5G-EPICENTRE

5G ExPerimentation Infrastructure hosting Cloud-nativE Netapps for public proTection and disaster RElief

Innovation Action – ICT-41-2020 - 5G PPP – 5G

Innovations for verticals with third party services

D4.7: Integration, Verification and Testing Report final version

Delivery date: January 2024

Dissemination level: Public

Project Title:	**5G-EPICENTRE - 5G ExPerimentation Infrastructure hosting Cloud-nativE Netapps for public proTection and disaster RElief**
Duration:	**1 January 2021 – 31 December 2023**
Project URL	https://www.5gepicentre.eu/

This project has received funding from the European Union's Horizon 2020 Innovation Action programme under Grant Agreement No 101016521.

www.5gepicentre.eu

Document Information

Deliverable	D4.7: Integration, Verification and Testing Report final version
Work Package	WP4: Platform integration and VNF development
Task(s)	T4.4: End-to-end platform integration activities T4.5: Lab testing, prototyping and validation
Type	Report
Dissemination Level	Public
Due Date	M34, October 31, 2023
Submission Date	M37, January 31, 2022 (Amendment)
Document Lead	Konstantinos C. Apostolakis (FORTH)
Contributors	Stefania Stamou (FORTH) Anna Maria Spagnolo (IST) Luigi D'Addona (IST) Jorge Marquez Ortega (UMA) Almudena Díaz Zayas (UMA) Fatemehsadat Tabatabaeimehr (CTTC)
Internal Review	Anna Maria Spagnolo (IST) Luigi D'Addona (IST) Elvina Gindullina (HPE)

Document history

Version	Date	Changes	Contributor(s)
V0.1	01/12/2023	Initial deliverable structure	Konstantinos Apostolakis (FORTH)
V0.2	14/12/2023	Template content preparation referring to the platform architectural elements identified last in D1.4, distribution to all partners.	Konstantinos Apostolakis (FORTH)
V0.3	18/12/2023	Input text for the integration and testing of the 5G-EPICENTRE Portal	Konstantinos Apostolakis (FORTH) Stefania Stamou (FORTH)
V0.4	09/01/2024	Input text for the integration and testing of the Analytics Engine, and Analytics Aggregator.	Anna Maria Spagnolo (IST) Luigi D'Addona (IST)
V0.5	11/01/2024	Input text for the integration and testing of the Experiment Coordinator and 5G Traffic Simulator.	Jorge Marquez Ortega (UMA) Almudena Díaz Zayas (UMA)
V0.6	18/01/2024	Input text for the integration and testing of the Karmada Federation layer and Service placement plug-in.	Fatemehsadat Tabatabaeimehr (CTTC)
V1.0	25/01/2024	Internal Review Version, final formatting and proof-reading (quality check).	Konstantinos Apostolakis (FORTH)
V1.1	30/01/2024	1st version with suggested revisions	Elvina Gindullina (HPE)
V1.2	30/01/2024	2nd version with suggested revisions	Anna Maria Spagnolo (IST) Luigi D'Addona (IST)
V2.0	31/01/2024	Final version for submission	Konstantinos Apostolakis (FORTH)

Project Partners

Logo	Partner	Country	Short name
	AIRBUS DS SLC	France	**ADS**
	NOVA TELECOMMUNICATIONS SINGLE MEMBER S.A.	Greece	**NOVA**
	Altice Labs SA	Portugal	**ALB**
	Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.	Germany	**HHI**
	Foundation for Research and Technology Hellas	Greece	**FORTH**
	Universidad de Malaga	Spain	**UMA**
	Centre Tecnològic de Telecomunicacions de Cata-lunya	Spain	**CTTC**
	Istella SpA	Italy	**IST**
	One Source Consultoria Informatica LDA	Portugal	**ONE**
	Iquadrat Informatica SL	Spain	**IQU**
	Nemergent Solutions S.L.	Spain	**NEM**
	EBOS Technologies Limited	Cyprus	**EBOS**
	Athonet SRL **(Participation ended)**	*Italy*	*ATH*
	RedZinc Services Limited	Ireland	**RZ**
	OptoPrecision GmbH	Germany	**OPTO**
	Youbiquo SRL	Italy	**YBQ**
	ORamaVR SA	Switzerland	**ORAMA**
	Hewlett-Packard Italiana Srl	Italy	**HPE**

List of abbreviations

Abbreviation	Definition
5GTSM	5G Traffic Simulation Manager
AF	Application Function
API	Application Programming Interface
CLI	Command Line Interface
CORS	Cross-Origin Resource Sharing
DL	Deep Learning
GA	Grant Agreement
HSPF	Holistic Security and Privacy Framework
ILP	Integer Linear Programming
IP	Internet Protocol
JSON	JavaScript Object Notation
K8s	Kubernetes
KPI	Key Performance Indicator
MQTT	Message Queueing Telemetry Transport
NF	Network Function
NS	Network Service
QoE	Quality of Experience
QoS	Quality of Service
RBAC	Role-Based Access Control
REST	Representational State Transfer
TCP	Transmission Control Protocol

TM	Technical Manager
UC	Use Case
(G)UI	(Graphical) User Interface
VPN	Virtual Private Network

Executive summary

The present report summarizes the activities of the 5G-EPICENTRE Consortium with respect to system integration (Task 4.4) and testing (Task 4.5), undertaken for the period M24-M37 (*i.e.*, after delivery of the preceding report D4.6: *"Integration, Verification and Testing Report preliminary version"*). It follows up on prior documentation on the integrated prototype (D4.5: *"5G-EPICENTRE experimentation facility final version"*), and constitutes a reporting of the final roadmaps and activities for carrying out the system integration and corresponding testing activities throughout the aforementioned timeframe. The content in this report refers to the platform architectural elements identified in D1.4: *"Experimentation requirements and architecture specification final version"*, and particularly reports on integration and testing with respect to the interfaces that partners responsible for the different interdependent components have defined. These are either individually reported in the component's standalone deliverable, or in the API reference documentation available in D4.5.

The delivery of this report concludes the partners' activities in Work Package (WP) 4, and constitutes a compendium of the integration work with regards to implementing the 5G-EPICENTRE experiment e-ordering platform. It hence accounts how the partners developed a novel aggregator of four independent (*i.e.*, characterized by different 5G standalone implementation and technologies) testbed facilities, federated under a typical Karmada control plane architecture, and specifically exposed for Public Protection and Disaster Relief (PPDR) vertical system experimentation.

Table of Contents

List of Figures 10
List of Tables 11
1 Introduction 13
1.1 Mapping of project's outputs 13
2 5G-EPICENTRE system integration report 15
2.1 5G-EPICENTRE Portal 15
2.1.1 Functional description 15
2.1.2 Information flow 15
2.1.3 Interdependencies with other components 16
2.1.4 Integration roadmap 17
2.1.5 Issues encountered and roadmap deviations 19
2.2 Experiment Coordinator 19
2.2.1 Functional description 19
2.2.2 Information flow 19
2.2.3 Interdependencies with other components 20
2.2.4 Integration roadmap 21
2.2.5 Issues encountered and roadmap deviations 23
2.3 5G Traffic Simulator 23
2.3.1 Functional description 23
2.3.2 Information flow 23
2.3.3 Interdependencies with other components 24
2.3.4 Integration roadmap 25
2.3.5 Issues encountered and roadmap deviations 26
2.4 Karmada (Federation Layer) 26
2.4.1 Functional description 26
2.4.2 Information flow 27
2.4.3 Interdependencies with other components 27
2.4.4 Integration roadmap 28
2.5 Service placement 29
2.5.1 Functional description 29
2.5.2 Information flow 29
2.5.3 Interdependencies with other components 30
2.5.4 Integration roadmap 30
2.5.5 Issues encountered and roadmap deviations 30
2.6 Analytics Engine 30
2.6.1 Functional description 30
2.6.2 Information flow 31
2.6.3 Interdependencies with other components 31
2.6.4 Integration roadmap 32
2.6.5 Issues encountered and roadmap deviations 33
2.7 Analytics Aggregator 33
2.7.1 Functional description 33
2.7.2 Information flow 33
2.7.3 Interdependencies with other components 34
2.7.4 Integration roadmap 34
2.7.5 Issues encountered and roadmap deviations 35

3 5G-EPICENTRE system testing and validation report ... 36
3.1 5G-EPICENTRE Portal functionality validation ... 37
3.1.1 Network Service Repository integration testing ... 37
3.1.2 Experiment Coordinator integration tests ... 39
3.1.3 Analytics Aggregator integration tests ... 41
3.2 Experiment Coordinator functionality validation ... 41
3.2.1 5GTSM integration testing ... 42
3.2.2 Publisher integration testing ... 42
3.2.3 Network Service Repository integration tests ... 44
3.2.4 Karmada integration testing ... 44
3.3 5G Traffic Simulator functionality validation ... 45
3.3.1 Testbed integration testing ... 45
3.3.2 Publisher integration testing ... 46
3.3.3 5GTSM – Remote iPerf Agent integration testing ... 47
3.4 Karmada (Federation layer) functionality validation ... 48
3.4.1 Kubernetes testbed cluster integration testing ... 48
3.5 Service placement functionality validation ... 49
3.5.1 Karmada (Federation layer) integration testing ... 49
3.6 Analytics Engine functionality validation ... 50
3.6.1 Publisher integration testing ... 50
3.7 Analytics Aggregator functionality validation ... 51
3.7.1 KPI Monitor integration testing ... 51
3.7.2 QoS/QoE Monitor integration tests ... 52
4 Conclusions ... 54
References ... 55

List of Figures

Figure 1: 5G-EPICENTRE Portal data flow. 16
Figure 2: Experiment Coordinator data flow. 20
Figure 3: 5G Traffic Simulator data flow. 24
Figure 4: Karmada Federation Layer data flow. 27
Figure 5: Service placement data flow. 29
Figure 6: Analytics Engine data flow. 31
Figure 7: Analytics Aggregator data flow. 33

List of Tables

Table 1: Adherence to 5G-EPICENTRE's GA Tasks' Descriptions 13
Table 2: 5G-EPICENTRE Portal connection with other 5G-EPICENTRE components 16
Table 3: 5G-EPICENTRE Portal – Network Service Repository final integration report 17
Table 4: 5G-EPICENTRE Portal – Experiment Coordinator final integration report 18
Table 5: 5G-EPICENTRE Portal – Analytics Aggregator final integration report 18
Table 6: Experiment Coordinator connection with other 5G-EPICENTRE components 20
Table 7: Experiment Coordinator – 5GTSM final integration report 21
Table 8: Experiment Coordinator – Publisher final integration report 22
Table 9: Experiment Coordinator – Network Service Repository final integration report 22
Table 10: Experiment Coordinator – Karmada Federation Layer final integration report 22
Table 11: 5G Traffic Simulator connection with other 5G-EPICENTRE components 24
Table 12: 5GTSM connection with other 5G-EPICENTRE components 25
Table 13: 5GTSM – Testbeds final integration report 25
Table 14: Remote iPerf agents – Publisher final integration report 25
Table 15: 5GTSM – Remote iPerf agents final integration report 26
Table 16: Karmada connection with other 5G-EPICENTRE components 28
Table 17: Karmada – Testbed K8s cluster final integration report 28
Table 18: Service placement plugin connection with other components 30
Table 19: Service placement– Karmada final integration report 30
Table 22: Analytics Engine connection with other 5G-EPICENTRE components 31
Table 23: Analytics Engine – Publisher final integration report 32
Table 24: Analytics Aggregator connection with other 5G-EPICENTRE components 34
Table 25: Aggregator – KPI Monitor final integration report 34
Table 26: Aggregator – QoS/QoE Monitor final integration report 35
Table 27: Test matrix listing the types of functional testing carried out, and reported in this Section. 36
Table 28: 5G-EPICENTRE Portal – Network Service Repository integration testing. 36
Table 29: 5G-EPICENTRE Portal – Network Service Repository integration testing. 37
Table 30: 5G-EPICENTRE Portal – Network Service Repository integration testing fail cases. 39
Table 31: 5G-EPICENTRE Portal – Experiment Coordinator integration testing. 39
Table 32: 5G-EPICENTRE Portal – Experiment Coordinator integration testing fail cases. 40
Table 33: 5G-EPICENTRE Portal – Analytics Aggregator integration testing. 41
Table 34: 5G-EPICENTRE Portal – Analytics Aggregator integration testing fail cases. 41
Table 35: Experiment Coordinator – 5GTSM integration testing. 42

Table 36: Experiment Coordinator – 5GTSM integration testing fail cases. 42
Table 37: Experiment Coordinator – Publisher integration testing. 43
Table 38: Experiment Coordinator – Publisher integration testing fail cases. 43
Table 39: Experiment Coordinator – Network Service Repository integration testing. 44
Table 40: Experiment Coordinator – Network Service Repository integration testing fails cases. 44
Table 41: Experiment Coordinator – Karmada integration testing. 44
Table 42: Experiment Coordinator – Karmada integration testing fail cases. 45
Table 43: 5GTSM – Testbed integration testing. 45
Table 44: Remote iPerf agent – Publisher integration testing. 46
Table 45: Remote iPerf agent – Publisher integration testing fail cases. 46
Table 46: 5GTSM – Remote iPerf Agent(s) integration testing. 47
Table 47: 5GTSM – Remote iPerf Agent(s) integration testing fail cases. 47
Table 48: Cross-testbed federation – Testbeds' Kubernetes's cluster integration testing. 48
Table 49: Service placement plugin – Karmada (Federation layer) integration testing. 49
Table 50: Service placement – Karmada integration testing fail cases. 50
Table 53: Analytics Driver – Publisher integration testing. 50
Table 54: Analytics Driver – Publisher integration testing fail cases. 51
Table 55: Aggregator – KPI Monitor integration testing. 51
Table 56: Aggregator – KPI Monitor integration testing fail cases. 52
Table 57: Aggregator – QoS/QoE Monitor integration testing. 52

1 Introduction

This deliverable represents the final report on 5G-EPICENTRE partner activities regarding system integration, testing and validation. All reported items have been carried out in the context of Tasks T4.4: "*End-to-end platform integration activities*" and T4.5: "*Lab testing, prototyping and validation*", and correspond to the timeline between delivery of the precursor deliverable D4.6 *"Integration, Verification and Testing Report preliminary version"* (M24) and the present document's re-planned delivery date (M37). All activities reported in the present document have been coordinated under the supervision of the Technical Manager (TM) of the project.

