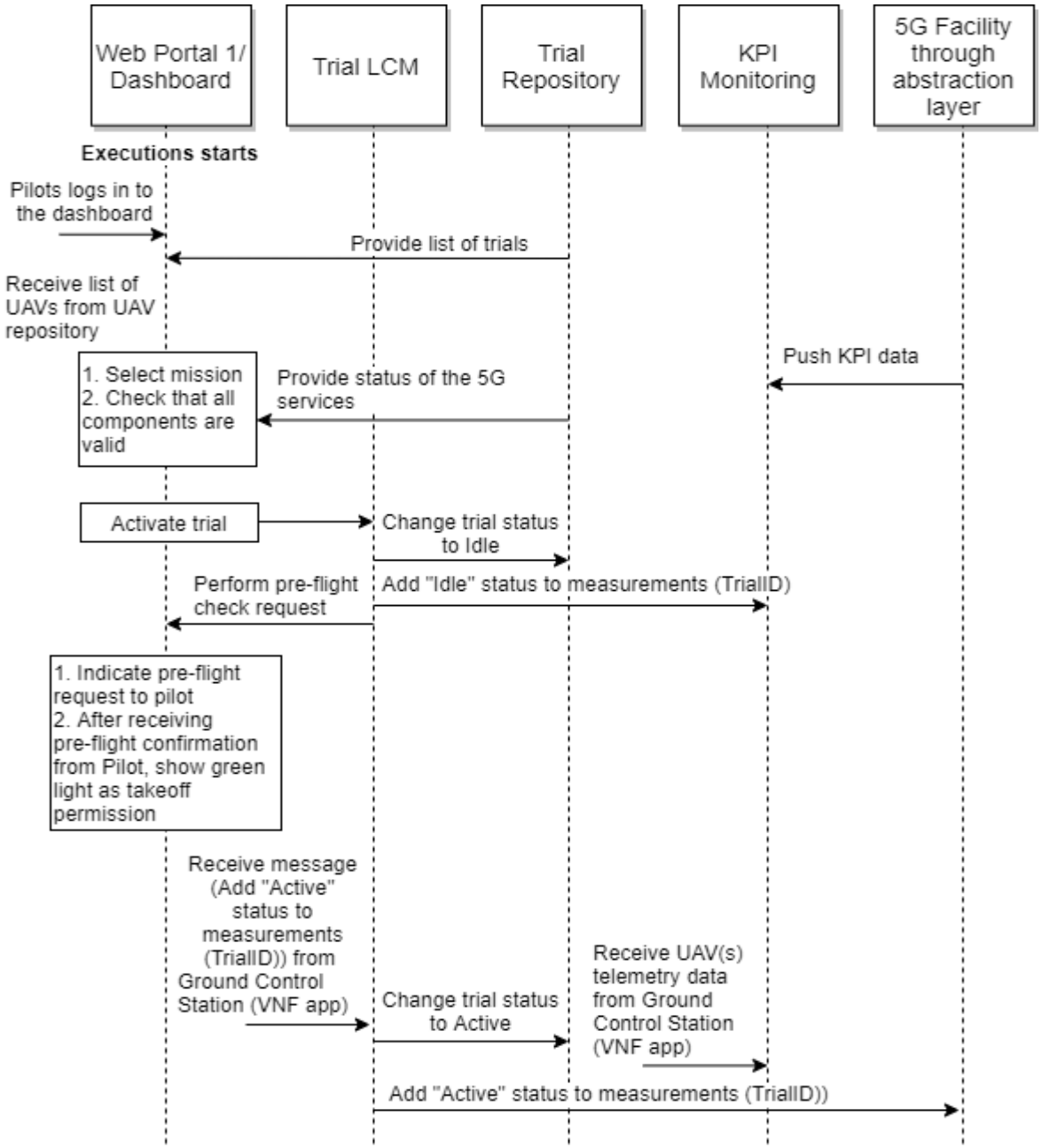

Figure 4. Trial activation.

Process shown in Figure 4 illustrates, how the trial is activated from the user/experimenter perspective. To activate the trial it is required, that facility is already provisioned and ready to be used, which is described earlier, in Figure 3. When user (pilot, experimenter) wants to start the trial execution, he logs into the Web Portal 1, selects the appropriate mission and can confirm the status of the trial. If it is OK, he can manually activate the trial, which causes to change the trial status to Idle. In parallel, LCM requests KPI monitoring to add label “idle” to the measurements. In next step LCM instructs the user to perform the necessary pre-flight checks of the UAV. When the drone is ready to fly, mission can be started. When mission is being started, the LCM receives also request to mark the KPI measurements with label “Active”. From now on the mission is executed. After the UAV starts, the telemetry data can be also received through the U-Space Adapter.

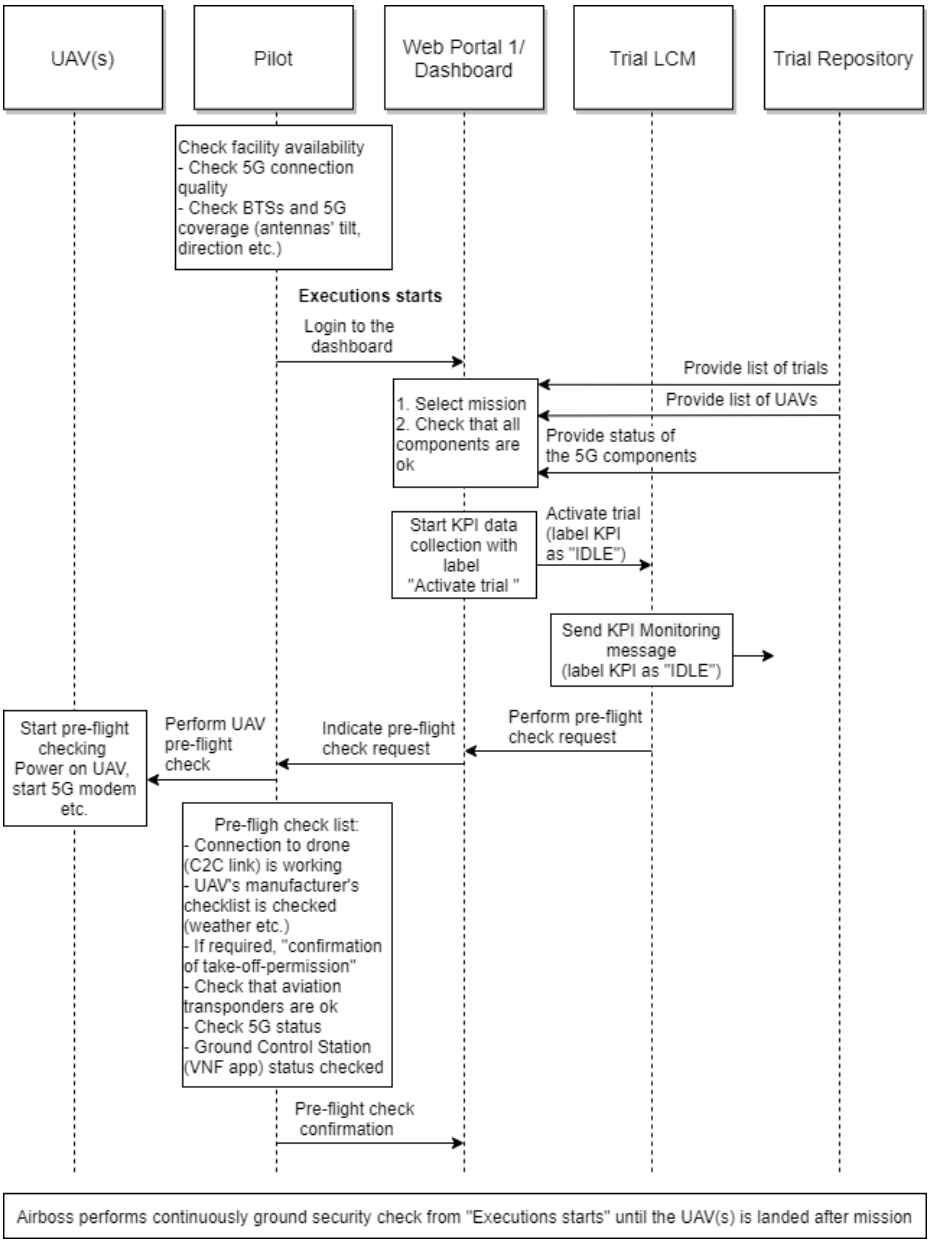

Figure 5. Pre-flight checks.

Before the mission can be started, some checks should be performed. At first it should be verified, whether the appropriate 5G service and radio coverage are available. This process is shown in Figure 5. At first step Pilot should check, whether the 5G service is available. Then there is a process of activation, which is in more details described in the diagram of Figure 4. After successful activation Pilot receives the request to perform pre-flight checks. This includes checks mentioned on the diagram and in principle is UAV specific. After pre-flight is passed, Pilot confirms it to the LCM through the Web Portal 1/Dashboard.

After successful checks and receiving take-off permission (whenever necessary), the mission can be started. The whole mission execution process is shown in Figure 6.

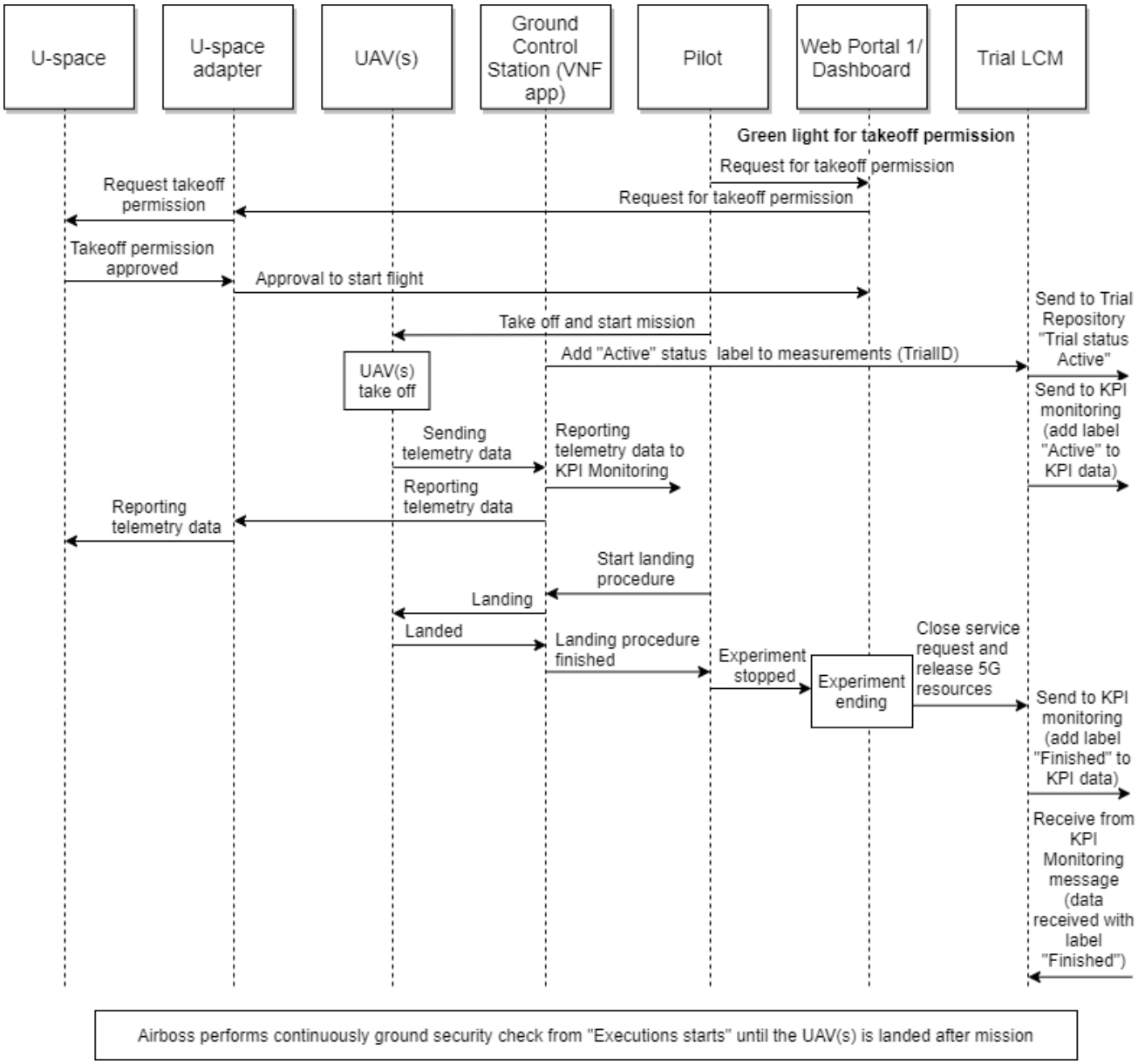

Figure 6. Sequence of steps from UAV take off to landing.

The initial sequence of requesting/granting flight permission shown in Figure 6 is optional and depends on the operational flight conditions. Take-off of the drone triggers setting the KPI labelling to the Active state. Just after the take-off also telemetry data transmission related to the UAV is started. Telemetry data received from the UAV is forwarded to the UTM system as well as to the KPI Monitoring module. When the mission is about to be finished, pilot starts the landing procedure. When it is finished, Pilot also triggers through the Web Portal 1/Dashboard the closing request to the LCM. The LCM in first step requests to change the KPI measurements labelling to “Finished”. Further steps are described in the next section.

