

5G EVE

5G European Validation platform for Extensive trials

Deliverable D4.1 Experimentation Tools and VNF Repository

Project Details

Call	H2020-ICT-17-2018
Type of Action	RIA
Project start date	01/07/2018
Duration	36 months
GA No	815074

Deliverable Details

Deliverable WP:	WP4
Deliverable Task:	Tasks T4.1, T4.2, T4.3, T4.4
Deliverable Identifier:	5G_EVE_D4.1
Deliverable Title:	Experimentation tools and VNF repository
Editor(s):	Giada Landi (NXW), George Loukas (WINGS)
Author(s):	<p>Giada Landi, Juan Brenes, Giacomo Bernini, Pietro Giardina (NXW)</p> <p>Mauro Femminella, Paolo Giaccone, Matteo Pergolesi, Gianluca Reali (CNIT)</p> <p>Jaime Garcia-Reinoso, Luca Cominardi, Carlos Guimaraes, Pablo Serrano, Jonathan Almodovar, Cristina Quintana Jordan (UC3M)</p> <p>Ramón Pérez, Aitor Zabala (TELC)</p> <p>Vincenzo Suraci, Silvia Canale, Yuri Maria Chianese, Graziano Ciucciarelli, Gerardo Corsaletti, Oriano Biscuola (A2T)</p> <p>K. Trichias, P. Demestichas, K. Tsagkaris, G. Loukas, V. Stavroulaki, D. Meridou, O. Zekai, E. Kostara (WINGS)</p>
Reviewer(s):	M. Molla (ERI-ES), L. Roullet (NOK-FR)
Contractual Date of Delivery:	31/10/2019
Submission Date:	31/10/2019
Dissemination Level:	PU
Status:	Final
Version:	1.0
File Name:	5G EVE – D4.1 Experimentation tools and VNF repository

Disclaimer

The information and views set out in this deliverable are those of the author(s) and do not necessarily reflect the official opinion of the European Union. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

Deliverable History

Version	Date	Modification	Modified by
<i>V0.1</i>	<i>21/05/2019</i>	<i>First ToC</i>	<i>G. Landi</i>
<i>V0.2</i>	<i>30/05/2019</i>	<i>Update ToC</i>	<i>All authors</i>
<i>V0.3</i>	<i>04/07/2019</i>	<i>Integrated first round of contributions</i>	<i>G. Loukas, All authors</i>
<i>V0.4</i>	<i>22/07/2019</i>	<i>Integrated second round of contributions</i>	<i>G. Loukas, All authors</i>
<i>V0.5</i>	<i>05/09/2019</i>	<i>First integrated and complete version</i>	<i>G. Landi, All authors</i>
<i>V0.6</i>	<i>18/09/2019</i>	<i>First internal review and consistency check</i>	<i>G. Landi, All authors</i>
<i>V0.7</i>	<i>02/10/2019</i>	<i>Update of software information. Added introduction, executive summary, conclusion</i>	<i>G. Landi, All authors</i>
<i>V0.8</i>	<i>04/10/2019</i>	<i>Integration and review</i>	<i>G. Landi, All authors</i>
<i>V0.9</i>	<i>16/10/2019</i>	<i>Integrated internal review comments</i>	<i>G. Landi, M. Molla, L. Roullet</i>
<i>V1.0</i>	<i>28/10/2019</i>	<i>Final version addressing internal review comments</i>	<i>G. Landi, All authors</i>

Table of Content

LIST OF ACRONYMS AND ABBREVIATIONS	7
LIST OF FIGURES	8
LIST OF TABLES	10
EXECUTIVE SUMMARY	12
1 INTRODUCTION	13
2 EXPERIMENTS IN 5G EVE	15
2.1 DESIGN, REQUEST AND EXECUTION OF EXPERIMENTS IN 5G EVE	15
2.2 5G EVE PORTAL: ACCESS POINT FOR 5G EVE ACTORS	20
2.3 5G EVE EXPERIMENTATION TOOLS FOR EXPERIMENT DEFINITION AND MONITORING	22
2.3.1 <i>Browse and look-up tool</i>	22
2.3.2 <i>Intent-based interface tool</i>	23
2.3.3 <i>Experiment monitoring and data collection tools</i>	23
2.3.4 <i>Ticketing System</i>	23
2.4 5G EVE REPOSITORY: CATALOGUES FOR EXPERIMENT BLUEPRINT AND DESCRIPTORS	24
3 INTERACTIONS BETWEEN 5G EVE PORTAL ELEMENTS.....	27
3.1 HIGH-LEVEL INTERACTIONS BETWEEN 5G EVE USERS AND 5G EVE PLATFORM.....	27
3.1.1 <i>Experiment design and definition</i>	27
3.1.2 <i>Experiment preparation</i>	28
3.1.3 <i>Experiment execution</i>	30
3.2 INTERNAL WORKFLOWS OF 5G EVE PLATFORM FOR EXPERIMENTS MANAGEMENT	30
3.2.1 <i>Experiment design and definition phase</i>	31
3.2.2 <i>Experiment preparation phase</i>	38
3.2.3 <i>Experiment execution phase</i>	43
4 EXPERIMENTATION TOOLS, INCLUDING INTENT-BASED INTERFACES	54
4.1 EXPERIMENT MONITORING AND MAINTENANCE & RESULTS COLLECTION	54
4.1.1 <i>Purpose/Description of the tool</i>	54
4.1.2 <i>Tool Design / Architecture</i>	54
4.1.3 <i>Tool Implementation</i>	56
4.1.4 <i>Interaction with other tools/layers</i>	59
4.2 TICKETING SYSTEM.....	60
4.2.1 <i>Purpose/Description of the tool</i>	60
4.2.2 <i>Tool Design / Architecture</i>	60
4.2.3 <i>Tool Implementation</i>	61
4.2.4 <i>Interaction with other tools/layers</i>	61
4.3 BROWSE AND LOOK-UP.....	63
4.3.1 <i>Purpose/Description of the tool</i>	63
4.3.2 <i>Tool Design / Architecture</i>	64
4.3.3 <i>Tool Implementation</i>	64
4.3.4 <i>Interaction with other tools/layers</i>	66
4.4 INTENT BASED INTERFACE	67
4.4.1 <i>Purpose/Description of the tool</i>	67
4.4.2 <i>Tool Design / Architecture</i>	69
4.4.3 <i>Tool Implementation</i>	69
4.4.4 <i>Interaction with other tools/layers</i>	75
5 5G EVE CATALOGUE SOFTWARE PROTOTYPE.....	76
5.1 CATALOGUE COMPONENTS AND FUNCTIONALITIES	76
5.2 INFORMATION MODELS.....	77
5.2.1 <i>Vertical Service Blueprint</i>	78
5.2.2 <i>Context Blueprint (CB)</i>	79

5.2.3 <i>Experiment Blueprint (ExpB)</i>	80
5.2.4 <i>Metrics</i>	81
5.2.5 <i>KPI</i>	82
5.2.6 <i>Test Case Blueprint</i>	82
5.2.7 <i>Experiment Descriptor (ExpD)</i>	83
5.2.8 <i>VNFD, PNFD, and NSD</i>	83
5.2.9 <i>Blueprint Java library</i>	84
5.3 CATALOGUE INTERFACES	85
5.3.1 <i>5G EVE Portal Catalogue REST API</i>	85
5.3.2 <i>5G EVE I/W framework Catalogue REST API</i>	91
5.4 SOFTWARE DESIGN	92
5.4.1 <i>Installation guide</i>	96
5.4.2 <i>Licenses and dependencies</i>	97
6 CONCLUSIONS	99
ACKNOWLEDGMENT	100
ANNEX A – SERVICE BLUEPRINTS DESIGN AND IMPLEMENTATION MANUAL	101
ANNEX B - VERTICAL SERVICE BLUEPRINT YAML FILE EXAMPLE	105
ANNEX C – DATA COLLECTION AND EXPERIMENTATION MONITORING TOOLS SURVEY AND SELECTION	107
ANNEX D – VIDEO LOAD GENERATOR FOR VIDEO KPIS MONITORING	113
ANNEX E – SOFTWARE REPOSITORIES	117
REFERENCES	118

List of Acronyms and Abbreviations

<i>Acronym</i>	<i>Description</i>		
<i>AGV</i>	Automated Guided Vehicle	<i>MEC</i>	Multi-Access Edge Computing
<i>AI</i>	Artificial Intelligence	<i>ML</i>	Machine Learning
<i>API</i>	Application Programming Interface	<i>MSO</i>	Multi-Site Orchestrator
<i>CB</i>	Context Blueprint	<i>MTC</i>	Machine Type Communication
<i>CPU</i>	Core Processing Unit	<i>NBI</i>	North Bound Interface
<i>CRUD</i>	Create Read Update Delete	<i>NFV</i>	Network Function Virtualization
<i>CSS</i>	Cascading Style Sheets	<i>NFVO</i>	Network Function Virtualization Orchestrator
<i>DB</i>	Database	<i>NSD</i>	Network Service Descriptor
<i>DF</i>	Deployment Flavour	<i>PNF</i>	Physical Network Function
<i>E2E</i>	End to End	<i>RBAC</i>	Role Based Access Control
<i>eMBB</i>	Enhanced Mobile BroadBand	<i>REST</i>	REpresentational State Transfer
<i>EPC</i>	Evolved Packet Core	<i>SDN</i>	Software Defined Networking
<i>ETSI</i>	European Telecommunication Standard Institute	<i>SDO</i>	Standard Development Organization
<i>ExpB</i>	Experiment Blueprint	<i>SLA</i>	Service Level Agreement
<i>ExpD</i>	Experiment Descriptor	<i>SNMP</i>	Simple Network Management Protocol
<i>GPS</i>	Global Positioning System	<i>TCB</i>	Test Case Blueprint
<i>GUI</i>	Graphical User Interface	<i>URL</i>	Uniform Resource Locator
<i>HTML</i>	Hyper Text Markup Language	<i>URLLC</i>	Ultra-Reliable Low-Latency Communication
<i>HTTP</i>	Hyper Text Transfer Protocol	<i>VIM</i>	Virtual Infrastructure Manager
<i>IBN</i>	Intent Based Networking	<i>VM</i>	Virtual Machine
<i>IBNS</i>	Intent Based Networking System	<i>VNF</i>	Virtual Network Function
<i>ID</i>	Identifier	<i>VNFD</i>	Virtual Network Function Descriptor
<i>IDMS</i>	Intent Driven Management System	<i>VNFFG</i>	Virtual Network Function Forwarding Graph
<i>IL</i>	Instantiation Level	<i>VSF</i>	Vertical Service Blueprint
<i>I/W</i>	Interworking	<i>VSD</i>	Vertical Service Descriptor
<i>JPA</i>	Java Persistence API	<i>WP</i>	Work Package
<i>KPI</i>	Key Performance Indicator		
<i>MANO</i>	Management and Orchestration		

List of Figures

Figure 1: Relationship between Blueprints and Descriptors	17
Figure 2: Experiment phases, actors involved and main actions.....	20
Figure 3: High-level architecture of 5G EVE repository.....	25
Figure 4: High-level procedure for experiment design and definition phase	28
Figure 5: High-level procedure for experiment preparation phase.....	29
Figure 6: High-level procedure for an experiment execution.....	30
Figure 7: Onboarding of a new VNF Package provided by a VNF developer	32
Figure 8: Onboarding of a VNFD in a 5G EVE site	32
Figure 9: Onboarding of context blueprint.....	33
Figure 10: Onboarding of vertical service blueprint.....	34
Figure 11: Definition and onboarding of experiment blueprint (1).....	35
Figure 12: Definition and onboarding of experiment blueprint (2).....	36
Figure 13: Onboarding of 5G EVE blueprints across 5G EVE catalogues	38
Figure 14: ExpD onboarding (1)	39
Figure 15: ExpD onboarding (2)	40
Figure 16: Experiment scheduling.....	42
Figure 17: Preparation of experiment environment.....	43
Figure 18: Experiment instantiation	45
Figure 19: Experiment execution	47
Figure 20: Subscription to the topics used for experiment monitoring and performance analysis purposes. ...	50
Figure 21: Delivery and management of monitoring information during the experiment execution.	51
Figure 22: Withdrawal of the topics used for experiment monitoring and performance analysis purposes	53
Figure 23: Improved version of the Data Collection Manager architecture with the Experiment Monitoring and Maintenance & Results Collection toolchain.	55
Figure 24: The Elastic Stack toolchain with Beats and Kafka integration.	57
Figure 25: Function distribution for the Beats + Kafka + Elastic Stack architecture.	57
Figure 26: Urban mobility 5G data flows analysis use case, visualization tool graphs	58
Figure 27: Urban mobility 5G data flows analysis use case, geo points graphs.....	59
Figure 28: State of bugs in Bugzilla	61
Figure 29: Option to create a new ticket in Bugzilla.....	62
Figure 30: Graphical representation of an NSD	63
Figure 31: Graphical representation of a VNFD	63
Figure 32: 5G EVE Portal Catalogue – main page.....	65
Figure 33: 5G EVE Portal Catalogue: List of experiment blueprints.....	65
Figure 34: Detailed view of an experiment blueprint.....	66
Figure 35: Intention through Free Text format.....	70

Figure 36: Users are prompted to provide missing fields.....	71
Figure 37: Guided Selection services	71
Figure 38: Intention through Guided Selection	72
Figure 39: Information about the Blueprint created	73
Figure 40: Expected metric values of the experiment	74
Figure 41: Help Page	74
Figure 42: Graphical representation of a VSB example.....	78
Figure 43: Graphical representation of a CB for artificial background traffic.	80
Figure 44: Graphical representation of an ExpB example.....	81
Figure 45: High-level software design of 5G EVE Portal catalogue.....	93
Figure 46: 5G EVE Portal catalogue: software structure	94
Figure 47: High-level software design of 5G EVE I/W framework catalogue	95
Figure 48: 5G EVE I/W framework catalogue: software structure	96
Figure 49: Service sequence for the ASTI AGV use case.....	101
Figure 50: csQCA for data collection tools	111
Figure 51: csQCA for experimentation monitoring tools.....	112
Figure 52: Video KPIs.....	114
Figure 53: Monitoring system for video-related KPIs.....	114

List of Tables

Table 1: 5G-EVE actors	15
Table 2: 5G-EVE experiment phases	18
Table 3: Services provided by 5G EVE Portal to the corresponding actors.....	20
Table 4: Mapping between roles and CRUD actions on catalogues' information elements	25
Table 5: Tools analysed for the ticketing system	61
Table 6: Libraries used in browse and look up tool	64
Table 7: Browse and look up tool access to CRUD actions offered by 5G EVE catalogues	67
Table 8: Features of 5G EVE catalogues.....	76
Table 9: Vertical Service Blueprint fields with relevant description	79
Table 10: Context Blueprint fields with relevant description.....	80
Table 11: Experiment Blueprint fields with relevant description.....	81
Table 12: Metric fields with relevant description.....	82
Table 13: KPI fields with relevant description.....	82
Table 14: Test Case Blueprint fields with relevant description.....	83
Table 15: Experiment Descriptor fields with relevant description	83
Table 16: Mapping between NFV interfaces and architectures (NFV-IFA) and relevant solutions (NFV-SOL)	84
Table 17: 5G-EVE Portal Catalogue REST API.....	85
Table 18: REST API – GET /portal/catalogue/ctxblueprint<?[filter_parameters]>.....	86
Table 19: REST API – POST /portal/catalogue/ctxblueprint.....	86
Table 20: REST API – GET /portal/catalogue/ctxblueprint/<id>	87
Table 21: REST API – DELETE /portal/catalogue/ctxblueprint/<id>.....	87
Table 22: REST API – GET /portal/catalogue/vsblueprint<?[filter_parameters]>	87
Table 23: REST API – POST /portal/catalogue/vsblueprint	88
Table 24: REST API – GET /portal/catalogue/vsblueprint/<id>	88
Table 25: REST API – DELETE /portal/catalogue/vsblueprint/<id>	88
Table 26: REST API – GET /portal/catalogue/expblueprint<?[filter_parameters]>.....	89
Table 27: REST API – POST /portal/catalogue/expblueprint.....	89
Table 28: REST API – GET /portal/catalogue/expblueprint/<id>	89
Table 29: REST API – DELETE /portal/catalogue/expblueprint/<id>.....	90
Table 30: REST API – GET /portal/catalogue/expdescriptor<?[filter_parameters]>	90
Table 31: REST API – POST /portal/catalogue/expdescriptor	90
Table 32: REST API – GET /portal/catalogue/expdescriptor/<id>.....	90
Table 33: REST API – DELETE /portal/catalogue/expdescriptor/<id>	91
Table 34: I/W framework Catalogue REST APIs	91
Table 35: 5G EVE Portal and I/W framework Catalogue software dependencies.....	96

Table 36: 5G EVE Portal and I/W framework Catalogue library dependencies	97
Table 37: 5G EVE Portal and I/W framework Catalogue repositories.....	97
Table 38: Software licenses of 5G EVE catalogues	97
Table 39: Example of service blueprint parameters	101
Table 40: State of the art on data collection	107
Table 41: State of the art on experimentation monitoring tools	109
Table 42: evaluated features of data collection tools.....	110
Table 43: evaluated features of experimentation monitoring tools	111
Table 44: Software repositories for D4.1 portal components.....	117

Executive Summary

This deliverable provides a first release of the 5G EVE portal tools for the design, specification, management and monitoring of experiments that verticals can execute over the 5G EVE platform to validate the behaviour and performance of their services in realistic 5G environments. The deliverable consists of two major components: (i) this document, which describes the concepts around 5G EVE experiments, providing also details about some of the portal components in support of them, and (ii) a number of software tools that help the 5G EVE experimenters in the different phases of the experiment definition, execution, and validation.

The document introduces 5G EVE experiments, identifying their phases (from the initial design, up to the execution and validation steps) and roles of the different 5G EVE actors during a 5G EVE experiment typical lifecycle. Focus is on the different actions that can be performed through 5G EVE Portal, which provides a unified access to all 5G EVE services, across the entire multi-site platform. The expected interactions between 5G EVE actors and 5G EVE platform, through the Portal, are presented in detail for each experiment phase. Moreover, the deliverable reports also the design of the internal workflows regulating the communications among the functional components of the portal and the interactions with the other entities of the 5G EVE platform.

The deliverable provides also the description and the documentation for the experimentation tools in the scope of the Task T4.1 and Task T4.4 activities. In particular, this deliverable covers the following components: (i) the tools for experiment monitoring and results collection and storage; (ii) browse and look-up tools and (iii) the intent-based interface in support of experiment definition; (iv) the trouble ticketing tool to help experimenters and facilitate their interaction with the dedicated 5G EVE support teams; and (v) the VNF catalogue storing 5G EVE offerings, in terms of blueprints and descriptors of services and experiments. Some of these components are based on widely used open source software, which have been customized and configured to meet the specific requirements of the 5G EVE platform. Others have been developed in the context of the project reusing and extending, where possible, software prototypes resulting from previous phase 1 and phase 2 projects.

1 Introduction

The 5G EVE Portal provides a web-based, unified access point for verticals and experimenters to all the functionalities made available by 5G EVE platform. Adopting an *Experiment-as-a-Service* model, the 5G EVE platform enables the validation of vertical services in different 5G scenarios through the execution of experiments in realistic and customizable environments, built dynamically to represent a variety of operational conditions. The Portal includes several tools that facilitate the design, definition, execution, monitoring and evaluation of experiments, providing also a web interface that guides through the different steps of the experiment lifecycle.

5G EVE experiments can be designed in a flexible and powerful manner, which allows to easily customize the target infrastructure deployment of the service, the characteristics of the execution environments, the metrics and Key Performance Indicators (KPIs) to be evaluated, as well as the detailed steps to be executed on the service components. Accessing the 5G EVE Portal, the verticals can define their experiments, schedule and launch their execution, follow their progress and visualize their results and KPIs. Experiment definition and modelling is facilitated through the portal tools, using intent-based interfaces or pre-defined experiment templates, called blueprints, which can be easily browsed, selected and customized directly from the web portal.

This deliverable describes the concepts around 5G EVE experiments, identifying actions assigned to the different actors (verticals, experiment developers, experimenters, etc.) and the different phases that regulate an experiment lifecycle. Starting from these high-level concepts, the document presents the roles and the functionalities of the different portal components during the design, preparation and execution of the 5G EVE experiments. The deliverable is specifically focused on a subset of the portal components, developed in the context of Task T4.1 and Task T4.4. It covers the monitoring tools, the browse and look-up tool, the intent-based interface, the trouble ticketing tool and the portal catalogue (or VNF repository). For each of these components, the deliverable provides an initial software release and the associated documentation. In the context of Task T4.2, all these modules, delivered so far as stand-alone entities, will be integrated within the overall 5G EVE Portal and, where needed, they will be further refined and customized to meet the requirements of the 5G EVE system.

The deliverable D4.1 thus consists of two main parts:

- The first release of the software prototypes of the experimentation tools and the portal catalogue, ready to be integrated in the 5G EVE Portal; the software is available in open repositories, as reported in Annex E.
- This document, which provides an overview of 5G EVE experiments and 5G EVE Portal functionalities, the specification of its internal components and the associated interactions and, finally, the documentation of the software prototypes.

The document is structured as follows. Section 2 describes the concept of a 5G EVE experiment, explaining its phases, the actions that can be performed by the different 5G EVE actors during its design, preparation and execution, as well as the mechanisms and services provided by 5G EVE Portal to support all the steps of the experiment lifecycle. The same section also analyses the role of the Portal catalogue and experimentation tools in providing and implementing the 5G EVE Portal functionalities.

Section 3 is focused on the interactions and workflows regulating the processes of 5G EVE platform, with special attention to 5G EVE Portal entities. The section initially defines the procedures that different categories of 5G EVE users need to follow during the entire experiment lifecycle, presenting their high-level interactions with the 5G EVE Portal components exposing external interfaces. Starting from this analysis of the “external” behaviour of the 5G EVE Portal, the section proceeds with the specification of the “internal” workflows of the system, presenting the message sequence diagrams that model the interaction between the components of the portal and with the other entities of the 5G EVE platform.

Section 4 and 5 describe the experimentation tools and the portal catalogue, respectively. For each component, the document reports the functional architecture and describes its software implementation, providing a link to the software repositories where the prototypes can be downloaded. A special focus is also on the interactions with the other components of the portal, providing an initial guideline for the future integration activities to be performed in Task T4.2. Section 5 provides also the specification of the information models adopted to represent

the blueprints and the descriptors for the definition and customization of the experiments and their sub-components. Examples are provided in the annexes, to help external experiment developers in the design of their own experiments.

2 Experiments in 5G EVE

In the 5G EVE framework, vertical industries (defined as “verticals” from now on) can test and evaluate the performance of their vertical services in a realistic 5G environment that replicates the characteristics of an operational 5G infrastructure. This approach allows the verticals to verify the compliance of the relevant network-related Key Performance Indicators (KPIs) with the expected service performance, before the roll-out of the services in the real operational networks and their delivery to the final customers. Moreover, the flexibility of the 5G EVE framework allows the verticals to validate the behaviour of their services in different conditions (e.g. changing the service request load, the traffic load, the network conditions in terms of delay and jitter, etc.). This is fundamental to support the tuning of the service in the pre-roll-out phase to guarantee its performance in different kinds of operational environments and conditions, as well as the definition of suitable SLAs and convenient service offers to be proposed to the customers.

In 5G EVE, the deployment, configuration, execution and evaluation of vertical services in the target site facilities are performed through a set of “experiments”. Therefore, an experiment represents the execution and the validation of the applications associated to a vertical service, running in one or more 5G EVE site facilities in a dedicated virtual environment, which is dynamically instantiated and configured to reflect the characteristics and the conditions of the target operative settings. In the following subsections we describe how the 5G EVE platform can be used to define and request experiments, to configure them and launch their execution, as well as to monitor them and collect their results and relevant KPIs.

2.1 Design, request and execution of experiments in 5G EVE

5G EVE portal is the architectural entity that provides the single point of access to the 5G EVE system and services for the verticals, wrapping all the backend functionalities and tools composing the 5G EVE platform. Through the 5G EVE portal, verticals will be able to define their experiments, launch their execution, verify progress and finally retrieve validation results. Since verticals have usually a service- and application-oriented expertise, without any specific know-how on networking issues and physical/virtual infrastructure configuration, 5G EVE portal provides mechanisms to facilitate declaration of experiments, e.g. through the selection of experiment templates from a set of pre-defined blueprints or intent-based service modelling.