Reflecting the Integration roadmap established in D4.1: *"Integration plan and framework"*, and re-iterating from D4.6, ***system integration*** (which shall henceforth be referred to simply as *integration*) in the context of 5G-EPICENTRE Task T4.4, refers to the process of interlinking different technological components toward facilitating a uniform system (*i.e.*, the 5G-EPICENTRE Platform). Following up on the final platform integration overview, reported in D4.5 (Section 2 in that document), the means by which integration is addressed is through the definition and implementation of well-defined interfaces (*i.e.*, with concretely established inputs and outputs). Such interfaces follow the *consumer-producer paradigm*, allowing a component to either expose, or consume methods exposed by other components, so that the exchange of data can be facilitated through pre-specified sets of parameters. Through the 5G-EPICENTRE integration approach that follows the microservices architecture paradigm, two components (services) are considered to be "integrated" when either one service is able to consume the interfaces exposed by the other.

As the penultimate document on the project-developed 5G-EPICENTRE platform and its development / integration activities, the present deliverable departs from the structure of D4.6 (and the more general processes described therein), and hence shall report on the technical content describing the different components and their interdependencies, together with information on the roadmap to integrate the component within the overall 5G-EPICENTRE developed solution. The document will further describe the undertaken actions and results regarding system integration testing, *i.e.*, testing carried out to verify integrity and consistency of the platform components' intercommunication. The majority of this information reflect the contents reported in deliverable D4.5: "*5G-EPICENTRE experimentation facility final version*" (M30). Whereas that document provides the full APIs' reference documentation, this deliverable emphasizes the actual 5G-EPICENTRE partners' activities in the context of Tasks 4.5 and 4.6, including the plans and roadmaps established to execute the activities therein.

The rest of the deliverable is structured as follows: Section 2 presents an overview of the approach to integration for each of the 5G-EPICENTRE platform's interdependent components. Section 3 then describes the integration testing activities, recounting followed procedures, test methodology for each integration method, and brief insight into the test outcomes, which led to overall system improvements. Finally, Section 4 concludes the deliverable.

1.1 Mapping of project's outputs

The purpose of this Section is to map 5G-EPICENTRE Grant Agreement (GA) commitments, within the formal Task description, against the project's respective outputs and work performed.

Table 1: Adherence to 5G-EPICENTRE's GA Tasks' Descriptions

5G-EPICENTRE Task	Respective Document Chapters	Justification
T4.4: End-to-end platform integration activities	Section 2.x.1	For each component (Section 2.x), this Section provides a brief description, and elaborates on the

"This Task will deal with the integration of the modules developed in the technical WPs, according to the system architecture (T1.3) and use case requirements (T1.2)". *"[…]. Integration will hence be addressed using vertical methods in order to have functional entities and horizontal approaches so as to facilitate any necessary customization of the platform, which will iteratively integrate components resulting from technical WPs to deliver incremental releases of the 5G-EPICENTRE platform"*.		component's role in the overall 5G-EPICENTRE platform.
	Section 2.x.2	This Section briefly describes the information flowing through the component.
	Section 2.x.3	This Section briefly lists all interdependencies of the component with other components in the architecture.
	Section 2.x.4	This Section describes the integration roadmap followed for the integration between it, and each interdependent component.
	Section 2.x.5	This Section informs on whether minor, or serious issues (especially such that made partners divert from the roadmap) were encountered, describing them, and providing the countermeasure(s) that was/were applied.
T4.5: Lab testing, prototyping and validation *"The main aim of this task is to manage testing and validation of the separate components to be integrated into the final system. This will involve specifying a testing framework, which supports automated unit-testing (e.g. black box, white box, integration) and guidelines for testing to be used for individual component development."*. *"[…]. The Task will specify high-level tests for the components to be integrated in T4.4, as well as integration tests and method of validating the integrated system"*.	Section 3.x	For each component, this Section describes the approach to testing the component. It reports on the ways in which the APIs were tested. It further reports whether those tests were exploratory (*i.e.*, send a request and check that the response is correct, black-box or white-box); usability tests (*i.e.*, emulate the use of the system by a user); or ad-hoc tests (check for loopholes).

2 5G-EPICENTRE system integration report

The purpose of this Section is to report on the integration outcomes for each of the 5G-EPICENTRE individual components, describing their established connections, data flow, integration roadmap and a report on any problems encountered during integration. Relevant deliverables (and thereby, their corresponding Tasks and WPs) are listed, toward guiding interested readers to the latest API documentation referenced for each interdependency, *i.e.*, their available endpoints, methods, headers, parameters and expected responses. This information, generated in Task T4.4, has been crucial for defining the system integration testing workflow in Task T4.5 (reported in Section 3 of this document).

This report is to be considered as complementary to D4.5, and refers to the system information described in Section 4 of that document.

2.1 5G-EPICENTRE Portal

2.1.1 Functional description

The 5G-EPICENTRE Portal is a user-facing web application that allows users to interact with the 5G-EPICENTRE platform, offering different usage scenarios for: i) delegating network, or vertical application artefacts (Helm charts) to the platform repository, for use and re-use in experimentation activities; ii) reserving resources for an experiment execution, by specifying desired execution timeline; iii) regulating how an experiment is deployed, which artefacts to be installed in the available testbeds' Kubernetes clusters, as well as chained network applications that should execute in parallel; and iv) collect and visualize measurements from an experiment carried over the platform. The complete functional description of this component is elaborated in deliverable D3.2 *"5G EPICENTRE Front-end components"*.

2.1.2 Information flow

The 5G-EPICENTRE Portal interfaces with the Network Service Repository, thereby operating as a web-based, graphical UI (GUI) client, to view all available Helm charts that can be deployed on top of the 5G-EPICENTRE federated testbeds infrastructure, as well as perform simple actions, such as adding to, replacing, or deleting a resource (provided the user has the necessary rights to these actions). The interaction enables the Portal users to inspect, and subsequently, specify which artefacts to deploy during an experiment execution request, *i.e.*, vertical application Helm charts uploaded by experimenters themselves, as well as network application/network function (NF)/application function (AF) Helm charts to incorporate into the experiment execution request, offered by the 5G-EPICENTRE Consortium vertical representatives (*i.e.*, use case owners). These are then forwarded to the Experiment Coordinator component through the experiment descriptor exchange structure over the Experiment Run API endpoint. At any given time, the Portal can be used by any user with a proper role-based access control (RBAC) authentication token (*i.e.*, either the experimenter themselves, or the testbed administrator of the platform elected to host the experiment), to request a scheduled experiment be cancelled.

Finally, during an experiment execution, the 5G-EPICENTRE Platform receives real-time information on experiment analytics, including pre-specified and user-specific Key Performance Indicators (KPIs), traffic parameters and detected anomalies. The information is structured in rich graphical representations, and stored in the Portal's own database for keeping a persistent record of the experiment execution for later use.

The graphical representation of the described data flow is illustrated in Figure 1.

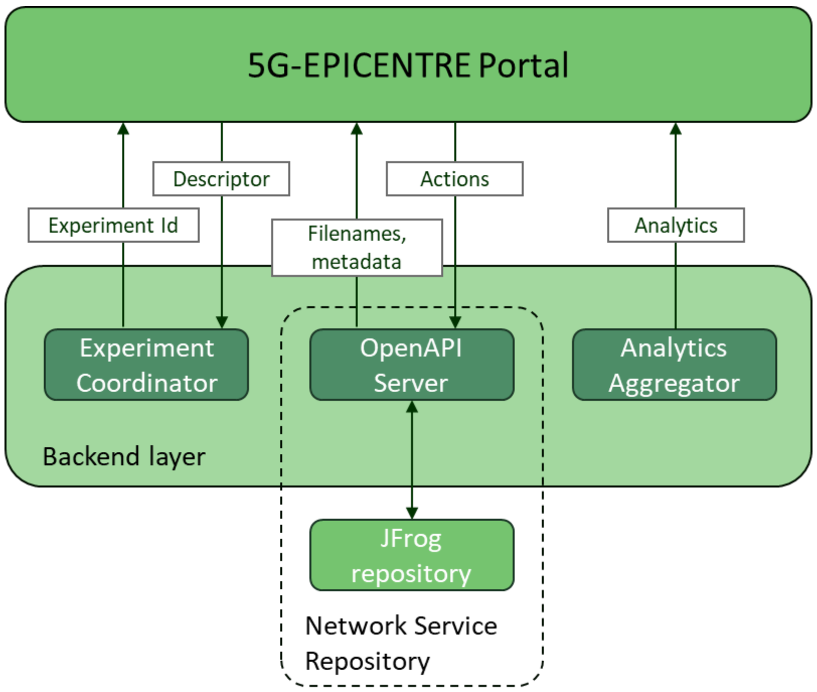

Figure 1: 5G-EPICENTRE Portal data flow.

2.1.3 Interdependencies with other components

The 5G-EPICENTRE Portal (a component developed by FORTH) shares interdependencies with the following 5G-EPICENTRE functional entities: Experiment Coordinator (UMA); Network Service Repository (IQU); Analytics Aggregator (IST). Table 2 below, provides an overview of the connections established with other platform functional elements, and briefly elaborates on their interfaces.

Table 2: 5G-EPICENTRE Portal connection with other 5G-EPICENTRE components

Component	Connection	API protocol	Data exchange	Comments
Experiment Coordinator	TCP/IP	REST	JSON	The 5G-EPICENTRE Portal consumes the '*Experiment Run*' and '*Experiment Cancel*' APIs exposed by the Experiment Coordinator, to either request the execution of an experiment on a pre-designated point in time or cancel a particular experiment execution request identified by a unique identifier. Minimal user interface (UI) dashboards have been implemented, which enable enriching the experiment descriptor payload (exchanged between the Portal and the Experiment Coordinator over the Experiment Run endpoint), with all the necessary deployment parameters to enable activation (*i.e.*, deployment) and chaining of pre-specified network

				applications with an experiment request involving vertical-specific artefacts. The final documentation of these APIs, can be found in deliverable D2.5 *"5G-EPICENTRE Experiment execution"*, whereas these chainings will be reported in detail in the revised version of deliverable D4.2 *"Network functions implementation"*.
Network Service Repository	TCP/IP	REST	JSON	The 5G-EPICENTRE Portal consumes all APIs exposed by the Network Service Repository's OpenAPI server, which enable operations, such as viewing of all filenames in the repository (to allow user to browse and select desired Helm charts to deploy), retrieval of Helm chart metadata (similarly, to display artefact information to the Portal user), deletion of a file, and uploading of a file in the Network Service Repository. The final documentation of these APIs can be found in deliverable D4.3 *"5G-EPICENTRE Experiment execution"*.
Analytics Aggregator	TCP/IP	MQTT	JSON	The 5G-EPICENTRE Portal is subscribed to the topic exchange queue published to by the Analytics Aggregator, thereby asynchronously receiving payloads corresponding to the processed analytics information generated in real-time for a particular experiment under test (specified by its unique identifier). The final documentation of this API can be found in deliverable D2.6 *"5G-EPICENTRE Analytics Engine"*.

2.1.4 Integration roadmap

Table 3 below, describes the integration roadmap followed for the integration between the 5G-EPICENTRE Portal and the Network Service Repository, along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 3: 5G-EPICENTRE Portal – Network Service Repository final integration report

Roadmap	Description
Integration format	REST API integration.
Integration activities	• Development of Network Service Repository OpenAPI server (IQU). • Testing of each API via mock services endpoints (IQU, FORTH).