2.2.2. KPI monitoring creation and measurement (DRR and team)

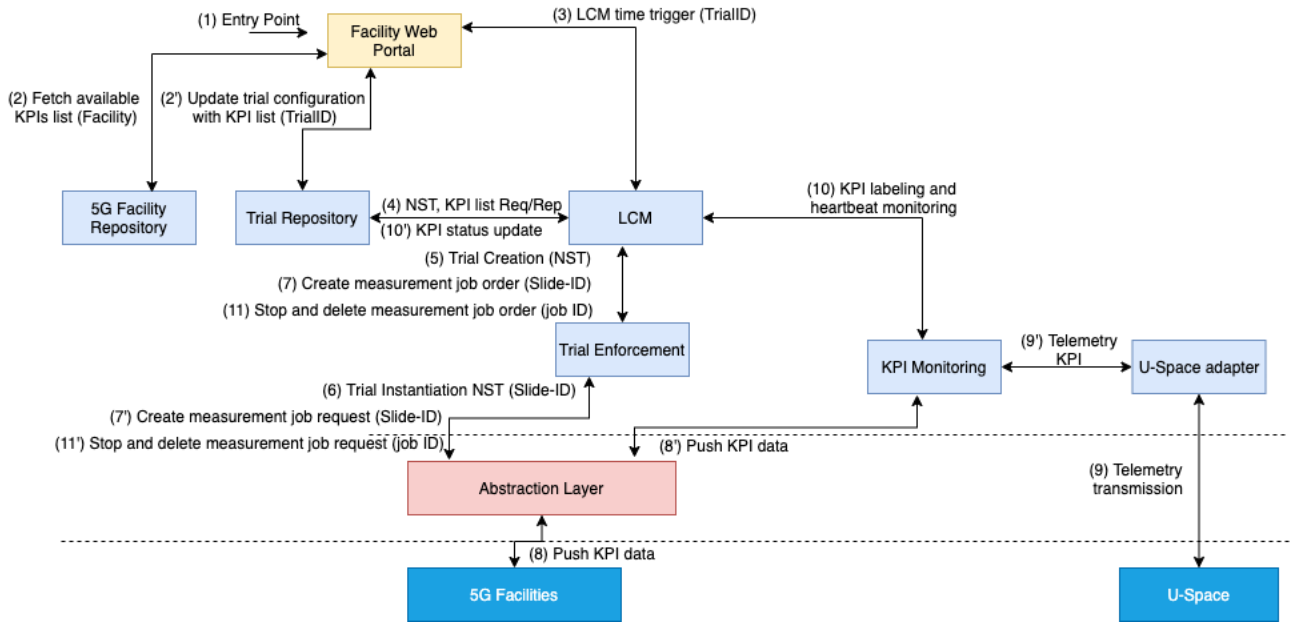

Figure 7. KPI measurement job creation and execution.

Figure 7 illustrates the workflow to create KPI measurements jobs and execute them. The first step (1) represents the login/switchover to the Facility Web Portal during the Trial definition phase (as described in section 2.2.1). At this stage, the corresponding Trial configuration information alongside with available KPI measurements list (2) is fetched. The Trial owner decides which KPIs from the available list will be measured during the Trial and this information is updated to the Trial Repository (2'). When the time comes to set up the trial (3), LCM fetches related Trial configuration information (4) from the Trial Repository: NST configuration and KPI list to be created as measurements' jobs at the facility. After slice enforcement (5) and instantiation (6), Slice-ID is provided by the facility. This Slice-ID, together with the list of KPIs is used to create measurements jobs at the facility (7). After successful measurement job creation (identified by job ID), the KPI measurements data stream is directed to the Abstraction Layer (8) and from here forwarded to the KPI Monitoring (8'). In parallel, after the mission is started, the corresponding telemetry data stream is provided by UTM system to the U-Space adapter (9) and forwarded as KPI measurements to the KPI monitoring (9'). During the trial execution LCM directs KPI Monitoring (10) to label measurements with labels corresponding to the appropriate trial execution status (Idle, Active, Finished). The LCM also performs periodic heartbeat monitoring, to confirm that valid KPI data is provided. After the Trial is completed, LCM requests the Trial Enforcement to stop and delete the measurement job (11).

In Figure 8 sequence diagram of the KPI setting up process is presented. The KPI collection setup starts, when the LCM sends the create measurement job request to the Trial Enforcement. Then the request is forwarded to the respective facility through the abstraction layer. After the measurement job is created, the KPI data streaming is started. Facility sends (pushes) the monitoring data stream to the facility adapter, which in turn forwards this stream to the KPI monitoring. The link to the KPI list is created by KPI Monitoring module and passed to the LCM to be stored for further reference in the Trial Repository. During the streaming of the KPI data, LCM actively monitors the continuity of the process by sending heartbeat requests to the KPI Monitoring.

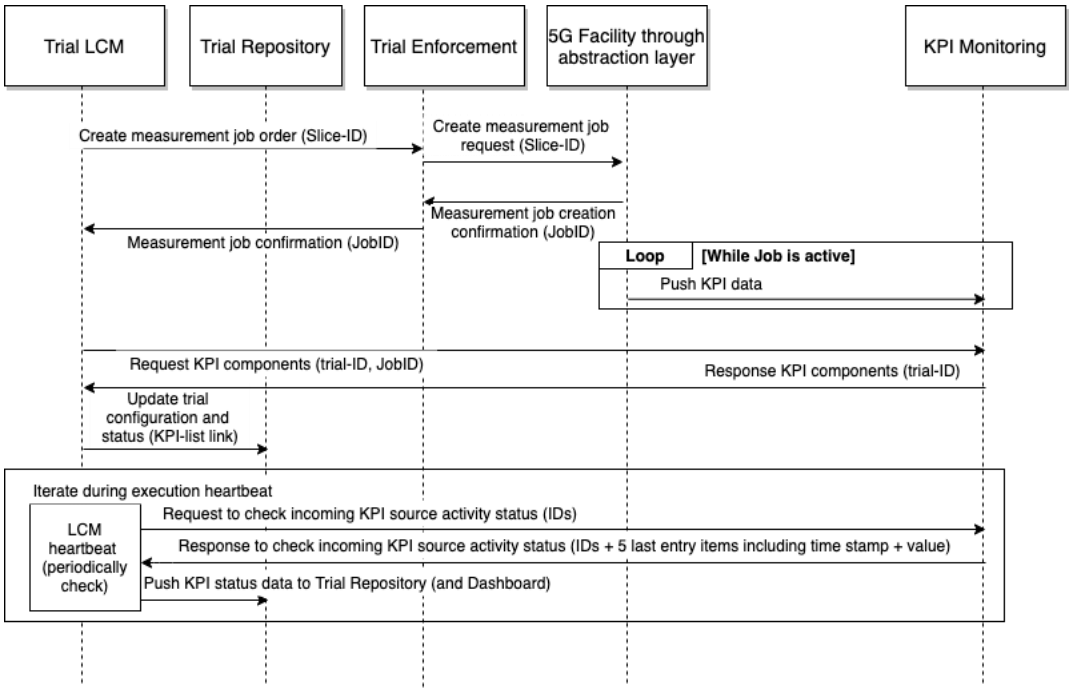

Figure 8. Setup of KPI collection.

2.2.3. Trial deletion

The Trial deletion can be performed by the Trial owner at any time by login in to the Web portal and deleting (i.e. marking as deleted) the Trial instance from the Trial Repository. No other action is required/performed with the Trial, which hasn't been set up yet (i.e. instantiated). Such a deleted trial wouldn't be instantiated.

The Trial configuration will also be removed from the facility after the Trial execution completion. This process is shown in Figure 9.

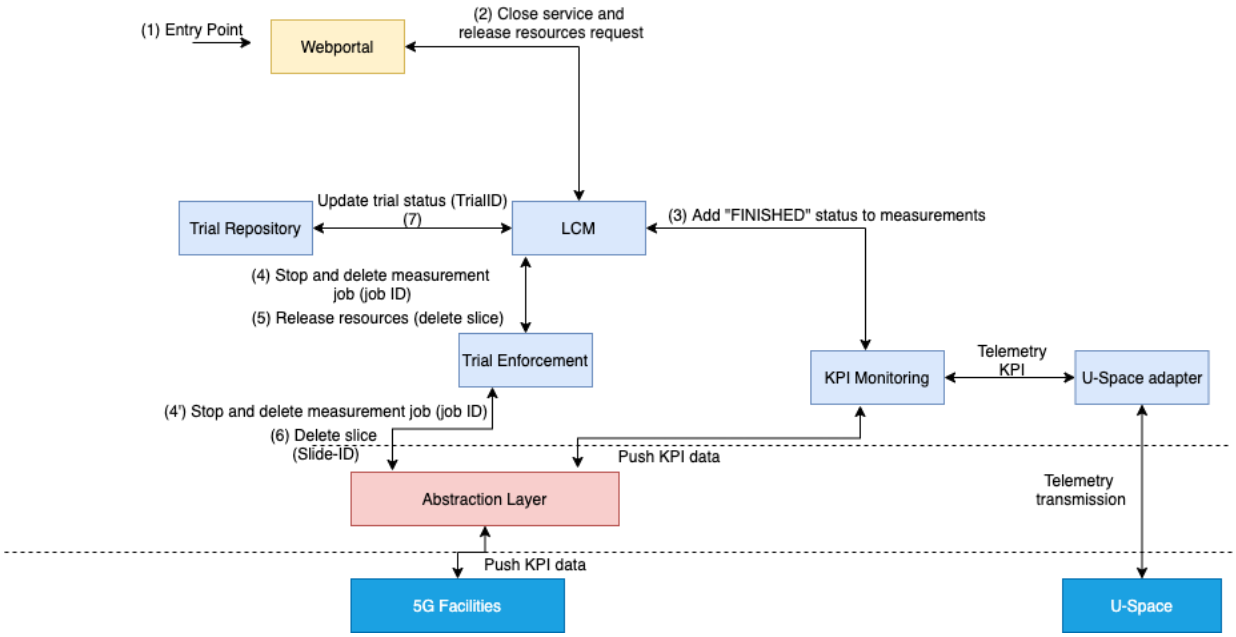

Figure 9. Trial deletion process.

The action is triggered by the experimenter/trial operator/trial owner through the Web portal (1). An appropriate request from Web portal is sent to the LCM (2). This triggers the resource release process managed by the LCM. In a subsequent step, as described in section 0, LCM requests the KPI Monitoring to label the KPI data with the “FINISHED” status (3). After this step, the LCM sends the stop and delete measurement job order to the Trial Enforcement (4), which sends the respective request to the facility through the Abstraction Layer (4’). Then the request to the Trial Enforcement to release the facility resources is sent (5). Slice decommissioning request is sent by Trial Enforcement (6). Appropriate status of the Trial is also updated to the Trial Repository (7).

The process flow for trial decommissioning is shown also in Figure 10. When the user (trial operator) triggers the “stop experiment” request, LCM starts to decommission the trial:

1. In the first step it updates the measurements data streaming to the “FINISHED” state (as mentioned and described in section 0).
2. Next LCM sends the request to delete the measurement job identified by JobID.
3. In the next step LCM sends to the Trial enforcement request to close the service and release 5G resources. It is forwarded through the abstraction layer to the facility.
4. Finally facility confirms the release of resources to the Trial Enforcement and further to the LCM.
5. LCM confirms this by updating the trial status in Trial Repository to Finished.

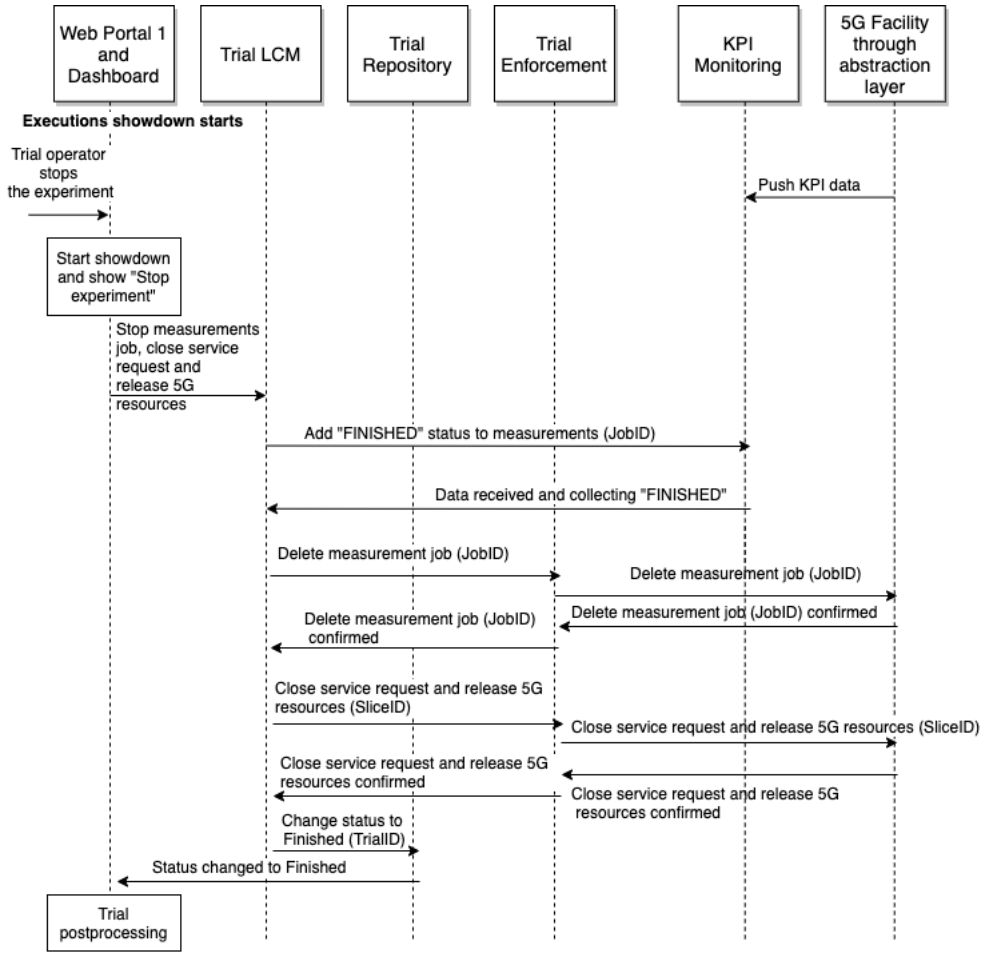

Figure 10. Service decommissioning.

2.2.4. Flight Plan rejected

It might happen that after submission of the operational flight plan for validation, it will be rejected by the UTM. In principle, it would require the adjustment of the operational flight plan according to the remarks provided by the UTM. After the adjustment is made, it should be re-submitted for validation to the UTM. It can take few iterations to get the final acceptance – the process should be repeated until it is successfully completed. The corresponding process is shown in Figure 11.