The portal backend will then take care of translating the service-oriented requirements from the verticals into formal definition of vertical service and experiment settings (defined as “context”) that will run in 5G EVE facilities to execute the experiment. Definition of such elements involves specification of network service descriptors, Virtual Network Function (VNF) packages for the single components of vertical services or experiments’ context, and descriptors of physical functions available in each of the 5G EVE facilities. These entities must be modelled and specified by different actors, each of them with a specific kind of expertise and a specific role in the whole value chain. For example, the definition and configuration of the physical equipment installed in a given 5G EVE facility is typically a responsibility of the site manager. Similarly, verticals can define high-level specification of their own services. However, encoding of experiment requirements in formal blueprints requires further knowledge (e.g. about the 5G EVE infrastructure) and expertise (e.g. about the modelling of NFV network services) that can be provided by a different actor.

In 5G EVE we have identified several different actors, represented in Table 1, each of them characterized by a specific role in the planning, preparation, and execution of the experiments. The role of the actors has a direct impact on his/her access rights to the 5G EVE portal services, defining a subset of actions that can be performed by the given actor.

Table 1: 5G-EVE actors

Actor	Description	Target actions
Anonymous user	User not authenticated to the Portal	No actions are allowed
VNF provider	Actor who provides the VNF packages for the vertical applications.	<ul style="list-style-type: none"> Uploading of VNF packages in 5G EVE portal.

		<ul style="list-style-type: none"> • Issuing of requests (via 5G EVE portal) for VNF packages on-boarding to specific sites.
Vertical	Actor with the knowledge of the service to be tested, including SLAs and service components	<ul style="list-style-type: none"> • Support in the definition of the vertical service blueprints and related experiments.
Experiment developer	<p>Actor responsible for specifying the blueprints associated to an experiment, as well as the associated NFV network services descriptors. This user has the knowledge about the 5G EVE infrastructure and expertise about NFV network service modelling. Moreover, it interacts with the vertical to receive information about the details of the target service.</p>	<ul style="list-style-type: none"> • Modelling and definition of blueprints for vertical services, experiment contexts and experiments. • Modelling and definition of NFV Network Service Descriptors (NSD) associated to the experiment blueprints, as well as related translation rules. • Definition and development of mechanisms for collecting application-level metrics from the vertical service. • On-boarding of blueprints via 5G EVE Portal. • On-boarding of NFV Network Service Descriptors via 5G EVE Portal.
Experimenter	Actor responsible for the request of an experiment and the assessment of its results. He/she defines characteristics of an experiment starting from its blueprint, requests the deployment of related virtual environment and experiment execution and analyzes results and KPIs.	<ul style="list-style-type: none"> • Selection of blueprints for the target vertical service and experiment, via 5G EVE Portal. • Configuration of the experiment, according to the blueprint parameters, via 5G EVE Portal. • Issuing of requests, via 5G EVE Portal, for scheduling the execution of the experiment in the 5G EVE facilities. • Monitoring of the experiment execution, via 5G EVE Portal. • Visualization of experiment measurements, KPIs and results, via 5G EVE Portal.
Site manager	Actor responsible for the management of the infrastructure and the orchestration systems in a particular site.	<ul style="list-style-type: none"> • Pre-provisioning and configuration of the physical infrastructure in the managed site. • Validation and on-boarding of VNF packages. • Preparation of resources needed to deploy an experiment.
System administrator	Actor who has access to all tools provided by the 5G-EVE project.	<p>All actions, including management functions (e.g. configuration of users and permissions, visualization of system logs, etc.).</p> <p>Note: the system administrator may have a limited access to the management system of the single site facilities.</p>

In the lifecycle of an experiment, we can distinguish four major phases: (a) experiment design and definition, (b) experiment preparation, (c) experiment execution and (d) experiment results analysis.

The **experiment design and definition phase** is the initial phase where the experiments for a given vertical service are planned and formalized, with the collaboration and off-line interaction of verticals, VNF providers and experiment developers. This phase has the objective of identifying the main components of the service, their interactions, the target execution environment(s) and condition(s), the elements to be monitored and the relevant KPIs, as well as the detailed steps to run the experiment. As result of this phase, the blueprints of vertical services (**Vertical Service Blueprint – VSB**), contexts (**Context Blueprint – CB**), test cases (**Test Case Blueprint – TCB**) and experiments (**Experiment Blueprint – ExpB**) are produced and on-boarded in the system.

The blueprints are a sort of “templates” that define the high-level features of services, execution environments and experiments, respectively. Such blueprints include also several variable parameters that can be specified by the experimenters during the *experiment preparation* phase for tuning the specific experiment instance. An experiment blueprint filled with concrete values assigned to its variable parameters is called **Experiment Descriptor (ExpD)** and it corresponds to a deployment model for the instantiation of an experiment, summarizing the characteristics of the target virtual environment where the experiment needs to run. Figure 1 shows the main relationship between the aforementioned Blueprints and the Experiment Descriptor, highlighting at the same time the phases where each of them are created and used.

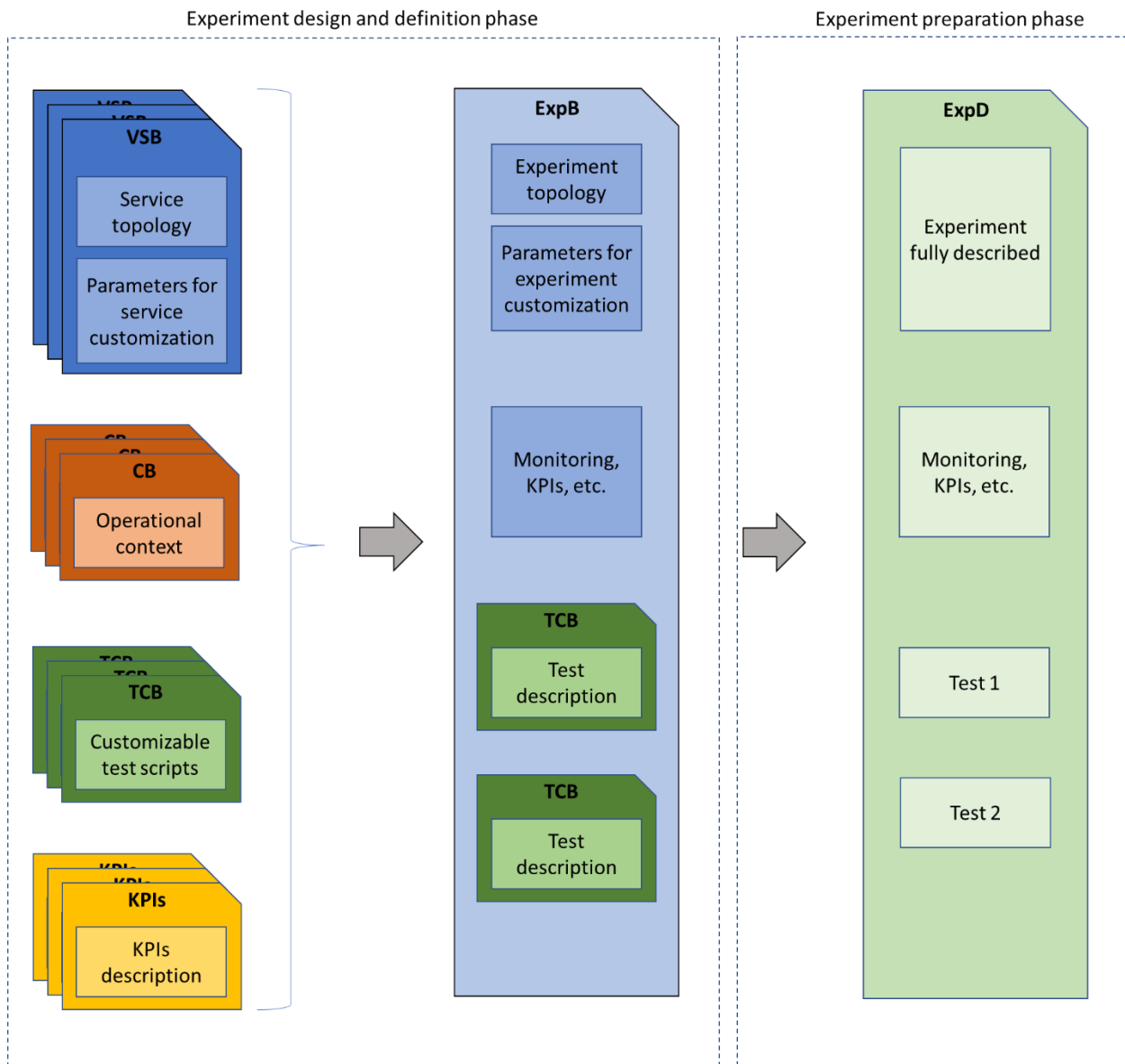


Figure 1: Relationship between Blueprints and Descriptors

Together with the blueprints, the experiment design and definition phase also produces a set of VNF Packages for the virtual applications of the vertical service and one or more NFV Network Service Descriptors (NSD) that describe potential deployments of the experiments in target 5G EVE facilities. VNF Packages are on-boarded on target 5G EVE sites, subject to a preliminary validation of the packages by the site owners. NSDs are also on-boarded in the system, in particular on the catalogue, together with “translation rules” that describe how to automatically map Experiment Descriptors into an equivalent NFV Network Service instance.

The following **experiment preparation phase** involves mostly experimenters and site managers and has the objective of preparing an environment in the 5G EVE facility suitable for the execution of a given experiment. At the beginning, the experimenter uses 5G EVE portal to select and configure target ExpB, specifying all needed parameters to generate a valid ExpD. The ExpD is then on-boarded in the portal and used as baseline for the request of the experiment. Since in most of cases the execution of an experiment requires manual setting and configuration of the physical infrastructure at 5G EVE sites, as well as the reservation of dedicated resources, the experiment execution needs to be scheduled in a specific time period that is agreed between experimenter and involved site manager(s). Once the experiment is scheduled, it is a task of the site manager(s) to guarantee that the target physical infrastructure(s) will be able to host the experiment as planned. This typically involves some off-line actions where the site manager configures the settings of the site facility.

The final phase is related to the **experiment execution**. This phase is managed through 5G EVE portal in an automated manner and involves mostly the experimenters. In case of failures or misconfigurations at the physical infrastructure, it may require manual intervention of the site manager; also in case of failures or errors in deployed VNFs, it would require the manual intervention of an expert. The initial step of the experiment execution is the instantiation and configuration of the virtual environment where the experiment will run. This step is triggered through 5G EVE portal and involves the active role of the Interworking (I/W) Framework, which will be responsible for instantiation and configuration of the NFV Network Service instances corresponding to the target ExpD. When virtual environment is up and running, applications and the scripts defined in the experiment specification will be executed according to the test plan. During the experiment execution, the monitoring metrics will be collected through dedicated tools of the system and, optionally, visualized through the portal allowing the experimenter to follow the progress of the experiment. The results of the experiment will be available on the portal at the end of the experiment.

After experiment execution, diagnostics tools under development in WP5 will help experimenters in the evaluation of the service during the **experiment results analysis** phase. However, this additional fourth phase will be omitted in the rest of the document, since not directly related to the tools developed in WP4.

Table 2 presents the three experiment phases defined in the 5G EVE project, detailing the main actors involved in each phase and the major actions as well. Figure 2 shows a similar information but using a schematic view, where the blue tones of the arrows represent the three experiment phases.

Table 2: 5G-EVE experiment phases

Experiment phase	Major actions	Involved actors
Experiment design and definition	<ul style="list-style-type: none"> • Design of experiment components, target execution environments and detailed test steps. • Definition of VNF Packages and NSDs. • Validation and on-boarding of VNF Packages in 5G EVE sites. • Definition of mechanisms for collecting application-level metrics from the vertical service. • Definition of VSBs, CBs and ExpBs. 	Verticals VNF providers Experiment developers Site managers

	<ul style="list-style-type: none"> On-boarding of VSBs, CBs, ExpBs, NSDs (and associated translation rules) in 5G EVE platform. 	
Experiment preparation	<ul style="list-style-type: none"> Definition and on-boarding of ExpD. Scheduling of experiment execution. Preparation and configuration of resources in the target 5G EVE site. 	Experimenters Site managers
Experiment execution	<ul style="list-style-type: none"> Instantiation and configuration of the virtual environment to run the experiment. Execution of scripts and applications to run the experiment. Collection and elaboration of monitoring data. 	Experimenters
Experiment results analysis	<ul style="list-style-type: none"> Analysis of experiment results through diagnostic tools (out of WP4 scope) 	Experimenters

As explained before, the experiment design and definition phase starts with the verticals providing requirements to both VNF providers and Experiment developers (step (1) in Figure 2). On the one hand, VNF provider(s) creates VNF(s) that has to be uploaded using 5G EVE Portal, requesting on-boarding to the proper site managers by using ticketing system, or simply ticketing system from now on (not shown in Figure 2). On the other hand, Experiment developers create all Blueprints and NSDs offline to finally onboard them using the 5G EVE Portal. These two last steps are shown in Figure 2 as step (2), which finalizes the first experiment phase. An Experimenter then can select a previously-defined Blueprint, shown as step (3) in Figure 2. After selecting the proper ExpB, an Experimenter must complete the required fields presented by 5G EVE Portal, shown in step (4) in Figure 2. It is important to highlight that these two steps (3 and 4) can be done in one step if the Experimenter uses intent-based tool. Previous actions trigger the Experiment Lifecycle Manager (a Portal component further described in section 3), which will generate proper tickets to all Site Managers that will be involved in the experiment (step (5) in Figure 2). Site Managers must check the Experiment Descriptors to analyse all resources requested in their particular sites. If the requested resources can be satisfied, Site Manager setup everything in their sites, modifying the state of the experiment at the end of this step (6), which concludes the experiment preparation phase. In the experiment execution phase, experimenters can start the execution of the experiment (step (7) in Figure 2). While the experiment is running, or when it is concluded, experiments can monitor the generated metrics (step (8) in Figure 2). Although it is not shown in Figure 2, the 5G EVE Portal will include other experimentation tools, which will be designed and implemented in WP5, that will be used to show and validate the KPIs defined and requested by the Experiment developers during the Experiment design and definition phase.

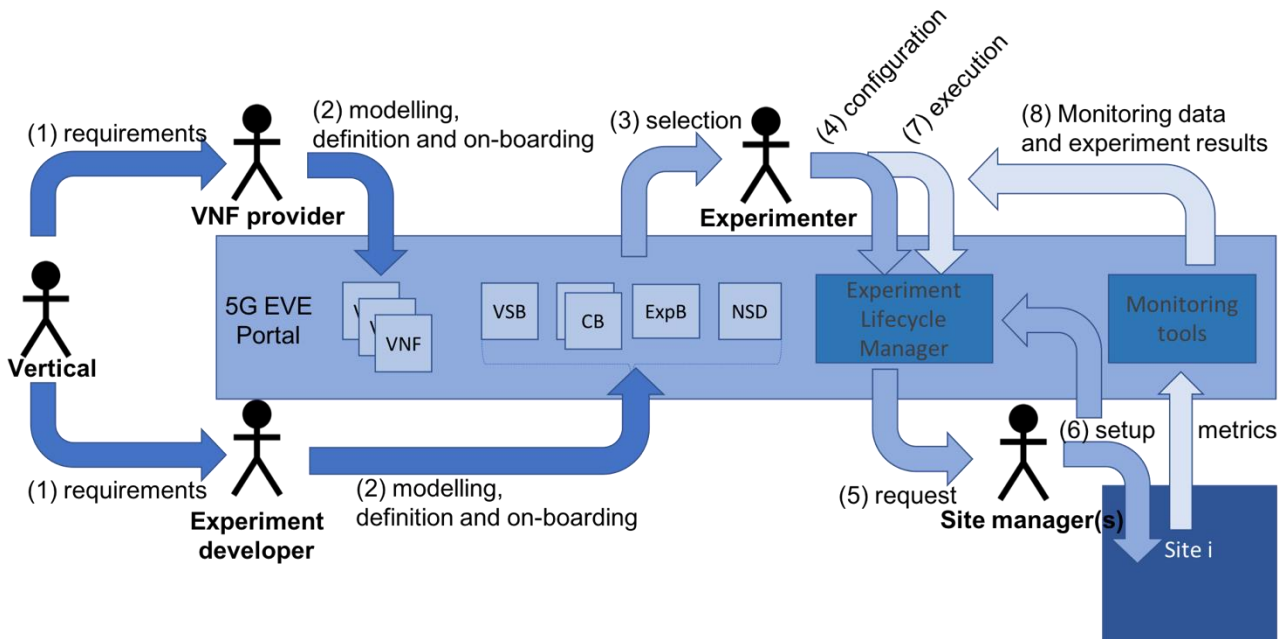


Figure 2: Experiment phases, actors involved and main actions

The lifecycle of an experiment is modelled through different states, as follows:

- scheduling: the experimenter has provided all requested information and the experiment must be approved (and configured) by all site managers involved.
- accepted: all site managers have analysed the resources to execute the parameters and everything is OK at their sites.
- ready: all resources to run the experiment are in place.
- instantiating: after the experiment is in the ready state, and the time is within the proposed period, it can be instantiated either automatically or manually by the experimenter, depending on the configuration of the experiment
- configuring: all components of the experiment have been instantiated but some of them require extra configuration.
- running: tests are running.
- terminating: all tests have finished, and the experiment is terminating.
- terminated: the experiment has finished successfully.
- failed: there was a problem during the execution of the experiment. An error code is provided.

2.2 5G EVE portal: access point for 5G EVE actors

After carefully analysing involved actors’ requirements presented before, and steps to define, run and analyse an experiment, services that have to be offered by the 5G EVE Portal are identified and shown in Table 3. Some of these services will be provided by internal tools or the Portal Catalogue, while other services will be provided by other components that will be provided by the Portal itself. The first column of Table 3 identifies how the corresponding service will be provided, which could be an experimentation tool, a VNF repository or the Portal. This deliverable presents the purpose, design and implementation of the first two layers, while deliverable D4.2 will include information about the Portal itself.

Table 3: Services provided by 5G EVE Portal to the corresponding actors

Layer	Actor	Service
Portal	Anonymous User	Signup, login to Portal

Tool / Portal Catalogue	Experiment/ Developer	Browse and look up tool to view the components this actor has permission to inspect.
Portal	Experiment/ Developer	Upload VSB, ExpB and Test Case Blueprint with the corresponding permissions/visibility. These actions will finish with an OK or Error with a given code.
Portal	Experimenter	List available VSB based on the provided query
Tool	Experimenter	List required/missing parameters for the VSB provided
Portal	Experimenter	Post parameters to create a service for the selected VSB and returns an experiment identifier (ExpID), which is in an incomplete state.
Portal	Experimenter	List available ExpB based on the provided (<i>incomplete</i>) experiment identifier (ExpID).
Portal	Experimenter	Get required parameters of the provided ExpB identifier.
Portal	Experimenter	With the ExpB selected, experimenters have to add all necessary tests. If successful, the provided ExpID state changes to <i>completed</i> .
Portal	Experimenter	With the ExpID (must be in <i>completed</i> state), this actor creates an Experiment Deployment, which includes the sites to deploy each component and a tentative date and time.
Portal	Experimenter/ Site Manager	These actors have access to the status of the experiment deployment object: pending, accepted, rejected, pre-provisioned, ready, running, finished, failed or deleted.
Portal	Experimenter	Experimenters could execute and accept experiments.
Portal	Experimenter	Experimenters can visualize detailed information about the result of the instantiation process, like the management IP addresses of (some) of the components involved in the experiment.
Tool	Experimenter	Experimenters could monitor metrics while experiments are running.
Tool	Experimenter	Experimenters can visualize the results of their executed (finished) experiments.
Portal	Experimenter	Experimenters could delete pending, accepted, finished or failed experiments.
Tool	Experimenter	Add/view/comment tickets
Portal	VNF provider	Users who can upload and request the onboarding of VNFD to a list of trial sites. Results could be OK or Error with code. This step may generate tickets to the site managers.
Tool	Site manager	Add/view/modify tickets
Portal	Site manager	Modify the status of an experiment deployment object for a given site.
Portal /Tool/ Portal Catalogue	System administrator	Administrators can execute all services available in the system

2.3 5G EVE experimentation tools for experiment definition and monitoring

To define an experiment, different actions need to take place by different actors, as summarized in Table 2. First, Experiment Developers need to upload the necessary VSBs, Context Blueprints (CBs), Test Case Blueprints (TCBs), and Experiment Blueprint (ExpB) with the corresponding permissions/visibility, in order to make available all the necessary “tools” needed to create an experiment. This task is performed in parallel while the vertical provides experiment developer with all the details of the service that he wishes to deploy. When all aforementioned items are uploaded, experimenters can select desired VSB from a list. In the VSB there are missing parameters, that need to be filled, to create a service descriptor for the selected VSB. Then, the Experimenter can select an ExpB from the ones available for target VSB. When the Experimenter provides all the required parameters for the given ExpB, the result is an Experiment Descriptor (ExpD) that fully defines all the necessary tests. With a completed ExpD, the experimenter can request the scheduling of the experiment deployment and execution, defining the sites to deploy each component and proposing a tentative date and time.

After an experiment is deployed, following the steps described in previous sections, experimenters have the ability to check the status of the experiment deployment (scheduling, accepted, ready, instantiating, configuring, running, terminating, failed). They can also monitor metrics and KPIs, while the experiment is running, or even visualize the results of the single test cases executed for their experiments. Assistance from 5G EVE support team can be requested in case of problems during any of the different phases.

To facilitate all these actions related to the experiment definition and monitoring, 5G EVE platform, and its Portal in particular, provides a number of *Experimentation Tools* that assist the experimenters and, more in general, all the 5G EVE actors in their tasks. This deliverable provides the first release of a subset of these tools, in particular: (i) the **browse and look-up tool** and (ii) the **intent-based tool**, both of them in support of experiment definition; (iii) the **experiment monitoring tools**, to enable the collection and storage of metrics during the experiment execution; (iv) the **ticketing system**¹ to assist experiment designers and experimenters providing an efficient communication channel towards the 5G EVE support teams; and (v) the **5G EVE catalogue** storing blueprints and descriptors to simplify the definition of the experiments. The first four tools are briefly introduced in the following subsections highlighting their role in the experiment lifecycle, while their implementation is described in section 4. The 5G EVE catalogue is introduced in section 2.4 and its implementation is reported in section 5.

2.3.1 Browse and look-up tool

The browse and look-up tools provide a web-based graphical interface, to be accessed through the 5G EVE Portal, that allows experiment developers and experimenters to easily define and customize their own experiments and effectively reuse experiment “templates” and “components” already available in the 5G EVE catalogue. This tool facilitates navigation of 5G EVE library and selection of blueprints more relevant for experiments specification. For example, during the definition of a new experiment blueprint, the experiment developer can easily visualize the context blueprints available for the target site and select the ones most suitable according to the desired operational conditions. Similarly, experimenters can browse the list of available experiment blueprints and visualize their characteristics and components.

The browse and look-up tool can be also used to visualize the NFV descriptors (e.g. Network Service, VNF and PNF Descriptors) available in different 5G EVE sites, with graphical representations of their elements and how they are interconnected.

¹ The 5G EVE Ticketing System was defined as “Trouble-Ticketing” system in the 5G EVE Description of Action. In this deliverable, we use the more general term “Ticketing” system, since the tool is used not only to signal issues and bugs, but also to enable a smooth interaction between experimenters (or any kind of consumer of 5G EVE services) and the 5G EVE support team during the normal lifecycle of experiments.

2.3.2 Intent-based interface tool

Intent-based tools provided in 5G EVE Portal provides an alternative mechanism to define and customize experiments, adopting an approach based on the natural language. This tool supports two different mechanisms that helps the experimenters to easily express their intentions. The former one is based on free text, where experimenters can simply write a sentence in plain English language to describe the characteristics of the desired experiment. Such description is then automatically translated by the intent-based tool into a blueprint or descriptor schema. The latter mechanism is based on guided selection and filling of the fields that defines an experiment, through a set of web pages that guides the experimenter during the entire procedure.