	• Refinement, finalization, and deployment at UMA (integration with actual/real endpoints, UMA, IQU). • System integration test (UMA, FORTH).
Integration testing	Section 3.1.1

Table 4 similarly describes the roadmap followed for the integration between the 5G-EPICENTRE Portal and the Experiment Coordinator, along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 4: 5G-EPICENTRE Portal – Experiment Coordinator final integration report

Roadmap	**Description**
Integration format	REST API integration.
Integration activities	• Development of the first version of the experiment descriptor data model for 5G-EPICENTRE (refinement of prior descriptor template from 5GENESIS, UMA) • Development of northbound Experiment Coordinator APIs exposed towards the 5G-EPICENTRE Portal (UMA). • Testing of each API via mock services endpoints (UMA). • Integration with actual/real endpoints (UMA) • Testing of APIs via established endpoints (UMA, FORTH). • First system integration test (UMA, FORTH). • Development of refined version of the experiment descriptor data model for 5G-EPICENTRE (integration of Network Intrusion Detection network application parameters, UMA, ONE). • Second system integration test (UMA, ONE, FORTH). • Development of final version of the experiment descriptor data model for 5G-EPICENTRE (integration of additional network application parameters, UMA, EBOS, IST). • Development of refined final version of the experiment descriptor data model for 5G-EPICENTRE (integration of network application, application function, network function parameters from UC owners, UMA, UC owners) **[pending]**. • Final system integration test (UMA, FORTH, EBOS, IST, UC Owners) **[pending]**.
Integration testing	Section 3.1.2

Finally, Table 5 describes the roadmap followed for the integration between the 5G-EPICENTRE Portal and the Analytics Aggregator, along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 5: 5G-EPICENTRE Portal – Analytics Aggregator final integration report

Roadmap	**Description**
Integration format	RabbitMQ MQTT broker topic exchange integration.
Integration activities	• Setup RabbitMQ MQTT broker (UMA, IST)

	• Development of the first version of the Analytics Aggregator data model (IST) • Integration with actual/real endpoints (UMA, IST). • Testing of API via established endpoints using mock data (IST, ADS, UMA, FORTH). • Development of refined version of the analytics aggregator data model (IST, ONE). • System integration test (IST, UMA, ONE, FORTH). • Development of refined final version of the analytics aggregator data model (IST, UC owners). • Final system integration test (IST, UMA, FORTH, UC Owners).
Integration testing	Section 3.1.3

2.1.5 Issues encountered and roadmap deviations

Minor issues were encountered and dealt with effectively, without significantly affecting the integration roadmaps in each case. Some service unavailability issues were encountered due to the need to re-integrate APIs with different real endpoints, at UMA side, without however affecting the integration procedure (requests were simply re-routed to the newest endpoints communicated). Regarding the Network Service Repository integration, changes in the Jfrog pricing model forced IQU to self-host the Jfrog repo in a private server (similarly requiring a re-integration with new real endpoints).

2.2 Experiment Coordinator

2.2.1 Functional description

The Experiment Coordinator is the element inside the 5G-EPICENTRE architecture, in charge of coordinating the life cycle of the experiments running on the platform. The Experiment Coordinator can: (1) schedule the execution and deployment of use cases from both first (project Use Cases – UCs) and third parties; (2) deploy the mentioned uses cases in any of the 4 testbeds that belong to the platform (through the federated Karmada synchronization layer); (3) deploy the Network Intrusion Detection Network Application through the Holistic Security and Privacy Framework (HSPF) module (see D2.8 *“Cloud-native Security Specifications Final Version”*); as well as (4) execute the traffic generation in any of the integrated testbeds, through the 5G Traffic Simulation Manager (5GTSM).

2.2.2 Information flow

The Experiment Coordinator receives the necessary information from the Portal, in the form of a descriptor, to download the Helm chart indicated from the Network Service Repository. Once downloaded, the Helm chart is deployed in the selected testbed and namespace through the Karmada federation layer, and the *“experiment_id”* is sent back to the Portal. Once the Helm chart is deployed, the Publisher of the corresponding testbed will be updated with the *“experiment_id”* and the *“netapp_id”* parameters, associated with the experiment for its correct identification in a RabbitMQ queue (the measurements will be published by the vertical in a RabbitMQ exchange, and from there, they are forwarded to the Analytics Engine). The Experiment Coordinator can also initiate the traffic generation in the desired testbed, by means of the 5GTSM instance deployed in it. This traffic will be generated according to the profile (light, moderate, disaster) indicated in the experiment descriptor. Finally, if selected, the security-oriented Network Application for intrusion detection (*i.e.*, HSPF) will be deployed on the selected testbed, and with the chosen microservices (as indicated in the experiment descriptor).

The graphical representation of the described data flow is illustrated in Figure 2.

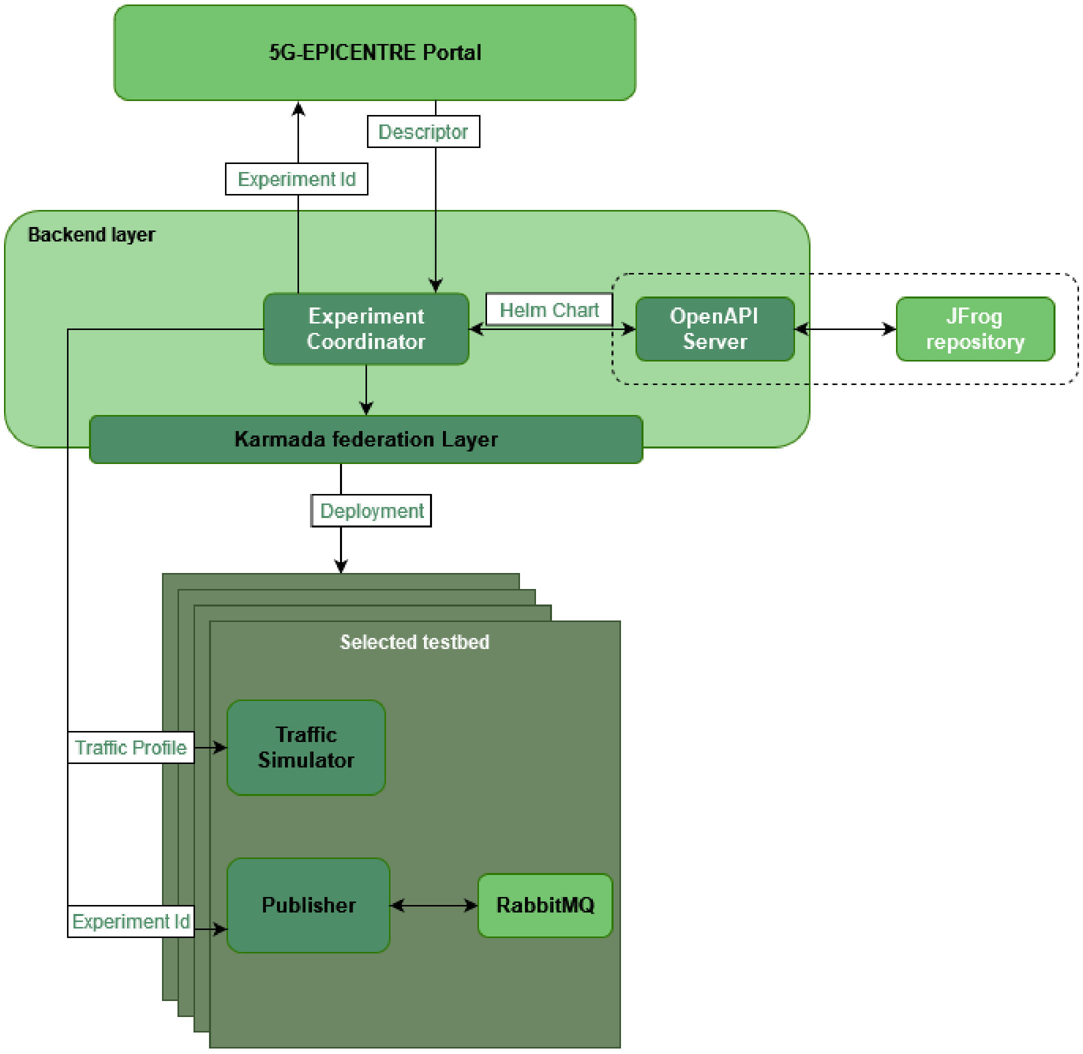

Figure 2: Experiment Coordinator data flow.

2.2.3 Interdependencies with other components

The Experiment Coordinator (a component developed by UMA) has interdependencies with the following elements of the platform: 5GTSM (UMA), Publisher (UMA), Network Service Repository (IQU) and Karmada federation layer (CTTC). Table 6 below, provides an overview of the connections established with other platform functional elements, and briefly elaborates on their interfaces.

Table 6: Experiment Coordinator connection with other 5G-EPICENTRE components

Component	Connection	API protocol	Data exchange	Comments
5GTSM	TCP/IP	REST	JSON	The Experiment Coordinator consumes the start and stop endpoints of the 5GTSM, which in turn accesses the corresponding endpoints of the remote iPerf agents (see D2.5 for a more detailed

				description). It also consumes end-points for aggregation and deletion of relevant information from the remote iPerf agents.
Publisher	TCP/IP	REST	API	This Integration with the Publisher is done to provide experiments' metadata to it. In this way, the Experiment Coordinator can communicate the *"experiment_id"* and *"netapp_id"* fields of the experiment to be executed to the Publisher. This in turn allows the identification of the measurements to be published from the experiment in a RabbitMQ exchange.
Network Service Repository	TCP/IP	REST	API	The Experiment Coordinator can retrieve Helm chart from the Network Service Repository using its REST API. This allows it to obtain the deployment file specified by the Portal via the descriptor, enabling the deployment of its contents on the chosen testbed." or something like that.
Karmada Federation Layer	TCP/IP	REST	API	For the deployment of the different use cases in each of the testbeds, the Experiment Coordinator makes use of the Karmada federation layer. By modifications in the propagation policies files included in each deployment use case, the Experiment Coordinator is able to select the correct testbed and namespace.

2.2.4 Integration roadmap

Table 7 below, describes the integration roadmap followed for the integration between the Experiment Coordinator and the 5GTSM, along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 7: Experiment Coordinator – 5GTSM final integration report

Roadmap	Description
Integration format	REST API and RabbitMQ MQTT broker topic exchange integration.
Integration activities	• Development of the 5GTSM (UMA). • Development of the Remote iPerf agents used by 5GTSM (UMA). • Integration of the 5GTSM in each testbed belonging to the platform (UMA, ALB, CTTC, HHI).

	• Testing of traffic generation in each testbed and its correct publication in the corresponding RabbitMQ queue (UMA, all testbeds). • System integration test (UMA, all testbeds).
Integration testing	Section 3.2.1

Table 8 below, describes the integration roadmap followed for the integration between the Experiment Coordinator and the Publisher, along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 8: Experiment Coordinator – Publisher final integration report

Roadmap	Description
Integration format	REST API and RabbitMQ MQTT broker topic exchange integration.
Integration activities	• Development of the Publisher (UMA). • Integration of the Publisher in each testbed belonging to the platform as well as the RabbitMQ broker (UMA, all testbeds). • Testing the correct publication of the KPIs of an experiment in the RabbitMQ queue with the correct *"experiment_id"* and *"netapp_id"* (UMA, all testbeds). • System integration test (UMA, all testbeds).
Integration testing	Section 3.2.2

Table 9 below, describes the integration roadmap followed for the integration between the Experiment Coordinator and the Network Service Repository, along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 9: Experiment Coordinator – Network Service Repository final integration report

Roadmap	Description
Integration format	REST API integration.
Integration activities	• Development of the OpenAPI server (IQU). • Integration of the OpenAPI in the backend layer hosted at UMA. • Testing of each API via mock services endpoints (IQU, UMA). • System integration test (UMA, FORTH).
Integration testing	Section 3.2.3

Finally, Table 10 below, describes the integration roadmap followed for the integration between the Experiment Coordinator and the Karmada Federation Layer, along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 10: Experiment Coordinator – Karmada Federation Layer final integration report

Roadmap	Description
Integration format	REST API integration.

Integration activities	• Development of the Karmada federation layer (CTTC). • Integration of the different clusters in Karmada (CTTC). • Installation of the necessary tool to manage Karmada such as kubectl[1], karmadactl[2], *etc*. (CTTC). • Implementation of the necessary certificates and configurations for the use of Karmada (UMA). • Testing of resources propagation through the different testbeds that are part of Karmada (UMA, all testbed owners).
Integration testing	Section 3.2.4

2.2.5 Issues encountered and roadmap deviations

With respect to the Publisher, small modifications had to be made, due to the addition of new fields in the structure of the messages throughout the project, but these modifications have not affected the roadmap established for its integration. Regarding the Network Service Repository, it has been necessary to deal with the problems of the JFrog repository pricing model (previously mentioned in Section 2.1.5). The integration with Karmada has had to deal with the different paradigms adopted by each testbed, when implementing their clusters. In addition, small modifications have had to be made in each use case, in order to indicate which resource propagation policy should be followed, depending on which testbed is targeted.

2.3 5G Traffic Simulator

2.3.1 Functional description

Under the term “5G Traffic Simulator”, we refer to a sub-system within the 5G-EPICENTRE architecture, in charge of generating simulated 5G traffic in the platform. It is composed of two functional elements, *i.e.*, the 5GTSM and the Remote iPerf Agents.

The 5GTSM is a simple interface, whose function is to orchestrate the remote iPerf Agents under its domain. This component maintains information about the agents it controls (*e.g.*, address, id, *etc*.), and sends REST API requests to them. The remote iPerf Agents are the components in charge of traffic generation, using the iPerf tool[3] (both version 2 and version 3). They are able to act as client or server, and generate traffic with the desired specifications. They are also able to publish such measurements about network traffic in the RabbitMQ queue, indicated in the established format.

Ideally, the 5GTSM will be maintained as a static, centralized element within the testbed, and Agents will act within containers dynamically.