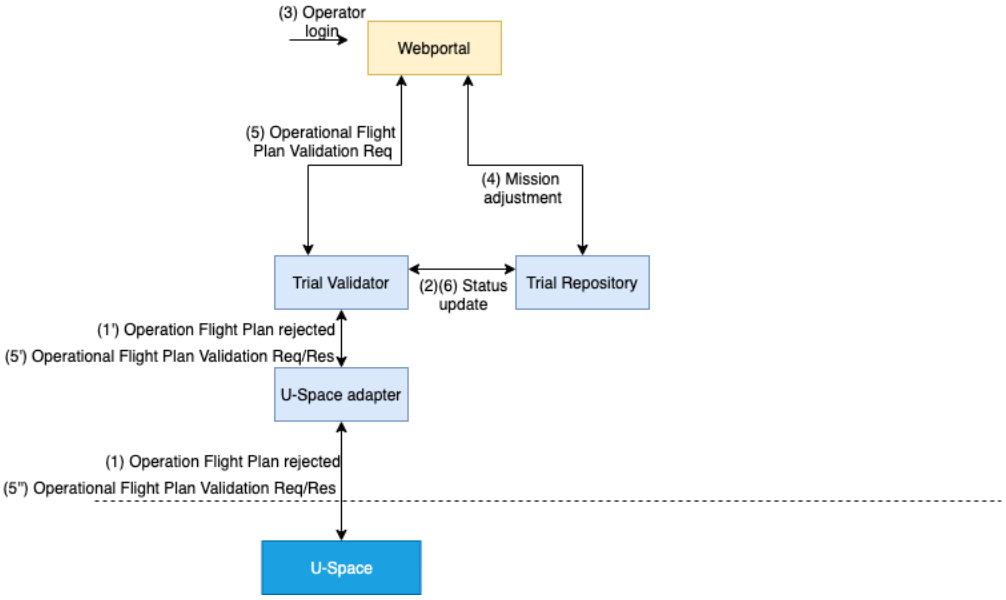

Figure 11. Operational flight plan rejection.

The U-Space adapter can check the status of the submitted operation flight plan, and whenever it receives the status update – rejection (1), it passes the response to the Trial Validator (1'). The Trial Validator updates the Trial status accordingly in the Trial Repository (2). When the operator logs into the Web portal (3), he can fetch the rejection response from the Trial Repository and, based on this, update the operational flight plan (4). After the operational flight plan is updated, it can be submitted for validation through the Trial Validator (5, 5', 5") – the cycle is repeated until the operation flight plan is accepted and at this final stage, the Trial Validator updates the Trial status accordingly in the Trial Repository (6).

3. 5G!DRONES COMPONENTS (ARCHITECTURE, MECHANISMS AND INTERFACES)

3.1. Web Portal 1

Web Portal 1 is an entry point to the Trial Controller. The experimenter interacts through it with the system. However, the drone mission is not controlled through Web Portal 1 – this is done by Ground Control Station. There is a plan to allow a preview of the active mission in the detailed Dashboard view, “as it’s seen and configured” by a Pilot. This will rely on embedding the GCS view into a detailed Dashboard.

3.1.1. Components and functions

To login to the Web Portal, the user needs to enter their credentials, which are verified by Identity and Access Manager (IAM). In our implementation it is based on an open source software called Keycloak (see [2] for more details). After successful authentication, the experimenter gets access to the graphical user interface. The main components are:

- Basic Dashboard – it lists the experiments with basic information, like name, planned date, their status. In this place it is possible to create or modify a plan, verify its status, access the Web Portal 2 to configure the facility setup (through NST), launch the manual check list before the trial execution.
- UAV module – used to define new UAVs and browse existing ones, which are used in the experiments. Parameters required for UTM service validation are mandatory; others are optional.
- Operational Flight Plan module – to define the trial parameters, including date and time, UAV operator, pilots, used facility, used UAVs, the geographical area where tests will be conducted and other parameters. The minimum set of parameters required by UTM service is mandatory, additional parameters, which can be useful for the experimenter, are optional.
- Detailed Dashboard is used to display detailed information about the experiment. During the ongoing live test, it will show the streaming of GCS application as seen by the pilot and indicating the actual status of the flight. In addition, the Dashboard allows to watch screen streams from various applications used by drone operators: video stream, map application view (s), and more. After the completion, it will allow to see the statistics and KPIs, browse the results and logs, and download data for further analysis and replay the screen video to analyse different situations in the past. One of the screen views of the Dashboard is described in Figure 12 below. The top left screen is a map application to understand the environment where the drone is flying. The lower left screen is the GCS screen, which shows the details of the drone's flight route. The upper right screen is the video stream or photo drone. The lower right screen is Web portal2.

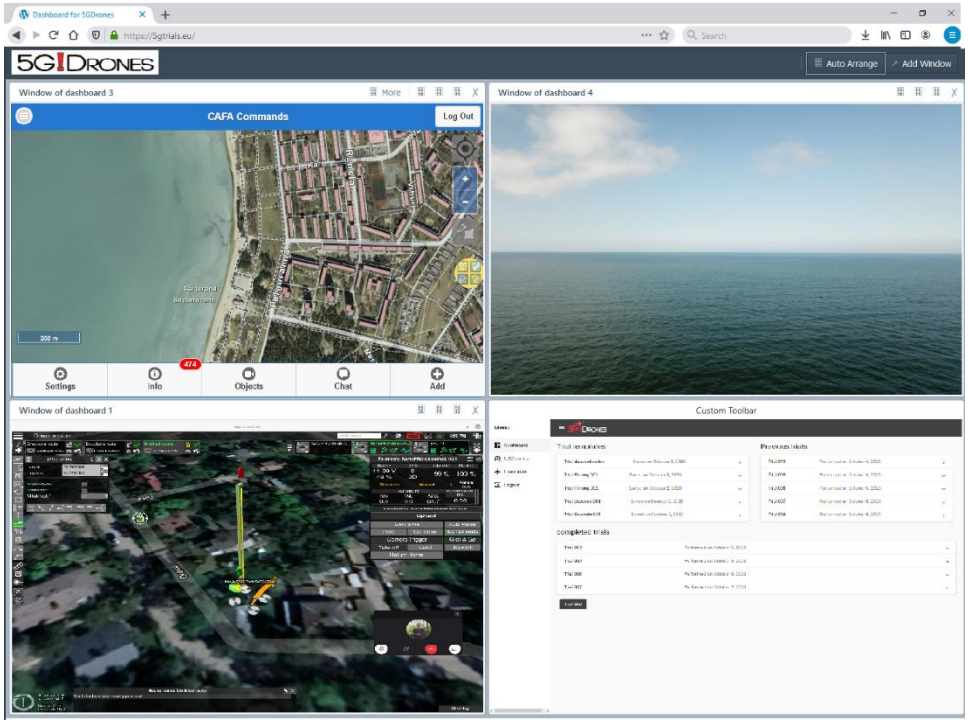

Figure 12. Screen views of the Dashboard vers01 in March 2021.

- Templates are used to facilitate the plan creation and avoid mistakes – they should be used if we want to define multiple tests with identical subset of parameters
- User Information is a module where users can manage their personal data and change their password

Trial Validator status visualisation for a particular plan is also hosted in the Web Portal 1. It is possible to see the status of a plan, from the moment of validation request until start of the experiment.

3.1.2. API

Web Portal is not exposing API functions, but consumes APIs provided by other components: Trial Repository, IAM, Dashboard and Trial KPI Component.

The most complex and demanding part of APIs definition was the Operational Flight Planning. When defining the APIs for it, we had to follow the requirements defined in the European Commission draft document on a regulatory framework for the U-space. Article 6 (4) is stating that:

"Before each individual flight, the UAS operator shall submit an UAS flight authorisation request to its U-space service provider, through the UAS flight authorisation service referred to in Article 10, in compliance with Annex IV."

The mentioned above Annex IV is specifying the information elements, which are mandatory for U-Space to analyse and issue the UAS flight authorisation. Part of them is UAV specific and another part is describing what is planned to be done (in space and time) and by who (UAV Operator/Pilot).

After the careful analysis and consultations with subject matter experts, we have identified the mandatory parameters. Table 1 and Table 2 contain the parameters required in the plan, submitted to U-space service provider.

Table 1. UAV mandatory parameters.

Call Sign
Class
Max weight (g)
Endurance (min)
Max Horizontal Speed (m/s)
Serial Number
UAS Operator Registration
Emergency Procedures
Tracker Type
Tracker ID
Camera

Table 2. Operation Flight Plan mandatory parameters.

Start time
End time
Mode of operation
Category of operation
Type of flight
Connectivity type
Principal UAS Operator registration number
Geoinformation
Call Sign (drone name)
Pilot in command name
Pilot in command license

It should be noted, that the Operational Flight Plan can include more than one UAV owned by different UAS Operators – for that reason we have a field called “Principal UAS Operator” in Operation Flight Plan, and it’s possible optionally to add more “secondary” UAS Operators to the plan.

Type of flight can be: Regular operation, Special operation.

Mode of operation can be: Indoor, VLOS, EVLOS, BVLOS, IFR, VFR.

Additionally, we have defined a number of other fields, which are optional to specify (not essential for U-space service provider to issue flight authorisation), but they can be important for experimenter for his trial.

3.1.3. Workflows

The workflows below show the interaction of the Web Portal 1 with other components. They are Web Portal 1 centric and may not reflect all the messages exchange between other nodes presented in the flow.

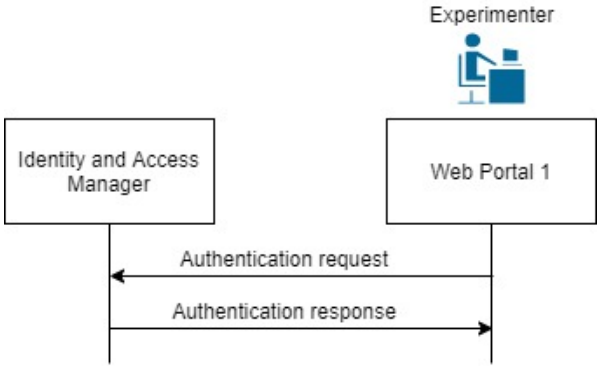

Figure 13. User authentication.

Figure 13 above shows the process of accessing the Web Portal 1 and Trial Controller by the experimenter. IAM module is verifying the user credentials and allowing or denying the access.

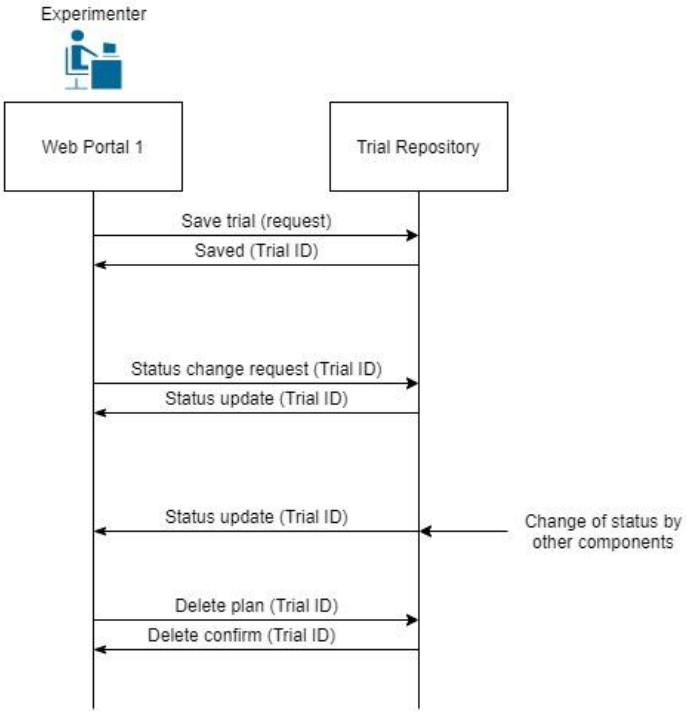

Figure 14. Trial repository – Web Portal updates.

Figure 14 shows the Web Portal 1 – Trial Repository relation and messages exchange. User can save the trial to Trial Repository at any stage of plan creation – the minimum condition is to give the plan a name. When all mandatory fields are given and a plan is saved to the Trial Repository, a user can submit this plan for validation in U-Space.

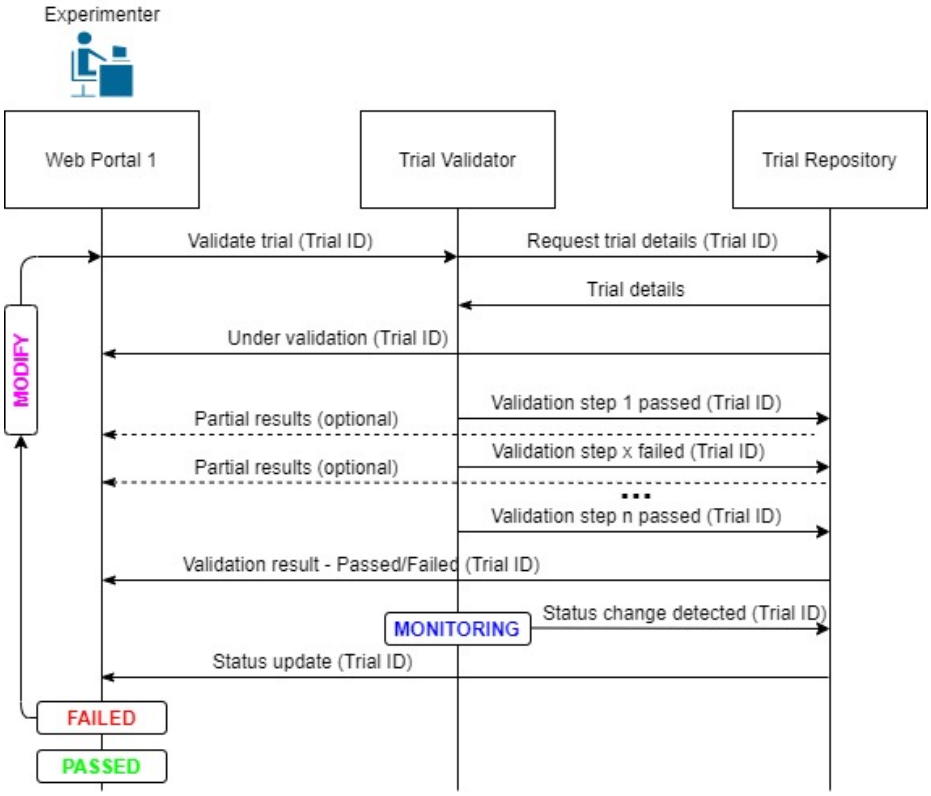

Figure 15. Web Portal 1 - Trial Validator.