As for the browse and look-up tool, the intent-based tool is designed to simplify the process of experiment definition and customization, allowing an easy access to the experimentation services provided by the 5G EVE platform.

2.3.3 Experiment monitoring and data collection tools

The experiment monitoring tools offered by the 5G EVE Portal provide mechanisms to efficiently collect and store different kinds of metrics and KPIs during the execution of the experiment. Such metrics can be related to network infrastructure or to applications, so that experimenters can evaluate performance of their services at different levels. The monitoring tools include also a dashboard for visualization of the monitoring data, with the possibility to apply filters or build graphs.

The monitoring tools consist of three major entities: “*Data Collection, Aggregation and Pre-processing*”, “*Data Indexing and Storage*” and “*Data Visualization*” components placed in the Portal. These entities are grouped in one single experimentation tool called “*Experiment Monitoring and Maintenance & Results Collection*”, which is based on the ELK² stack, as further specified in section 4.1. The main responsibilities to be fulfilled by this tool are: (i) collect data from different experiments, logging it in a homogeneous way and doing a preliminary data manipulation; (ii) save the pre-processed data and perform different filtering operations in order to be easily analysed afterwards; and (iii) display the selected data for validation by either a human operator or an interoperating information system.

Moreover, this tool interacts with other components that participates in the experiment monitoring and performance analysis, as depicted in section 3.2.3.3, which are the following:

- The “*Data Collection Manager*” component, placed in the I/W framework, which is in charge of managing and coordinating the data sources so that they can provide their data to experiment monitoring and data collection tools. The data received by the Data Connection Manager is provided by “*Data shippers*” deployed in the VNFs/PNFs that belongs to a given experiment: a lightweight framework that eases the process to ship data to a higher layer data collector.
- The “*Results Analysis and Validation*” component, located in the Portal, which calculates the KPIs and results to be displayed through the “*Experiment Monitoring and Maintenance & Results Collection*” tool based on the metrics received from the Data Collection Manager. This component is under the scope of WP5.

2.3.4 Ticketing System

In 5G EVE, a Ticketing System is provided to simplify interaction between different 5G EVE actors and 5G EVE support teams during experiments entire lifecycle, from design up to execution, e.g. to request scheduling, or request support during different phases. All 5G EVE actors can create, view or comment additional tickets whenever required, with policies regulating level of visibility to different tickets based on actor’s profile.

5G EVE Portal will include a ticketing system that will be used in two possible ways:

² <https://www.elastic.co/what-is/elk-stack>

- Automatically by Portal elements. Some elements will use the interface provided by the ticketing system to automatically generate tickets to certain products. For example, it is foreseen that when an experimenter schedules an experiment, this action will create a ticket including the information provided by the experimenter, which will be submitted to the proper site managers.
- Manually by some actors. Actors like experimenters may issue tickets to site managers or system administrators, where they can inform about different issues, like bugs, problems with the Portal, etc.

After analysing different alternatives, 5G EVE project has selected Bugzilla³ as the main tool to implement the ticketing system. These are the reasons for this selection:

- It is an open-source project, covered by the Mozilla Public License⁴.
- It is a mature software, with more than 20 years of experience.
- Installable on many platforms, like in Linux, Windows and Mac.
- Advanced search capabilities, bug list in multiple formats, time tracking, automatic duplicate bug detection, private attachments and comments, etc.
- REST API available.

One of the most important features of Bugzilla for the 5G EVE project is the availability of the REST API. This will be used by other components to create, modify, delete, etc. tickets, but this will be used by the Portal to offer an integrated view to manually perform the same operations by a human user.

2.4 5G EVE repository: catalogues for experiment blueprint and descriptors

5G EVE repository⁵ includes two major catalogues, as shown in Figure 3: portal catalogue and I/W framework catalogue. I/W framework catalogue, developed in the context of WP3⁶, is placed within 5G EVE I/W Framework and manages VNF descriptors (VNFD), PNFs (PNFD) and NFV Network Services (NSD) and PNF instances available in the overall 5G EVE multi-site facility. The portal catalogue, developed in WP4, on the other hand, provides a “global” catalogue service for all the different kinds of descriptors and blueprints handled in 5G EVE, independently on the specific architecture layer where they are managed. It wraps I/W framework catalogue to provide a (regulated) access to the NFV descriptors in the different sites and manages directly repositories located at portal backend for vertical service, context and experiment blueprints and descriptors.

Thus, the two catalogues manage two different kinds of information models, as follows:

- Portal catalogue:
 - Vertical Service Blueprints (VSB), Context Blueprints (CB) and Test Case Blueprints (TCB)
 - Experiment Blueprints (ExpB) and Experiment Descriptors (ExpD)
- I/W framework catalogue:
 - VNF Descriptors (VNFD)
 - PNF Descriptors (PNFD)
 - NFV Network Service Descriptors (NSD)
 - PNF Instances

³ <https://www.bugzilla.org/>

⁴ <https://www.mozilla.org/en-US/MPL/>

⁵ The 5G EVE repository (or “catalogue”, as often referred to in the rest of the deliverable) was defined as “VNF repository” in the 5G EVE Description of Action. In this deliverable we omit the “VNF” word, since the 5G EVE catalogues store not only VNF descriptors, but also a much wider set of entities, like blueprints and descriptors for experiments, vertical services, test cases etc.

⁶ The I/W framework catalogue is developed in the context of WP3 but documented in this deliverable for completeness.

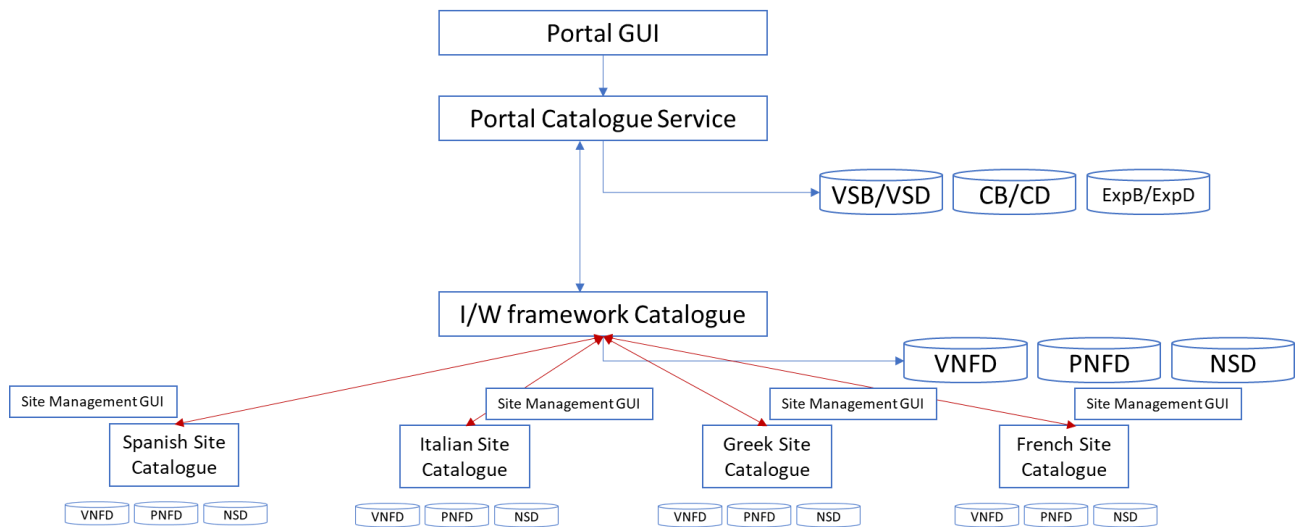


Figure 3: High-level architecture of 5G EVE repository

It should be noted that a 5G EVE user always accesses catalogues through 5G EVE Portal and, thus, the access to VNFs, PNFs, NSDs, and PNF Instances is “mediated” through portal catalogue. This approach allows to manage the coordination between on-boarding and retrieval of NFV descriptors and the overall workflows related to the management of vertical service and experiment blueprints. This coordination is handled in the internal logic of the portal catalogue. On the other hand, I/W framework catalogue is responsible for maintaining synchronization between centralized view of the descriptors available in the multi-site I/W framework and local repositories in each 5G EVE site. Moreover, I/W framework catalogue is also in charge of translating between information models adopted at the northbound interface (NBI) of I/W framework and site-specific models and languages adopted by NFV MANO stacks deployed in each facility (and vice-versa), such as ONAP, OSM, vendor specific NFV platforms. In particular, NBI of the I/W framework catalogue (initially introduced in D3.2 [6]) is based on the ETSI NFV SOL 005 [1] specification, with NSD and VNF modelled according to ETSI NFV SOL 001 [2] (TOSCA models) or SOL 006 [3] (YANG models).

Both catalogues offer a REST API on their north-bound interface to enable programmable access to repositories. However, through the browse and look-up tool (see also Section 4.3), users can also access catalogue services using a web-based graphical interface.

An important feature of 5G EVE catalogues is the management of the authorizations according to the role of the users, following the principles of the Role Based Access Control (RBAC). In 5G EVE, the kind of actions that can be performed on the entities available in the catalogues, modelled as Create, Read, Update, Delete (CRUD) actions, depends on the specific role of the user. For example, the experimenter role can only read experiment blueprints, but cannot create, update or delete them. On the other hand, he/she has full control on its own experiment descriptors. The experiment developer role has full access, in read and write mode, to experiment blueprints, but does not have any access to experiment descriptors. Associating users to multiple roles, it is possible to achieve a deep granularity in the configuration of the permissions associated to the different entities managed in the 5G EVE catalogues.

Mapping between user roles’ and CRUD actions (classified as read/write) permitted on each of the information elements managed by the catalogue are represented in Table 4. The system administrator is omitted, since he/she implicitly has full access in read/write mode to all the repositories. The detailed workflows related to on-boarding and queries of the different information elements during the lifecycle of an experiment are defined in section 3.2.

Table 4: Mapping between roles and CRUD actions on catalogues’ information elements

	VNF Provider	Vertical	Experiment developer	Experimenter	Site Manager
VSB	--	Read/Write	Read/Write	Read	--
CB	--	--	Read/Write	Read	--
TCB	--	--	Read/Write	Read	

ExpB	--	--	Read/Write	Read	--
ExpD	--	--	--	Read/Write	--
VNFD	-- (***)	--	Read	Read	Read/Write (*)
PNFD	--	--	Read	Read	Read/Write (*)
NSD	--	--	Read/Write	Read	Read (**)

(*) The on-boarding of VNFDs and PNFDs from the site manager is performed on the specific site and the related descriptors are automatically loaded from here to the centralized I/W framework catalogue.

(**) In general, a site manager can on-board NSDs in its own site. However, NSDs managed by 5G EVE platform are defined by an experiment developer during the experiment design and they are on-boarded through the portal and, from here, to the I/W framework catalogue that distribute them to the target sites. For this reason, these NSDs are “owned” by an experiment developer, who is the one with the rights to modify or remove them (through the I/W framework catalogue).

(***) The VNF Provider is not allowed to directly on-board VNFDs, since a manual validation of the VNF packages is required at the site level.

3 Interactions between 5G EVE Portal elements

This section will focus on describing the role of the experimentation tools and the catalogues in 5G EVE, analysing their interactions with 5G EVE users and 5G EVE architectural components. As already stated in the previous section, some services provided by 5G EVE Portal to all possible actors are implemented in tasks 4.1 and 4.4, while other services will be designed and implemented in the remaining tasks of work package 4. The former will be explained in detail next, while the latter will be presented in deliverable D4.2 [4] in December 2019.

3.1 High-level interactions between 5G EVE users and 5G EVE platform

In this section the manner of interaction between the 5G EVE users and the 5G EVE platform and its tools is explained. The way to design, schedule, deploy, execute and monitor an experiment over the 5G EVE platform is detailed in the following sub-sections, providing a handy user manual for future verticals collaborators of the 5G EVE project, and allowing for an efficient use of the resources of the 5G EVE site facilities.

3.1.1 Experiment design and definition

This is the first stage of an experiment, where verticals, VNF providers and Experiment developers prepare all the required material to allow experimenters to request the experiment in the next stage explained in 3.1.2. Site managers are also involved in this stage to on-board (or not) the VNFs prepared and uploaded to the portal by the VNF providers.

Figure 4 shows a high-level view of all interactions between the main actors involved in this stage (in white boxes) and all components provided by the 5G EVE platform (in yellow boxes). This figure shows the five main steps required to design and define an experiment, which will be explained next.

The first two steps can be executed in parallel and consist in providing the details of the service from the vertical to the VNF provider and to the Experiment Developer correspondingly. In the first step, the VNF provider receives the details of the service from the vertical so to create the service VNF, which is specific for the service provided by a given vertical. In 5G EVE, each site has its own Management and Orchestration (MANO) platform implementation, so the VNF provider must prepare all descriptors required by all sites where the experiment will be executed. In the second step, the Experiment Developer receives information from the vertical about the service.

In the third step, the VNF provider uploads the corresponding VNF packages to the Portal, selecting the sites where these packages must be on-boarded. The Portal contacts with the ticketing system (see Section 0), which generates the proper tickets to the corresponding site managers to notify them about the requirement of the VNF provider. Site managers must check and validate the integrity and functionality of the received VNFs. If the VNFs pass this validation, the site managers on-board the VNF locally in their sites. This information is then propagated towards the I/W framework, which in turn is available to the Portal.

An Experiment developer, based on the information provided by the vertical and the services provided by 5G EVE platform detailed in the catalogue, has to prepare in the fourth step (i) all descriptors defined by the 5G EVE project: Vertical Service Blueprint (VSB) (see Annex B), Context Blueprints (CB), Test Case Blueprints⁷(TCB) and Experiment Blueprints (ExpB); and (ii) descriptors defined by other standard developing organizations (SDO) like the Network Service Descriptor (NSD) defined by ETSI NFV. Finally, the Experiment Developer requests the on-boarding of all descriptors in the fifth step, shown in Figure 4. The Portal provides a service to validate the correctness of all these descriptors.

⁷The specific format of Test Case Blueprints is under definition in WP5.

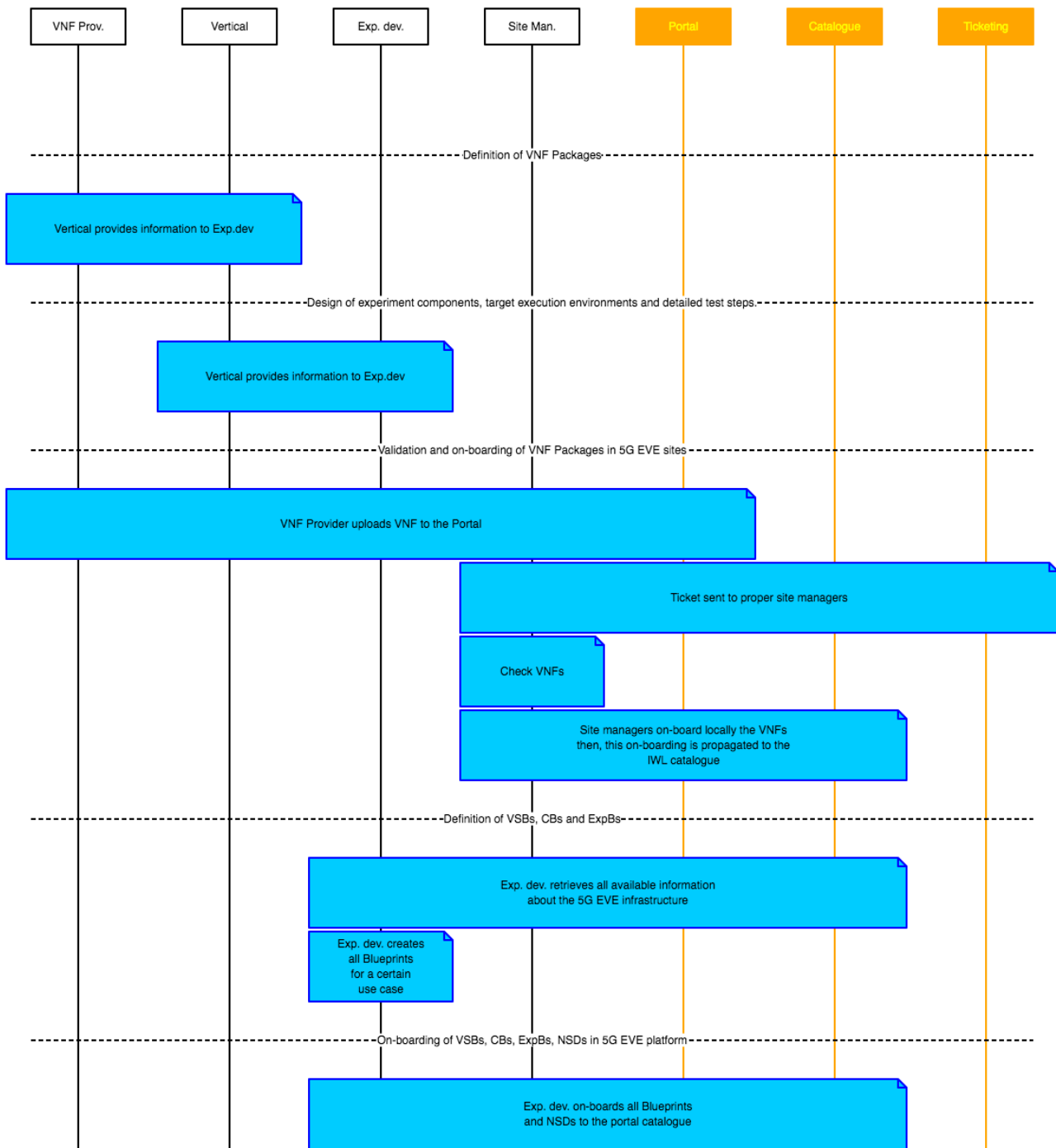


Figure 4: High-level procedure for experiment design and definition phase

3.1.2 Experiment preparation

The **experiment preparation phase** allows the experimenters to describe in detail the kind of experiment they want to run and to schedule that in a suitable period, so that site manager(s) can prepare and configure the virtual environment at the selected 5G EVE site(s). As already explained in section 2.1, the description of an experiment is specified through an Experiment Descriptor ExpD that an experimenter can declare using the 5G EVE portal. In 5G EVE there are two main mechanisms to define an experiment:

- Using a **Service/Experiment Blueprint**, which consists of a service/experiment template defined by an experiment developer in a previous stage. On the one hand, a service blueprint is a high-level template used to capture the service requirements specified by the vertical. It describes the vertical service processes in terms of functional components and their service sequence. In addition, it contains the

fields to define the service constraints, Service Level Agreements (SLA) and other vertical service requirements, as well as fields for the vertical service outputs in terms of monitoring parameters like metrics and KPIs. On the other hand, an experiment Blueprint includes other types of information required to prepare a testbed where the service can be tested in a wide set of environments, parameters, emulators, probes, etc. Several experiment Blueprints can be associated to a single service Blueprint. Both the service blueprint and its experiment blueprints are defined for a particular use-case and includes several experiment alternatives that the experimenter must select. In this document we present the Service Blueprint and particularly, a manual for experiment developers, to design these Blueprints in Annex A.

- **Intent-based tool**, which is an application that maps an experiment intent provided by the experimenter in free format to an Experiment Blueprint and its open parameters. The application interacts with the experimenter requesting missing parameters, if any. More information about the intent-based tool can be found in section 4.4.

In both cases, the result of the procedure is an ExpD that can be used to request the instantiation and execution of the experiment.

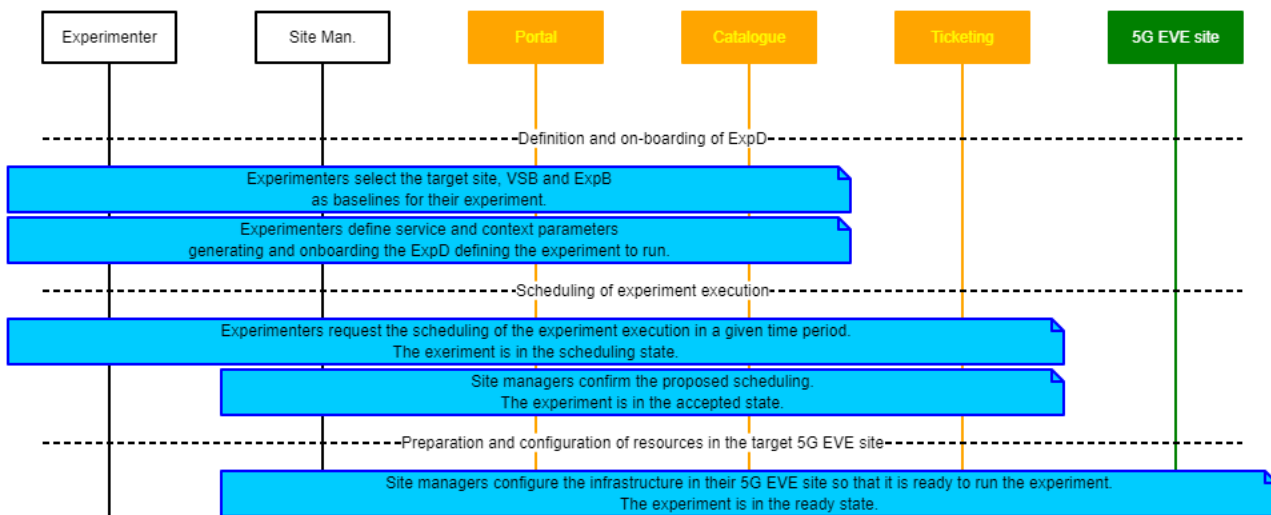


Figure 5: High-level procedure for experiment preparation phase.

Figure 5 shows the high-level interactions between the 5G EVE actors and the different components of the 5G EVE platform during the three main steps of the experiment preparation phase, assuming an approach based on service and experiment blueprints.

The first step is triggered by the experimenter, who accesses a guided procedure provided by the portal to select the ExpB and to complete the parameters for the definition of the ExpD. The experimenter can initially choose the 5G EVE sites where he/she wants to run the experiment, select one of the VSBs available for that site and, finally, select the desired ExpB from the ones associated to the VSB. Then, the portal visualizes the parameters that can be configured in both the VSB and CBs referred by the ExpB, so that the experimenter is able to define the service and context parameters for the specific configuration of the experiment. This configuration is translated by the portal in a new ExpD, which is then onboarded in the catalogue. It should be noted that this initial step related to the creation of the ExpD may alternatively be performed through the “intent-based tool”, further described in section 4.4.

The second step allows experimenter and site manager to schedule the experiment in a suitable time period. The action is started by the experimenter, who requests the scheduling of the experiment proposing a possible time slot through the portal. The experiment is thus in the *scheduling* state. The request is automatically included in a new ticket, managed through the ticketing tool and addressed to the site manager. The same ticket is used to confirm the proposed period or converge on a different one and the experiment moves to the *accepted* state.

The third and final step involves the preparation of the environment at the target 5G EVE site and it is performed manually by the site manager. For example, it may configure the required PNFs or reserve quotas of virtual resources for the given experiment. Once this step is completed, the experiment is updated on the portal so that it becomes in *ready* state and the following execution phase can be started.

3.1.3 Experiment execution

When both the experiment is in the ready state, and the date within the scheduling period, an experimenter can request the instantiation and configuration of the corresponding experiment (this can be automatically triggered by the system if the experimenter has requested this service in the previous phase). As shown in Figure 6, the final action of this first step is started by an element in the Portal, which contacts the I/W framework to request the instantiation of such experiment. When all components of the experiment are up and running, the I/W framework answers back the Portal with the details of the deployed service.

In the second step of the experiment execution, an element of the Portal starts the execution of all tests included in the ExpD in sequence. Experimenters can access monitoring information while the experiment is running (step 3 in Figure 6) or after all tests have finished (step 4 in Figure 6).

It is important to notice that, after the execution of all tests included in the ExpD, Experimenters may check the KPI validation information available through the Portal. The details of such information will be described by work package 5 of the 5G EVE in a different deliverable.