2.3.2 Information flow

The 5GTSM is instantiated on each testbed as a service, alongside the Publisher. When the testbed receives the request to generate traffic, it sends to the remote iPerf Agents (instantiated in the testbed in the form of docker containers) the request to generate traffic with the specified configuration. Finally, the result of the measurements about the generated traffic are published in the RabbitMQ queue of each testbed, through the Publisher component.

[1] https://kubernetes.io/docs/reference/kubectl/
[2] https://karmada.io/docs/next/reference/karmadactl/karmadactl-commands/karmadactl_index/
[3] https://iperf.fr/

The graphical representation of the described 5G Traffic Simulator internal and external data flows, is illustrated in Figure 3.

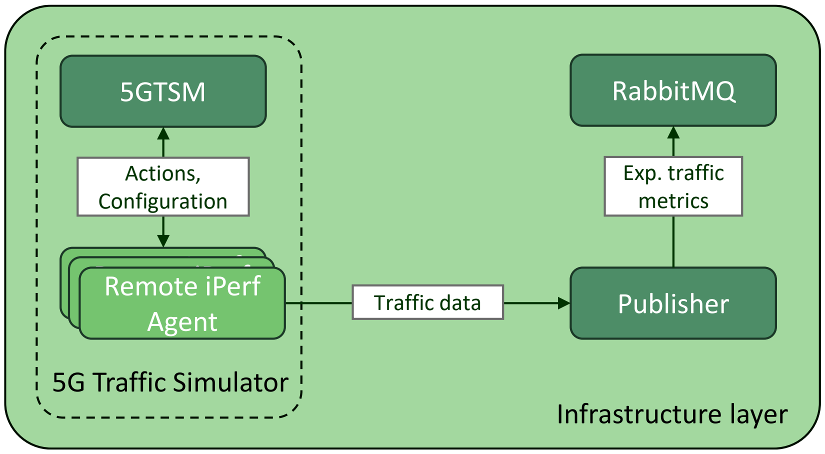

Figure 3: 5G Traffic Simulator data flow.

2.3.3 Interdependencies with other components

The 5G Traffic Simulator (a system developed by UMA) is instantiated in each testbed federated under the 5G-EPICENTRE platform. It has interdependencies with the testbed on which it is hosted, and the Publisher that is hosted on the corresponding testbed (UMA). Furthermore, its internal architecture entails interdependencies between the 5GTSM and the remote iPerf Agents it controls. Table 11 provides an overview of the connections established between the 5G Traffic Simulator (as a sub-system of the 5G-EPICENTRE platform) with its interdependent components, briefly elaborating on their interfaces. Internal integration points occurring within the 5G Traffic Simulator are listed in Table 12.

Table 11: 5G Traffic Simulator connection with other 5G-EPICENTRE components

Component	Connection	API protocol	Data exchange	Comments
Testbed	TCP/IP	REST	JSON	The 5GTSM exposes endpoints for starting the remote iPerf Agents already instantiated in the testbed. The Agent is identified by an id, previously stored in the 5GTSM, as well as its address and parameters. The testbed is also capable of adding new remote iPerf Agents, as well as removing them from the 5GTSM.
Publisher	TCP/IP	MQTT	JSON	The integration with the Publisher is done through the remote iPerf Agents. These Agents capture the console lines resulting from traffic generation, and send them to the Publisher, which is in charge of the publication of those messages in the RabbitMQ queue in the corresponding testbed.

Table 12: 5GTSM connection with other 5G-EPICENTRE components

Component	Connection	API protocol	Data exchange	Comments
Remote iPerf Agents	TCP/IP	REST	API	The integration of the 5GTSM with the remote iPerf Agents is done by means of a POST request, from the 5GTSM containing a JSON with the necessary configuration, to the /Iperf endpoint of each Agent. This JSON contains information about the iPerf parameters that the Agent will use (used to establish the client or server role, among others), as well as the Agent's credentials.

2.3.4 Integration roadmap

Table 13 below, describes the integration roadmap followed for the integration between the 5G Traffic Simulator sub-system and the testbeds, along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 13: 5GTSM – Testbeds final integration report

Roadmap	Description
Integration format	REST API integration.
Integration activities	• Development of the 5GTSM (UMA). • Integration of the 5GTSM in each testbed belonging to the platform (UMA, all testbed owners). • Testing the addition and removal of Agents in the 5GTSM (UMA). • Testing of REST API calls to Agents (UMA). • System integration test (UMA, all testbed owners).
Integration testing	Section 3.3.1

Table 14 similarly describes the integration roadmap followed for the integration between the Remote iPerf Agents and the Publisher, along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 14: Remote iPerf agents – Publisher final integration report

Roadmap	Description
Integration format	REST API integration.
Integration activities	• Development of the Remote iPerf Agents (UMA). • Integration of the Agents in the testbeds (UMA, all testbed owners). • Testing of the communication between the Agents acting as client, and the agents acting as server (UMA). • System integration test (UMA, all testbed owners).

Integration testing	Section 3.3.2

Finally, Table 15 below, describes the integration roadmap followed for the integration between the 5GTSM and the remote iPerf Agents (internal integration for the 5G Traffic Simulator sub-system), along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 15: 5GTSM – Remote iPerf agents final integration report

Roadmap	Description
Integration format	REST API integration.
Integration activities	• Development of 5GTSM (UMA). • Development of the Remote iPerf Agents (UMA). • Correct identification of agents by the 5GTSM (UMA). • Testing of the correct sending and reception of the parameters for traffic generation by the 5GTSM and the Agents (UMA). • Establishment of clients and servers between Agents by the 5GTSM, and generation of traffic between them (UMA). • System integration test (UMA, all testbed owners).
Integration testing	Section 3.3.3

2.3.5 Issues encountered and roadmap deviations

Regarding the remote iPerf Agents, some problems have been encountered when handling the format provided by the iPerf tool in its version 2 and 3. In order not to affect the roadmap, it was decided to program the agents to support both formats, *i.e.*, to select the version of the tool to be used (this can be done when instantiating the Agents).

Another issue encountered with the Agents is the need to create "ephemeral" Agents, since they should not save the state from one experiment to another. For this reason, we have chosen to integrate the agents in the form of docker containers. Therefore, it is possible to add the desired configuration when instantiating them in the selected testbed.

Finally, a problem has been encountered when trying to run the Agents on mobile devices. To solve this, we have made use of an app developed by UMA, that will be executed through the OpenTAP[4] tool, and which has the necessary plugins for the publication of messages in the RabbitMQ queue, in the correct format.

2.4 Karmada (Federation Layer)

2.4.1 Functional description

The 5G-EPICENTRE Federation Layer, leveraging Karmada as its core component, orchestrates distributed resources across multiple Kubernetes (K8s) clusters in various testbeds. Karmada is essential for managing cluster lifecycles and resources, and functions as a multi-cluster management solution. It enables seamless operation and management of cloud-native applications across geographically distributed environments, without the need

[4] https://opentap.io/

for changes to the applications. Each testbed is treated as a point-of-presence, with the Federation Layer facilitating integrated and unified management through standard APIs.

2.4.2 Information flow

As previously mentioned, the Experiment Coordinator triggers the Karmada API server to deploy the Helm chart of the Network Service (NS) to the designated cluster. Upon receiving the deployment manifest, Karmada's binding controllers generate the appropriate binding object. Then Karmada's scheduler processes the workload, based on active plugin(s). Note that, the service placement component within 5G-EPICENTRE is comprehensively detailed in D2.4, and is beyond the scope of this Section.

Once Karmada identifies the target testbed and cluster, it creates the deployment manifest. Following this, the API Server of the K8s cluster in the targeted testbed is invoked, to initiate the deployment. The selection of the desired node within the K8s cluster is managed internally by K8s.

The graphical representation of the described data flow is illustrated in Figure 4.

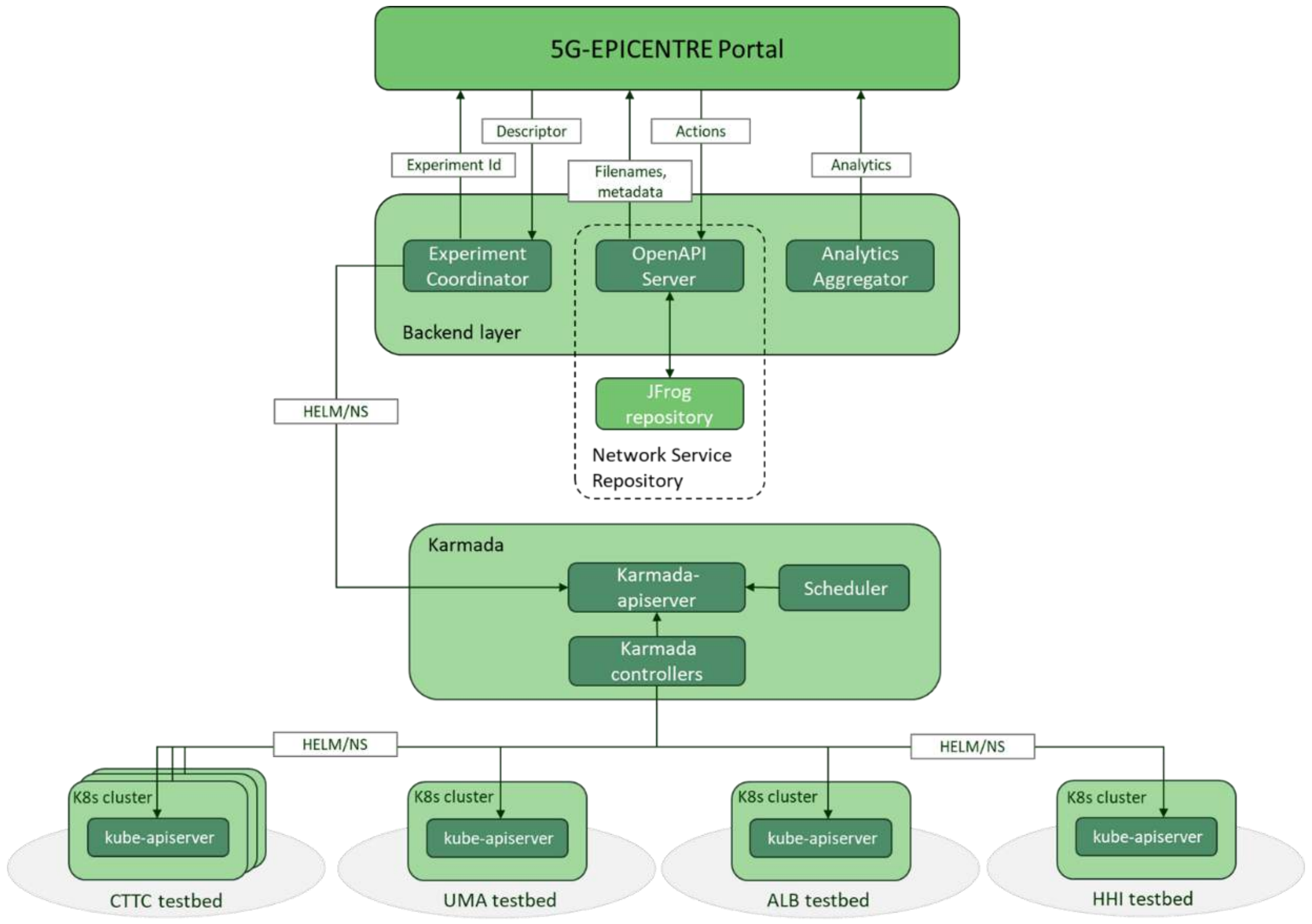

Figure 4: Karmada Federation Layer data flow.

2.4.3 Interdependencies with other components

For seamless deployment from the Portal to the testbed cluster, cross-testbed federation uses a two-tiered interconnection system. The upper layer involves the Experiment Coordinator consuming the Karmada API for deploying services. Meanwhile, the lower layer focuses on integrating with each individual testbed.

As discussed in D4.5, the federation integrates four geographically dispersed testbeds within the project. Each testbed has K8s cluster(s), each possessing different physical characteristics. These clusters are connected to Karmada in a 'push mode', wherein Karmada actively monitors the clusters' statuses, and deploys manifests. This interaction predominantly occurs through Karmada's API Server and its controllers, which maintain direct communication with the K8s API servers of the affiliated clusters.

This architecture adopts a centralized approach, with the Karmada control plane exerting direct influence, and prompt responsiveness over the member clusters. The federation layer is deployed at the CTTC testbed, and to ensure the smooth coordination and communication between CTTC and each remote testbed, a Virtual Private Network (VPN) is established with each partner. The control plane is tasked with the distribution of workloads, enforcement of policies, and the maintenance of the intended state across federated resources in each cluster.

Table 16 below, provides an overview of the connections established with other platform functional elements, and briefly elaborates on their interfaces.

Table 16: Karmada connection with other 5G-EPICENTRE components

Component	Connection	API protocol	Data exchange	Comments
Testbed K8s cluster(s)	TCP/IP	REST	YAML/JSON	The K8s cluster(s) in each testbed is registered, as member cluster, to the federation. Therefore, the status of each cluster is regularly updated within Karmada. Additionally, Karmada propagates the deployments across the member clusters.