Figure 15 shows the role of Trial Validator. A plan with all defined mandatory parameters can be sent to Trial Validator for further processing. Trial Validator sends the partial status updates (option) or final result to Trial Repository, which updates the status to Web Portal 1. If the result is negative, the experimenter needs to make corrections according to indications given in the failure message.

Trial Validator keeps monitoring the plan until execution start time to detect the changes, which can prevent its implementation or require modifications.

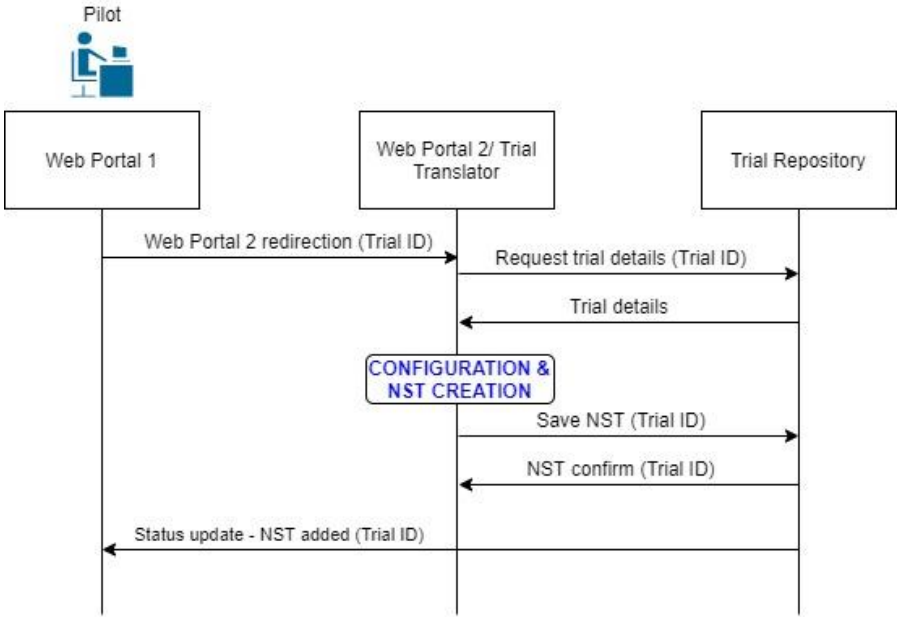

Figure 16. Web Portal 1- Web Portal 2 flow.

Figure 16 - when the experimenter wants to configure the 5G facility and create the NST, he is redirected to Web Portal 2, specific for each facility. When the result is saved in Trial Repository, Web Portal 1 is updated about successful NST creation.

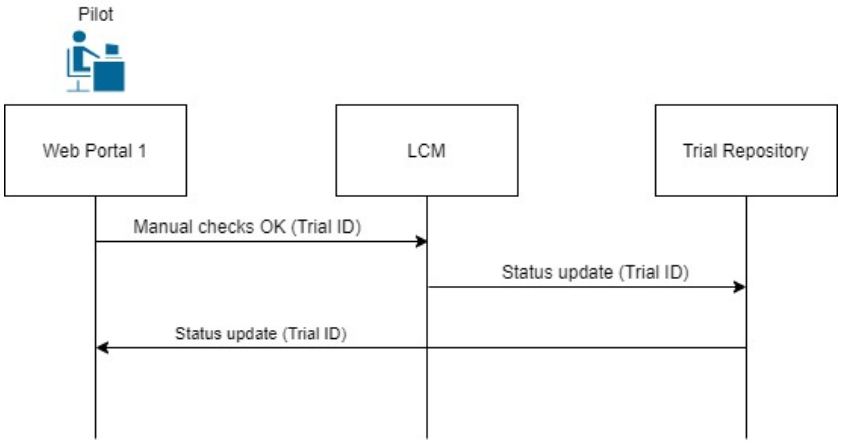

Figure 17. Web Portal 1 – LCM.

Figure 17 - in the trial's due time, the Pilot is verifying and confirming the check's list. When all checks are positive, he can start the trial execution – the message "Manual checks OK" with Trial_ID is sent to LCM.

3.2. Trial Translator

3.2.1. Components and functions

As indicated earlier, the Trial Translator uses a web portal to ease the creation of NST needed by the facility to create and instantiate the network slice that will run the trial. The translator maps the trial requirements in terms of computation and networking entered in the web portal 1 by the trial owner to information understandable by the 5G facility in a specific format, i.e., NST. The NST includes attributes and meta-data on the network slice (e.g., the start date and end date, slice owner, type of slice, etc.) and information on each sub-slice composing the network slice, i.e., RAN sub-slice, Edge/Cloud sub-slice, transport sub slice. It should be noted that no standard defines the NST format and attributes, but examples of Generic Slice Template (GST) are provided by GSMA in [3]. Accordingly, each facility has its own Trial Translator (Web portal 2).

The Trial Translator comprises (1) a front-end that interacts with the trial owner to automatically generate the NST from the information entered. It is called by Web portal 1 through an HTTP redirection along with Trial ID; (2) a back-end that sends the NST along with the Trial ID to the Trial Repository using an API call.

3.2.2. API

The Trial Translator has no API, but rather interfaces. A Web Graphical User Interface (GUI) allows the Trial Owner to define NST to be used to run the trial on a 5G facility after a HTTP redirection from Web portal 1. Table 3 highlight the exposed API by the Trial Translator.

Table 3. API exposed by the Trial Translator.

Exposed API	Component	Mandatory parameter(s)
Web GUI	Trial Owner	Information needed to generate the facility NST
HTTP redirection	Web portal	Trial ID

3.2.3. Workflows

The Trial Translator is stateless, which means that it is called only the time needed to generate the NST. The workflow to generate the NST is shown in Figure 18. The Web portal redirects the trial owner to Trial Translator using a HTTP redirection with the Trial ID as a parameter. The Trial Owner fills the forms required to generate the facility NST. Then, the NST is pushed to the Trial Repository using the Trial ID, therefore updating the trial's entry with the generated NST to be fetched later by the LCM to implement the trial.

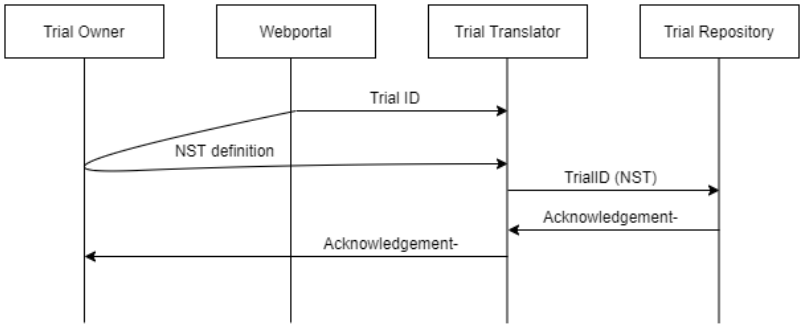

Figure 18. NST definition and creation.

3.3. Trial Repository

3.3.1. Components and functions

The trial repository (also referred as repositories' module) is a module which is used to store different information about the trials to be performed. It exposes APIs used by different modules including the web portal, the trial translator, the trial validator, and the LCM module. At the trial creation phase, the web portal uses the interfaces of the trial repository to create and store a new trial, which includes the definition of the operational plan of the flight, in addition to other information such as the target facility, the start date, end date, etc. As specified in subsection 3.1, the operation plan covers UAV-related information of the trial such as the target UAVs, the geolocation of the operation, the associated pilot, etc. The trial translator, which is a facility-dependent component, will also use the API exposed by the trial repository to insert and update the NST. Furthermore, the interfaces provided by the trial repository allow the update of the trial status by the trial validator and the retirement of this information by the LCM module. The database schema of the trial repository is depicted in Figure 19.

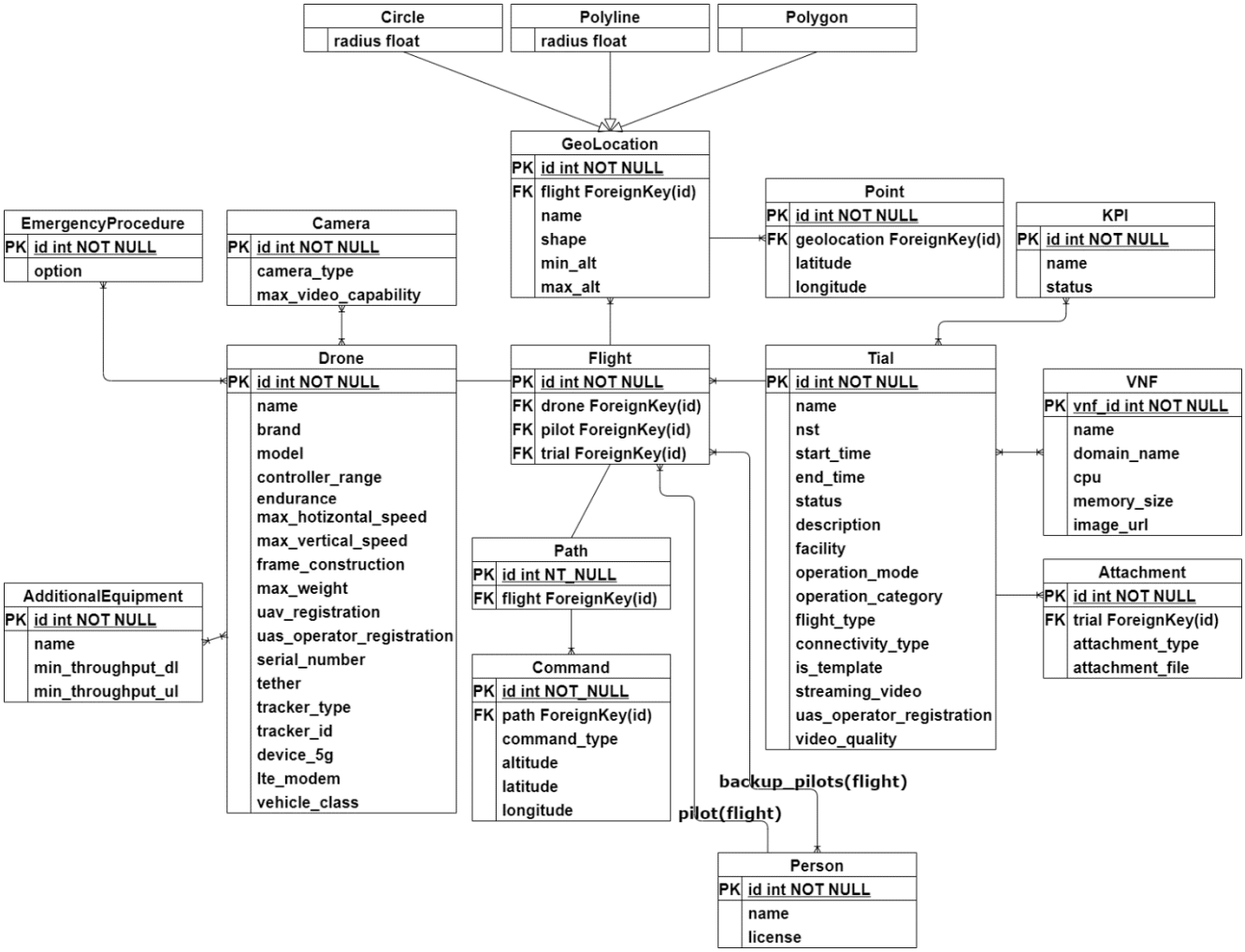

Figure 19. Database schema of the trial repository.

As described in D2.2 [4], the trial repository includes two components (services) which are:

- Database service,
- Web service.

The database service stores the different information and operates relational database (postgreSQL). As for the webservice, it uses the database and exposes the possible operations as APIs. The latter are detailed in D2.2. The module exploiting these interfaces are described in the next subsection.

3.3.2. API

The trial repository module exposes a number of APIs which are used by different components, as summarised in Table 4.

Table 4. APIs exposed by the Trial Repository.

Exposed API	Component	Mandatory parameter(s)
Trial Repository (HTTP) - (CRUD) create/read/update/delete trial	Web portal1	- Trial ID (the trial id will be returned during the creation phase)
Trial Repository (HTTP) - Update NST	Trial translator	- Trial ID
Trial Repository (HTTP) - Retrieve trial information	LCM	- Trial ID
Trial Repository (HTTP) - Update trial status	Trial validator	- Trial ID

3.3.3. Workflows

The trial repository is a central point that stores information required by different modules. It therefore appears in the workflows of the web portal, the trial translator, the trial validator, and the LCM. The interaction between the database and the webservice is depicted in Figure 20. As described earlier, the webservice exposes APIs allowing to perform operations on the database. First, the webservice connects to the database prior to any request. This is performed using port 5432 in case of PostgreSQL. Upon the establishment of the connection, the webservice can expose the APIs to receive the HTTP requests. Each received HTTP request will be mapped to the corresponding format of the database and communicated to the latter. In the other hand, once a database response is received, the webservice maps it back to an HTTP response, so it can be returned via the corresponding API.