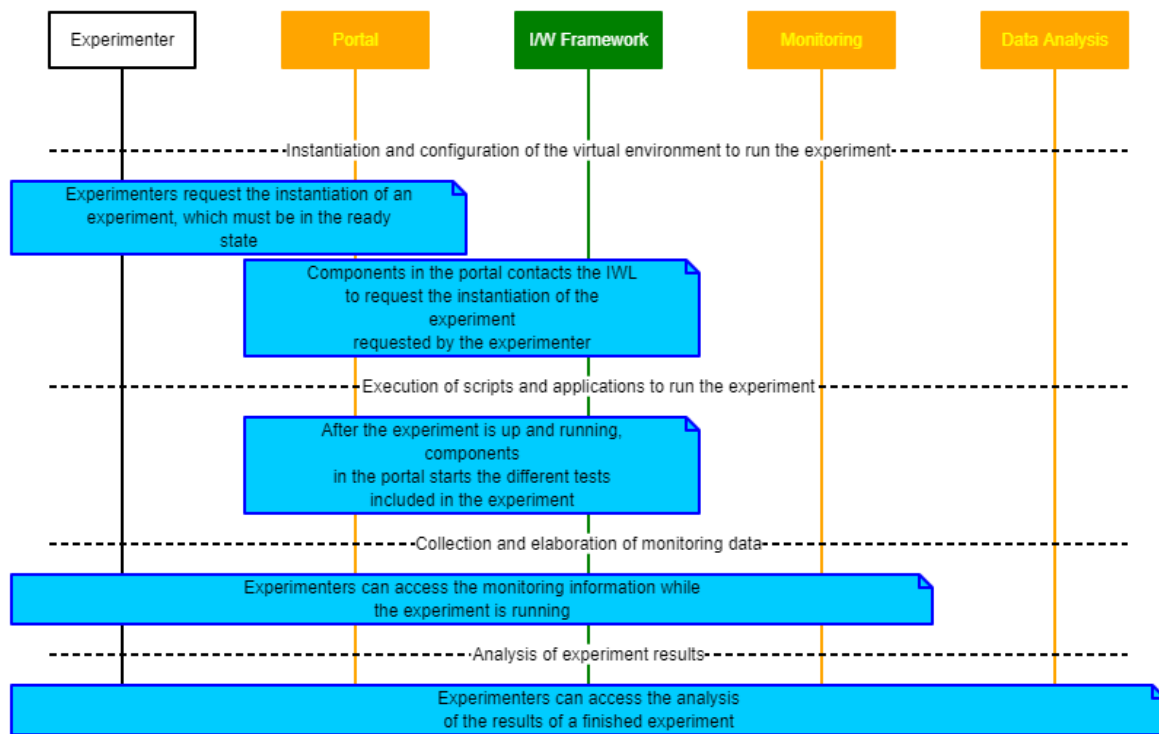


Figure 6: High-level procedure for an experiment execution

3.2 Internal workflows of 5G EVE platform for experiments management

This section defines the major workflows, internal to the 5G EVE platform, related to the different phases of the experiment lifecycle. The workflows will focus on the interactions between the 5G EVE portal components, with special attention to the experimentation tools and the catalogues, and the other entities of the 5G EVE platform.

3.2.1 Experiment design and definition phase

As mentioned in section 2.1, the experiment design and definition phase allows the VNF providers and the experimenter developers, with the support of the verticals, to onboard the blueprints of vertical services, contexts and experiments, together with the associated VNF packages, NSDs and translation rules, in the 5G EVE platform. The high-level interactions between the different actors and the 5G EVE Portal or the 5G EVE sites components has been described in section 3.1. Starting from these user-to-system interactions, in this section we analyse in detail the related internal communications within the 5G EVE Portal.

The major experiment design actions that require the active involvement of the 5G EVE platform are the following:

- Onboarding of VNF Packages from VNF Providers
- Definition and on-boarding of Context Blueprints (CBs)
- Definition and on-boarding of Vertical Service Blueprints (VSBs)
- Definition and on-boarding of Experiment Blueprints (ExpBs)

In the following subsections, the workflows for the major procedures in the experiment design actions are described in detail.

3.2.1.1 Onboarding of VNF Packages from VNF Providers

The workflow for the onboarding of VNF packages in a generic site i is represented in Figure 7.

The VNF developer uploads the VNF Package on a temporary repository provided by 5G EVE portal, so that it becomes available to a public URL (messages 1 and 2) for the time needed to be onboarded in the target sites. The VNF package typically includes the image with the developed application, the VNFD and several scripts and metadata for the management of the application. It should be noted that the format of VNF packages and VNFDs are dependent on the specific site where the VNFs will be deployed. When VNF package is available on 5G EVE temporary repository, the VNF developer requests its onboarding on the target site, using 5G EVE Ticketing tool (message 3)⁸. The tool dispatches the ticket to the site manager (message 4), who downloads the VNF package from the temporary repository (message 5) and processes it. The VNF image is extracted from the package (step 6) and manually validated (step 7), for example verifying the licenses of the software, the compatibility of the virtual machine requirements with the capabilities of the virtual environment offered by the site and validating the performance of the virtual application when running in the target system. If the VNF validation gives a positive result, the VNF image is uploaded on the Virtual Infrastructure Manager (VIM) installed in the operational infrastructure of the site i (message 8) and the associated VNFD is onboarded in the NFVO catalogue deployed on the same site (message 9). This last action triggers the procedures for the VNFD onboarding, as discussed in the following.

The workflow shown in Figure 8 describes the specific steps of the VNFD onboarding. The process starts when the manager of a generic site i issues the VNFD onboard request (message 1), containing the full VNFD. The catalogue in site i processes the request validating the VNFD format and assigning a new identifier `VnfPkgIn-fold` (called `X_ID` in the picture) to the VNFD, which is then stored in the catalogue. The identifier is sent back to the site manager in the VNFD onboard response (message 2).

As consequence of the subscription performed by the I/W framework catalogue during the initial synchronization between I/W framework and sites catalogues (see in D3.2 [6], section 4.3.3.1, for further details), the local catalogue notifies the I/W framework catalogue about the new VNFD, providing its identifier. Finally, the I/W framework catalogue retrieves the VNFD by issuing a query to the site catalogue (messages 4 and 5) and update its repository adding the new VNFD with a reference to site i .

⁸ This request can also be sent automatically by the portal after the VNF packages have been successfully uploaded in the temporary repository.

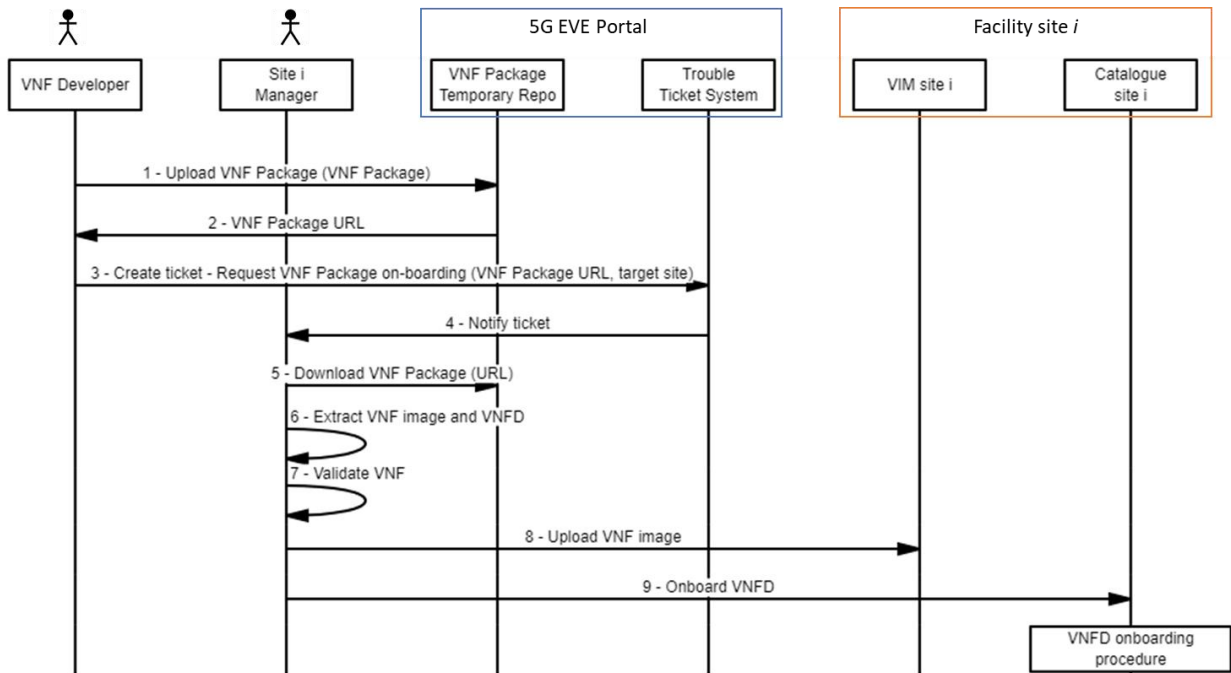


Figure 7: Onboarding of a new VNF Package provided by a VNF developer

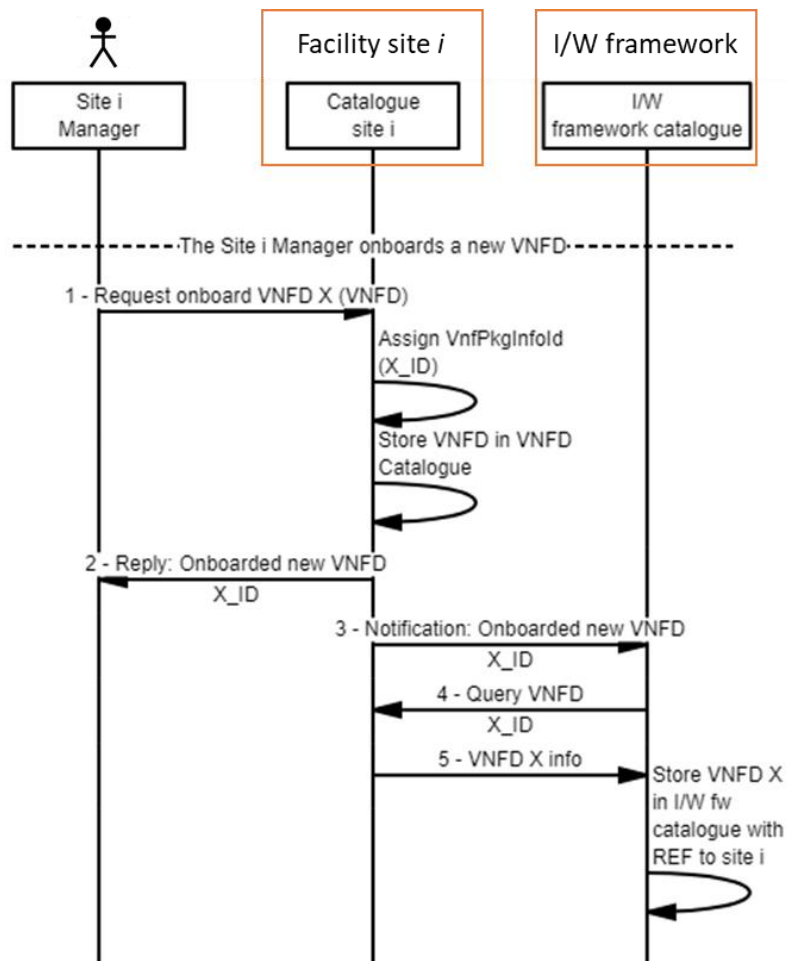


Figure 8: Onboarding of a VNFD in a 5G EVE site

3.2.1.2 Onboarding of context, vertical service and experiment blueprints in 5G EVE Portal

The definition and onboarding of blueprints requires communications among the catalogues at the different layers of the 5G EVE platform, to guarantee the synchronization of the blueprints and associated descriptors.

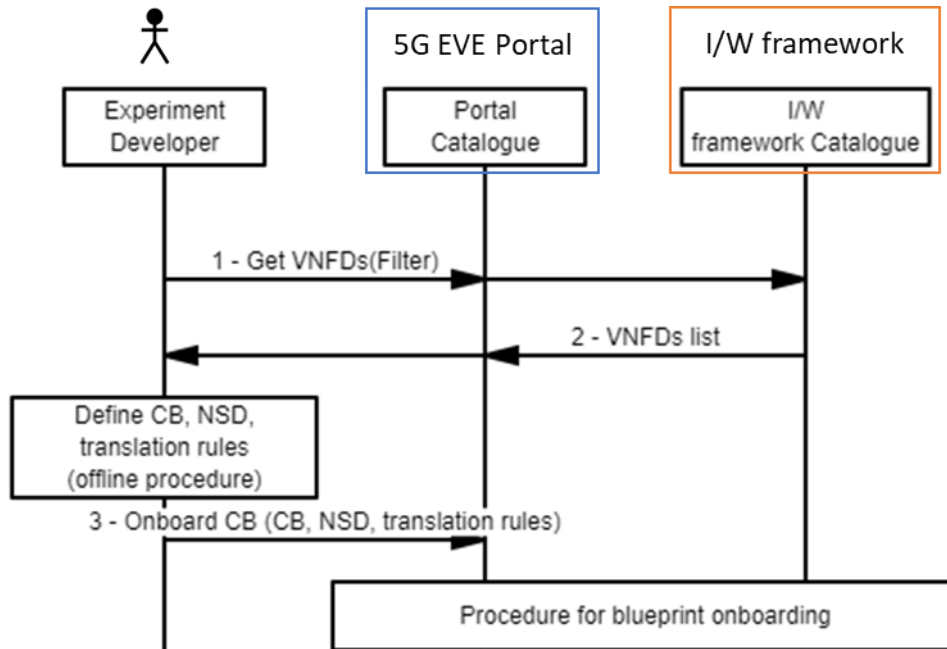


Figure 9: Onboarding of context blueprint.

The first step for the definition of a complete experiment blueprint consists in the definition of several context blueprints that will be adopted in the experiment, followed by the definition of the blueprint for the vertical service associated to the experiment. Finally, the experiment blueprint can be onboarded with references to these context and vertical service blueprints. Each blueprint may include several metrics and KPIs to be collected and evaluated for the validation of the experiment.

Figure 9 shows the workflow for the onboarding of a context blueprint. A context blueprint is typically defined by an experiment developer, based on the VNFDs made available for monitoring tools, network probes or traffic generators in the various 5G EVE sites. Here we assume the VNFDs are already onboarded in the system by the site managers, following the procedure described in section 3.2.1.1.

The CB onboarding procedure starts with the retrieval of the relevant VNFDs. This is requested with a query (step 1) to the 5G EVE Portal catalogue, which can optionally specify a suitable filter. For example, the experiment developer may want to limit the research to a given site or to a specific type of functionality, e.g. the generation of background traffic. Internally, the Portal Catalogue will interact with the I/W framework Catalogue, where the VNFDs are stored, to collect and return the list of requested VNFDs (step 2). Using the VNFDs information, the experiment developer produces the CB, together with the associated NSD and the translation rules to map them together. The generated NSD includes the references to one or more VNFDs available at the sites. Finally, all the resulting elements (CB, NSD and translation rules) are onboarded in the Portal Catalogue (step 3). This request will trigger a generalized procedure for the onboarding of a blueprint in the catalogues, presented in detail in section 3.2.1.3, which coordinates the storage of the different elements in the relevant catalogues.

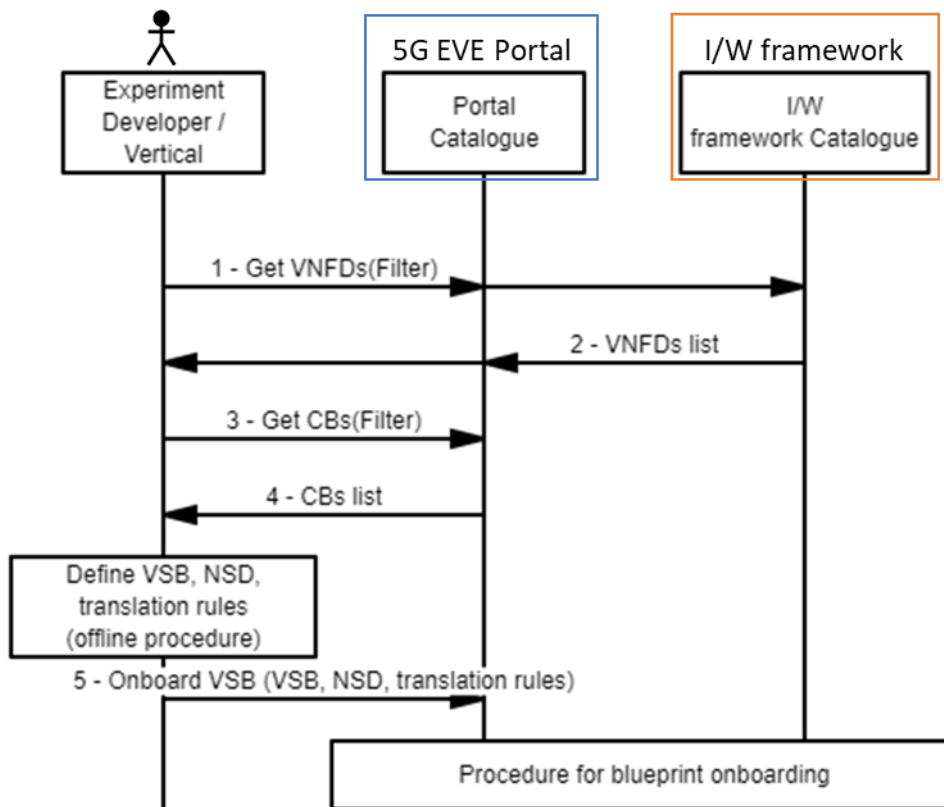


Figure 10: Onboarding of vertical service blueprint.

The workflow for the onboarding of a VSB, shown in Figure 10, is very similar to the CB onboarding. The main actor is the experiment developer, who can be potentially supported by the vertical. The initial steps, as in the previous workflow, are needed to collect all the information required to produce the VSB and the associated NSDs and translation rules. In this case, the experiment developer needs to retrieve the VNFDs of the service-specific applications, as defined by the VNF providers, and the context blueprints. Some of the CBs, in fact, will be referred in the VSB as potential contexts applicable to the given vertical service, to help the experimenter in the configuration of the experiment. Both VNFDs and CBs are queried on the Portal Catalogue (step 1 and step 3); VNFDs are returned through a query to the I/W framework catalogue (step 2), while CBs are returned directly from the Portal Catalogue (step 4). Finally, the experiment developer requests the onboarding of the VSB (step 5), triggering the generalized procedure for blueprint onboarding at the catalogues (ref. section 3.2.1.3).

The workflow for the definition and onboarding of an ExpB involves an additional component of the 5G EVE portal, called Experiment Blueprint Builder⁹, which is designed to help the experiment developer in the experiment design. The Experiment Blueprint Builder provides a “wizard” that guides the user in the different steps of the ExpB definition, wrapping the interaction with the Portal Catalogue and showing a graphical interface where the user can select, complete and upload the information needed to build the ExpB. The detailed workflow is represented in Figure 11 and Figure 12.

⁹ The Experiment Blueprint Builder is out of the scope of this deliverable and it will be documented in D4.3.

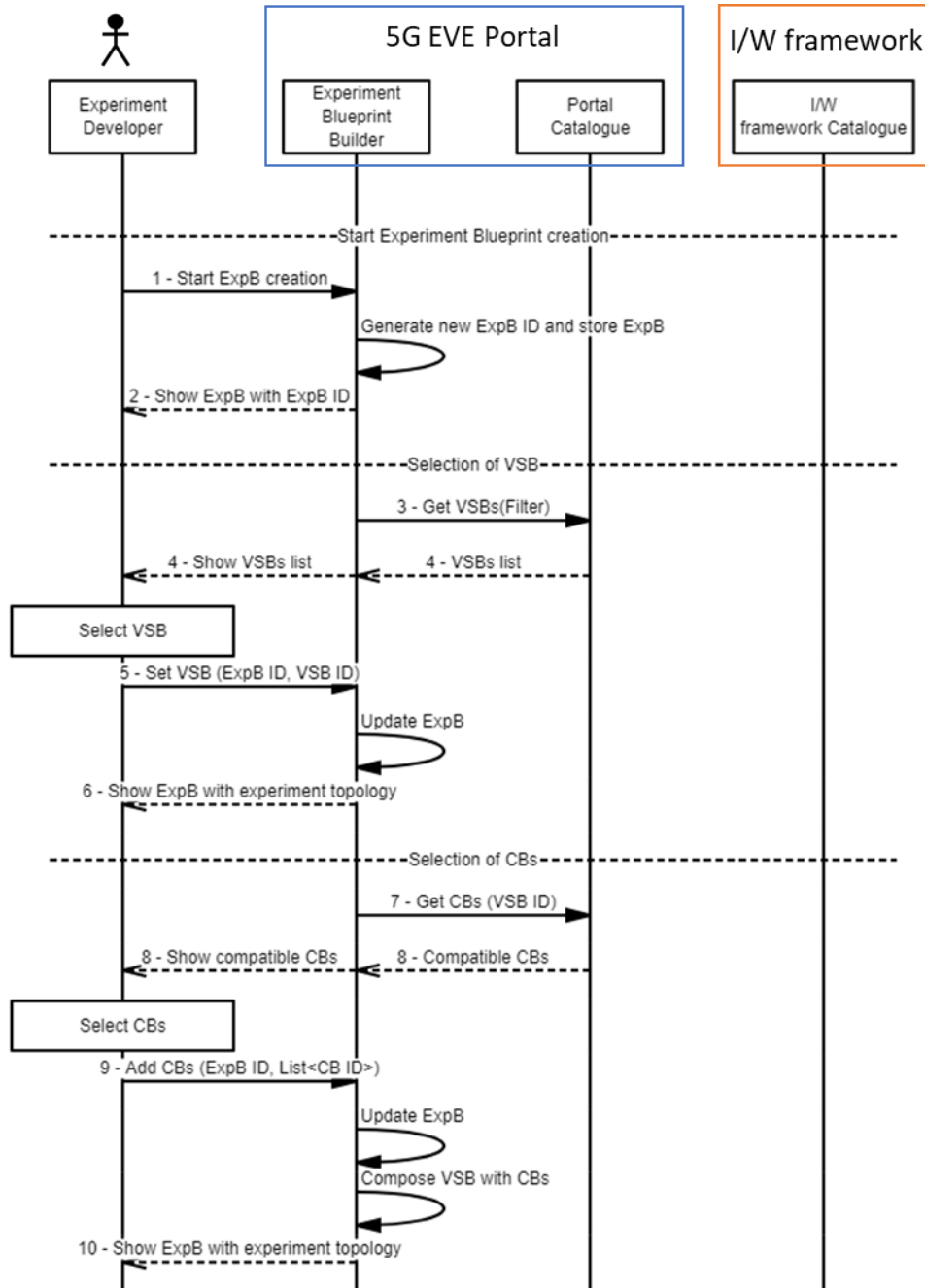


Figure 11: Definition and onboarding of experiment blueprint (1).

The experiment developer requests the creation of a new ExpB to the Experiment Blueprint Builder (step 1), which generates an empty ExpB with a new identifier and stores it locally (step 2). As explained in section 2.1, an experiment blueprint needs to be associated to a VSB and, optionally, to several CBs. For this reason, the Experiment Blueprint Builder interacts with the Portal Catalogue to retrieve the available VSBs (step 3) and the compatible CBs (step 7), showing their lists on the graphical interface (steps 4 and 8 respectively) and allowing the experiment developer to select the desired ones (steps 5 and 9). The ExpB is updated accordingly at the Experiment Blueprint Builder, composing the VSB with the selected CBs and showing the resulting experiment topology on the graphical interface (step 10).