2.4.4 Integration roadmap

Table 17 below, describes the integration roadmap followed for the integration between the Karmada Federation Layer and the K8s cluster in the 5G-EPICENTRE testbeds, along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 17: Karmada – Testbed K8s cluster final integration report

Roadmap	Description
Integration format	Karmada API integration.
Integration activities	• Set up Karmada control plane in CTTC testbed (CTTC). • Configure necessary credentials of each testbeds’ K8s cluster(s), for authentication and authorization (CTTC, UMA, ALB, HHI). • Set up a secure communication channel via proper VPN connections between Karmada and each of the cluster testbeds (UMA, ALB, HHI). • Join testbeds’ cluster(s) to the federation environment (CTTC). • Conduct checks to ensure that each Kubernetes cluster has correctly joined the Karmada control plane (CTTC).
Integration testing	Section 3.4.1

2.5 Service placement

2.5.1 Functional description

The 5G-EPICENTRE service placement module is in charge of selecting the best testbed's cluster to host the service, while the service requirement is fulfilled. For the final integration of service placement, the measurement approach has been streamlined, by evaluating both latency and available CPU resources in the topper component, which allows more integrated and holistic understanding of system performance.

The module resides in the cross-testbed federation layer in the form of newly drafted plugin for the Karmada scheduler, to process the workload based to optimization approach (more details can be found in deliverable D2.4). Because of this dedicated development, the integration and testing of this plug-in are treated in separate (to Karmada) Sections.

2.5.2 Information flow

Upon receipt of the deployment request by Karmada from the Experiment Coordinator (or an admin/user in test scenarios), Karmada's internal components initiate the processing of the workload. As the process unfolds, the Scheduler is activated, which in turn triggers the cluster resource plugin. The plugin extracts the necessary information from Karmada's APIs. For the specific latency-aware scenario, the measured latency to the cloud cluster is measured periodically by a metrics tracker (see D2.4), and published via RabbitMQ in the 'application' topic exchange. The plugin can subscribe to the topic and pull the metrics.

Specifically, a metrics tracker (referenced in D2.4) periodically measures latency to the cloud cluster and publishes this data to the 'application' topic on a RabbitMQ topic exchange queue. The plugin subscribes to this topic to access the metric data. After gathering all essential information, the plugin makes an HTTP request to the Optimizer, which houses an ILP solver. This solver calculates the most suitable cluster for hosting the service in question. Then, the Optimizer returns the target cluster of the service at request, to the plugin for the subsequent steps.

The graphical representation of the described data flow is illustrated in Figure 5.

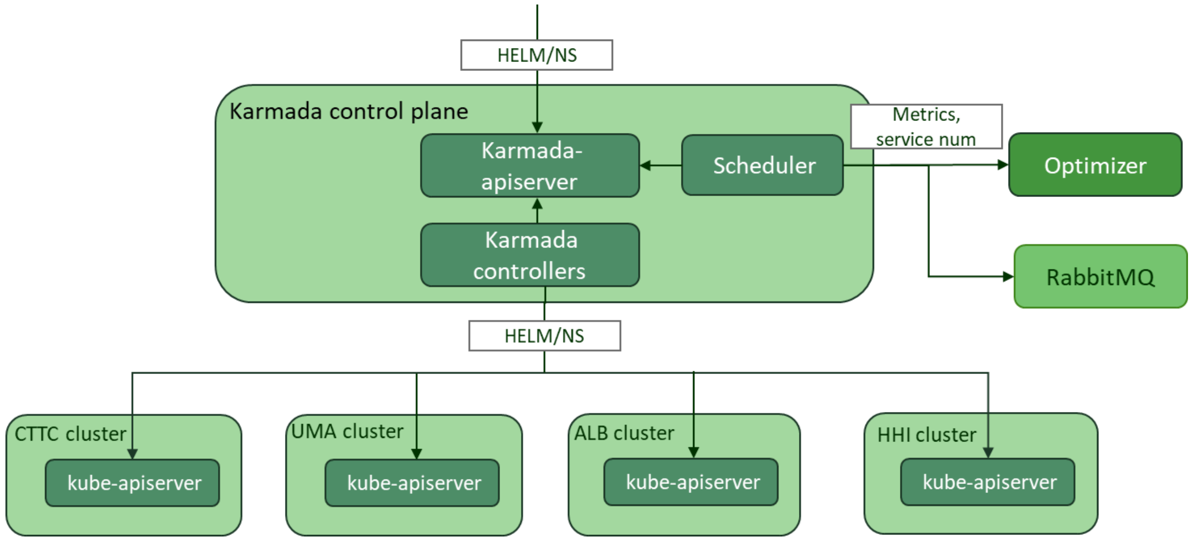

Figure 5: Service placement data flow.

2.5.3 Interdependencies with other components

The service placement module, developed using the plugin approach as new feature for the Karmada federation, has connectivity with the Karamad API server. Table 18 below, provides an overview of the connections established with other functional elements, and briefly elaborates on their interfaces.

Table 18: Service placement plugin connection with other components

Component	Connection	API protocol	Data exchange	Comments
Karmada	TCP/IP	REST	JSON	The service placement is developed as a new plugin (cluster resource) for Karmada's internal scheduler component in the federation layer. The plugin consumes the APIs provided by Karmada system to access the necessary information related to the member clusters.

2.5.4 Integration roadmap

Table 19 below, describes the integration roadmap followed for the integration between the service placement plugin and Karmada, along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 19: Service placement– Karmada final integration report

Roadmap	Description
Integration format	REST API integration
Integration activities	• Deployment of the Karmada as cross-testbed federation (CTTC). • Development the first version of cluster resource plugin (CTTC). • Testing service deployment (CTTC, all testbeds).
Integration testing	Section 3.5.1

2.5.5 Issues encountered and roadmap deviations

Initially, an incompatibility of Filter interface between the developed plugin and the new release of Karmada was identified, leading to errors during the image creation process for the Scheduler. The plugin has been updated to adapt with the new interface.

2.6 Analytics Engine

2.6.1 Functional description

The Analytics Engine is composed of three key modules deployed at each testbed: the Analytics Driver, the KPI Monitor, and the Quality of Service/Quality of Experience (QoS/QoE) Monitor. The Analytics Driver collects and pre-processes data generated by infrastructure and vertical applications, validates the data, and records it into an InfluxDB. The KPI Monitor and the QoS/QoE Monitor then process this data for KPI calculation and Deep Learning (DL)-based analysis for anomaly detection. The results of the data analytics tasks, performed at the testbed level, are then provided to the Analytics Aggregator module, deployed at the Back-end Layer. The complete functional description of the Analytics Engine is elaborated in deliverable D2.6 *"5G-EPICENTRE Analytics Engine"*.

2.6.2 Information flow

When an experiment is conducted on the 5G-EPICENTRE Platform, the Analytics Driver, which is part of the Analytics Engine, subscribes to the Publisher's message queue and receives both metrics and metadata. After validating the data, the Analytics Driver further publishes metrics to both the KPI Monitor and the QoS/QoE Monitor, which are other internal modules of the Analytics Engine. These modules are responsible for calculating and evaluating KPIs and identifying anomalies on the network metrics, based on the measurements. The analytics results are then provided to the Analytics Aggregator at the Back-end Layer.

The graphical representation of the described data flow is illustrated in Figure 6.

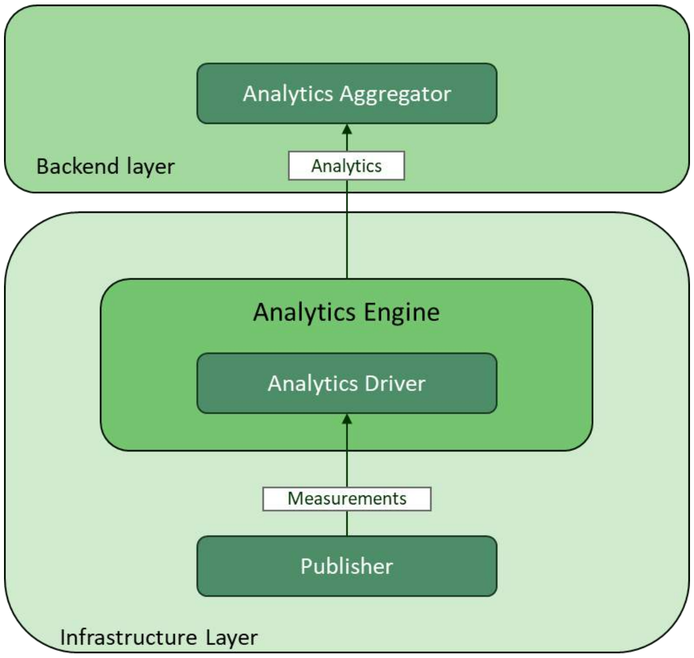

Figure 6: Analytics Engine data flow.

2.6.3 Interdependencies with other components

The Analytics Driver, a sub-component of the Analytics Engine (developed by IST), is interconnected with the Publisher (developed by UMA), a component present in each Testbed. The Publisher's role is to link various metrics with the appropriate experiment metadata for identification, to supply data to the Analytics Engine for analysis. It uses a message broker (*e.g.*, RabbitMQ), to publish metrics and metadata to a common topic exchange.

Table 20 below, provides an overview of the connections established by the Analytics Engine with other platform functional elements, and briefly elaborates on their interfaces.

Table 20: Analytics Engine connection with other 5G-EPICENTRE components

Component	Connection	API protocol	Data exchange	Comments
Publisher	TCP/IP	MQTT	JSON	The Analytics Driver is subscribed to the topic exchange queue published to by the Publisher, thereby asynchronously

				receiving payloads corresponding to the measurements generated in real-time for a particular experiment under test (measurements from the vertical applications, or the network; and security reports generated by the HSPF Network Application components – see also D2.8 *"Cloud-native Security Specifications Final Version"*).
Analytics Aggregator	TCP/IP	MQTT	JSON	The Analytics Engine components at each testbed, publish the processed analytical information (KPIs, statistics, HSPF reports and notification of anomalies) to a topic exchange queue subscribed to by the Analytics Aggregator.

2.6.4 Integration roadmap

Table 21 below, describes the integration roadmap followed for the integration between the Analytics Engine modules and the Publisher at each testbed, along with listing 5G-EPICENTRE partners' responsibilities in each step.

Table 21: Analytics Engine – Publisher final integration report

Roadmap	Description
Integration format	RabbitMQ MQTT broker topic exchange integration.
Integration activities	• Setup RabbitMQ MQTT broker (IST, UMA, Testbed owners). • Development of the first version of the data model for the output of the Publisher (IST, UMA, UC owners). • Integration with actual/real endpoints (IST, UMA, Testbed owners). • Testing of the communication via established endpoints using mock data (IST, UMA, Testbed owners). • Development of a refined version of the data model for the output of the Publisher (IST, UMA, UC owners). • System integration test (IST, UMA, Testbed owners). • Development of refined final version of the data model for the output of the Publisher (IST, UMA, UC owners). • Final system integration test (IST, UMA, UC Owners, Testbed owners).
Integration testing	Section 3.6.1

The roadmap for the integration between the components of the Analytics Engine deployed at each testbed, and the Analytics Aggregator module deployed at the Back-end layer is described in Section 2.7.

2.6.5 Issues encountered and roadmap deviations

During the integration of the Analytics Engine and the Publisher, minor challenges were encountered. However, these were resolved without causing significant disruptions to the integration plans. The main issue involved the need to modify the format of the messages exchanged on RabbitMQ. This change was necessary to accommodate more detailed information. Despite these challenges, the integration process was successfully carried out.

2.7 Analytics Aggregator

2.7.1 Functional description

The Analytics Aggregator (or simply "Aggregator") module is an external component linked to the 5G-EPICENTRE Analytics Engine, and which is deployed at the Back-end Layer (hosted at the UMA testbed). It is responsible for gathering data produced by the components of the Analytics Engine, that are installed at each Testbed (*i.e.*, part of the Infrastructure Layer), and make them available to the 5G-EPICENTRE Portal for visualisation (in the Front-end Layer). The complete functional description of this component is elaborated in deliverable D2.6 *"5G-EPICENTRE Analytics Engine"*.

2.7.2 Information flow

When an experiment is conducted on the 5G-EPICENTRE Platform, the Aggregator consolidates the processed data from both the KPI Monitor and QoS/QoE Monitor at each testbed. This data is then supplied to the Front-end tool for visualization purposes. In more detail, the KPI Monitor forwards the calculated KPIs and statistics, based on the measurements from the vertical applications and infrastructure probes, to the Aggregator. Concurrently, the QoS/QoE Monitor carries out anomaly detection on the infrastructure data, and alerts the Aggregator about any potential anomalies that are detected.

The graphical representation of the described data flow is illustrated in Figure 7.