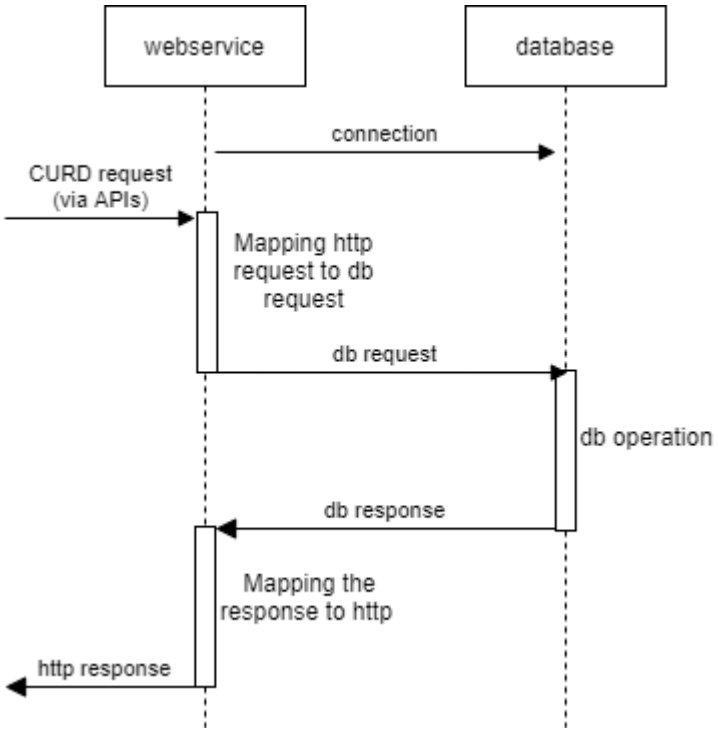

Figure 20. Interaction between trial repository internal components.

3.4. Trial Enforcement

3.4.1. Components and functions

Trial Enforcement (TE) is the module responsible for the execution of the trials as well as monitoring for the trial status in 5G facility. The Trial Enforcement uses the Abstraction layer in order to enforce requests from the Life Cycle Manager and essentially manage and monitor the life cycle of the Network Slice running the trial.

The TE runs on a docker-based container and has been developed in Python, powered by the FastAPI [5] framework. This module has a series of OpenAPI compliant REST APIs that enables the LCM to manage the Network Slices that are needed to be deployed during the trials. It has to be noted that the Trial Enforcement doesn't possess any initiative on its own and only acts after a direct request from the LCM, which then processes it and communicates with the underlying infrastructure, most notably the Abstraction Layer, in order to perform the aforementioned tasks.

Trial Enforcement provides REST APIs so that it can expose access to its functionalities. This includes Network Slice management options such as NS deployment, activation, deactivation, NS status, NS modification and finally termination. Furthermore, depending on the facility, it can provide access to 5G resource testing. Additionally, the Trial Enforcement functions include creating or deleting Measurement Jobs that are needed for the KPI collection.

3.4.2. API

A RESTful API has been developed on the Trial Enforcement module in order for the LCM to be able to communicate with it and issue the necessary commands relevant for the network slice creation and management. The exposed APIs are shown in Table 5.

Table 5. APIs exposed by the Trial Enforcement module.

Exposed API	Component	Mandatory parameter(s)
Network Slice Management (HTTP)	LCM	Trial ID, User ID, Network Slice Descriptor
Measurement Jobs	LCM	Trial ID, User ID, Measurement Job Description

3.4.3. Workflows

Trial Enforcement only executes its functions after an LCM request as can be seen in Figure 21. During trial execution, the LCM will request the Trial Enforcement to deploy the NST provided earlier by the experimenter to the requested facility. Then, it is the module's job to contact the Abstraction Layer, which is registered in the start-up configuration file, and send the Slice instantiation request to it. When the Slice is successfully deployed and activated, it performs a call-back request to the LCM to inform it and provide the necessary information gathered. At this stage, the LCM can make NS management requests to the Trial Enforcement such as modify the NS, de-activate or delete it, while it can also request 5G resource testing if it is supported. Finally, the LCM can request the creation of a Measurement job, which the Trial Enforcement will forward to the Abstraction Layer and respond with the confirmation of this and the ID of the job.

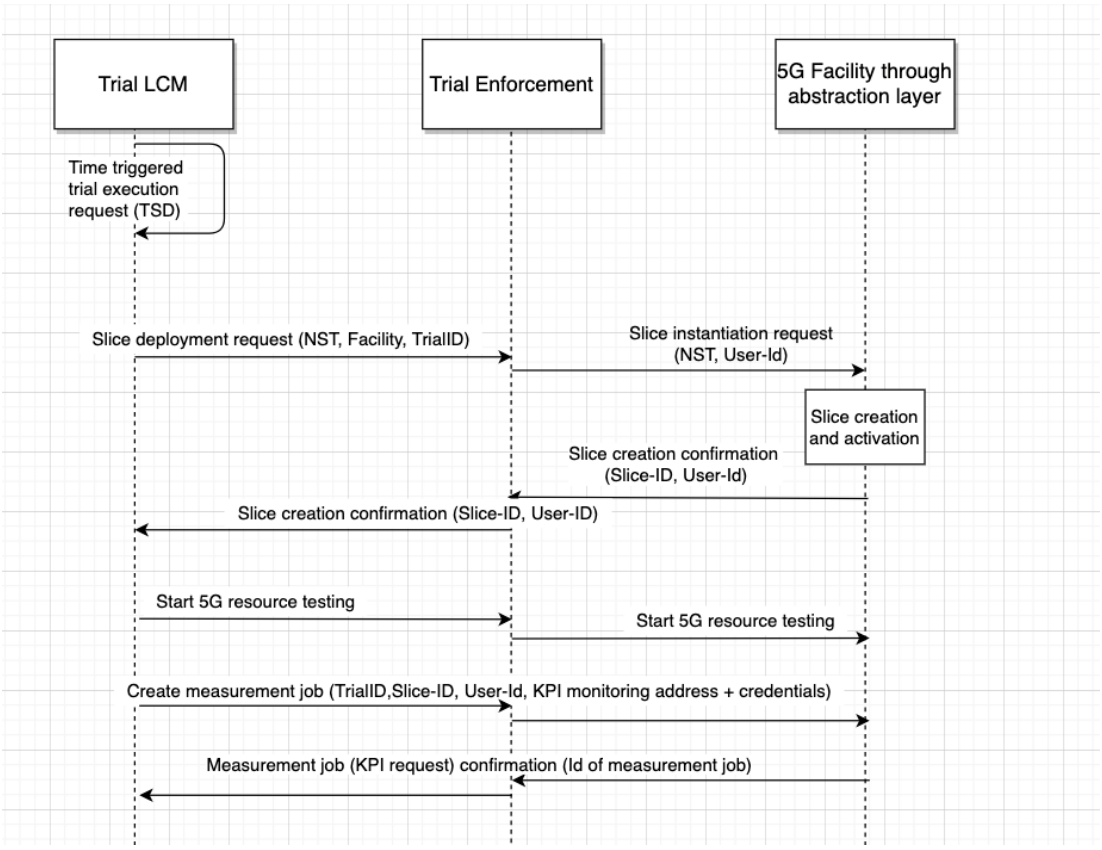

Figure 21. Trial Enforcement Workflow.

3.5. Trial Validator

Trial Validator provides the mechanisms to check and validate the Operational Flight Plan, facility resources and other elements important for conducting the experiment. The results are communicated to the experimenter in an understandable form, which should allow the corrections in case of negative verification. Trial Validator is also supposed to continuously monitor conditions for possibility of conducting the experiment since its positive validation until deployment.

3.5.1. Components and functions

Trial Validator consists of a frontend Web User Interface implemented in JavaScript as well as a Java based backend component. Figure 22 illustrates the Frontend Web User Interface of the Trial Validator.

The Trial Validator is accessing the Trial Repository to retrieve information about individual trials. This information is used to send a request to the UTM system to validate the flight plan and therefore reserve the airspace. The status of the flight plan as received from the UTM system will be displayed on the user interface as well as updated in the Trial Repository.

The Trial Validator does monitor the availability of the UTM system, the KPI component, the network slice (through Facility Abstraction Layer) and the Trial Repository. Shortly before the actual Trial flight starts, the Trial Validator gives the on-site personnel the possibility to check if all components that shall be reporting KPIs according to the information stored in the Trial Repository are already up and successfully reporting KPI data packets to the KPIC.
The Trial Validator is comparing expected KPIs (as retrieved from the Trial Repository) against the actual KPIs currently received by the KPI component. This is achieved by using the according KPIC interface. See Figure 23.

Figure 22. Frontend Web User Interface of the Trial Validator.

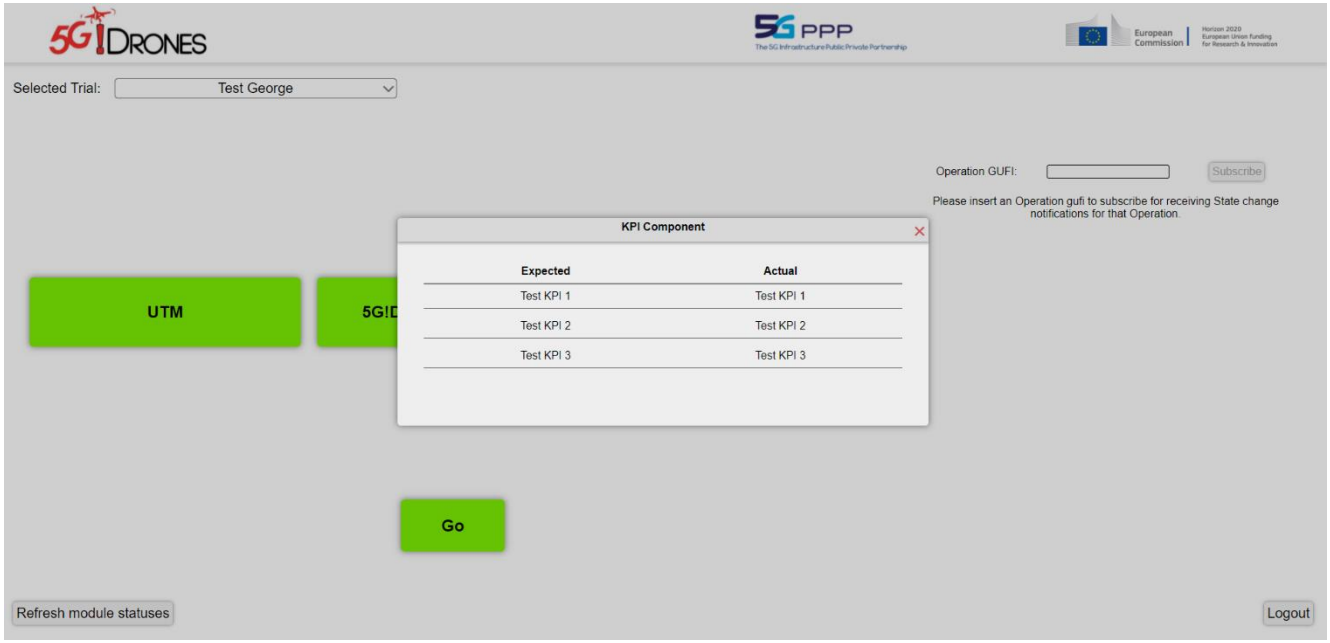

Figure 23. Trial Validator – KPI Comparison.

3.5.1. Trial Validator Frontend API

Trial Validator is not exposing API functions but instead makes use of provided APIs from other components like e.g., Trial Repository, KPI Component, Facility Abstraction and U-Space adapter.

3.5.2. Workflows

The sequence diagram depicted by Figure 24 shows the workflow for Trial Validator and its interaction with other components in the 5G!Drones ecosystem. In principle, the Trial Validator will hand over the monitoring responsibilities to Life Cycle Manager once the UAV takes off from ground.

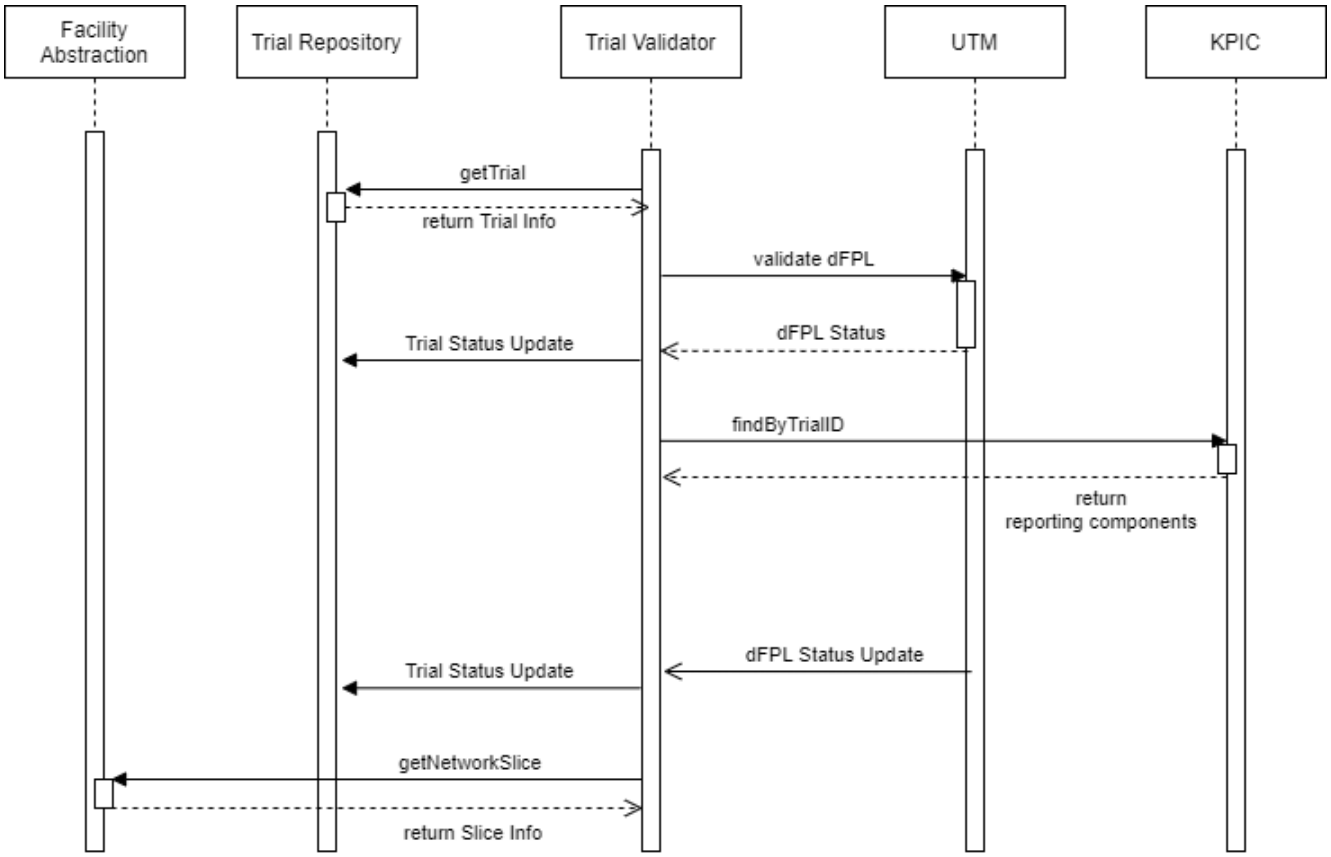

Figure 24. Workflow sequence diagram for the Trial Validator.