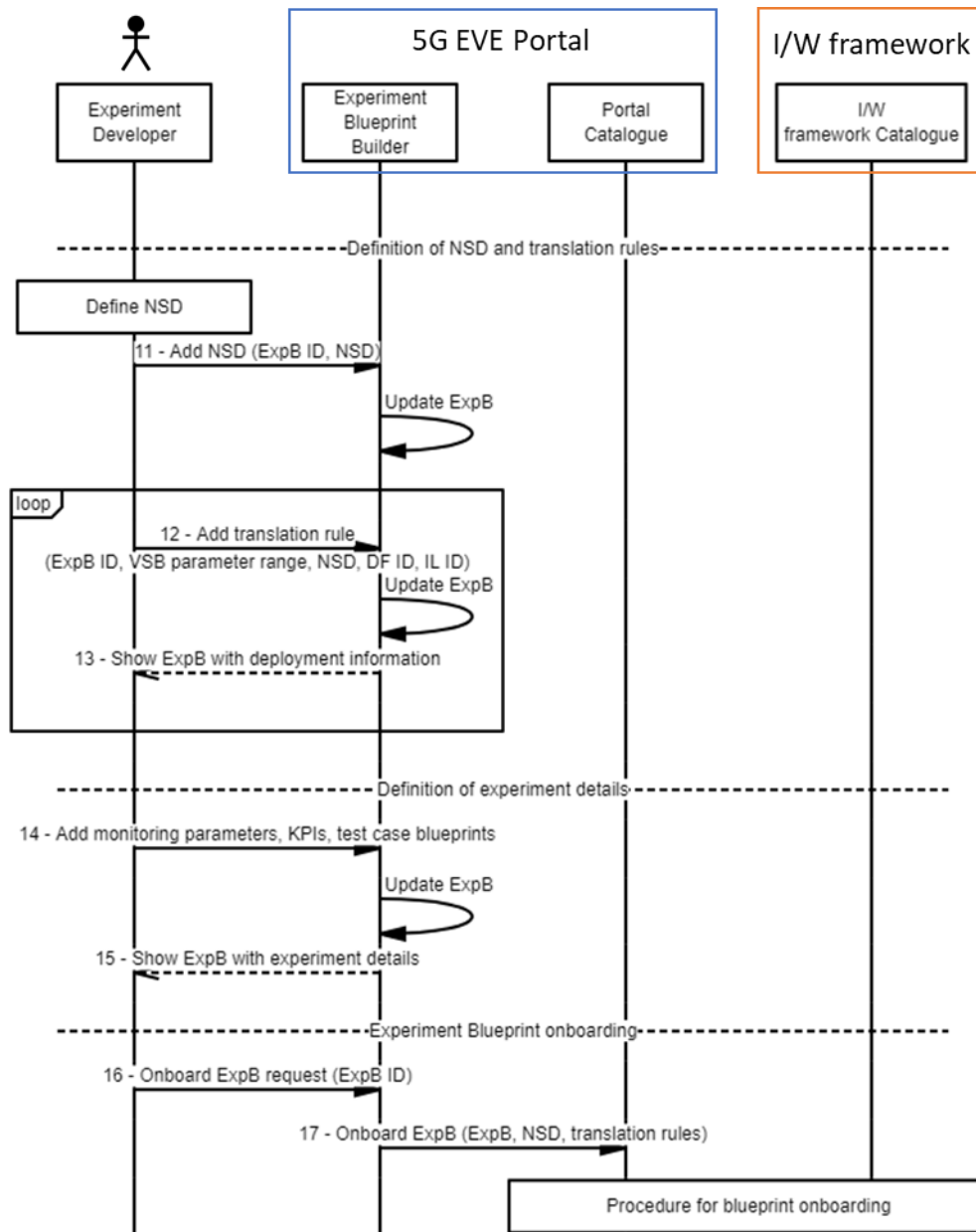


Figure 12: Definition and onboarding of experiment blueprint (2).

After this initial part of the procedure, the experiment developer needs to specify how the experiments based on such blueprint will be deployed in the 5G EVE infrastructure, for different ranges of the associated service parameters. The initial step is thus the definition of a suitable NSD to associate to the ExpB (step 11). In this initial phase of the project we assume that the NSD is built manually by the experiment developer. However, a dedicated tool to facilitate the auto-generation and composition of NSDs starting from their blueprints is currently under development. Beyond the NSD, the experiment developer needs to indicate also the translation rules, specifying how different VSB parameter ranges are mapped on specific deployments of the given NSD (step 12). Such deployments are specified through pair of NSD fields “Deployment Flavour” (DF) and “Instantiation Level” (IL) [5].

The last part of the ExpB definition allows the experiment developer to define additional details about the experiment execution, like the type of metrics and KPIs to be measured or the test scripts to run, defined in the Test Case Blueprints (step 14). The specific model and format of such experiment parameters is currently under discussion in WP5 and will be refined during the Experiment Blueprint Builder implementation. Finally, when the ExpB is completed, the experiment developer requests its onboarding in the 5G EVE platform (step 16) through the Experiment Blueprint Builder that, in turn, sends a request to the portal catalogue providing the

generated ExpB and the associated NSD and translation rules (step 17). This request triggers the blueprint onboarding procedures at the catalogues level, as specified in section 3.2.1.3.

3.2.1.3 Onboarding of a generic blueprint across 5G EVE catalogues

In the general case of a blueprint onboarding, valid for VSBs, CBs and ExpBs, the onboarding request is issued from the experiment developer and it includes three elements: (i) the blueprint itself, (ii) the corresponding NSD and (iii) the translation rules to map a VSD/CD/ExpD based on the given blueprint into the triple $\langle \text{NSD ID}; \text{DF ID}; \text{IL ID} \rangle$ that identifies the deployment of such descriptor. The blueprint and the translation rules are stored at the Portal catalogue, while the NSD is stored at the I/W framework and, from here, onboarded on the target site. The entire procedure needs to be properly coordinated to guarantee the consistency between the information available at the two catalogues, as well as the suitable references maintained at the Portal catalogue to link the blueprint with the associated NSD at the I/W framework. Figure 13 represents the workflow for the onboarding of an experiment blueprint at the catalogues level, if the corresponding NSD must be onboarded on the catalogue of a single site i . Exactly the same workflow is applicable also to the onboarding of context and vertical service blueprints. It should be noted that, while the original source of the request is always the experiment developer, the actual formatting and dispatch of the request is typically mediated through the portal. For example, in the case of the ExpB, the request is compiled by the Experiment Blueprint Builder, as analysed in section 3.2.1.2. In the figure, we ignore this mediation to make the workflow more general and applicable to the different types of blueprints. Moreover, in this workflow we are considering a single target site for simplicity. However, the NSD may also be onboarded on multiple sites. This would require the interaction of the I/W framework Catalogue with more site catalogues, following the same procedure indicated in the workflow for site i .

As shown in the picture, the initial request (message 1) includes the ExpB, the NSD and a set of translation rules. The request is processed at the Portal Catalogue, which is the entity responsible to coordinate the onboarding of the different elements in the specific repositories. As first step, the target site i is extracted from the ExpB (step 2). The target site represents the 5G EVE facility where the NFV network service will be deployed and, as such, it is the site where the NSD will need to be onboarded. The Portal Catalogue sends a request to the I/W framework Catalogue to request the onboarding of the NSD, specifying also this target site (message 3). The I/W framework Catalogue validates the NSD format and content, also verifying that the descriptors of the VNFs and PNFs referred by the NSD are available at the target site (step 4). If everything is validated, the I/W framework Catalogue generates an ID for the NSD (step 5) and stores it in the local repository (step 6). The ID assigned to the NSD by the I/W framework Catalogue, which is also returned in the response to the Portal Catalogue (message 7), is the ID that will be used globally in the 5G EVE platform to refer the NSD itself.

To make the NSD available also at the target site, the I/W framework Catalogue requests the onboarding of the NSD in the catalogue of the site's NFVO (message 8), performing a translation between the ETSI NFV SOL 001 format adopted internally and the site-specific descriptor format. The catalogue at site i will in turn generate a local identifier, in this case X_loc_ID (step 9). This ID is returned in the response to the I/W framework Catalogue (message 10), which stores the association between the IDs assigned on the two catalogues (step 11). Using an asynchronous task, the NSD is also stored at the site catalogue (step 12) and, when the procedure is terminated, a notification is sent back to I/W framework catalogue (message 13), as consequence of the subscription between I/W framework and site catalogues performed during the initial synchronization (ref. to in D3.2 [6], section 4.3.3.1, for further details). At the I/W framework catalogue the status of the NSD is updated to keep trace of the remote onboarding (step 14) and a notification is sent back to the Portal Catalogue (message 15), confirming the successful termination of the NSD onboarding procedure.

As final steps, the Portal Catalogue onboards the ExpB, generating a new identifier for the ExpB (step 16) and storing the ExpB content and the translation rules, including also a suitable reference to the global identifier assigned to the NSD (step 17). Finally, the ExpB identifier is returned to confirm the successful ExpB onboarding within the entire chain of 5G EVE catalogues.

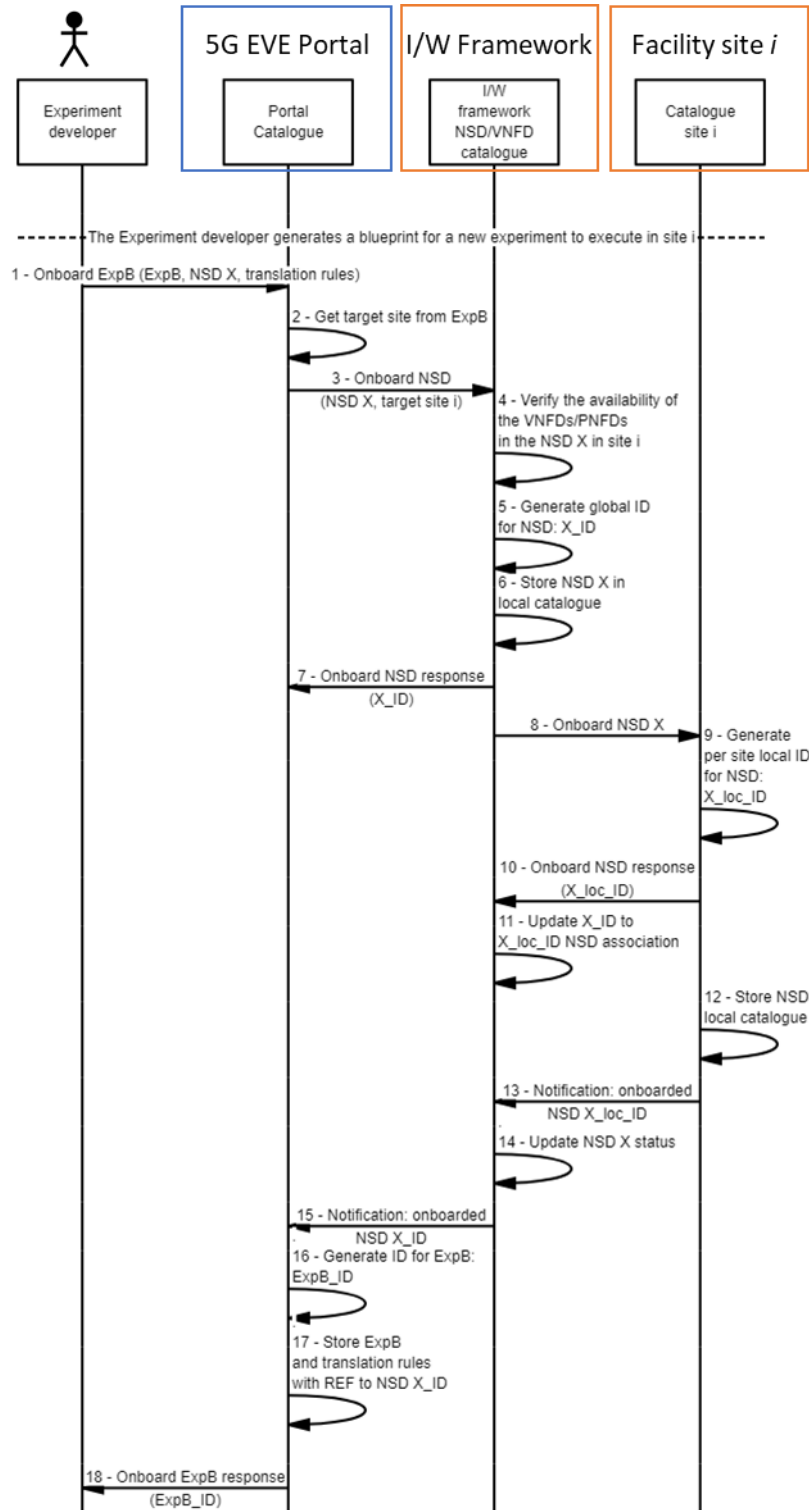


Figure 13: Onboarding of 5G EVE blueprints across 5G EVE catalogues

3.2.2 Experiment preparation phase

Section 3.1.2 has presented the high-level actions for the request and preparation of an experiment, highlighting the interactions between the involved actors and the components of the 5G-EVE platform. The major procedures to request an experiment are the following:

- Definition and on-boarding of Experiment Descriptors (ExpDs)
- Scheduling of the experiment execution
- Preparation and configuration of the target 5G EVE site(s)

For each of them, we present the detailed workflows in the following subsections.

3.2.2.1 Definition and onboarding of experiment descriptors

This section presents the workflow that allows an experimenter to define and onboard an experiment descriptor that will be used to run the desired experiment. In this workflow we assume that the ExpD is defined starting from the selection of an ExpB, followed by the specification of the related parameters. However, this is not the only option offered by the 5G EVE platform: the ExpD can also be created using an intent based interface. In both the cases, the resulting ExpD is then onboarded in the Portal Catalogue and made available for the following steps.

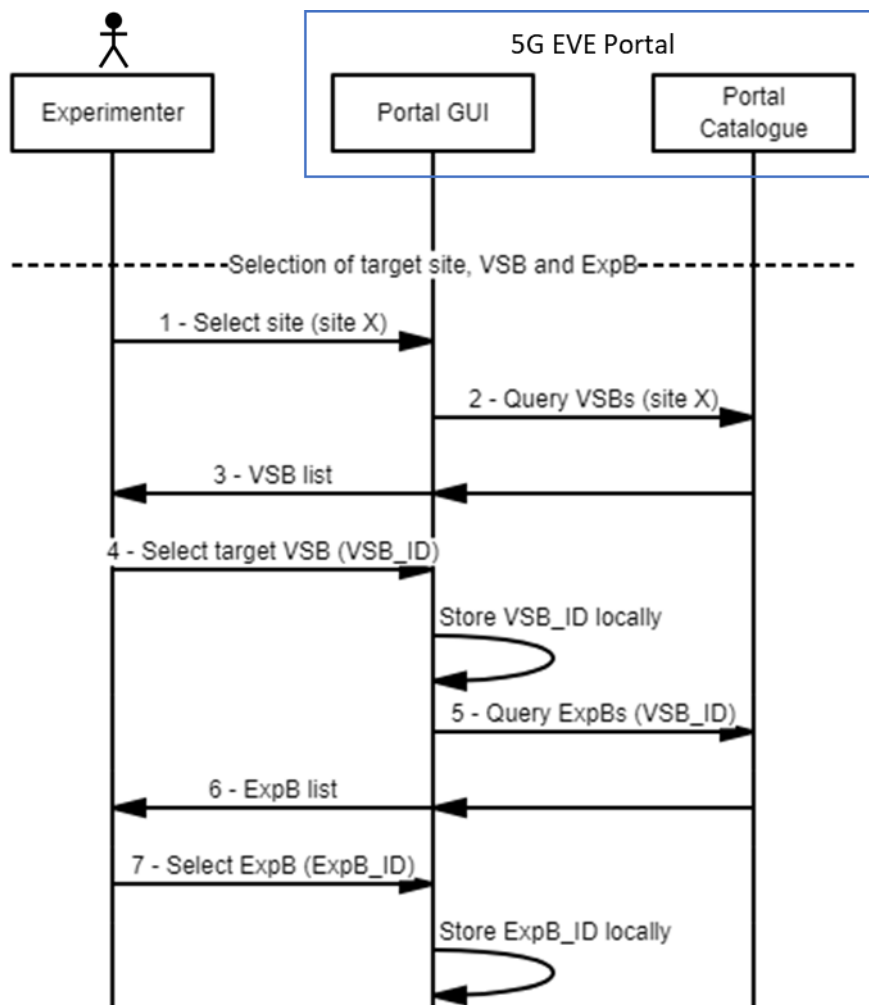


Figure 14: ExpD onboarding (1)

The overall procedure for the definition and onboarding of an ExpD is structured in two major parts. The former allows the experimenter to select the target ExpB, as shown in the workflow represented in Figure 14. The latter allows the experimenter to complete the ExpB with the required parameters values, thus obtaining a complete ExpD, which is then onboarded in the Portal catalogue. The entire procedure is mediated through the Portal GUI, guiding the user through an ordered sequence of steps.

As shown in Figure 14, when the experimenter accesses the ExpD creation page on the Portal GUI, he/she can select the target site where to run the experiment (message 1). Here we are assuming a single site experiment, but the concept can be easily extended to a multi-site one. The Portal GUI sends a query (message 2) to the

Portal catalogue to retrieve the list of VSBs that can be deployed on the given site. The returned list (message 3) is shown to the user that, in turn, selects the target VSB (message 4). The internal logic of the GUI starts to build the information associated to the ExpD, storing locally the relevant elements, in this case the VSB identifier. A query is then issued to the Portal Catalogue for retrieving all the ExpBs associated to the given VSB (message 5). Again, the returned list is shown to the user (message 6), who selects the desired ExpB (message 7). The identifier of the selected ExpB is stored locally at the Portal GUI. This completes the first part of the procedure, related to the selection of the ExpB.

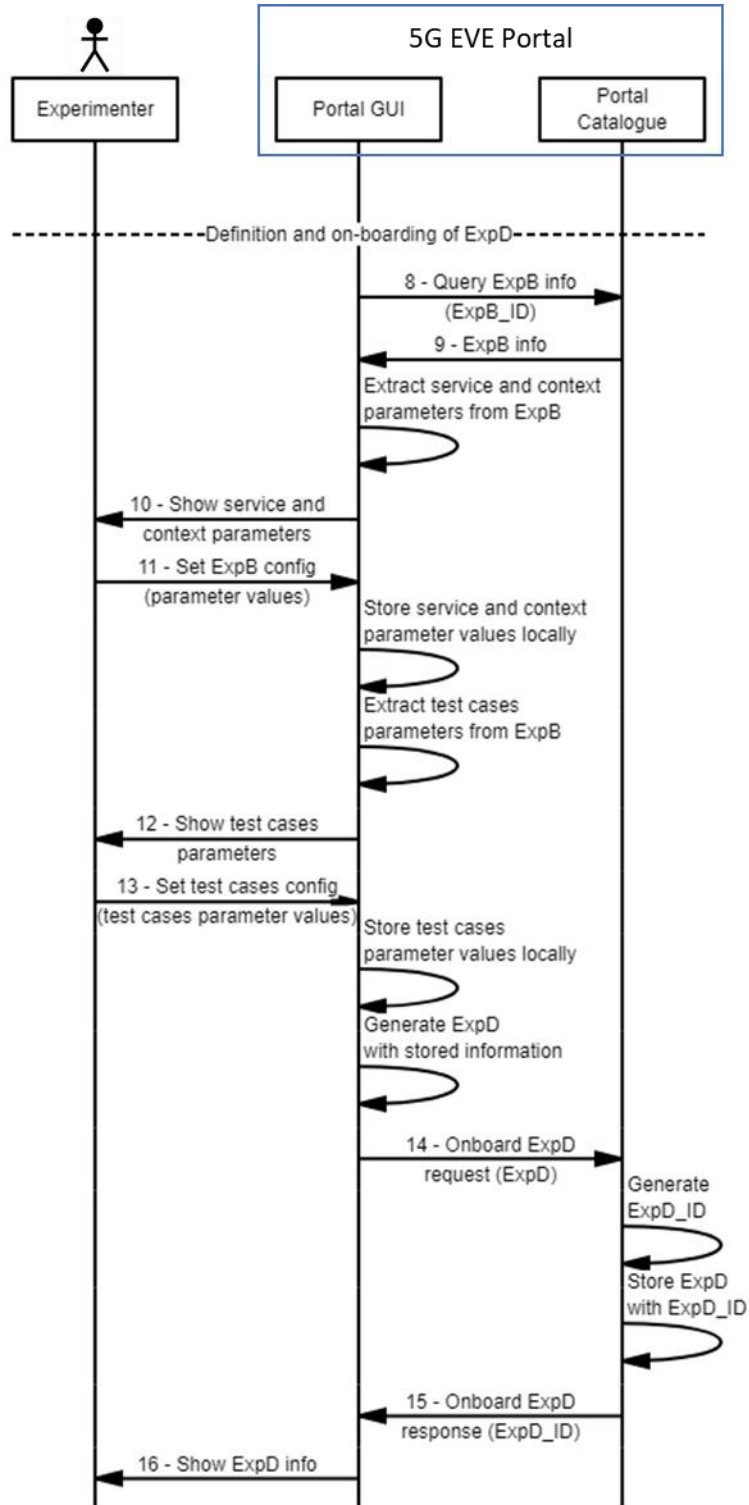


Figure 15: ExpD onboarding (2)

The second part of the procedure, represented in Figure 15, starts with the processing of the selected ExpB. The ExpB details are retrieved from the Portal Catalogue (messages 8 and 9). The Portal GUI extracts the configurable parameters related to the vertical service and to the experiment context, showing them to the user (message 10). The experimenter fills out the required information defining the experiment configuration (message 11), which is stored locally at Portal GUI. Then the parameters of the test cases included in the ExpB are extracted and shown to the user (message 12). The test cases specification provided by the experimenter (message 13) is also stored locally at the GUI. All the collected information is then used to compile and generate a full ExpD, which is then onboarded at the Portal Catalogue (message 14). Here, a new identifier is generated for the given ExpD and everything is stored in the Portal Catalogue repository. The identifier is finally returned to the Portal GUI (message 15), where the ExpD is updated and its complete information is sent back to the experimenter (message 16).

3.2.2.2 Scheduling of experiment execution

This section describes the workflow to schedule an experiment and reach an agreement between experimenter and site manager about a period suitable for the execution of the experiment in the given site. The procedure is structured in two parts: in the former the experimenter requests the scheduling of the experiment proposing a possible period; in the latter the site manager verifies the feasibility of the request and, if positive, confirms the experiment scheduling. In the workflow presented in Figure 16 we assume a positive verification; however, in case of mismatching with the availability of the site resources, the scheduling can be further negotiated.

The scheduling request procedure is coordinated through a component of the 5G EVE Portal, called Experiment Lifecycle Manager¹⁰, which is responsible to manage the entire lifecycle of a given experiment, from its creation to its termination, and to maintain up-to-date information about the experiment itself. Moreover, the procedure is also assisted through the usage of the 5G EVE ticketing system, which provides an efficient mechanism to enable the interaction between different users of the 5G EVE platform (in this case the experimenter and the site manager) on a specific topic. The ticketing system can be used for negotiating a possible date of the experiment, in case the time period initially proposed by the experimenter does not fit with the current scheduling of the resources at the target site.

As shown in Figure 16, the procedure begins with a schedule experiment request (message 1) sent to the Experiment Lifecycle Manager, where the experimenter specifies the identifier of the target ExpD and proposes a potential time slot for the execution of the experiment. The Experiment Lifecycle Manager retrieves the information about the ExpD interacting with the Portal Catalogue (messages 2 and 3) and it generates a new experiment entry in its internal repository. This experiment includes a new assigned identifier (defined as *Exp_ID* in the picture) and it is in the *scheduling* state. The identifier of the experiment is then returned to the experimenter in the schedule experiment response (message 4). It should be noted that such experiment identifier is unique in the 5G EVE platform and it can be used to query or refer to the experiment itself in the different phases of its lifecycle, still interacting with the Experiment Lifecycle Manager.

The Experiment Lifecycle Manager interacts with the ticketing system creating a new ticket (message 5), addressed to the manager of the site where the experiment will run. This ticket will include a request for the experiment scheduling, reporting information about the experiment, like its identifier, its ExpD and the proposed time slot. The identifier of the ticket will be returned in the response (message 6) and stored in the experiment entry for future references. In parallel, the ticketing system will dispatch the ticket to the site manager.

The site manager will start an offline and site-specific procedure to verify (i) the feasibility of the experiment in the target site and (ii) the compatibility of the proposed period with the availability of the resources at the site. If both verifications are positive, the site manager interacts with the Experiment Lifecycle Manager (which acts as the single point of access for all the operations related to the experiments) for updating the experiment status and confirming its acceptance (message 8). The Experiment Lifecycle Manager updates its internal repository with the new experiment state (*accepted*) and interacts with the ticketing system to update the status of

¹⁰ The Experiment Lifecycle Manager is out of the scope of this deliverable and it will be documented in D4.2.

the original ticket (message 9). As result, the ticket is notified to the experimenter, who becomes aware of the experiment acceptance (message 10).