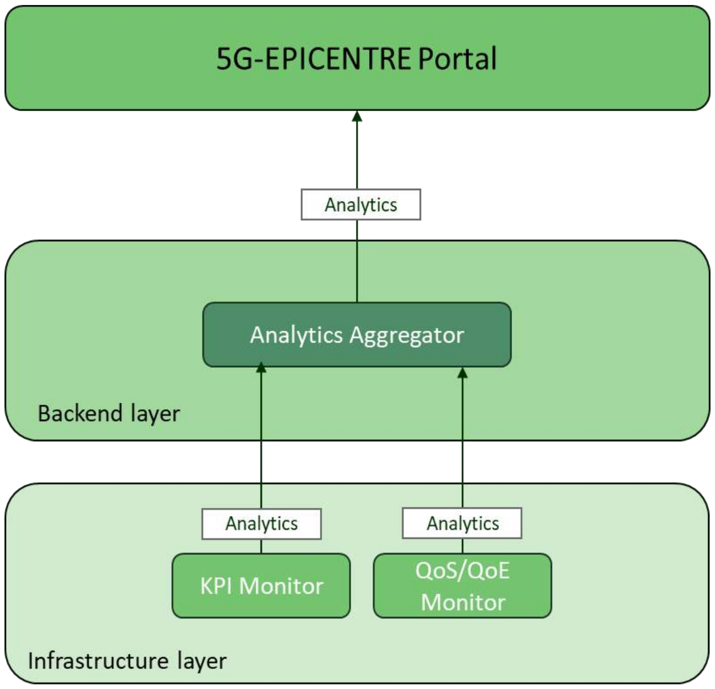

Figure 7: Analytics Aggregator data flow.

2.7.3 Interdependencies with other components

The Analytics Aggregator (a component developed by IST) shares interdependencies with the KPI Monitor and the QoS/QoE Monitor, which are two internal modules of the Analytics Engine deployed at each testbed; and with the 5G-EPICENTRE Portal (see Section 2.1).

Table 22 below, provides an overview of the connections established with the other platform functional elements, and briefly elaborates on their interfaces.

Table 22: Analytics Aggregator connection with other 5G-EPICENTRE components

Component	Connection	API protocol	Data exchange	Comments
KPI Monitor	TCP/IP	MQTT	JSON	The Aggregator is subscribed to the topic exchange queue published to by the KPI Monitor, thereby asynchronously receiving payloads corresponding to the processed analytics information generated in real-time for a particular experiment under test (KPIs from the vertical applications or the network infrastructure, statistics and security reports generated by the HSPF).
QoS/QoE Monitor	TCP/IP	MQTT	JSON	The Aggregator is subscribed to the topic exchange queue published to by the QoS/QoE Monitor, thereby asynchronously receiving payloads corresponding to the notifications of potential anomalies detected on network measurements.
Portal	TCP/IP	MQTT	JSON	The Aggregator publishes the processed analytical information to a topic exchange queue subscribed to by the Portal.

2.7.4 Integration roadmap

Table 23 below, describes the integration roadmap followed for the integration between the Aggregator and the KPI Monitor, along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 23: Aggregator – KPI Monitor final integration report

Roadmap	Description
Integration format	RabbitMQ MQTT broker topic exchange integration.
Integration activities	• Setup RabbitMQ MQTT broker (IST, UMA). • Development of the first version of the data model for the output of the KPI Monitor (IST).

	• Integration with actual/real endpoints (IST). • Testing of the communication via established endpoints using mock data (IST). • Development of a refined version of the data model for the output of the KPI Monitor (IST). • System integration test (IST, UMA). • Development of refined final version of the data model for the output of the KPI Monitor (IST, UC owners). • Final system integration test (IST, UMA, FORTH, UC Owners, testbed owners).
Integration testing	Section 3.7.1

Table 24 similarly describes the roadmap followed for the integration between the QoS/QoE Monitor and the Aggregator, along with listing 5G-EPICENTRE partner responsibilities in each step.

Table 24: Aggregator – QoS/QoE Monitor final integration report

Roadmap	Description
Integration format	RabbitMQ MQTT broker topic exchange integration.
Integration activities	• Setup RabbitMQ MQTT broker (IST, UMA). • Development of the first version of the data model for the output of the QoS/QoE Monitor (IST). • Integration with actual/real endpoints (IST). • Testing of the communication via established endpoints using mock data (IST). • Development of a refined version of the data model for the output of the QoS/QoE Monitor (IST). • System integration test (IST, UMA). • Development of refined final version of the data model for the output of the QoS/QoE Monitor (IST). • Final system integration test (IST, UMA, FORTH).
Integration testing	Section 3.7.2

Finally, the roadmap followed for the integration between the 5G-EPICENTRE Portal and the Analytics Aggregator is described in Section 2.1.4.

2.7.5 Issues encountered and roadmap deviations

We faced some minor challenges, but managed to resolve them without major disruptions to our integration plans. For example, to ensure the connectivity of the Aggregator with both the Portal (deployed in the Front-end) and the KPI Monitor and QoS/QoE Monitor (deployed in the Infrastructure Layer) it was necessary to:

1. make the Aggregator use a public Internet Protocol (IP) address (provided by UMA, where the module is hosted);
2. adjust the network policies on the testbeds, in order to enable the KPI Monitor and QoS/QoE Monitor to access the Aggregator RabbitMQ MQTT broker exchange.

3 5G-EPICENTRE system testing and validation report

Testing and validation activities run in parallel to the iterative module integration processes with the aim to support them, by validating component functionality at both the unit and (sub)system level. In the context of Task 4.5, testing focuses at establishing a feedback loop with the individual Task in charge of component / service / module development, to ensure that testing results are used towards refining technical development and introducing enhancements to the overall 5G-EPICENTRE infrastructure.

At the unit level, activities carried out in the context of Task 4.5 receive input from the 5G-EPICENTRE testing framework elaborated in D4.4 (Section 5), which outlines the provisions in accordance to which the automated and manual testing activities take place. At the core of the unit testing lies the test documentation (D4.6), which in 5G-EPICENTRE has produced both a test plan document and test cases documentation, covering how tests for individual components will be carried out to validate both components in accordance to the test plan guidelines.

At the system level, in this Section we describe the approach to the final **integration** and **system-level** tests of each of the components elaborated in Section 2. To describe the ways in which the APIs were tested, Table 25 delivers a test matrix, which elaborates, in a unified manner, the kind of tests carried out.

Table 25: Test matrix listing the types of functional testing carried out, and reported in this Section.

	Testing request in isolation	Test a series of requests in sequence (*i.e.*, check that the response of one request is used as param to another request)	Manual testing with UI (*e.g.*, curl, postman, Portal) (data integrity and consistency check)
Basic positive test: check basic (*i.e.*, expected) functionality and acceptance criteria	Single API request with correct payload, to check that the response is the one expected.	Chain of several API requests one after another, with each having the correct payload, to check that the chain of requests yields the expected behaviour.	Single API request with correct payload using a UI tool to make/visualize the request, to check that even different systems still behave as expected when invoking the API.
Extended positive test: same as above, but with additional optional parameters and functionalities	Single API request tested several times with different correct payloads, to check that each yields the response expected.	Chain of several API requests one after another, with each having different (but correct) payloads, to check that each chain of requests yields the expected behaviour.	Single API request tested several times with different correct payloads using a UI tool to make/visualize the request, to check that even different systems still behave as expected when invoking the API.
Negative testing: test API with valid or invalid input to check graceful handling of errors	Single API request with incorrect (*e.g.*, format is correct, but value is not correct) or invalid (wrong format of the request parameters, or typo) payload, to	Chain of several API requests one after another, with one or more having incorrect (*e.g.*, format is correct, but value is not correct) or invalid (wrong for-	Single API request with incorrect (*e.g.*, format is correct, but value is not correct) or invalid (wrong format of the request parameters, or typo) payload, us-

	check that the API gracefully handles the error.	mat of the request parameters, or typo) payload, to check that the API gracefully handles the error.	ing a UI tool to make/visualize the request and check that the API gracefully handled the error.
Destructive testing: intentionally try to break the API e.g., send huge payload)	Single API request with anomalous payload, to check robustness of the API.	Chain of several API requests one after another, with one or more having anomalous payload, to check robustness of the API.	Single API request with anomalous payload, using a UI tool to check robustness of the API.

The following Sections present all the various test cases for the integration between components identified and listed in Section 2. Each test case is identified in the following manner:

$$\boldsymbol{M}(.\boldsymbol{m})$$

The M number corresponds to the incremental number of the test case for the component. If the test resulted in a PASS test result, only this number is listed. For every FAIL test case (until the PASS mark is achieved), the m number is used to count the number of tries (after contingency action was taken). For each FAIL mark given, we provide specific details to the failure encountered, its identified cause, and countermeasure applied.

3.1 5G-EPICENTRE Portal functionality validation

5G-EPICENTRE Portal validation testing has been carried out ad-hoc, by checking the developed solution for integrity and stability of interdependent component APIs integration (see Section 2.1.3). It further aimed at testing system usability and user experience, by emulating the intended way in which the software is meant to be used by its intended end users. A black-box testing approach was followed, by sending the requests to each API and verifying that the expected output is received.

3.1.1 Network Service Repository integration testing

Table 27 below, describes the integration tests carried out between the 5G-EPICENTRE Portal and the Network Service Repository, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 27: 5G-EPICENTRE Portal – Network Service Repository integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1	Extended positive test of 'Get All' endpoint in isolation (using a single API call), using Postman for data model integrity and consistency check.	Authentication header must be setup manually in Postman, using the test username and password credentials provided by IQU.	The response should contain the list of filenames in the repository (200 status code).	1) PASS
2.1 **2.2**	Extended positive test of 'Get File endpoint in isolation (using a single API call), using Postman,	Authentication header must be setup manually in Postman, using the	The response should contain the contents of the "Chart.yaml" file (200 status code).	1) FAIL 2) PASS

	for data model integrity and consistency check.	test username and password credentials provided by IQU. The file must be in an archive (*e.g.*, .zip) file format.		
3.1 **3.2**	Basic positive test of 'Get All' endpoint in isolation (using a single API call), using 5G-EPICENTRE Portal UI environment to both execute requests via the API, and validate the response payload (data model integrity and consistency check).	User must be authenticated in the Portal as an experimenter, and must navigate to the third tab of the "Create a New Experiment" page, where the relevant UI is available.	The Portal should receive the list of filenames in the response body, and thereby construct a visual list of available artefacts in the UI.	1) FAIL 2) PASS
4	Basic positive test of 'Get File' endpoint in isolation (using a single API call), using 5G-EPICENTRE Portal UI environment to both execute requests via the API, and validate the response payload (data model integrity and consistency check).	User must be authenticated in the Portal as an experimenter, and must navigate to the third tab of the "Create a New Experiment" page, where the relevant UI is available. YAML file of the Helm chart must be in the correct format (listing services, etc.).	The Portal should receive the contents of the "Chart.yaml" file in the response body, and thereby construct a visual record of the Helm artefact upon selection (select metadata should be visible in the record).	1) PASS
5	Multi-step basic positive test of 'Put File' and 'Get All' endpoints using 5G-EPICENTRE Portal UI environment to execute chain of requests via the API, and validate the response payload (consistency check, *i.e.*, success of the first request can be verified in the second request).	User must be authenticated in the Portal as an experimenter, and must navigate to the 'Delegate Artefact' page, and submit an artefact via the UI form.	After executing the test, and by navigating to the third tab of the "Create a New Experiment" page, the user can verify that the Helm chart uploaded via 'Put File' is among the results returned in the list of filenames in the 'Get All' request response.	1) PASS
6	Multi-step basic positive test of 'Delete File' and 'Get All' endpoints using	User must be authenticated in the Portal as an experimenter, and must	After executing the test, and by navigating to the third tab of the "Create	1) PASS

	5G-EPICENTRE Portal UI environment to execute chain of requests via the API, and validate the response payload (consistency check, *i.e.*, success of the first request can be verified in the second request).	navigate to the 'My Artefacts' page, and delete a specific artefact using the UI button provided.	a New Experiment" page, the user can verify that the Helm chart deleted via 'Delete File' is no longer among the results returned in the list of filenames in the 'Get All' request response.	

Problems encountered, which led to specific tests failing, are reported in Table 28.

Table 28: 5G-EPICENTRE Portal – Network Service Repository integration testing fail cases.

Test case no	Failure encountered	Cause	Contingency
2.1	Response payload contains wrong metadata information.	Incorrect YAML file format in the Helm chart package being retrieved.	Ensured all Helm charts are uploaded with specific "Chart.yaml" file structure (added as a prerequisite for all ensuing tests). This proper structure must be communicated to experiment applicants for their experiment artefact upload.
3.1	The request is not allowed by the server's Cross-Origin Resource Sharing (CORS) configuration.	CORS request is missing the required 'Access-Control-Allow-Origin' header.	Configured API server to return the HTTP headers required by the CORS standard.