3.6. Life Cycle Manager

3.6.1. Components and functions

Lifecycle Manager (LCM) is a component that handles the high-level execution of a trial lifecycle. In order to perform the selected scenarios, LCM communicates with the other 5G!Drones components, sets up the required services, and runs required 5G!Drones components with associated UAV operators.

Lifecycle Manager performs scheduled jobs that trigger further activities, which comprise of automated sequences of requests for preparing trial resources and state updates during the preparation, flight execution, and post-trial phases. The sequences are described in section 2.2 *(Trial call flows)*. The required functionalities of LCM are presented in deliverable D1.6: *5G!Drones system architecture refined design* [6].

3.6.1.1. Architecture

The internal architecture of LCM consists of the following main components, as outlined by Figure 25: LCM API, Scheduler, and Execution Engine. Scheduler is responsible for scheduling the trials, which in turn trigger activities in Execution Engine.

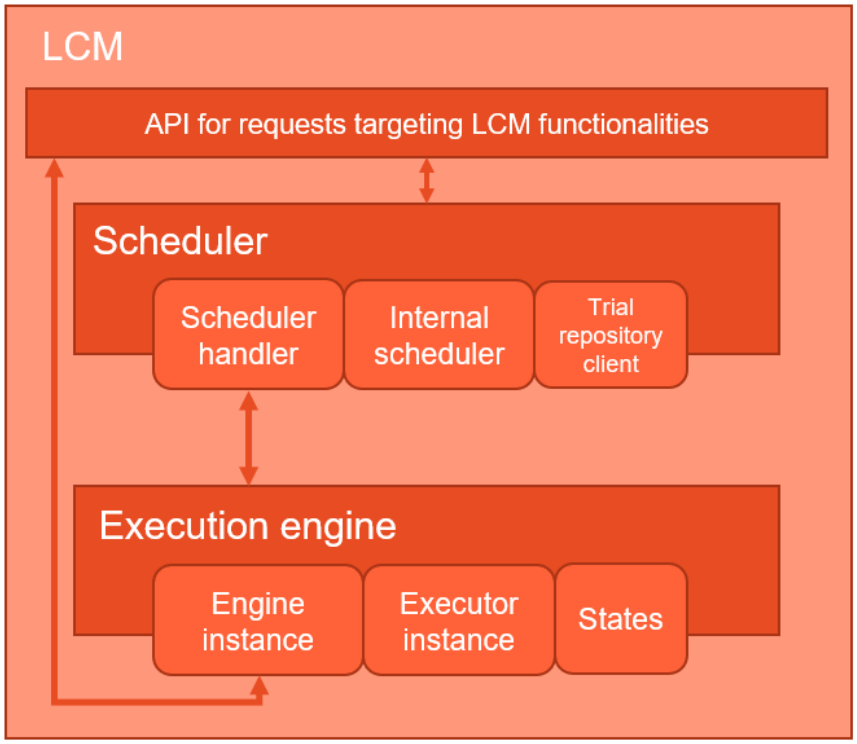

Figure 25. High-level architecture of the internal components of LCM.

3.6.1.2. Scheduler

Scheduler functions as the main component of LCM and is active alongside LCM API once an LCM service has been set up. Scheduler listens to the calls from the LCM API and performs various activities accordingly. The essential function of Scheduler is to manage the scheduling of trials, and to call Execution Engine once a trial start time has been reached to begin the actual activities required to be performed by LCM during trial preparation and execution phases. In addition, calls to LCM API that target restoration and deletion of Engine instances are routed via Scheduler.

Scheduler consists of three internal components: Scheduler Handler, Internal Scheduler, and Trial repository client. Scheduler Handler oversees the high-level execution of LCM, and forwards selected calls from LCM API to the correct target components. Internal Scheduler manages the scheduled jobs, and Trial repository client is utilised for retrieving the information required for scheduling trials from Trial Repository.

In Scheduler, trial scheduling is performed utilising scheduled jobs. The required information including trial start time is requested from Trial Repository by Scheduler, and the trial is scheduled according to the information. When reaching the set start time, Scheduler calls the Execution Engine component and initialises an Engine instance, in which the activities for setting up required services and managing the trial preparation, execution, and post-trial phases are performed. Via LCM API, a trial scheduling can be removed from Scheduler if necessary.

3.6.1.3. Execution Engine

Execution Engine refers to the set of one or multiple Engine instance complexes, each of which is responsible for managing a single trial. An Engine instance is created by a call from Scheduler once a start time for a specific trial has been reached.

Engine instance complexes consist of three component instances: Engine, Executor, and States. Engine and Executor instances are responsible for managing the high-level execution of an Engine

complex. States instance includes the implementation of the communication and other functionalities required to perform the designed sequences for trial preparation, execution, and post-trial phases.

An Engine instance complex begins its execution associated with a trial by retrieving required trial information from Trial Repository. For trial preparation, an Engine instance communicates with Trial Enforcement to set up 5G services. To ensure the availability of KPI measurements for the trial, Trial Enforcement communicates with 5G facility through abstraction layer to create a measurement job as requested by LCM. Additionally, LCM communicates with KPI monitoring by requesting the KPI components.

After the setup of required services, LCM listens to requests from Web portal/Dashboard and Ground Control Station (VNF app), forwarding them to the relevant 5G!Drones components. The requests mainly target changing the status of trial execution. Once a trial has been executed, LCM forwards the request to close and release the 5G services to Trial Enforcement.

In case of facing a failure during execution, an Engine instance complex can be restored via LCM API. Such situation may occur if an Engine instance complex fails to communicate with another 5G!Drones component. Engine instance complex backs up the state of the Engine instance complex regularly, and the information is used to restore and resume the execution from the latest checkpoint.

3.6.2. API

LCM API exposes several endpoints that target various functionalities of LCM, as shown in Table 6. The API includes endpoints for controlling the internal components of LCM: scheduling a trial, removal of a scheduling, and restoration and removal of Engine instance complexes. The rest of the endpoints are utilised for the activities performed in the defined sequences during trial execution.

Table 6. APIs exposed by the LCM.

Exposed API	Component	Mandatory parameter(s)
LCM API (HTTP) - Trial scheduling	Web portal/Dashboard	Trial ID
LCM API (HTTP) - Setup of 5G services	Trial Enforcement	Trial ID, slice ID, user ID
LCM API (HTTP) - Change of trial status	Web portal/Dashboard, Ground Control Station (VNF app)	Trial ID, status
LCM API (HTTP) - Closing and releasing 5G services	Web portal/Dashboard	Trial ID

3.6.3. Workflows

3.6.3.1. Trial scheduling

Scheduling of a trial can be triggered once the required information including trial ID and start time are available. Scheduling is performed via a call to LCM API, providing the trial ID to LCM. Within LCM, the

scheduling is executed by Scheduler, which retrieves the start time based on the provided trial ID. Once the start time has been retrieved, Scheduler creates a scheduled job to trigger activities in LCM Execution Engine when the start time is due. Figure 26 depicts the sequence of scheduling of a trial on a high level within LCM.

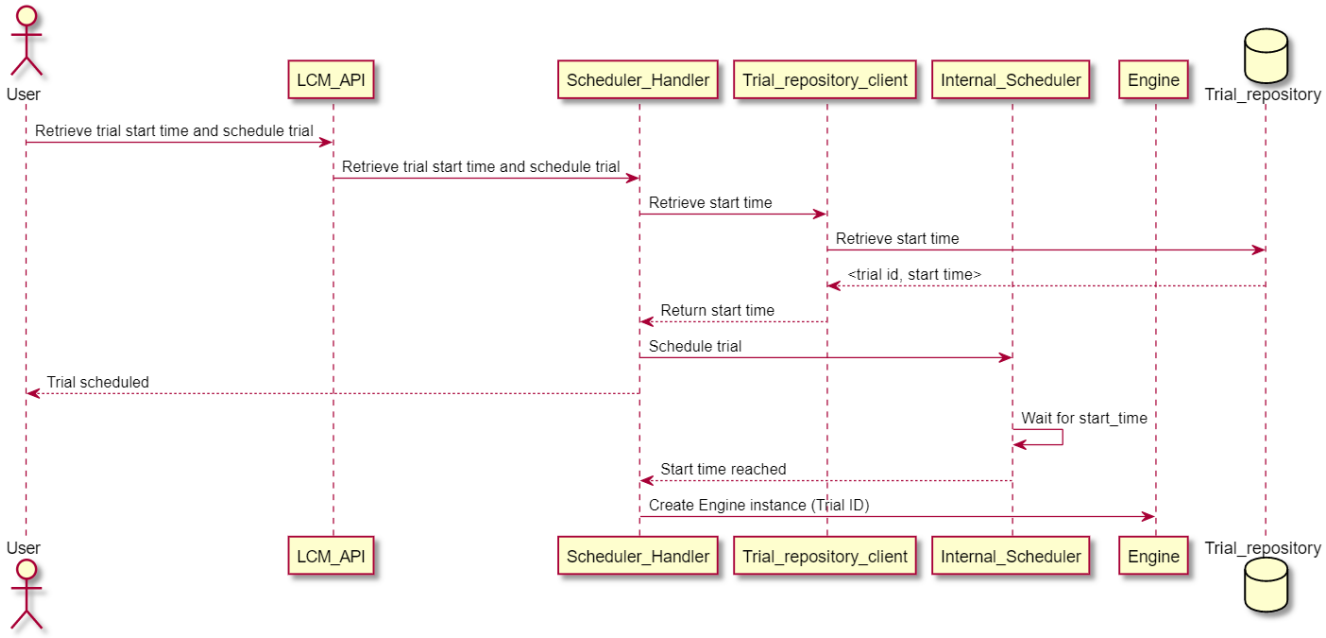

Figure 26. High-level sequence diagram for scheduling a trial in LCM.

3.6.3.2. Trial execution

Activities in LCM Execution Engine begin when a start time for a trial has been reached. A job scheduled by LCM Scheduler calls Execution Engine and initialises an Engine instance complex, triggering the sequences required for preparing and executing a trial. Figure 27 shows a general sequence of an Engine instance complex execution lifecycle.

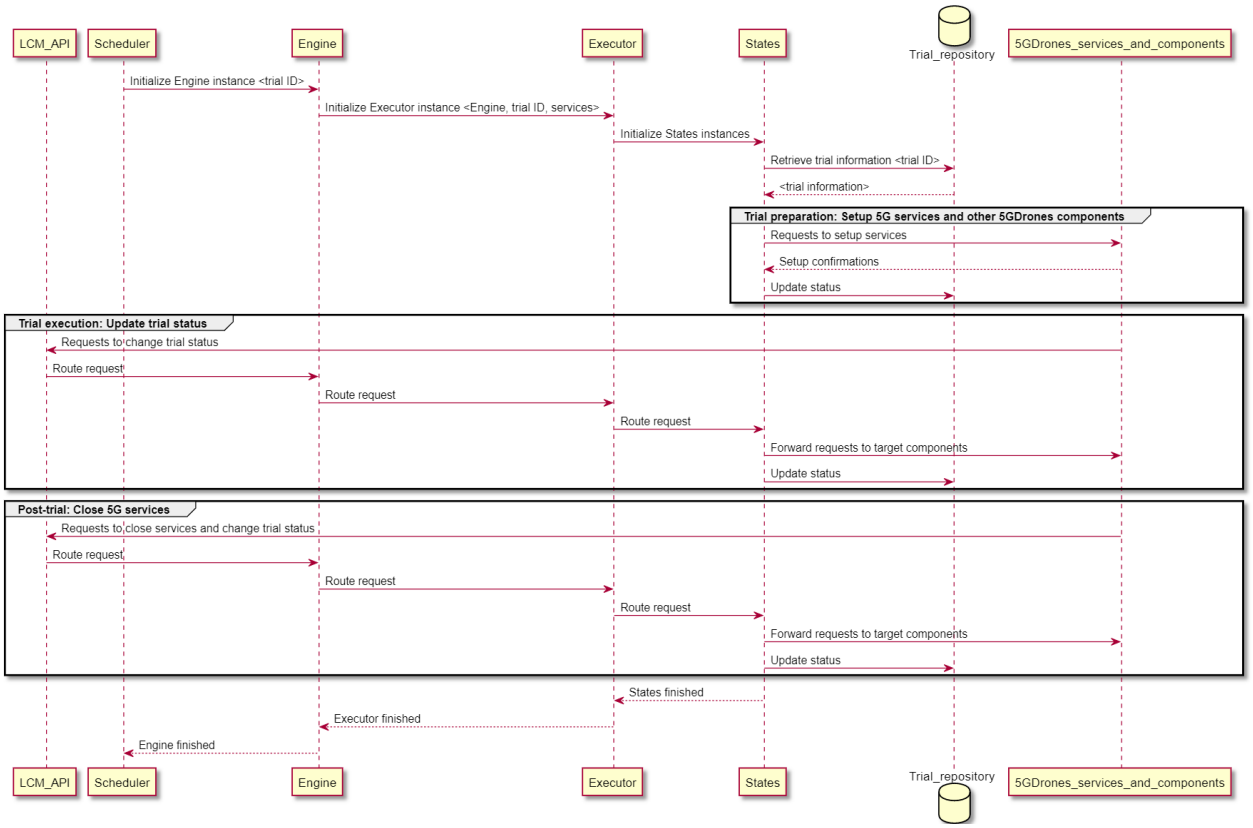

Figure 27. High-level execution lifecycle of an Engine instance complex in LCM.