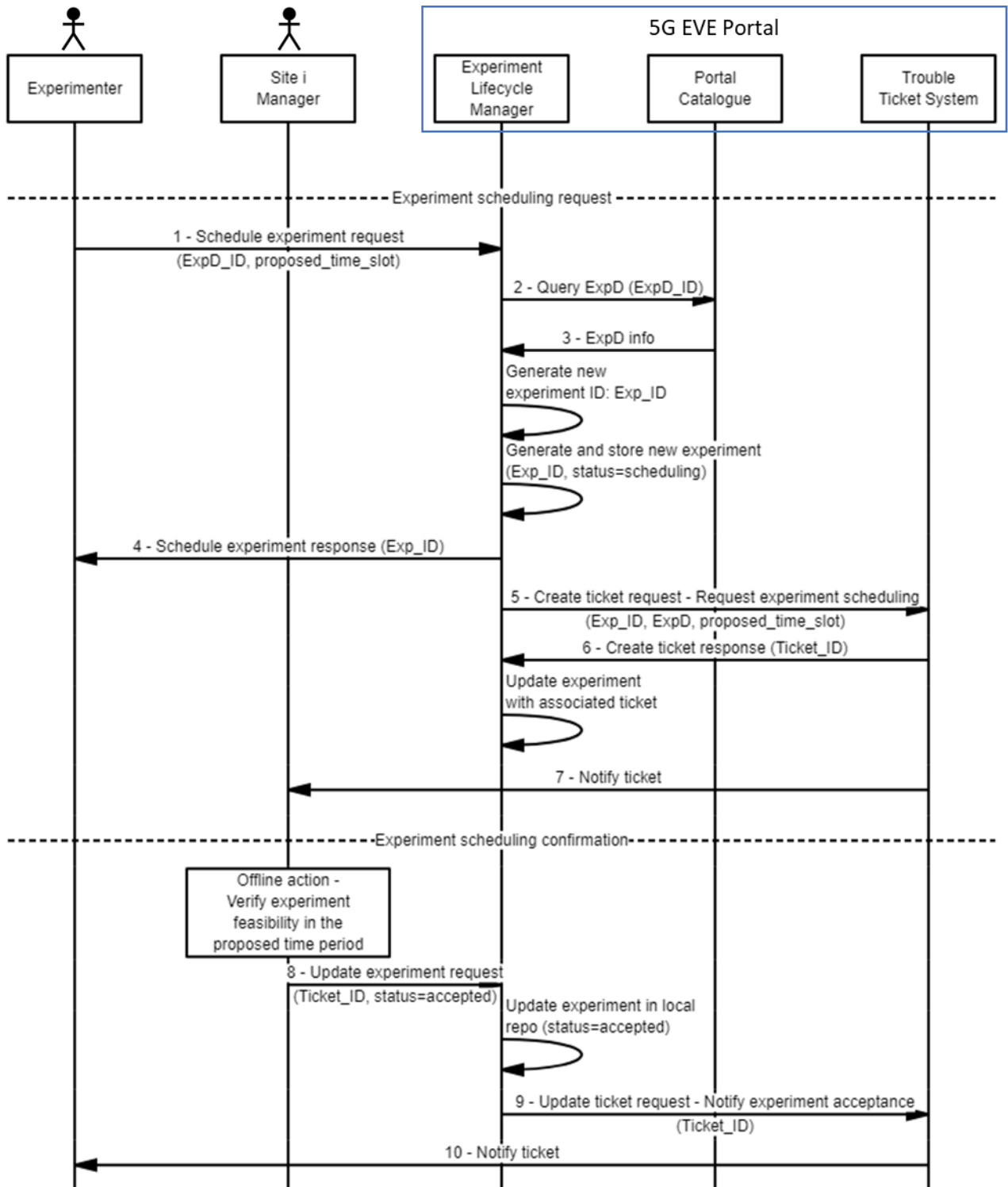


Figure 16: Experiment scheduling

3.2.2.3 Preparation of experiment environment

The preparation of the experiment environment is mostly an off-line procedure that is coordinated by the site manager and may involve a set of manual or automated actions for the deployment and configuration of the resources at the target site. These actions are independently on the 5G EVE platform and they are executed using the management systems and tools available in the specific 5G EVE facility. At the end of the procedure, the site is ready to host the components required to run the experiment and all the needed configuration are in place, so that the experimenter can proceed with the execution phase. At this stage, the only interaction needed with the 5G EVE platform is the notification about the change of the experiment state from *accepted* to *ready*, as shown in Figure 17. Once the preparation of the experiment environment is completed, the site manager interacts with the Experiment Lifecycle Manager to update the status of the experiment (message 1). The status is updates in the local repository. Moreover, a request is sent to the ticketing system (message 2) to update the ticket related to the status of the experiment. This action, in turns, triggers a notification to the experimenter who is informed about the readiness of the experiment environment.

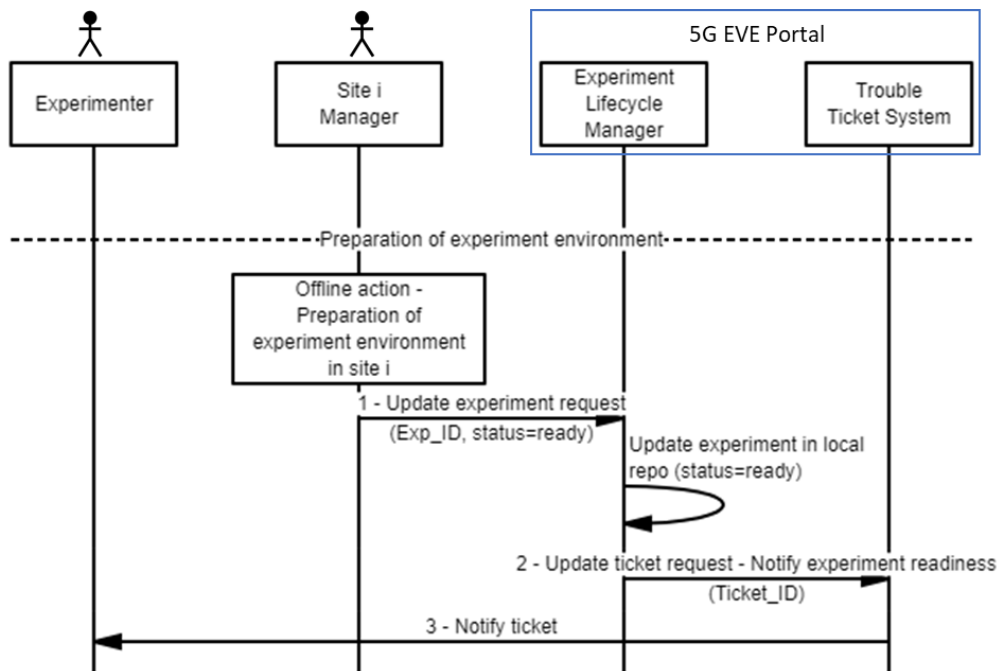


Figure 17: Preparation of experiment environment

3.2.3 Experiment execution phase

This section presents the internal workflows for the execution of an experiment over the environment prepared during the preparation phase. The major steps for the execution of the experiment are the following:

- Instantiation and configuration, where all the virtual components to run the experiment are instantiated and properly configured.
- Execution of the experiment test cases, where the scripts to run the experiments are configured with the right parameters and executed in the required order.
- Collection and elaboration of experiment data, which may happen in parallel with the test cases execution and allows to collect all the monitoring data to evaluate the performance of the service and verify the compliance with the expected KPIs.
- Analysis of the obtained results at the end of the experiment, through a set of tools of visualization tools.

3.2.3.1 Experiment instantiation

Instantiation and execution of an experiment is triggered by the experimenter through a single request towards the 5G EVE Portal and to the Experiment Lifecycle Manager. The experiment instantiation can be requested only after the experiment is in the *ready* state and when the current time is within the agreed period. This request starts a fully automated procedure that involves components at the 5G EVE Portal and at the I/W framework. At the beginning, the system instantiates and configures the virtual environment (experiment instantiation) with the virtual machines and virtual networks to run the experiment. Then, it executes the test cases and collects the monitoring data as required by the experiment specification (experiment execution). At the end of the procedure, the virtual environment is terminated, and the resources become available for further experiments. This section defines the workflow for the experiment instantiation, while section 3.2.3.2 focuses on the experiment execution.

The whole experiment instantiation-execution-termination procedure is coordinated by the Experiment Lifecycle Manager, while its enforcement is mostly managed by the I/W framework. The Multi-site Orchestrator under development in WP3 is responsible for the instantiation and termination of the NFV Network Services associated to the experiment virtual environment, during the experiment instantiation and termination steps. The Experiment Execution Manager, under development in WP5, is responsible to coordinate the different steps of the test cases execution and the collection of the associated results. Both these components are out of scope for WP4 and hence of this deliverable but are reported in the following workflows for clarity and completeness. It should be noted that the execution of processes reported in the workflow as “internal” to the I/W framework components may involve the interaction of such components to other entities belonging to the I/W framework or to the 5G EVE sites tools. Since these interactions are out scope for WP4, they are omitted in this document.

Figure 18 represents the workflow for the instantiation of the virtual environment associated to an experiment, where all the virtual functions of the NFV network service associated to the experiment are instantiated in the target site and properly interconnected. The workflow begins with a request from the experimenter to deploy the experiment (message 1). The request is processed by the Experiment Lifecycle Manager, which verifies the correct status of the given experiment: in fact, the experiment must be in the *ready* state and the time within the agreed period, to guarantee that the site environment is ready for its deployment and execution.

The Experiment Lifecycle Manager handles the deployment procedure in an asynchronous manner, updating the internal entry of the experiment to the *instantiating* state and replying with an acknowledgement (message 2). From this step on, the experimenter may interact with the Experiment Lifecycle Manager to query the status of the experiment and verify its status and evolution. In parallel, the Experiment Lifecycle Manager retrieves from its internal records the information related to the experiment (e.g. ExpD and related translation rules) to determine the proper deployment of the NFV network service needed to instantiate the experiment component. In particular, it elaborates the NSD, deployment flavour and instantiation level of the required network service and it sends an instantiation request (message 3) with the related identifiers to the Multi-Site Orchestrator (MSO) of the I/W framework. At the I/W framework the instantiation procedure is handled asynchronously. The MSO generates a new identifier for the requested network service (NS_ID in the picture), returning it in the response (message 4). It should be noted that this procedure may involve the exchange of multiple messages, i.e. with an initial request to create the NS_ID, followed by the actual instantiation request for the network service with that NS_ID.

At the Experiment Lifecycle Manager, the NS_ID is stored in the experiment entry of the local repository and it is used to monitor the evolution of the instantiation procedure, through a subscribe/notify mechanism. As soon as the NS_ID is returned, the Experiment Lifecycle Manager subscribe with the MSO in order to receive notifications about the status of the network service (message 5). In parallel, the MSO proceeds with the actual instantiation of the network service, with a set of interactions with the orchestrator at the target site facility and other components of the I/W framework (omitted in this workflow representation). When the instantiation is completed, the internal MSO record related to the network service is updated with the new state *instantiated*. A notification is sent to the Experiment Lifecycle Manager (message 6), where the internal status of the experiment is updated to *configuring*. This event triggers automatically the beginning of the experiment execution phase, described in the next section.

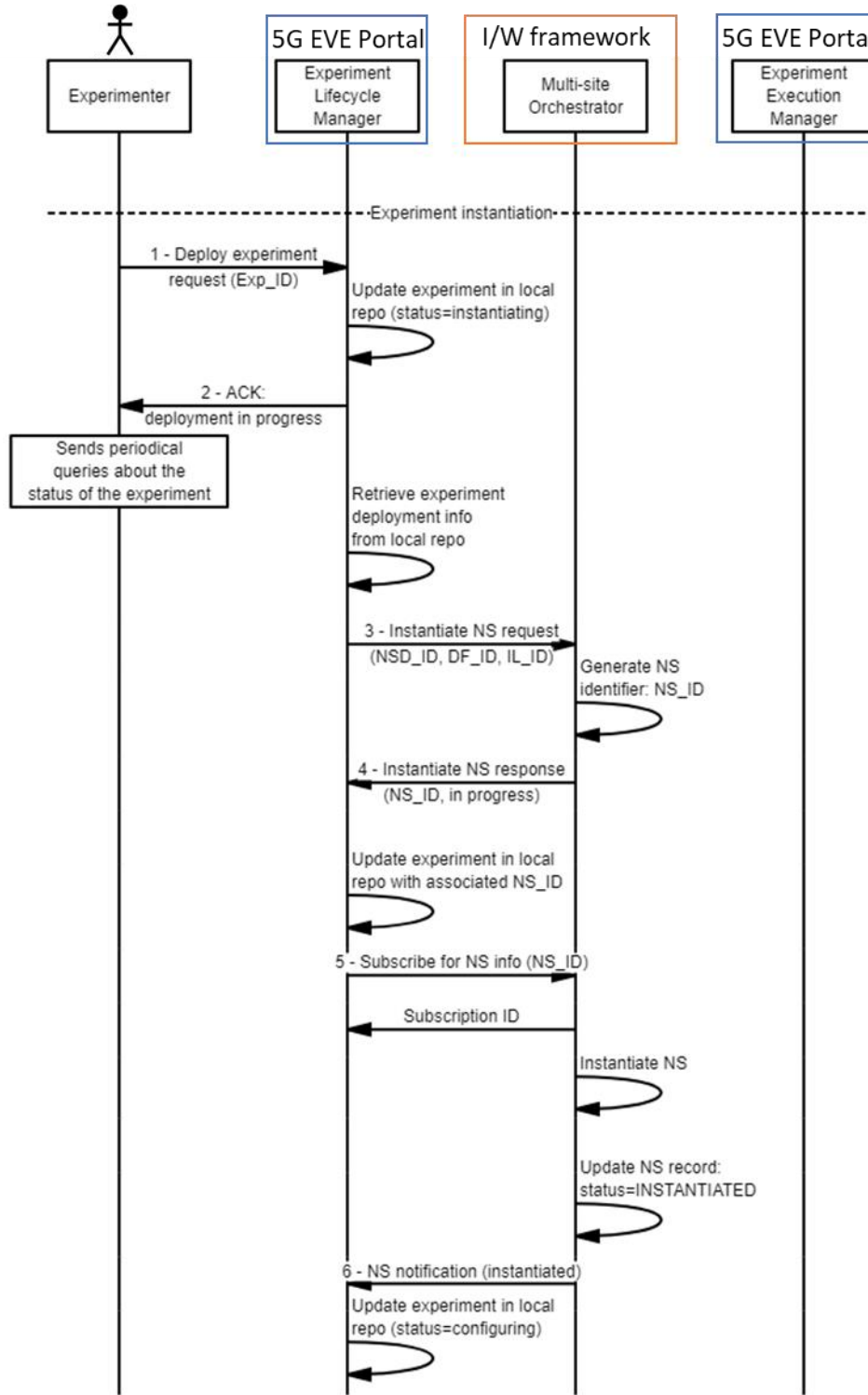


Figure 18: Experiment instantiation

3.2.3.2 Experiment execution management

The experiment execution, shown in Figure 19, is triggered by the Experiment Lifecycle Manager as soon as the virtual environment of the experiment is properly instantiated. The procedure is triggered with an execute experiment request (message 1) sent to the Experiment Execution Manager at the I/W framework. The request includes the identifier of the experiment, its ExpD and the information associated to the deployed NFV network service. The Experiment Execution Manager creates internally a new entity “experiment execution”, used to keep trace of the actual evolution of the experiment during its runtime and returns its identifier to the Experiment Lifecycle Manager (message 2). Such identifier will be used by the Experiment Lifecycle Manager to create a subscription with the Experiment Execution Manager for receiving notifications about the evolution of the running experiment (messages 3 and 4).

The first step of the experiment execution involves the configuration of the VNFs involved in the experiment. This action is managed by the Experiment Execution Manager, through the interaction with other I/W framework components (see D3.3 [7] for the description of the interaction between Experiment Execution Manager and Runtime Configurator at the I/W framework for VNF configuration). Once the VNFs are properly configured, the “experiment execution” status is updated to *running* and the system starts to execute the test cases and collect the related monitoring data. The details of this procedure are provided in WP5 deliverables. As soon as the experiment execution status changes, this is notified to the Experiment Lifecycle Manager (messages 5), which updates its internal information, so that the experimenter can be informed about the whole evolution. Further details, like the percentage of the executed tests or the temporary results, may also be provided based queries to the Experiment Execution Manager and made available for the experimenter through the Experiment Lifecycle Manager at the 5G EVE Portal.

When the experiment execution is completed, the related status at the Experiment Execution Manager is updated accordingly and notified as terminated to the Experiment Lifecycle Manager (message 6). As soon as the execution is terminated, also the virtual environment associated to the experiment can be terminated, to release the resources at the site facility. Thus, the Experiment Lifecycle Manager sends a message to the MSO for requesting the termination of the NFV network service (message 7). The network service is then terminated (message 8) and the status of the experiment is again updated at the Experiment Lifecycle Manager.

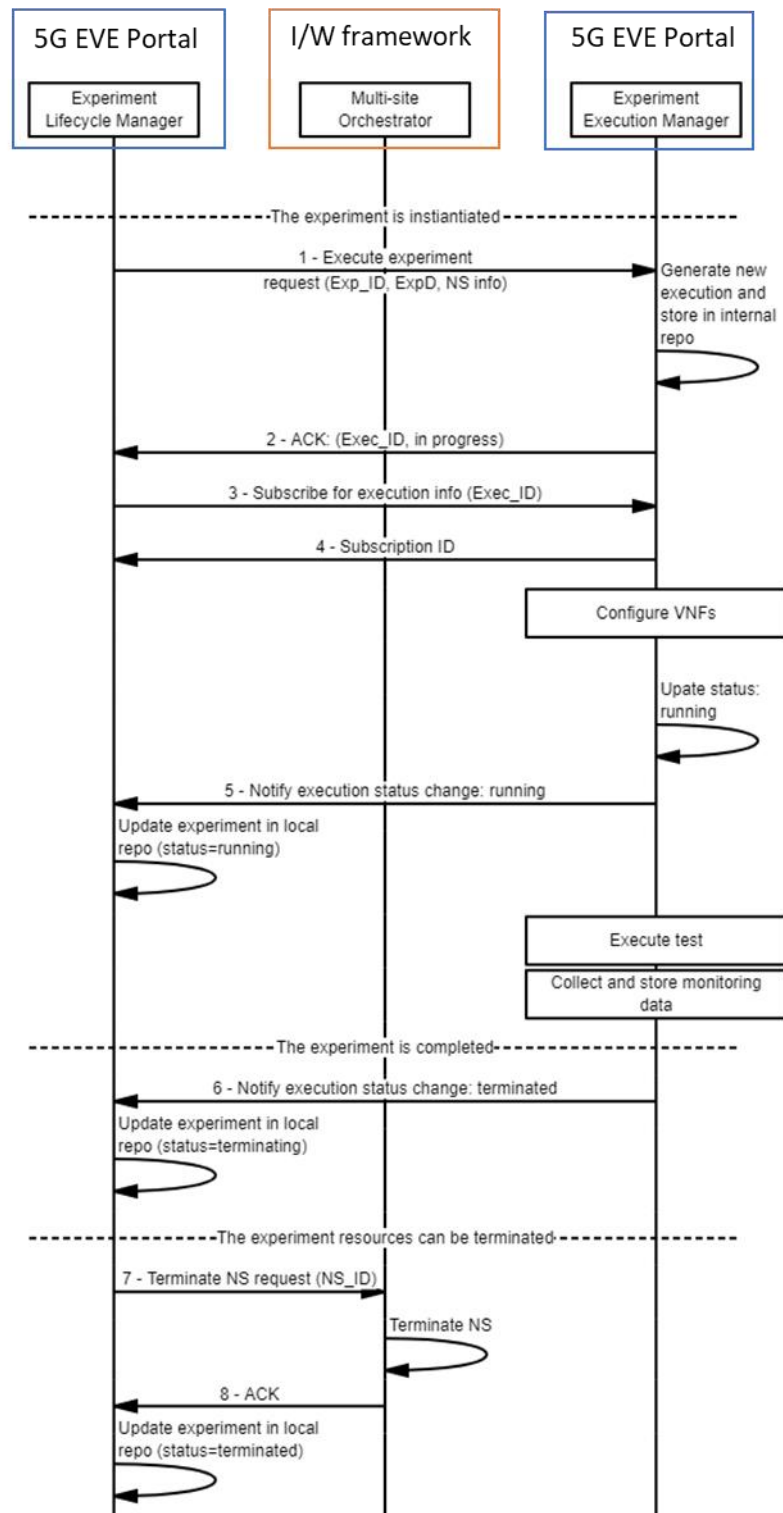


Figure 19: Experiment execution

3.2.3.3 Experiment monitoring and performance analysis

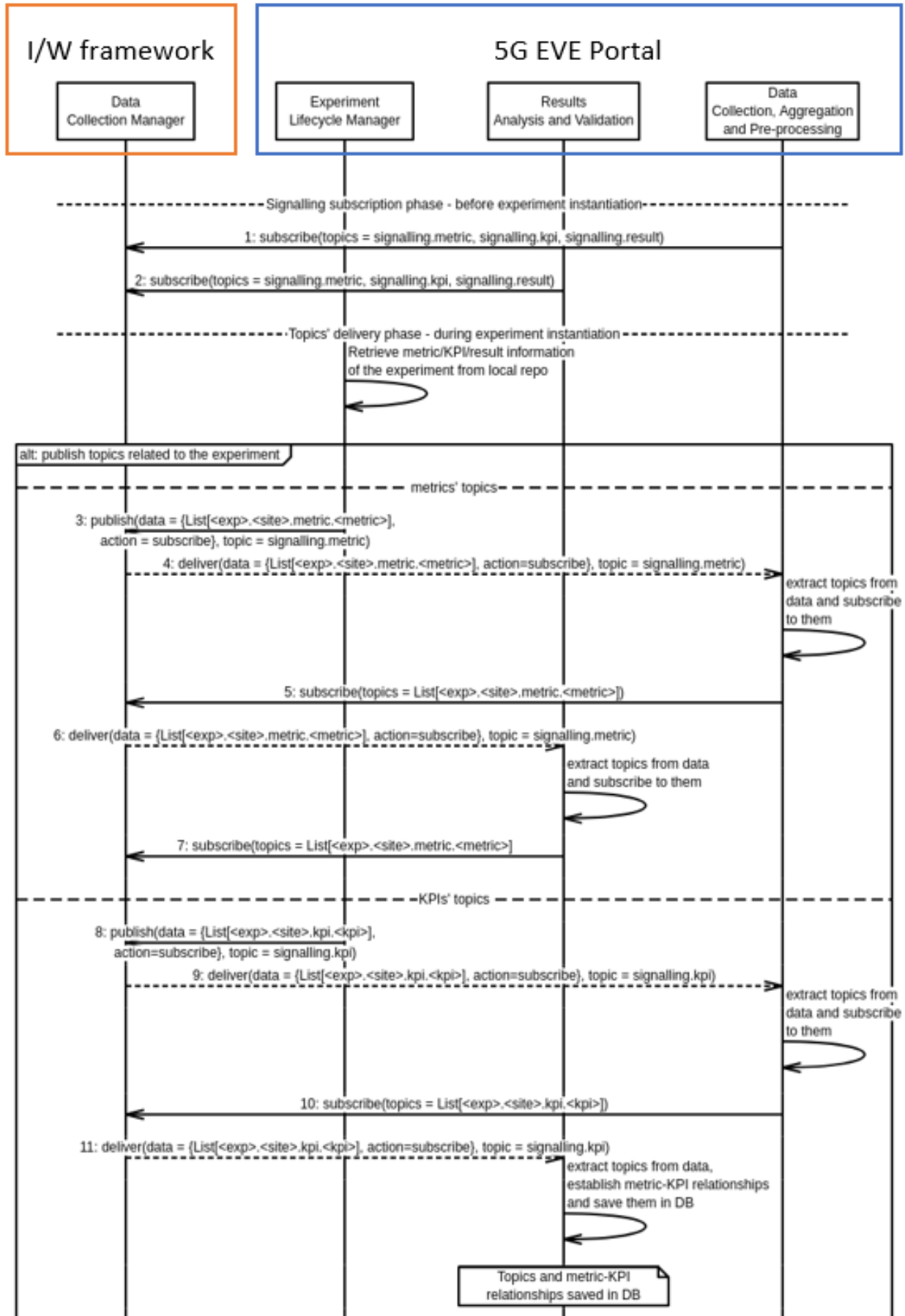
Regarding the experiment monitoring and performance analysis workflow, we can distinguish between three different phases, which are executed in specific stages of the experiment workflow.