3.1.2 Experiment Coordinator integration tests

Table 29 below, describes the integration tests carried out between the 5G-EPICENTRE Portal and the Experiment Coordinator, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 29: 5G-EPICENTRE Portal – Experiment Coordinator integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1	Extended positive test of 'Experiment Run' endpoint in isolation (using a single API call), using Postman for data model integrity and consistency check.	None.	The response should contain the execution id of the experiment that the Coordinator has queued (200 status code).	1) PASS
2.1 **2.2**	Extended positive test of 'Experiment Run' endpoint in isolation	The user must first authenticate in the Portal as an experimenter, and	The Portal should redirect the user to the 'My experiments' page,	1) FAIL 2) PASS

	(using a single API call), using 5G-EPICENTRE Portal UI environment to both execute the request via the API, and validate the response payload (data model integrity and consistency check).	must complete the "Create a New Experiment" process, and send the request by clicking on the submit button. The user must then authenticate in the Portal as a testbed administrator (for the corresponding testbed), and must 'accept' the submitted request via the reviewing interface (see also D3.2).	where the new experiment is shown as 'Accepted'. By checking the browser console, the returned execution id of the experiment that the Coordinator has queued, should be logged.	
3	Basic positive test of 'Experiment Cancel' endpoint in isolation (using a single API call), using Postman for data model integrity and consistency check.	None.	The response should contain the execution id of the experiment that the Coordinator has queued (200 status code).	1) PASS
4	Basic positive test of 'Experiment Cancel' endpoint in isolation (using a single API call), using 5G-EPICENTRE Portal UI environment to both execute the request via the API, and validate the response payload (data model integrity and consistency check).	The user must authenticate in the Portal as either an experimenter, or as a testbed administrator, and must navigate to the 'Experiments' page where the experiment can be cancelled with the corresponding button.	The Portal should refresh the 'My experiments' page, where the user can verify that the experiment is no longer among the list entries. By checking the browser console, the response code should be 200.	1) PASS

Problems encountered, which led to specific tests failing, are reported in Table 30.

Table 30: 5G-EPICENTRE Portal – Experiment Coordinator integration testing fail cases.

Test case no	Failure encountered	Cause	Contingency
2.1	The request is not allowed by the server's Cross-Origin Resource Sharing (CORS) configuration.	CORS request is missing the required 'Access-Control-Allow-Origin' header.	Configured API server to return the HTTP headers required by the CORS standard.

3.1.3 Analytics Aggregator integration tests

Table 31 below, describes the integration tests carried out between the 5G-EPICENTRE Portal and the Analytics Aggregator, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 31: 5G-EPICENTRE Portal – Analytics Aggregator integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1.1 **1.2**	Basic positive test of subscription to the RabbitMQ broker, for receiving mock analytics data payloads via MQTT.	RabbitMQ must be up and running on the host testbed. Some kind of metrics generation service (serving mock data) must be up and running on the host testbed	Connection successfully established. Mock data payloads must be received and logged in the Portal's backend console.	1) FAIL 2) PASS
2	Basic positive test of subscribing to the actual endpoint topic exchange, for receiving real analytics data payloads via MQTT and displaying them as visual graphs in the Experiment Insights page.	RabbitMQ must be up and running on the host testbed. UC must be deployed, up and running on the host testbed.	Analytics payloads must be received and logged in the Portal's backend console. Experiment report document is created inside the Portal's Mongoose database (see D3.2). Visualization components must be automatically created inside the Experiment Insights page.	1) PASS

Problems encountered, which led to specific tests failing, are reported in Table 32.

Table 32: 5G-EPICENTRE Portal – Analytics Aggregator integration testing fail cases.

Test case no	Failure encountered	Cause	Contingency
1.1	Connection is not established.	The specific MQTT library imported into the Portal backend is either incompatible with RabbitMQ (AMQ-based), or has unspecified issue to connect to the broker.	Replace the library.

3.2 Experiment Coordinator functionality validation

The validation of the Experiment Coordinator was performed manually. The focus was not so much to test the correct response of the endpoints it offers (*i.e.*, the experiment id in this case), but rather, the different configurations it can execute. For this purpose, tests have been carried out with the different use cases available in each of the associated testbeds. In addition, the correct deployment and operation of the HSPF Network Application has been tested, as well as the correct operation of the different components that interact with the Experiment Coordinator.

3.2.1 5GTSM integration testing

Table 33 below, describes the integration tests carried out between the Experiment Coordinator and the 5GTSM, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 33: Experiment Coordinator – 5GTSM integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1.1 **1.2**	Basic positive test of 'start' endpoint in isolation (using a single API call), using curl and Hoppscotch tools.	The remote iPerf agents that will be involved in traffic generation must be instantiated. The address and Id of these agents should be included in the 5GTSM iperf-host.json file.	The response should establish the server and client roles between the remote iPerf agents involved, and initiate traffic generation between them, with the chosen parameters (200 status code).	1)FAIL 2)PASS
2	Extended positive test of 'start' endpoint in isolation (using a single API call), using Hoppscotch for testing the possible parameters that can be used for the configuration of the remote iPerf agents.	The remote iPerf agents that will be involved in traffic generation must be instantiated. The address and Id of these agents should be included in the 5GTSM iperf-host.json file.	The response should be able to see (in the corresponding RabbitMQ queue) the traffic generation values, indicated in the parameters established.	2)PASS

Problems encountered, which led to specific tests failing, are reported in Table 34.

Table 34: Experiment Coordinator – 5GTSM integration testing fail cases.

Test case no	Failure encountered	Cause	Contingency
1.1	On some platforms, traffic is not displayed on a second-by-second basis, instead traffic is displayed at the end of the experiment run.	Some platforms create a buffer, to store the output of the iPerf command and display it either when the buffer overflows, or when the command ends.	By installing tools on the platform that force the iPerf command to be launched, without using buffers.

3.2.2 Publisher integration testing

Table 35 below, describes the integration tests carried out between the Experiment Coordinator and the Publisher, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 35: Experiment Coordinator – Publisher integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1	Basic positive test of 'Publish' endpoint in isolation (using a single API call), Hoppscotch and curl tools. Assignment of the experiment id, generated by the Experiment Coordinator, to the Publisher's metadata related to the experiment.	The experiment metadata must be included in the Publisher.	The experiment metadata is updated with the experiment id provided, and the KPIs published by the experiment can be viewed in the correct RabbitMQ queue, with that field assigned (200 status code).	1)PASS
2	Extended positive test of 'add experiment' endpoint in isolation (using a single API call), using Hoppscotch and curl for testing the updating of the possible metadata of an experiment.	The Publisher must contain the experiment metadata identified by the *"netapp_id"* field.	The Publisher correctly updates the *"experiment_id"* field in the experiment identified by the field *"netapp_id"*.	1)PASS
3.1 **3.2**	Extended positive test of 'add experiment' endpoint in isolation (using a single API call), using Hoppscotch and curl, for testing the addition of the possible metadata of an experiment.	The Publisher must not contain the metadata of the experiment to be added.	The Publisher adds the new metadata of the experiment in the Publisher, and publishes the messages to the RabbitMQ queue with the correct metadata.	1)FAIL 2)PASS

Problems encountered, which led to specific tests failing, are reported in Table 36.

Table 36: Experiment Coordinator – Publisher integration testing fail cases.

Test case no	Failure encountered	Cause	Contingency
3.1	The Publisher deletes all stored experiments when trying to add the metadata of a new one.	A bug in assigning the list of new experiments. Experiments lists were processed by reference, instead of by value.	The bug has been identified and fixed: the experiment list is locked by value, and then updated.

3.2.3 Network Service Repository integration tests

Table 37 below, describes the integration tests carried out between the Experiment Coordinator and the Network Services Repository, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 37: Experiment Coordinator – Network Service Repository integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1.1 **1.2** **1.3**	Extended positive test of 'Get File' endpoint in isolation (using a single API call), using Hoppscotch, for data model integrity and consistency check.	Authentication header must be setup manually in Postman, using the test username and password credentials provided by IQU. The file must be in an archive (*e.g.*, .zip) file format.	The response should contain the contents of the "Helm-chart.zip" file (200 status code).	1)FAIL 2)FAIL 3)PASS

Problems encountered, which led to specific tests failing, are reported in Table 38.

Table 38: Experiment Coordinator – Network Service Repository integration testing fails cases.

Test case no	Failure encountered	Cause	Contingency
1.1	The file does not download correctly. A corrupted version of the file is downloaded instead.	The file must be in .zip format and the file name must be in the form "{filename}.zip"	Assert that the file needs to have the correct format, as well as the format of its name.
1.2	The file is not decompressed in a folder with name format {filename}	The zip file does not contain a folder with the same name as the file (without the .zip extension). This causes the program to not be able to decompress it correctly.	Assert that the zip file needs to contain a folder with the same filename as the original file.

3.2.4 Karmada integration testing

Table 39 describes the integration tests carried out between the Experiment Coordinator and the Karmada Federation Layer, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 39: Experiment Coordinator – Karmada integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1.1 **1.2** **1.3**	Extended positive test of deploying use cases in the different testbeds	The target testbed must be attached to the Karmada federation layer.	The deployment is propagated in the indicated testbed and namespace, and works as expected.	1)FAIL 2)FAIL 3)PASS

	that make up the platform, as well as the different namespaces that make up the testbeds.	The namespaces must be created on the Karmada master node, and propagated to the joined testbeds.		

Problems encountered, which led to specific tests failing, are reported in Table 40.

Table 40: Experiment Coordinator – Karmada integration testing fail cases.

Test case no	Failure encountered	Cause	Contingency
1.1	Resources are not propagated correctly to the specified testbed and namespace	An extra file called "PropagationPolicy" is needed, to indicate the resource propagation policy.	The file has been added to each Helm chart that wants to interact with the platform, and has been set to be deployed before starting to deploy the rest of resources
1.2	The resources are propagated to the target testbed and namespace, but are not executed correctly.	The use cases do not have the credentials of each testbed for publishing to their RabbitMQ queue. In addition, they do not have the configurations of each testbed for their dynamic storage solutions (*e.g.*, StorageClass)	By modifying the Values.yml files of each Helm chart, the credentials of the RabbitMQ and StorageClass queues can be dynamically set by means of the Experiment Coordinator.

3.3 5G Traffic Simulator functionality validation

The validation of the 5G Traffic Simulator has been performed both manually and automatically. In addition to checking the correct operation of the different endpoints, several 4-hour stress tests were performed on the different endpoints.

3.3.1 Testbed integration testing

Table 41 below, describes the integration tests carried out between the 5GTSM and the testbeds, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 41: 5GTSM – Testbed integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1	Extended positive test of 'add_iperf_agent' endpoint in isolation (using a single API call), using Hoppscotch for testing the possible parameters that can be	None.	The response must contain the success status code. The file "iperf-hosts.json" must contain the credentials of	1)PASS

	used for the configuration of the remote iPerf Agents.		the newly added agent (Status code 200).	
2	Extended positive test of ‘remove_iperf_agent’ endpoint in isolation (using a single API call), using Hoppscotch for testing the possible parameters that can be used for the configuration of the remote iPerf Agents.	The Agent’s credentials must be found in the file “iperf-hosts.json”	The file “iperf-hosts.json” file should no longer contain the credentials of the selected agent.	1)PASS

3.3.2 Publisher integration testing

Table 42 below, describes the integration tests carried out between a remote iPerf Agent and the Publisher, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 42: Remote iPerf agent – Publisher integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1.1 **1.2**	Extended positive test of ‘Publish’ endpoint in isolation (using a single API call), using the remote iPerf Agents for testing the publication of KPIs related to traffic generation.	The Publisher must have access to the RabbitMQ queue, and the Agents must be able to generate traffic, and publish it in the correct format.	The display of messages in the RabbitMQ queue in the correct format, and with the correct identification.	1)FAIL 2)PASS

Problems encountered, which led to specific tests failing, are reported in Table 43.

Table 43: Remote iPerf agent – Publisher integration testing fail cases.

Test case no	Failure encountered	Cause	Contingency
1.1	The Publisher “loses” messages when many agents publish messages for a long time.	The Publisher suffers from “overheating”, when it must handle a large number of messages from several agents over a long period of time. This happens due to the creation of many threads for each one of them.	Making agents publish to a queue with topic “application”, instead of publishing to the Publisher’s endpoint. This frees up endpoint load. The Publisher will subscribe all messages in the queue with topic “application”, and publish them in the correct format to the specified queue,

			without suffering from “over-heating”.

3.3.3 5GTSM – Remote iPerf Agent integration testing

Table 44 below, describes the integration tests carried out between the 5GTSM and the remote iPerf Agents, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 44: 5GTSM – Remote iPerf Agent(s) integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1	Extended positive test of ‘Iperf’ endpoint in isolation (using a single API call), using Hoppscotch for testing the possible parameters that can be used for the configuration of the remote iPerf Agents.	The remote iPerf Agents that will be involved in traffic generation must be instantiated. The address and Id of these Agents should be included in the 5GTSM iperf-host.json file.	The response should establish the server and client roles between the remote iPerf Agents involved, and initiate traffic generation between them with the chosen parameters (200 status code).	1)PASS
2.1 **2.2**	Stress test on the generation of traffic between the different instantiated Agents. Tests were carried out on different traffic profiles for a duration of 4 hours.	The agents must be instantiated, and must have a connection to the Publisher.	The sample of results should be reflected in the corresponding RabbitMQ queue.	1)FAIL 2)PASS

Problems encountered, which led to specific tests failing, are reported in Table 45.

Table 45: 5GTSM – Remote iPerf Agent(s) integration testing fail cases.

Test case no	Failure encountered	Cause	Contingency
2.1	After a period of one hour, the messages stop appearing in the corresponding RabbitMQ queue in real time. After that period, the messages are published in blocks, or are completely lost.	The iPerf command uses a system of buffers for traffic generation. After a period of time, these buffers become saturated, and become susceptible of losing messages, or generating them incorrectly.	Using external tools, such as the Except[5] library (in the case of Linux), or the Winpty[6] tool (in Windows), it is possible to make the iPerf tool not use the buffers causing the issue.