3.6.4. Towards autonomous Life Cycle Management of trials

This section elaborates on enabling autonomous management of trials. Indeed, the LCM is the module responsible for handling and instructing the operations to be executed on top of the target facilities. This can range from instructing the creation of network slices, onboarding additional VNFs, requesting scaling-up/down resources, requesting the decommission of network slices, etc. This also raises questions around what actions to be selected by the LCM and its level of autonomy in selecting such actions. When it comes to automation, three levels can be distinguished which are:

- Manual: it includes no automation. It therefore requires human intervention to execute the actions.
- Automatic: it allows the automatic executions of actions based on predefined and stored behaviours.
- Autonomous: the executed actions are not predefined. This involves the use of Machine Learning (ML).

The LCM described earlier performs automatic behavior. The actions requested by the LCM do not require human intervention, but they are predefined. In order to pave the way to more advanced management that considers the dynamicity of the environment and the scalability of the trials, it is highly important to elaborate on the autonomous behavior of the LCM.

3.6.4.1. Enabling autonomous management via policies and machine learning

We introduce policy models allowing to express automatic behaviour in performing the management operations of the LCM. This also aims to pave the way to more autonomous behaviour where the LCM can learn the optimal policies to apply.

An LCM policy is defined so some actions are performed when the corresponding conditions are met. It is therefore defined as an association between conditions and actions. We thus propose the format depicted in Figure 28 to model an LCM policy. Here, the field “interval” reflects the time window to periodically check the validity of the conditions and can be adjusted depending the importance of the conditions. The field “conditions” holds the conditions’ part. A generic definition of a condition includes the target object (e.g., the target VNF), the measurement of interest (e.g., CPU), and the range of values, as depicted in Figure 28. As for the field “actions”, it specifies the actions to apply when the conditions are met. An action is related to a VNF and a network slice (a resource) and is defined by the operation to perform (e.g., VNF migration, VNF onboarding) and the associated fields. Note that Figure 28, also includes an example of the policy model format.

Figure 28. Proposed format of a policy model.

As mentioned earlier, the actions are related to VNFs and network slices. We summarise the current envisaged actions in Table 7. This table provides the possible combinations of operations related to NSs and VNFs. The operations related to the former are creation and decommissioning of network slices. As for VNFs, the possible operations are onboarding, offboarding, resource scaling and migration. Without losing in generality, this can be extended to cover more type of actions (e.g., slice migration).

Table 7. Possible combinations of actions.

Resource	Operations					
	Onboard	Offboard	Re-dimension (scale-up/scale-down)	Relocate	Create	Delete
Slice					X	X
VNF	X	X				
VNF-loc				X		
VNF-cpu			X			
VNF-ram			X			

The validity of LCM policy's conditions is checked based on monitored resources from the facility. Indeed, the KPI component of the trial controller allows the collection and the storage of measurements captured from the facilities. This module can also be extended to accommodate conditions and report when they are met. The proposed format for this model (watcher) is depicted in Figure 29. Here, the "interval" and the "conditions" fields correspond to the same field described in the policy model. As for the authentication field, it provides the information which will be used to communicate the validity of the conditions.

Figure 29. Proposed format for a watcher model.

Based on this model, the LCM can be more reactive and perform dynamic actions based on the monitored data captured from the facility. This is possible thanks to the consideration of the policy model. It can happen that conditions corresponding to different policies are met in the same time. In such a situation, two approaches can be distinguished which are:

i) applying the actions corresponding to all the policies whose conditions are met,

ii) considering a strategy to only target some policies out of those whose conditions are met.

In the next subsection, we consider a study case where the second approach is considered to one policy out of a set of policies whose conditions are met, in a way to enhance a target quality of service.

Figure 30 summarises the link between different components used to enable the policy-based model. The policy repository is used to store the different policies of the LCM. The resources monitoring component can be part of the KPI component of the trial controller and is mainly used to accommodate conditions and notify when they are met. As for the policy engine, it retrieves the policies from the policy repository, communicate their conditions to the resources monitoring, and introduces a logic in applying their actions when the corresponding conditions are met. Both the policy repository and the policy engine can reside in the LCM module.

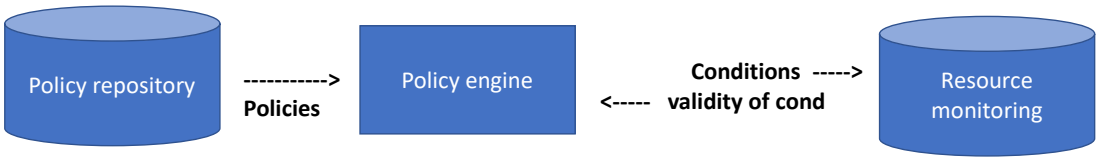

Figure 30. Link between Policy repository, policy engine and resource monitoring.

The consideration of the policy-based model introduces reactive and more dynamic management of the trials' lifecycles. However, it still provides automatic behaviour. The autonomous management can be enabled by using machine learning techniques. In this case, the policy engine will be empowered to take autonomous decisions on the policies to apply. This autonomy covers two aspects which are:

i) the selection of the policies to apply,

ii) the selection of the parameters of the corresponding actions of the chosen policies (e.g., the target MEC in case of VNF migration).

In order to better emphasise with the autonomous management of trials/experiments using machine learning, we consider a study case which is presented in the following subsection.

3.6.4.2. A study case

We have considered a study case for performing autonomous management of trials lifecycles. In this study case, a set of UAVs are being served by a facility to communicate to their software pilots deployed at distributed edge servers. The facility provides the possibility to apply different operations, among others the migration of the VNF software pilots to follow the mobility of the drones, and also to scale their resources. These operations can be triggered using the corresponding interfaces provided by the facility. The latter (the facility) also ensures the possibility to monitor the resources.

In this study case, the LCM gets three policies whose conditions are verified (which are migration, scaling-up and scaling-down of VNFs) and needs to autonomously develop a strategy to select the one associated with lowest downtime. Given the dynamicity of the environment (in terms of resources, configuration and deployed applications), this problem is complex, especially for a large network. In this regard, we developed a solution based on Deep Reinforcement Learning (DRL) to learn complex tasks and effectively takes decisions through the interaction with the environment based on trial and error processes. More precisely, a Reinforcement Learning (RL) agent interacts periodically with an environment, observes the current state S_t, then executes an action A_t, Subsequently, the agent will observe a new state S_{t+1} and receives a corresponding reward R_t. Unlike supervised and unsupervised ML algorithms, RL techniques do not require prior data set.

Table 8. Table of notations.

Notation	Description
$\mathcal{M}$	Set of MEC
$\mathcal{V}$	Set of VNFs
$\mathcal{P}$	Set of policies
C_m	Percentage of free CPU amount for the MEC $m \in \mathcal{M}$
R_m	Percentage of free RAM amount for the MEC $m \in \mathcal{M}$
D_m	Percentage of free Disk amount for the MEC $m \in \mathcal{M}$
c_v	CPU usage of the VNF $v \in \mathcal{V}$
r_v	RAM usage of the VNF $v \in \mathcal{V}$
x_v	MEC id where the VNF v is running
S_t	State of the system captured at the time t
A_t	The system action executed at the time t
R_t	The system reward at the time t

Let us respectively denote by $\mathcal{V}$ and $\mathcal{M}$ the set of VNFs and MECs. More notations are provided in Table 8. The proposed DRL model is defined by several elements which are the following:

State space: the system state is defined in a way to capture the feature of the current deployment. To this end, we consider the information related to the hosting servers and the deployed VNFs in defining the system state. More precisely, at a time step t, a state S_t is defined as

$$S_t = (C_m, R_m, D_m, c_v, r_v, x_v)_{m,v},$$

Where C_m, R_m, D_m, c_v, r_v, and x_v are features related to the state of the MEC, the state of the VNFs, as well as the location where these VNFs are running, as described in Table 8. As we can see, the state space therefore increases with the number of MEC as well as the number of VNFs. Note that the sate space is discrete.

Action space: the application of an action allows the transition from one state to another. In this study case, the action space is related to the LCM policies and are centered on resource scaling and migration of VNFs, as depicted in Table 9.

Table 9. The policies considered in the study case.

Policy actions	Description
VNF-Scaling-up	Scaling up RAM and CPU resource of a given VNF
VNF-Scaling-down	Scaling down RAM and CPU resource of a given VNF
VNF-migration	Migrate a VNF from the current edge host to a target edge host

As detailed in the previous section, the actions of a policy are characterised by the target resource and the associated fields (e.g., in case of migration, the resource is the VNF and the associated field is the target MEC). The action space is therefore related to the LCM policies and their associated fields, which makes it very huge. At a time step t, an action A_t is defined as

$$A_t = p_{v,a,f},$$

where $p_{v,a,f} \in \mathcal{P}$ is the selected LCM policy which specifies the target VNF $v \in \mathcal{V}$, the action to perform a (out of migration, scaling-up and scaling-down) as well as the associated field f. The agent explores various actions taken from the state space and learns from their associated reward. The proposed reward function is defined in the following.

Reward function: the goal is to select the policy associated with the lowest downtime. The reward function is therefore defined considering the execution time of the selected operation as

$$R_t = \begin{cases} \frac{1}{\mathcal{T}_t} \; if \; action \; is \; feasible \\ -\alpha_t \qquad\qquad otherwise, \end{cases}$$

where $\mathcal{T}_t$ correspond to the execution time of the selected operation, and α_t is a function used to penalise the reward when infeasible actions are decided.

DRL algorithm: in this study case, we consider two DRL algorithms, which are DQN and Advantage Actor Critic (A2C).

i) Deep Q-Networks: DQN relies on replaying experiences to break the correlation between subsequent time-steps and ensure a stable *learning*. In each batch size, DQN calculates the Temporal Difference (TD) error by taking the difference between Q-targets (which is the maximum value that can be captured from the next states) and the predicted Q-values.

ii) Advantage Actor-Critic: A2C allows to cope with the constraints related to the growth of action space. It has two main networks, mainly the Actor and the Critic networks. The Actor observes the environment and selects a given action by outputting a probability distribution across the action space. After that, the Critic evaluates the quality of the selected action regarding both the current state and the next state.

Table 10. DRL algorithm for LCM policy selection.

Algorithm: Deciding LCM policies
Inputs:
$\mathcal{S}$: states/features
Outputs:
Policy to take p_t at each episode t
$t = 0$
Repeat
$t = t + 1$
Get current state $S_t = (C_m, R_m, D_m, c_v, r_v, x_v)_{m,v}$
Select an action $A_t = p_t$
Execute action and get R_t, S_{t+1}
Learn
Until $t = Eps_{max}$

The overall algorithm for selecting LCM policies is described in Table 10. The execution of the algorithm is performed through several episodes, until reaching a maximum value $Eps_{\max}$. At each episode, the agent, DQN agent or A2C agent, gets the current state S_t which characterises the actual state of the MECs and the deployed VNFs. Based on this state, the agent selects an action and outputs the policy

to apply. The agent will also get the reward value and the next state. The agent can thereafter learn considering these parameters and the error value which is computed depending on the agent type.

Numerical evaluation

In order to evaluate the RL-based LCM policy management, we have performed simulation using python. Pytorch [7] has been used to model and train the DNN. We have also considered a dynamic environment that is characterised by 5 MECs with different deployments that change over the episodes. More precisely, the RAM size of each MEC is between 64 – 128 G while the number of CPU cores is between 64 – 128. We have also considered a varying number VNF that correspond to the software pilot used to control the drones.

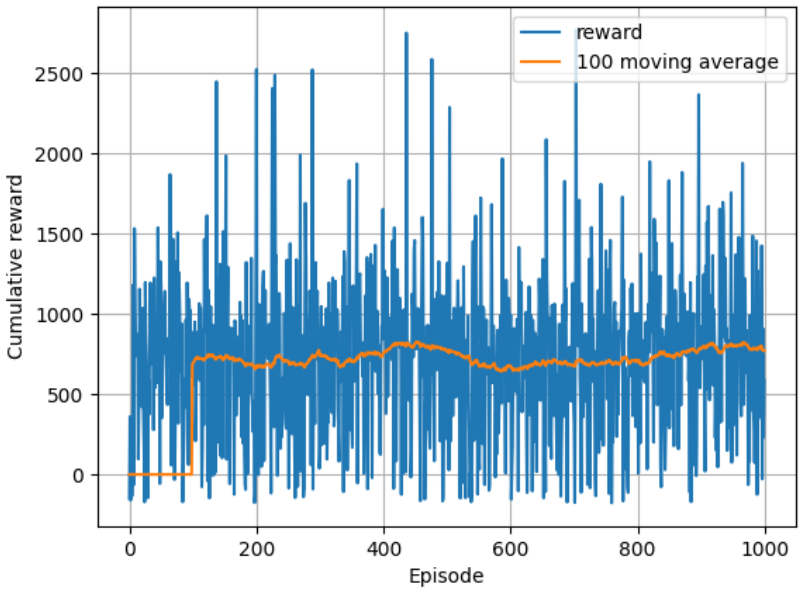

Figure 31. Evaluation of system reward (DQN algorithm).

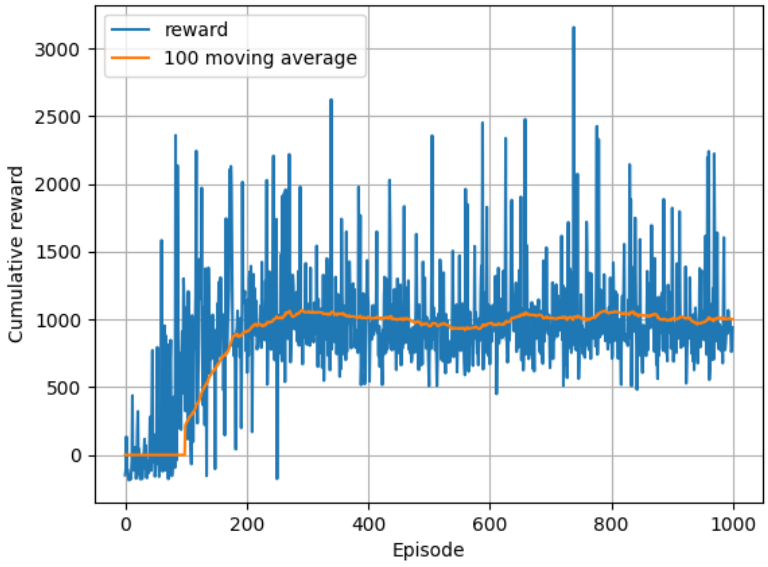

Figure 32. Evaluation of the system reward (A2C algorithm).