First, there is a **subscription phase**, where the components that perform functions related to monitoring and performance analysis are subscribed to the corresponding topics related to the experiment, depending on the specific experiment that is going to be executed and the metrics/KPIs/results to be monitored. The workflow for

this phase is depicted in the Figure 20. At the beginning both “*Data Collection, Aggregation and Pre-processing*” and “*Results Analysis and Validation*” components in the Portal subscribe with the “*Data Collection Manager*” at the I/W framework (messages 1 and 2). These subscriptions define the signaling topics that will be used to announce the topics to be (un)subscribed for a given experiment, being these new topics the ones that will be used in the delivery of the monitoring data for a given experiment. Monitoring data include metrics (data extracted from the monitored components in a given format), KPIs (formula to be applied for a set of metrics) and results (specific values related to the experiment, calculated from the metrics, which are interesting for experiment monitoring). This subscription step is executed before the experiment instantiation, because it will enable the exchange of topics to be (un)subscribed in future stages.

When the experiment is being instantiated, the Experiment Lifecycle Manager will obtain the metrics, KPIs and results that are expected to be monitored from its local repository, where it has all the information related to the experiment specification. That information will be published in the Data Collection Manager in a string-list format, where each string will be a unique identifier for the metric/KPI/result to be monitored. The string format is the following: `<exp>.<site>.[metric/kpi/result].<value>`, where `<exp>` is the experiment ID, `<site>` identifies the site facility where the data is extracted, `[metric/kpi/result]` indicates the type of information to which belongs the next parameter, and `<value>` set the name of the parameter to be monitored. Moreover, there is another parameter that is included in the data: `action = subscribe`, because the flow corresponds to the subscription phase (there is an un-subscription phase which uses the same signaling topics but doing different actions, so it is necessary to distinguish both phases with this parameter).

Even if the publication action performed by the Experiment Lifecycle Manager is similar for the three possible kinds of values to be monitored (messages 3, 8 and 12), the workflow for each case is slightly different. For metrics, both Data Collection, Aggregation and Pre-processing and Results Analysis and Validation components are finally subscribed to the topics provided by the Experiment Lifecycle Manager (messages 5 and 7 respectively). This action is repeated in the case of KPIs and results only for the Data Collection, Aggregation and Pre-processing component (messages 10 and 14 respectively). However, the Results Analysis and Validation component does not subscribe to the KPIs’ and results’ topics, but it establishes the relationships between metrics-KPIs and metrics-KPIs-results according to the information model defined in both blueprints and descriptors, in order to use these relationships afterwards, during the experiment execution, for calculating the KPI and result values from the metric values. This last process will be defined in WP5 context.



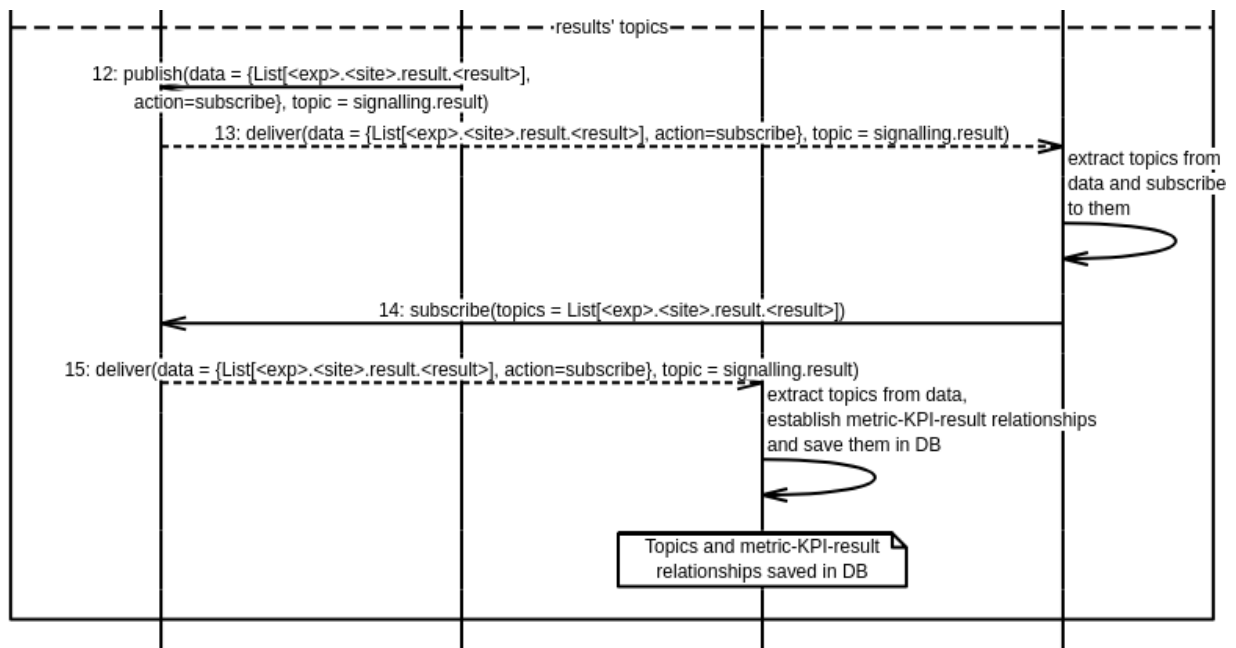


Figure 20: Subscription to the topics used for experiment monitoring and performance analysis purposes.

At this point, the components related to the experiment monitoring and performance analysis already know the topics which will be used during the experiment execution. In that phase, while the experiment is being executed, it will start the **monitoring and data collection phase**, presented in the Figure 21. Here the metrics that have been published by the distributed data shippers (with a previous log-to-metric transformation done by these components) in the Data Collection Manager (message 2) are delivered to the two components that are subscribed to the topics which correspond to each metric (message 3 and 4 respectively). In the case of the Data Collection, Aggregation and Pre-processing component, it will collect the metrics information and display it, a function that is carried out by the toolchain provided by the Experiment monitoring and Maintenance & Results Collection tool, described in the section 4.1.2.

Moreover, the Results Analysis and Validation component will process the metrics information and calculate the corresponding KPIs and results associated to these metrics, according to the relationships established in the signalling phase. This process will be defined in WP5 context. The information related to these KPIs and results will be published in the Data Collection Manager, using again its corresponding topics (messages 5 and 7). This information will be delivered to the Data Collection, Aggregation and Pre-processing component (messages 6 and 8), which again will collect all the information and display them.

This process is repeated for each metric defined in the experiment, which will be gathered each N seconds by the data shippers until the end of the experiment.

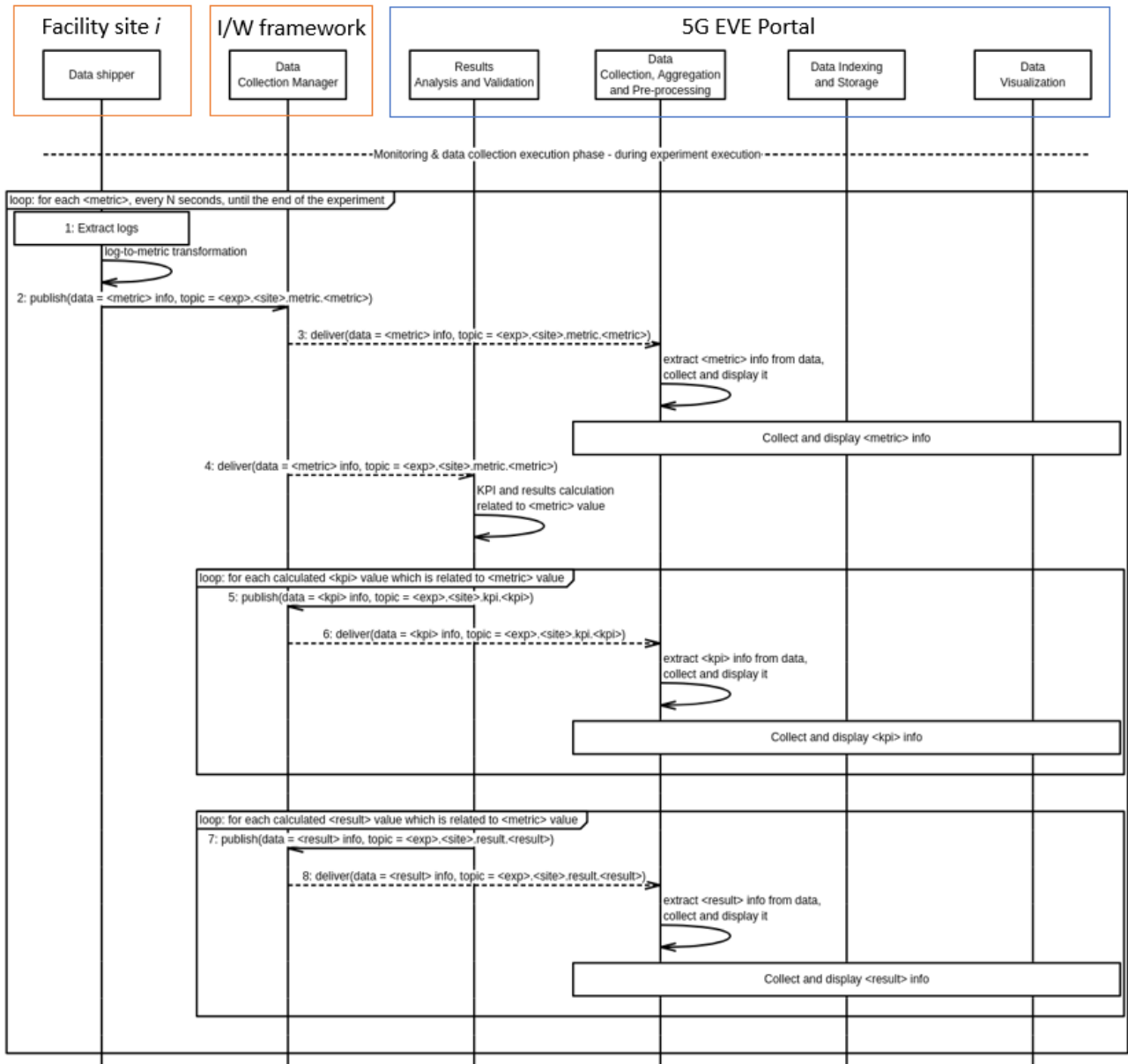
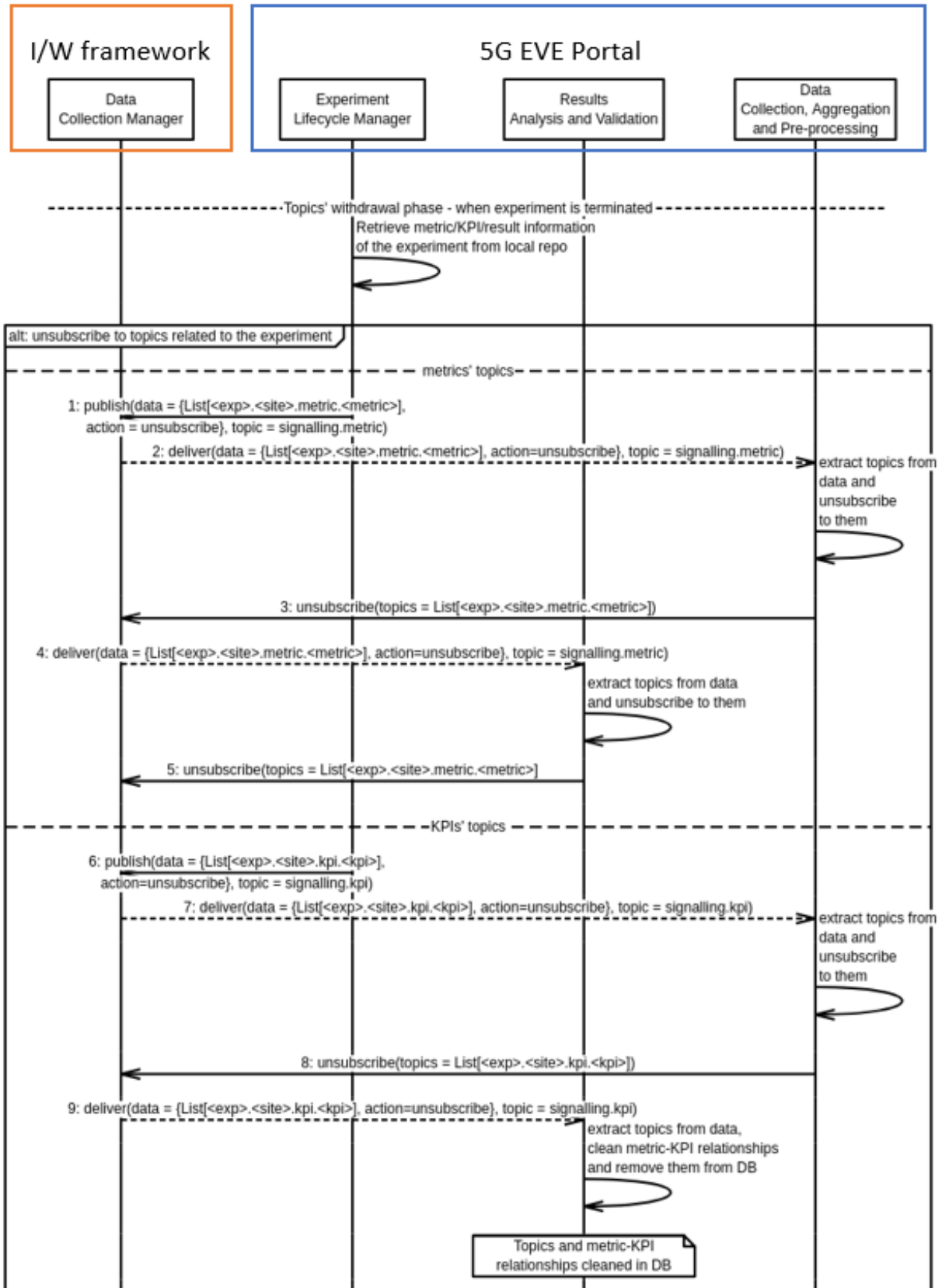


Figure 21: Delivery and management of monitoring information during the experiment execution.

Eventually, when the experiment is finished, a cleaning process is triggered. This **withdrawal phase**, showed in the Figure 22, is similar to the subscription phase in terms of workflow, but it performs the opposite operations.

In this case, the Experiment Lifecycle Manager retrieves again the metrics, KPIs and results that have been monitored during the experiment and publishes in the Data Collection Manager its corresponding topics identifiers, followed by the parameter *action = unsubscribe* (message 1). As a result, this will cause the withdrawal of the metric/KPI/result topics for the Data Collection, Aggregation and Pre-processing component (messages 2-3, 7-8 and 11-12 respectively), and the withdrawal of the metric topics for the Results Analysis and Validation component (messages 4-5). Moreover, the metric-KPI and metric-KPI-result relationships established in the signaling phase within the Results Analysis and Validation component will be cleaned (messages 9 and 13). In this way, it assures the returning to the initial state to all the components involved in this experiment monitoring and performance analysis workflow.



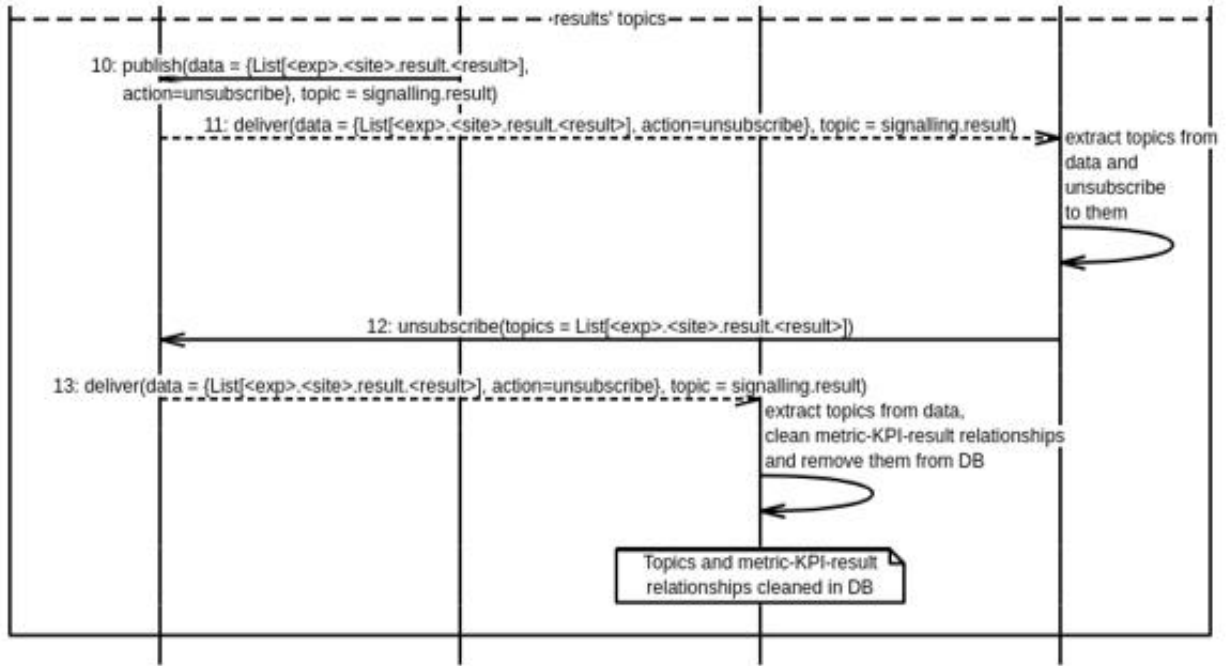


Figure 22: Withdrawal of the topics used for experiment monitoring and performance analysis purposes

4 Experimentation tools, including intent-based interfaces

This section describes the different experimentation tools that are available to the different 5G actors from the 5G Portal during the experiment lifecycle, the reason they are necessary, the architecture and the implementation of each one, and their interaction with other tools or layers.

4.1 Experiment monitoring and Maintenance & Results Collection

4.1.1 Purpose/Description of the tool

To provide experiment monitoring and maintenance and results collection functionalities, a toolset has been selected among the most popular, documented, supported and customizable tools available on the market (see Annex C). This toolset oversees collecting, maintaining and monitoring the experiment data. The experiment results are collected using well known data collection tools. The experiment data maintenance is achieved using data aggregation, processing, indexing and storage tools. The experiment data monitoring focuses on the representation of related metrics and KPIs, provided by proper visualization tools.

4.1.2 Tool Design / Architecture

In order to cope with the features described in the previous section, the following toolchain has been identified:

1. A **data collection, aggregation and pre-processing tool** to extract the experiment results, metrics and KPIs generated for a given experiment from a specific Publish-subscribe queue which is directly connected to the Inter-working Publish-subscribe queue provided by the Data Collection Manager component at the I/W framework.
2. A **data indexing and storage tool**, which can search and filter among all the data gathered by the data collection tool for obtaining the useful information to be displayed to the experimenters, according to what results they expect to visualize during and after the experiment execution.
3. A **data visualization tool** in charge of presenting those experiment results, metrics and KPIs with an intuitive GUI, being able to monitor the progress of the experiment in terms of that information displayed and allowing verticals to interact partially with the visualization tool in an online fashion (e.g. by choosing what kind of information they want to see in any moment, setting thresholds for some parameters...)

This toolchain is depicted in Figure 23, which presents an improved version of the Data Collection Manager architecture diagram included in deliverable D3.2 [6] – figure 18 with new considerations regarding the collection and monitoring framework to be implemented in the project.

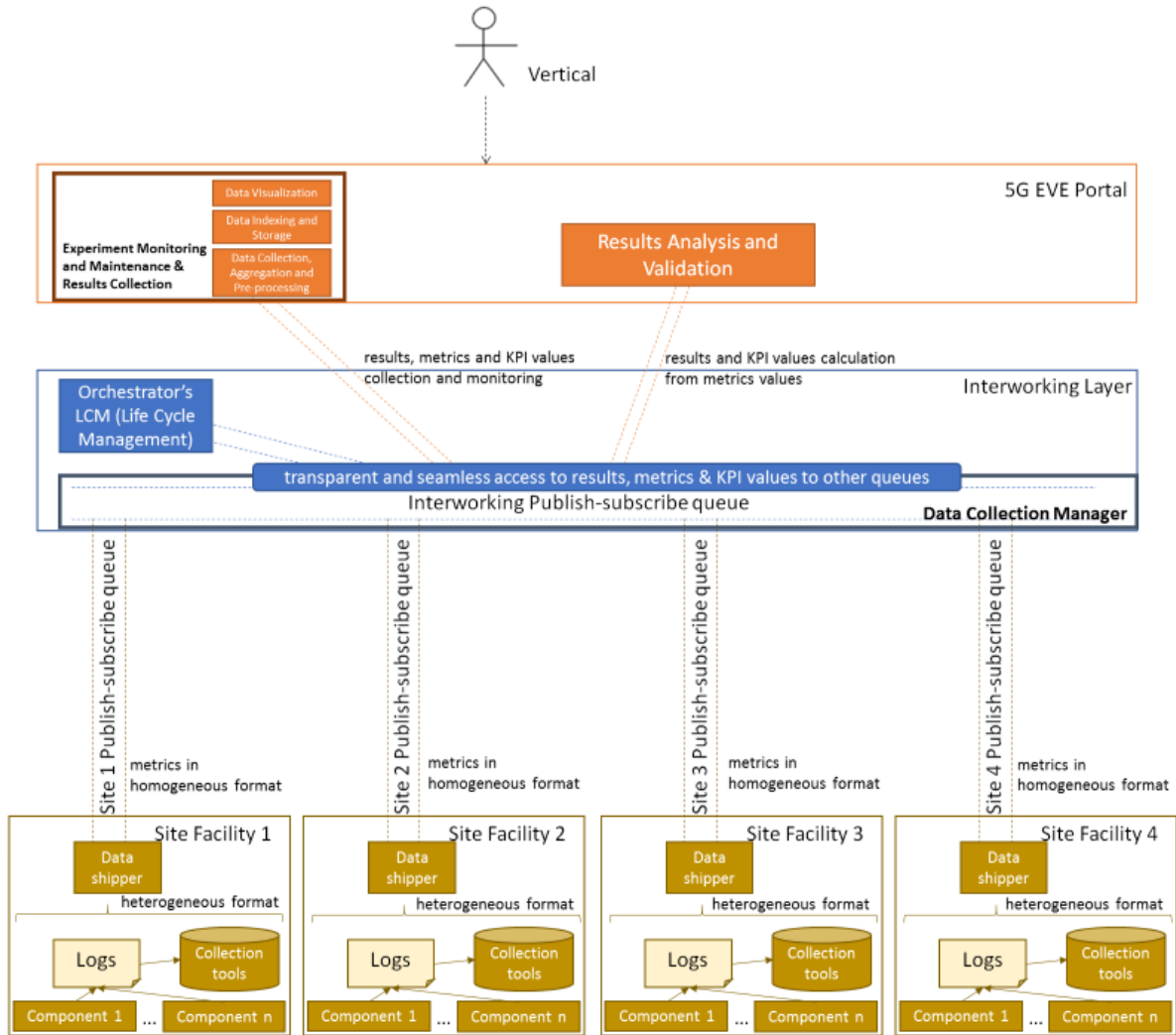


Figure 23: Improved version of the Data Collection Manager architecture with the Experiment Monitoring and Maintenance & Results Collection toolchain.