[5] https://linux.die.net/man/1/unbuffer
[6] https://github.com/rprichard/winpty

3.4 Karmada (Federation layer) functionality validation

First, we validated the correct access rights to the clusters via Karmada, which ensure that the clusters were correctly integrated and prepared to host the resources. This involved verifying that access controls and permissions were appropriately set and functioning as intended within the Karmada-managed environment as cluster status synchronization. Additionally, the testing process included the deployment of applications using Helm charts or NS across the clusters. This was achieved by using the 'kubectl' command tool, alongside 'karmadactl', which is a Command Line Interface (CLI) for Karmada control plan. The aim was to validate the reliability of Karmada in managing application lifecycles within a multi-cluster setup.

3.4.1 Kubernetes testbed cluster integration testing

Table 46 below, describes the integration tests carried out between the cross-testbed federation and the testbeds' K8s clusters, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 46: Cross-testbed federation – Testbeds' Kubernetes's cluster integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1	Basic positive test of retrieving Karmada's components in isolation (using single API request), using *kubectl*.	All necessary prerequisites, including the correct versions of Go, *Kubectl*, and *Karmadactl*, are available, and verified for compatibility.	All the components of Karmada are in the *running* status.	1)PASS
2	Basic positive test of 'ping' endpoint in isolation (using a single API call), using *ping* command.	VPN is established between CTTC testbed and each remote testbed in UMA, ALB, and HHI.	Successful ping to Kubernetes cluster in testbeds UMA, ALB, and HHI.	1)PASS
3	Basic positive test of retrieving registered Cluster object in isolation (using single API request), using *kubectl*.	Testcase number1. Testcase number 2. The necessary credentials foraccessing all remote testbeds' cluster(s) must be available.	All testbeds' cluster(s) are correctly registered.	1)PASS
4	Basic positive test of Kubernetes resource deployment in isolation (using a single API call), using *kubectl* and *karmadactl* commands.	The propagation policy object is prepared. The corresponding HELM/NS YAML files for deployment are prepared.	The HELM/NS with corresponding propagation policy are sent to Karmada. The HELM/NS is deployed successfully in the cluster specified in propagation policy.	1)PASS

5	Basic positive test of K8s resource deployment in isolation (using a single API call), using *kubectl* and *karmadactl* commands.	The Preconditions in testcase 4. The desired namespace in the specific testbed's cluster should be created (ALB).	The HELM/NS with corresponding propagation policy are sent to Karmada. The service is correctly deployed in the target namespace, in the specific cluster specified in propagation policy.	1)PASS

3.5 Service placement functionality validation

The process of service placement involved several manual validation test steps. Since this module takes place in Karmada, a crucial step of integrating was the development preliminary structure for the plugin, to allocate space for incorporating essential code. All verification tests were conducted by monitoring the log messages from the Karmada scheduler pod, and performing the cross-validation between the expected target cluster and the host cluster for the currently deployed pod.

3.5.1 Karmada (Federation layer) integration testing

Table 47 below, describes the integration tests carried out between the service placement and Karmada, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 47: Service placement plugin – Karmada (Federation layer) integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1.1 **1.2**	Basic positive test of K8s resource deployment in isolation (using a single API call), using *kubectl apply -f [file]* command.	Since the plugin is integrated in Karmada, it inherits all the prerequisite installation conditions of Karmada. Scheduler image, which contains new plugin, must be available and accessible for pulling from the repository.	Submitting YAML files of a service and propagation policy to Karmada. The expected outcome from this test is to see a running pod in the target cluster, that was fixed for this test.	1) FAILS 2) PASS
2	Basic positive test for the K8s client set initialization in isolation (using several APIs call), by using authentication credential.	All the required Go libraries must be implemented. Network access must be granted to K8s clusters.	The available CPU resources per each member cluster must be shown.	1)PASS

Problems encountered, which led to specific tests failing, are reported in Table 48.

Table 48: Service placement – Karmada integration testing fail cases.

Test case no	Failure encountered	Cause	Contingency
1.1	The pod did not deploy in the cluster.	Incompatibility of Filter interface in developed plugin and new release of Karmada.	The interface was updated with appropriately defined arguments, followed by the creation of a new image.

3.6 Analytics Engine functionality validation

The validation of the integration between the Analytics Engine modules and the Publisher at the testbeds was carried out using a black-box testing approach. This involved sending messages over a RabbitMQ broker exchange, and verifying that the received output matched the expected results.

3.6.1 Publisher integration testing

Table 49 below, describes the integration tests carried out between the Analytics Engine, more specifically the Analytics Driver component, and the Publisher, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 49: Analytics Driver – Publisher integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1	Basic positive test of subscription to the RabbitMQ broker, for receiving mock data payloads via MQTT from the Publisher module.	RabbitMQ must be up and running on the testbed. Some kind of metrics generation service (serving mock data) must be up and running on the testbed.	Connection successfully established. Mock data payloads must be received and printed in the Analytics Driver log.	1) PASS
2	Basic positive test of subscribing to the actual endpoint topic exchange, for receiving real data payloads via MQTT from the Publisher, including measurements about the network infrastructure.	RabbitMQ must be up and running on the testbed. The Analytics Driver must be up and running on the testbed. Measurements from the network infrastructure must be available on the testbed.	Data payloads must be received and printed in the Analytics Driver's log.	1) PASS
3.1 **3.2**	Basic positive test of subscribing to the actual endpoint topic exchange, for receiving	RabbitMQ must be up and running on the testbed.	Data payloads must be received and printed in the Analytics Driver's log.	1) FAIL 2) PASS

	real data payloads via MQTT from the Publisher, including measurements about the vertical application under test.	The Analytics Driver must be up and running on the testbed. UC must be deployed, up and running on the testbed.		

Problems encountered, which led to specific tests failing, are reported in Table 50.

Table 50: Analytics Driver – Publisher integration testing fail cases.

Test case no	Failure encountered	Cause	Contingency
3.1	No data received.	The vertical application was not compliant with the data format required by the Publisher, and the collected data was discarded.	The issue was resolved by correcting the data format.

3.7 Analytics Aggregator functionality validation

The validation of the integration between the Aggregator and the Analytics Engine internal Infrastructure Layer modules at the testbeds, was carried out using a black-box testing approach. This involved sending messages over a RabbitMQ broker exchange, and verifying that the received output matched the expected results.

3.7.1 KPI Monitor integration testing

Table 51 below, describes the integration tests carried out between the Aggregator and the KPI Monitor at the testbed level, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 51: Aggregator – KPI Monitor integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1	Basic positive test of subscription to the RabbitMQ broker, for receiving mock analytics data payloads via MQTT from the KPI Monitor module deployed at UMA testbed.	RabbitMQ must be up and running on the UMA testbed. Some kind of metrics generation service (serving mock data) must be up and running on the UMA testbed.	Connection successfully established. Mock data payloads must be received and printed in the Aggregator log.	1) PASS
2	Basic positive test of subscribing to the actual endpoint topic exchange, for receiving real analytics data payloads via MQTT from the	RabbitMQ must be up and running on the UMA testbed.	Analytics payloads must be received and printed in the Aggregator's log.	1) PASS

	KPI Monitor at UMA testbed.	KPI Monitor must be up and running on the UMA testbed. UC must be deployed, up and running on the UMA testbed.		
3.1 **3.2**	Basic positive test of subscribing to the actual endpoint topic exchange, for receiving real analytics data payloads via MQTT from the KPI Monitor at a different testbed.	RabbitMQ must be up and running on the UMA testbed. KPI Monitor must be up and running on the host testbed to be tested. UC must be deployed, up and running on the host testbed.	Analytics payloads must be received and printed in the Aggregator's log.	1) FAIL 2) PASS

Problems encountered, which led to specific tests failing, are reported in Table 52.

Table 52: Aggregator – KPI Monitor integration testing fail cases.

Test case no	Failure encountered	Cause	Contingency
3.1	No data analytics received from the testbed.	The specific KPI Monitor hosted on the testbed was not able to connect to the Aggregator's RabbitMQ hosted at the UMA testbed.	The network issue preventing the KPI Monitor from connecting to the RabbitMQ on the UMA testbed was resolved by the network administrator, who enabled the module to connect to an external IP.

3.7.2 QoS/QoE Monitor integration tests

Table 53 below, describes the integration tests carried out between the Aggregator and the QoS/QoE Monitor at the testbed level, to verify that the former properly fulfils expected functionality with respect to this interface.

Table 53: Aggregator – QoS/QoE Monitor integration testing.

Test case no	Test scenario and flow	Preconditions	Expected result	Test result
1	Basic positive test of subscription to the RabbitMQ broker, for receiving mock analytics data payloads via MQTT from the QoS/QoE Monitor module deployed at UMA testbed.	RabbitMQ must be up and running on the UMA testbed. Some kind of metrics generation service (serving mock data) must be	Connection successfully established. Mock data payloads must be received and printed in the Aggregator's log.	1) PASS

		up and running on the UMA testbed.		
2	Basic positive test of subscribing to the actual endpoint topic ex-change, for receiving real analytics data pay-loads via MQTT from the QoS/QoE Monitor at UMA testbed.	RabbitMQ must be up and running on the UMA testbed. QoS/QoE Monitor must be up and running on the UMA testbed. UC must be deployed, up and running on the UMA testbed.	Analytics payloads must be received and printed in the Aggregator's log.	1) PASS
3	Basic positive test of subscribing to the actual endpoint topic ex-change, for receiving real analytics data pay-loads via MQTT from the QoS/QoE Monitor at a different testbed.	RabbitMQ must be up and running on the UMA testbed. QoS/QoE Monitor must be up and running on the host testbed to be tested. Network measurements must be available at the testbed.	Analytics payloads must be received and printed in the Aggregator's log.	1) PASS

4 Conclusions

In this deliverable, we have deposited our reports on system integration (Section 2); and testing/validation and verification activities (Section 3), as these were carried out in the context of Task 4.4 and T4.5. Partner activities elaborated in these reporting documents were carried out to ensure that the final integrated platform reported in D4.5 both implements, and meets the required functional specifications.

The integration report references the platform architectural elements identified last in D1.4 (not the network applications - these will be reported in D4.2 revision), and particularly contextualizes the integration work with respect to the APIs that have either been individually reported in a technical WP deliverable, or in D4.5 (which has been updated, and is re-submitted in parallel to this report). For each component, we delivered a short summary of its role within the 5G-EPICENTRE architecture, described which APIs from the other components it makes requests to (along with connection type, API protocol and type of data exchanged), and to what purpose. We further elaborated on the integration format, thereby establishing a roadmap of collaborative actions, that partners in the project carried out to ensure that the system functionality has been delivered.

Cohesively, the testing and validation report references the information provided in Section 2, and outlines all the tests that 5G-EPICENTRE partners have carried out, to ensure the integration was achieved, and that the system functions as intended (for the evaluation stage with both first- and third-party experimenters, in WP5). It elaborates on the test scenarios and flows carried out (providing the test tools used), the preconditions for testing and the expected behaviour of the system for each flow. If deviations were encountered, these have been duly reported, along with applied countermeasures.

The deliverable has thus reported on the final status of the system integration and testing activities throughout the course of the integration Task lifetime since D4.6 (T4.4, M24-M37), as well as in the case of the system testing and validation Task (T4.5). It follows up completion of the final version of the 5G-EPICENTRE experimentation facility (D4.5), and concludes the activities in WP4. The focus of the partners will now shift towards the experimentation (WP5) and outreach activities (WP6), which will be supported by minimal technical work whenever necessary (in the context of those WPs).

References

[1] IEEE Standard for Software and System Test Documentation. (2008). *IEEE Std 829-2008*, 1-150. https://doi.org/10.1109/IEEESTD.2008.4578383.

[2] Sayadi, B., Chang, C.-Y., Tranoris, C., Iordache, M., Katsaros, K., Vilalta, R., … & Makropoulos, George. (2022). *Network Applications: Opening up 5G and beyond networks*. Zenodo. https://doi.org/10.5281/zenodo.7123919.

[3] Fu, L., Salvendy, G., & Turley, L. (2002). Effectiveness of user testing and heuristic evaluation as a function of performance classification. *Behaviour & information technology*, *21*(2), 137-143.

[4] Maguire, M., & Isherwood, P. (2018, July). A comparison of user testing and heuristic evaluation methods for identifying website usability problems. In *International Conference of Design, User Experience, and Usability* (pp. 429-438). Springer, Cham.

[5] Nielsen, J. (1994). *Usability Engineering*. Morgan Kaufmann.

[6] Nielsen, J. (1994). Enhancing the Explanatory Power of Usability Heuristics. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems* (pp. 152-158). 10.1145/191666.191729.

[7] Ntoa, S., Margetis, G., Antona, M., & Stephanidis, C. (2021). User experience evaluation in intelligent environments: A comprehensive framework. *Technologies*, *9*(2), 41.

[8] Fernandez, A., Insfran, E., & Abrahão, S. (2011). Usability evaluation methods for the web: A systematic mapping study. *Information and software Technology*, *53*(8), 789-817.

[9] Vermeeren, A. P., Law, E. L. C., Roto, V., Obrist, M., Hoonhout, J., & Väänänen-Vainio-Mattila, K. (2010, October). User experience evaluation methods: current state and development needs. In *Proceedings of the 6th Nordic conference on human-computer interaction: Extending boundaries* (pp. 521-530)