Figure 31 and Figure 32 show the evaluation of the reward function considering the DQN and the A2C algorithms, respectively. These evaluations have been performed throughout 1000 episodes. The two figures also include 100-episode average in an orange colour. Compared to the DQN algorithm, we can see that the A2C agent could learn to apply efficient policies throughout the different episodes. Indeed, the comparison of the 100-moving average of the cumulative reward shows that A2C achieves higher values. Furthermore, the minimal values of the cumulative rewards related to the A2C algorithm do not fall below 500 after a certain number of episodes at the opposite of that of the DQN algorithm. The latter shows a large fluctuation which can be translated into very poor decisions at any time. This is mainly due to the fact that A2C algorithm is more adapter to situations characterised by a large action space. This is translated into reduced downtime corresponding the chosen policies. This also demonstrates the effectiveness of the A2C agent in taking autonomous decision throughout the changing environment. We recall that the evaluation considers a changing environment at each episode, which requires from the agent to take adequate decision and construct the optimal policies.

3.7. Data Monitoring

The Key Performance Indicator Component (KPIC) provides an OpenAPI based Rest Interface to receive KPI datasets defined by the sending component. The KPIC is responsible to storing and providing the KPIs for further analysis. More details can be found in D2.3 [8]. Report on algorithms, mechanisms and tools for data analysis and visualisation.

3.7.1. Components and functions

The KPIC consists of 5 internal components, as illustrated by Figure 33. The KPI endpoint in the REST interface which receives requests and provides output according to the agreed API. The ActiveMQ component receives data messages from the KPI endpoint to queue and distribute them.
The KPI Service instance consumes the queued message and prepares it for persisting it into the ElasticSearch datastore.
From the datastore a Kibana instance can be used for analysis and reporting purposes.

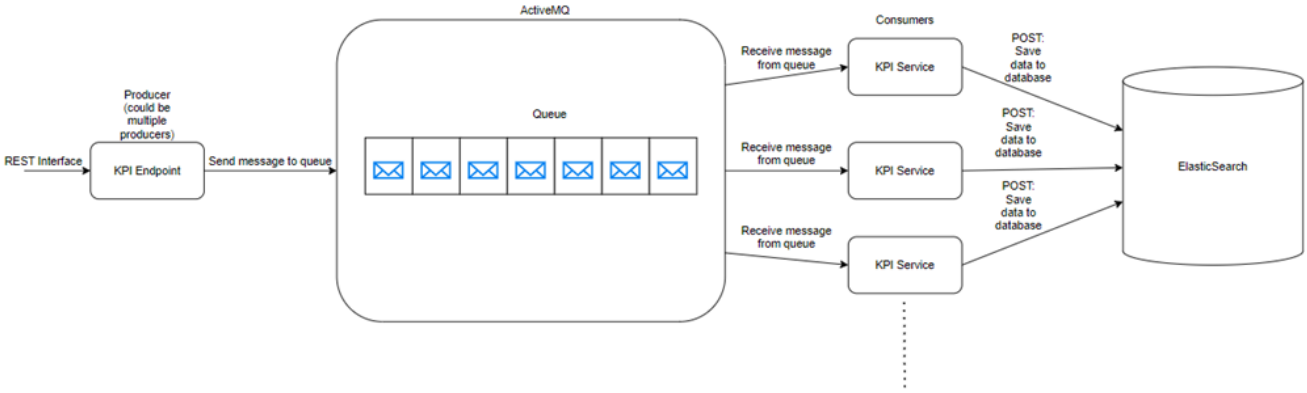

Figure 33. Data Monitoring – KPIC architecture overview.

3.7.2. API

The KPI Component exposes a number of APIs which are used by different components to provide and manage KPI data as described in Table 11.

Table 11. APIs exposed by the KPI component.

Exposed API	Component	Mandatory parameter(s)
KPI API (HTTPS) – get findByTrialID	LCM, Trial Validator	Trial ID
KPI API (HTTPS) - post KPIData	All KPI reporting components	KPI Data Entry (KPIDataID, Timestamp, DataObject)
KPI API (HTTPS) - post KPIDataJsonFile	All KPI reporting components	KPI Data Entries, as JSON file
KPI API (HTTPS) - get markings	Optional – to retrieve current markings	none
KPI API (HTTPS) - post markKPIData	Life Cycle Manager, all KPI reporting components to mark special events	Marking Data Entry
KPI API (HTTPS) - delete unmarkKPIData	Optional – to stop marking KPI data entries, Life Cycle Manager, all KPI reporting components	markingID

3.7.3. Workflows

The KPIC Client reports its business KPIs via the KPIData message to the KPI Component (KPIC). The corresponding workflow is described by Figure 34. Optionally, the KPI Client (as well as the Life Cycle Manager) may use the MarkKPIData call to set a tag to its reported KPI data messages. This allows easier analysis and reporting later on by marking specific business events during the Trials. Ending of the marking is possible by using unmarkKPIData.

In addition, the KPI Client may also use the KPIDataJsonFile call. This allows to provide KPI data packages via JSON file after the actual Trial, e.g. if no connectivity is available.

A KPIClient may retrieve current active reporting clients by using the getByTrialID call. The response consists of KPIC clients which were reporting for the defined TrialID. This allows to assure that all involved and needed clients are reporting KPIs.

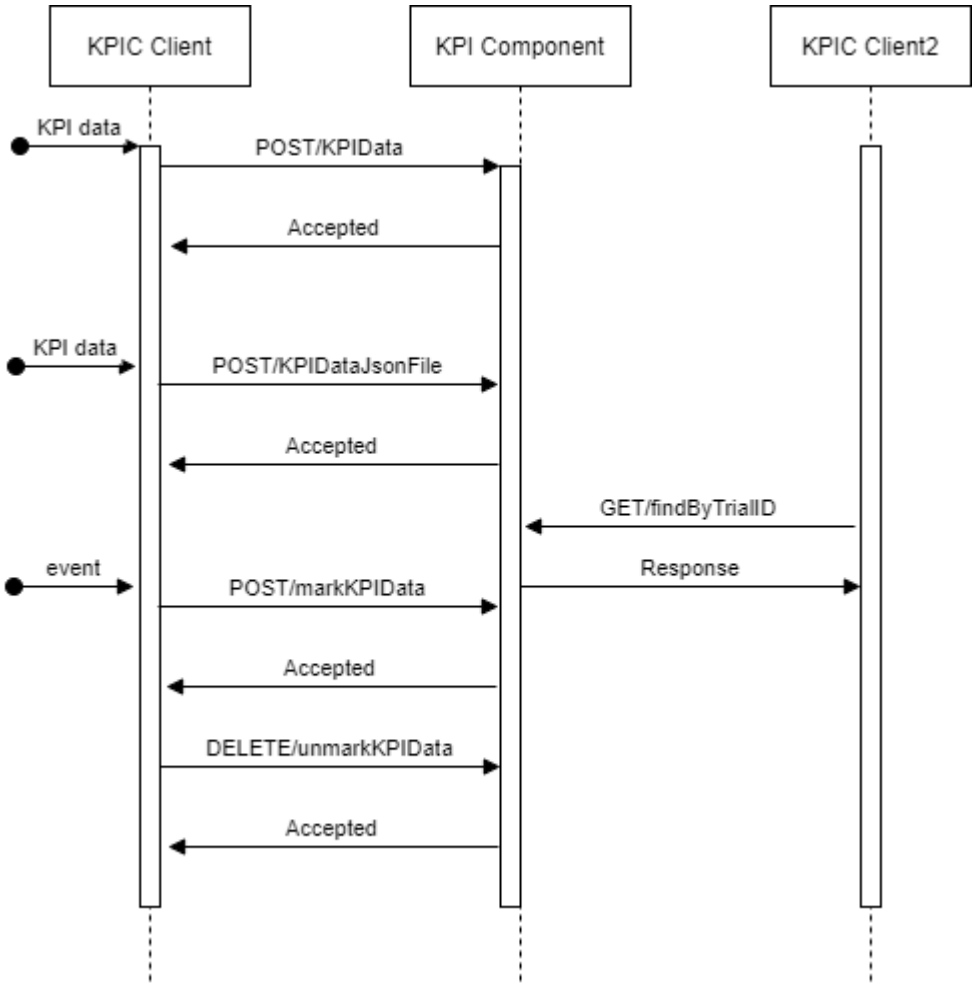

Figure 34. Data Monitoring – KPIC message workflow.

3.8. U-Space Adapter

The U-Space adapter acts as focal point for integration between U-Space and other components within the 5G!Drones project. A defined set of data can be exchanged via agreed interfaces.

3.8.1. Components and functions

The U-Space adapter is a logical summary of different interface endpoints to allow other Trial components to exchange information and data with the U-Space and its UTM components.

In general, the U-Space adapter offers capability to receive data from component as well as offering a subscription interface to provide data to interested parties.

3.8.2. API

The set of APIs exposed by the U-space adapter module are provided in Table 12.

Table 12. APIs exposed by the U-Space Adapter.

Exposed API	Component	Mandatory parameter(s)
HTTPS – put DFPL	Trial Validator, UTM systems	ID, contact, operation volume (start, end, altitudes, geography) submit time, uas registration.
HTTPS – subscribe dFPLS	Trial Validator, UTM systems	subscription endpoint
HTTPS – put Telemetry	Drone Operators	position
HTTPS – subscribe Telemetry	UTM systems	subscription endpoint
HTTPS – put Alert	UTM systems	free_text, message ID, type, operation plan IDs,
HTTPS – subscribe Alert	UTM systems, USSPs, Drone Operator	subscription endpoint
HTTPS – put UVR	UTM systems, USSPs	start, end, cause, geography, altitudes, message ID, submit time, type
HTTPS – retrieve UVRs	UTM systems, USSPs, Drone Operator	query (geometry)

3.8.3. Workflows

The following sequence diagram (Figure 35) shows an exemplary depiction of 2 components interacting with the U-Space adapter. Client a subscribes to be updated for a specific data type (e.g. drone telemetry). Client B provides the dataset (a drone position). Both Client A and the UTM system will be notified on the provided position. Client A will also be notified if the UTM system provided datasets.

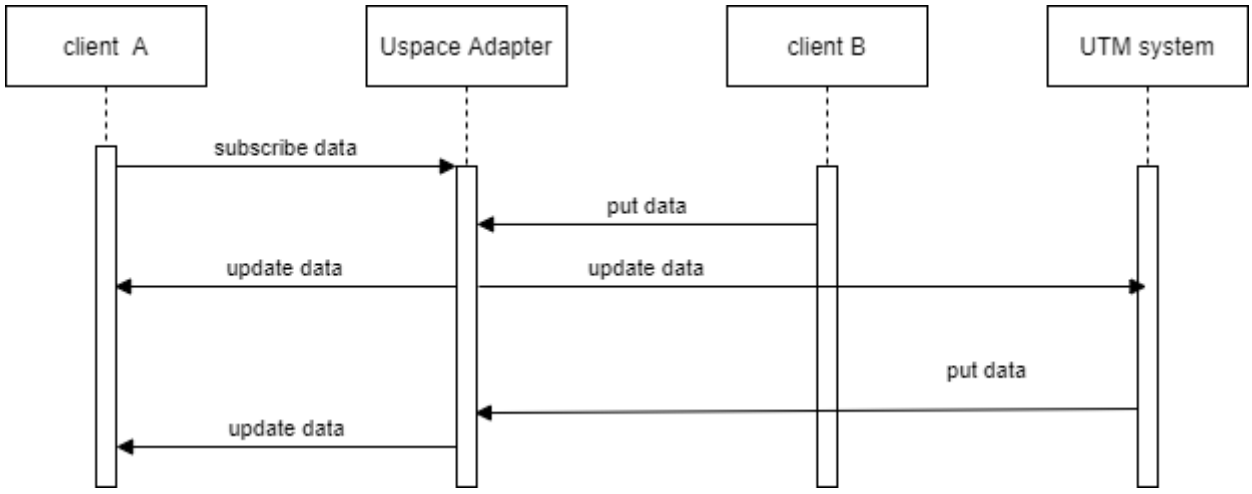

Figure 35. U-Space adapter workflow.

4. CONCLUSION

This deliverable described the final version of the trial controller and its related components. After presenting the trial deployment and deletion using a high-level workflow, the deliverable detailed all the components of the trial controller, namely Web portal, trial validator, U-Space adapter, trial translator, trial repository, LCM, trial enforcement and KPI monitoring. For each component, the deliverable provided:

1) a detailed description of its functions and architecture;
2) its exposed interfaces;
3) representative functions through workflows.

Furthermore, the deliverable also elaborated on performing autonomous management of the trials. To this end, a policy-based model is proposed along with a study case that considers reinforcement learning for automation. The implementation of the two algorithms, DQN and A2C, showed that an A2C agent could learn performing autonomous policies within the large action space.

Since the specifications of the trial controller are finalised, the implementation and integration phases can be finalised. While the implementation phase will be finalised in WP2, the integration will be finalised in WP4.

References

[1] 5G!Drones D2.1, “Initial definition of the trial controller architecture, mechanisms, and APIs,” H2020 5G!Drones project, 2020.

[2] www.keycloak.org.

[3] GSMA, “Generic Network Slice Template, version 3.0, NG.116.,” May 2020..

[4] 5G!Drones D2.2, “Initial Implementation of the Trial Controller,” H2020 5G!Drones project, 2021.

[5] https://fastapi.tiangolo.com/.

[6] 5G!Drones D1.6, “5G!Drones system architecture refined design,” H2020 5G!Drones project, 2021.

[7] pytorch, “pytorch website,” [Online]. Available: https://pytorch.org/. [Accessed 2021].

[8] 5G!Drones D2.3, “Report on algorithms, mechanisms and tools for data analysis and visualisation,” H2020 5G!Drones project, 2021.