Regarding the changes in the Data Collection Manager functionality, they will be explained in detail in the deliverable D3.3 [7]. In the case of the Experiment monitoring and Maintenance & Results Collection, as it was mentioned before, it will receive the results, metrics and KPIs from the Data Collection Manager by using a publish-subscribe queue connected to that component. In fact, it will be subscribed to the topics related to these results, metrics and KPIs, which are published by different components in the architecture:

- In the case of metrics (defined in Vertical Service, Context and Experiment Blueprints), they are delivered by the data shippers¹¹ installed in the different components which belong to a given experiment and are deployed in the site facilities. These data shippers implement the log-to-metric transformation function (as explained in deliverable D3.2 [6]).
- On the other hand, KPIs (defined in Experiment Blueprints) and results are provided by the Results Analysis and Validation component, developed in the context of WP5, which calculates these results after receiving the metrics values thanks to the subscription to the related topics

¹¹ Components in charge of obtaining the metrics from the VNFs/PNFs and resources that are used in the experiment (e.g. Beats, which is a specific solution from Elastic). Experiment developers are responsible for the definition of data shippers for the metrics defined in the blueprints. During the experiment instantiation, data shippers are loaded or configured via the Runtime Configurator (in case this Day-2 configuration is related to metrics/KPIs/results that have been described in the blueprints and is considered in the test description) at the I/W framework.

This behaviour is aligned with the experiment monitoring and performance analysis workflows already showed in section 3.2.3.3.

Moreover, there can be other components that may feed the publish-subscribe queue with new data to be collected and monitored; for example, the Multi-site Orchestrator placed in the I/W framework, which could provide new metrics with a connection from the Orchestrator's Lifecycle Manager to the Data Collection Manager.

4.1.3 Tool Implementation

After studying many monitoring and data collection tools as candidates to be used in the project, which can be reviewed in the Annex C, the baseline toolchain selected for the implementation of the Experiment Monitoring and Maintenance & Results Collection is the **Elastic (ELK) Stack**¹². It is composed by three main open-source products developed, managed and maintained by Elastic¹³, which perform the three main functions defined for the toolchain:

- **Logstash** for data collection, aggregation and pre-processing. It is a server-side data processing pipeline that collects data from multiple input sources simultaneously, executes different transformations and enhancements and then ships the data to various supported output destinations; in our case, Elasticsearch.
- **Elasticsearch** for data indexing and storage. It is an open source, distributed, RESTful, full-text and JSON-based search and analysis engine, based on the Apache Lucene search engine, which is easy to use, scalable and flexible.
- **Kibana** for data visualization. It is a visualization layer that works on top of Elasticsearch, providing users with the ability to analyse and visualize the data with charts and graphs.

The Elastic Stack can be installed using a variety of methods and on a wide array of different operating systems and environments. In fact, it can be installed locally, on the cloud, using Docker and configuration management systems like Ansible, Puppet, and Chef. The environment chosen for the installation of the Elastic Stack in the 5G EVE project will be described in future deliverables related to implementation issues.

For connecting the Elastic Stack to the Data Collection Manager placed in the I/W framework, the Logstash component will be connected to an **Apache Kafka broker**¹⁴, a technology which implements the publish-subscribe queue used in the Data Collection Manager and that will deliver the monitored information to the Elastic Stack following that paradigm. In the same way, the data shippers must publish the metrics into another Kafka broker to be delivered. These data shippers could be implemented with different technologies, if they are able to publish the information into Kafka. In the case of the Elastic Stack, they provide a specific tool for extracting data from the monitored components, which is called **Beats**. This product is a lightweight agent that is installed in these monitored components for collecting different types of data and forwarding them into the Elastic Stack through the Kafka bus. The overall toolchain, according to the selected implementation tools, is represented in Figure 24.

¹² <https://www.elastic.co/what-is/elk-stack>

¹³ <https://logz.io/learn/complete-guide-elk-stack/#intro>

¹⁴ <https://kafka.apache.org/>

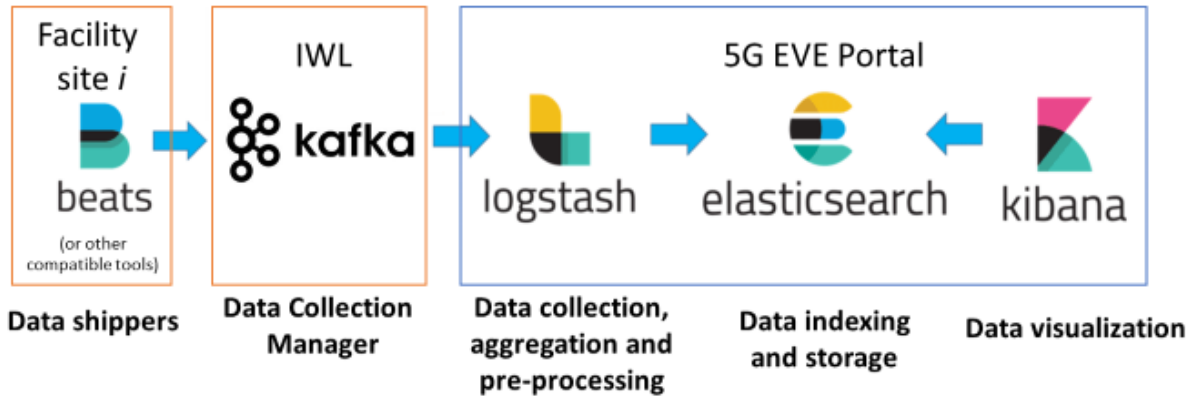


Figure 24: The Elastic Stack toolchain with Beats and Kafka integration.

A “dockerized” environment for testing the 5G EVE Monitoring & Data Collection tools is available in the following repository: <https://github.com/5GEVE/5geve-wp4-monitoring-dockerized-env>

Figure 25 represents an example of function distribution for the monitoring system, in a multi-site scenario with multiple generic VNFs and PNFs and a mobile network infrastructure.

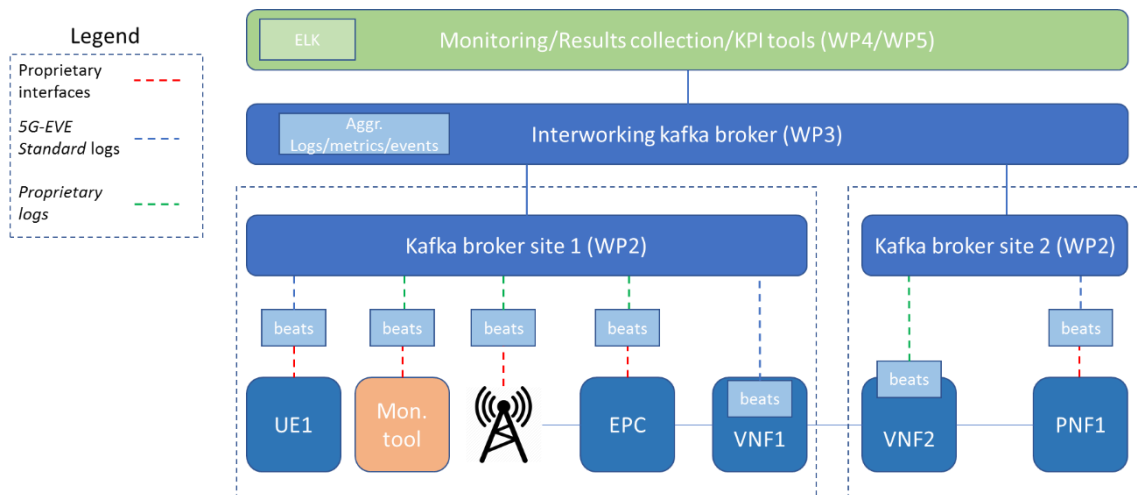


Figure 25: Function distribution for the Beats + Kafka + Elastic Stack architecture.

In the diagram, we can see that there are different components in each site that could be monitored (user equipment, base stations, the EPC, the VNFs deployed...), and the format of their logs may be different. The role of transforming these logs in unified metrics is carried out by the Beats components, which gathers data from the aforementioned components and send them to Kafka. After that, the workflow presented in section 3.2.3.3 is followed.

Note that this way of obtaining data from components is transparent for verticals, as most modules that play that role (gather logs/events for saving them in databases for a future use in a specific monitoring stack) have the option of publishing that data in Kafka. And for proprietary cases (such as the EPC or base stations), it would be possible to use other traditional alternatives like SNMP queries. In that case, a SNMP collector would be deployed for obtaining the data from legacy components, and another Beats component would extract that data from the SNMP collector to publish it into Kafka.

4.1.3.1 Example of tool implementation for the 5G-EVE transport vertical

In this section we report an example of tool implementation for the 5G-EVE transport vertical, with a specific focus on the “Urban mobility 5G data flows analysis” use case described in section 3.1 of deliverable D1.1 [8], and section 3.1.1 in deliverable D2.1 [9]. The main objective of the use case is the integration of 5G data and

mobility data from different transport operators to enhance sustainable multi-modality between railway network and other urban transportation services.

To collect mobility data, an Android app has been developed ad-hoc to record information from smartphone sensors (e.g., GPS, accelerometer, magnetometer and gyroscope). Running on the user’s phone, the app, through a simple graphical interface, permits users to start a new session, pause and resume the data collection, and label the transportation mode being performed, choosing between: still, walking, running, motorcycle, biking, car, bus, metro, tram and train. To analyse and represent the collected data, the whole aforementioned ELK stack has been implemented. Using Kibana, a customizable and interactive dashboard has been created, on which statistics and graphs are displayed on the different transportation modes. The main information displayed concerns features about the speed detected by GPS, the magnitude of the sensors and network data. Kibana allows to filter records by time intervals and also to filter data based on customizable parameters. Figure 26 and Figure 27 below show some examples of graphs built with Kibana visualization tool.

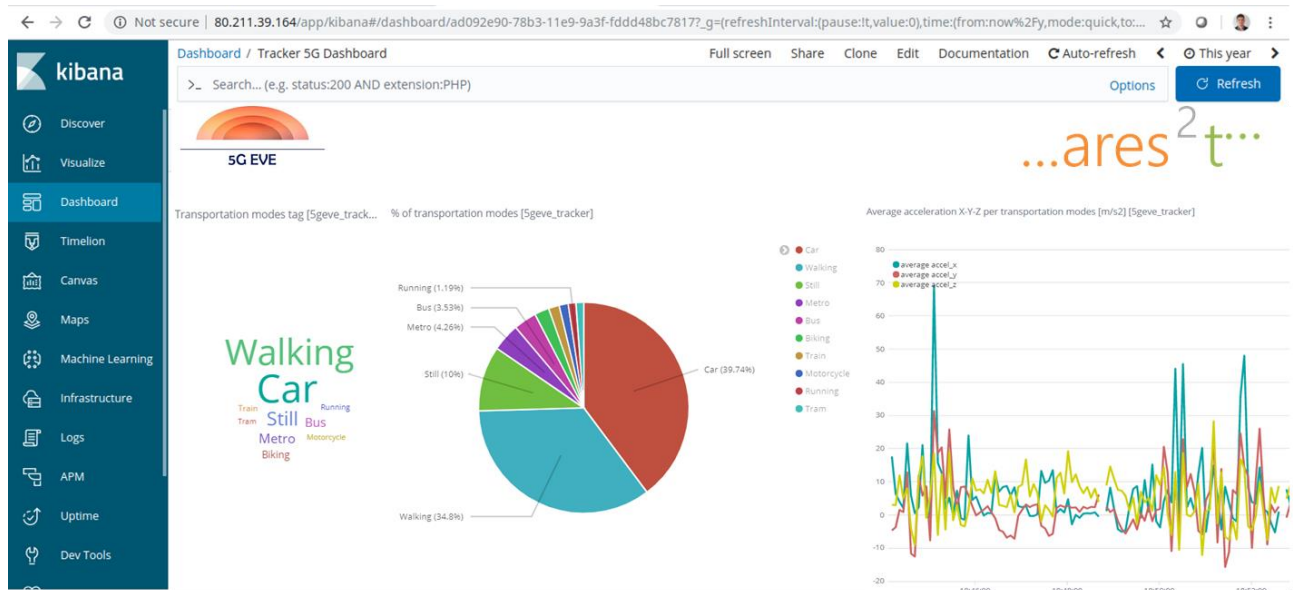


Figure 26: Urban mobility 5G data flows analysis use case, visualization tool graphs

Figure 26 reports, on the left, a words cloud showing the names of the transportation modes in various sizes, based on the number of records collected for each of them. In the middle, a classic pie chart is shown, in which each slice represents the percentage of records collected for each transportation mode. On the right, a line graph shows the trend of the three linear accelerometer axes (X, Y and Z) recorded during a 30 minutes session characterized by walking, running and still mobility modes.

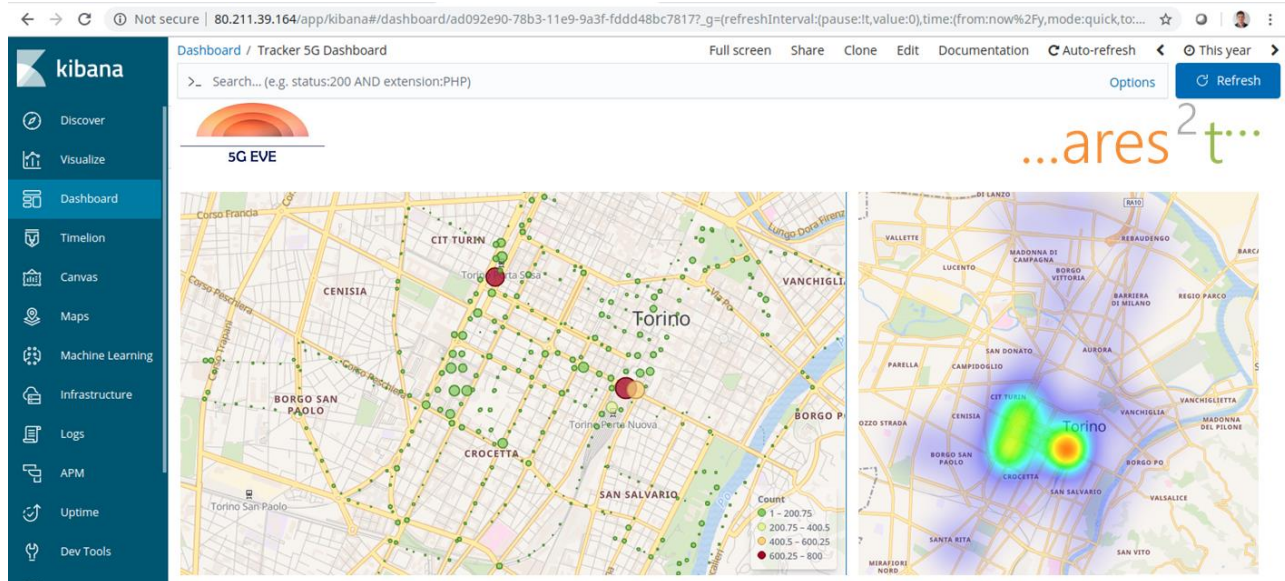


Figure 27: Urban mobility 5G data flows analysis use case, geo points graphs

Figure 27 shows two examples of geo points graphs. In Kibana it is possible to build fields of type `geo_point` that accept latitude-longitude pairs, provided in our case by 5G EVE Tracker app through the GPS information. Geo points can be used to build a coordinate map visualization. It displays a metric of a certain variable distributed on a geographic area. In our dashboard, geo-points have been represented with circles in different scales, as in the example on the left, that shows the number of records collected in Turin; other possibility is to represent data with heatmap points, as shown in the right example, which shows the speed detected by GPS. Kibana Geo Points graphs offer the possibility to the user to choose the area of interest and to fit data bounds.

The installation guide and the configuration files to reproduce the demo of the 5G EVE monitoring and visualization tools applied to the transport vertical use case are available at the following repository: <https://github.com/5GEVE/5geve-wp4-visualization-tool-transport-vertical>

4.1.4 Interaction with other tools/layers

As it has been described previously, the Experiment Monitoring and Maintenance & Results Collection tool-chain directly interacts with the Data Collection Manager (Kafka bus) from the I/W framework. In this way, the data in the Kafka bus is delivered by using Kafka topics, which group messages with a common format.

In this case, each metric, KPI and result will have a specific topic assigned, and the different components in the architecture only must publish/subscribe to the correct topics. For achieving this, it is necessary that the Experiment Monitoring and Maintenance & Results Collection tool knows in advance the topics to which it has to be subscribed, which are obtained automatically during the experiment definition phase, as depicted in section 3.2.3.3, by the Experiment Life Cycle Manager component through signalling topics. The different topics available in the 5G EVE architecture are described in the Topic framework, which is introduced in the deliverable D3.3 [7].

As a result, the Experiment Monitoring and Maintenance & Results Collection toolchain must be able to handle two different types of information exchanged through the Kafka bus, and the way of dealing with each case is slightly different:

- **Information provided by signaling topics:** this information is present during the subscription and withdrawal phases, and it is composed by a list of the corresponding metrics/KPIs/results topics ID to be subscribed for obtaining the monitored data. In this way, it is necessary to have a specific component within the Experiment Monitoring and Maintenance & Results Collection toolchain that performs the operations of: subscription to signaling topics, reception, processing and extraction of data published by the Data Collection Manager in these signaling topics (the list of the metrics/KPIs/results topics ID), and configuration of Logstash component in order to (un)subscribe it to these metrics/KPIs/results topics. The description and implementation of this component will be described in future WP4 deliverables

regarding the portal implementation, but the unique requirement for it is to be able to exchange information with Kafka (e.g. a simple Python program could achieve this).

- **Information provided by metrics/KPIs/results topics:** this information is present during the monitoring and data collection phase. The component from the toolchain that is responsible for managing the exchange of this information is Logstash, being configured during the (un)subscription phases as mentioned before. That information comes from different sources:
 - **Metrics** are published by data shippers (Beats or other components compatible with Kafka connection), which would parse the logs produced by the monitored components, apply a regex to that stream, produce data in a specific format (e.g. in JSON, according to a specific structure) and publish it in a specific Kafka topic.
 - **KPIs and results** are published by the Results Analysis and Validation component by transforming the metrics obtained from data shippers accordingly.

Note that the most useful information for verticals to be monitored and displayed are both KPIs and results from the experiment, but it is also delivered the metrics values for verticals and users that are interested in checking what values are obtained from a test execution.

This tool also interacts with the 5G EVE Portal GUI, as Kibana provides a specific GUI for data visualization. For that reason, the 5G EVE Portal must provide an access to the Kibana URL for allowing verticals to check their experiments' monitoring through the Kibana GUI., also allowing them to configure the dashboards and graphs they need from the data collected.

4.2 Ticketing System

4.2.1 Purpose/Description of the tool

This tool will be used for communication among different actors, mainly experimenters and the portal with site managers and system administrators. The main communications will be:

- Automatically notify errors/warnings in the normal operation of the 5G EVE Portal, the experimentation tools, the VNF repository or any other component of the Portal. This functionality can be extended to include other components of the 5G EVE like components of the I/W Framework and even the components of all trial sites.
- Actors like experimenters, experiment developers, VNF providers, etc. can manually create, modify or delete tickets to notify issues detected in the Portal.
- This tool will be used by the scheduling component when an experimenter creates a deployment object used to schedule an experiment. Experimenters will include information of the experiment like the ExpID, the site(s) to deploy (parts of) the experiment and the tentative days to execute the experiment. This action will issue a ticket to each of the site managers involved in the experiment will all the information necessary, so site managers can check the resources required by the experiment and other required parameters. Site managers can modify the tickets as they progress in the preparation of the experiment.
- This tool will be used by the VNF uploading component too, to issue a ticket after a VNF developer uploads a VNF, so the selected sites are notified that a VNF was uploaded, so them can check the VNF provided. Then, site managers can decide to on-board the VNF or not, modifying the status of the ticket correspondingly.

4.2.2 Tool Design / Architecture

Following the requirements defined in the previous section, the project has analysed the possibility to use completely or partially some existing tools to provide the aforementioned services. Table 5 presents the main results of the analysis of three of these tools: Mantis Bug Trackers, Bugzilla and Request Tracker. The main aspects analysed were focused on the dependencies of these tools with other components, the license and the availability of a REST API to access the services provided. The result of this analysis did not provide a strong argument to select one tool among the others, so the decision to use Bugzilla was motivated due to the good support and large community behind this solution.

Table 5: Tools analysed for the ticketing system

Name (webpage)	Dependencies	License	REST API
MANTIS Bug Tracker (http://www.mantisbt.org/)	PHP, ADOdb (MySQL, PostgreSQL, SQL Server, etc.)	Free, open-source	Yes
Bugzilla (https://www.bugzilla.org/)	MySQL and PostgreSQL recommended, perl, web server.	Free, open-source	Yes
Request Tracker (https://en.wikipedia.org/wiki/Request_Tracker)	Perl, MySQL, PostgreSQL, Oracle, SQLite	Open source (FOSS) and distributed under the GNU General Public License	Yes

Although the main goal of these tools, and Bugzilla in particular, is to keep track of the bugs detected in software (see Figure 28) these tools can be used to track any other events, like troubles or configuration requests in other environments. Bugzilla was designed to allow the modification of almost all states declared by default in the system, so it is flexible enough to modify its structure to adapt and integrate it with other tools.

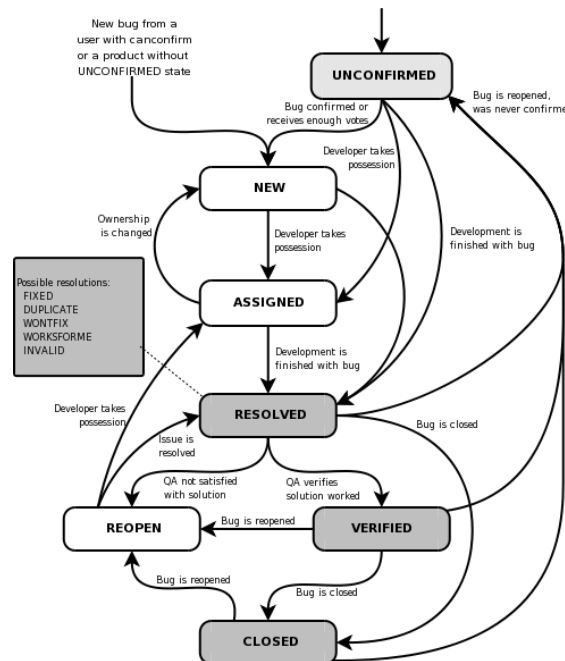


Figure 28: State of bugs in Bugzilla

4.2.3 Tool Implementation

Bugzilla is implemented in Perl (version 5.12 or higher recommended) while the supported databases are MySQL, PostgreSQL, Oracle and SQLite, although the first two are highly recommended. Thus, the 5G EVE project does not have to implement this tool but to use the services provided to integrate it with other tools.

4.2.4 Interaction with other tools/layers

As explained in previous sections, the ticketing system provides interfaces to other tools/components included in the Portal to request the creation and modification of tickets. Our plan is to use the REST API provided by Bugzilla to perform this integration. In order to test this possibility, and because most of the components of the

5G EVE Portal were not available during the first year of the project, we have decided to test this integration with an existing Portal provided by the 5GinFIRE project. This project has some similarities with the 5G EVE project, and they have performed some minimal integration with Bugzilla already.

In the 5GinFIRE project the Portal is available using the URL <https://portal.5ginfire.eu/#/> while the Bugzilla is decoupled using a different URL <https://portal.5ginfire.eu/bugzilla/>. Our plan in 5G EVE is to have a unified Portal where all services are offered through the same graphical interface, so our users do not need to have different accounts and different views depending on the service they want to access.

Based on these requirements, we have improved the integration of the 5GinFIRE Portal with Bugzilla, following the templates of the former and using the REST API of the latter.

When a user signs up in the portal, this information is also shared with Bugzilla to add the user to the database of the latter. This way, it is possible to use the same credentials in both sides, reducing the complexity of the integration.

After this step, the user has a new option available in the Portal to access the ticketing system. Figure 29 shows the integrated Portal of 5GinFIRE with the new service available in the top menu, highlighted in red in the Figure. This figure shows the interface to create a new ticket, where the user has to select a topic (product, following the Bugzilla terminology), a subtopic, a name for the ticket, the severity, the summary and a short description of the ticket.

It is important to highlight that this is just an example to integrate Bugzilla with an existing Portal, but the actual integration with the 5G EVE Portal, as well as the definitive fields and options in Bugzilla will be done in T4.2, and the first integrated version will be provided in D4.2, which will be delivered in December 2019. Anyway, the extension of the 5GinFIRE Portal to improve the integration with Bugzilla can be found under the following link: <http://vm-images.netcom.it.uc3m.es/5GEVE>

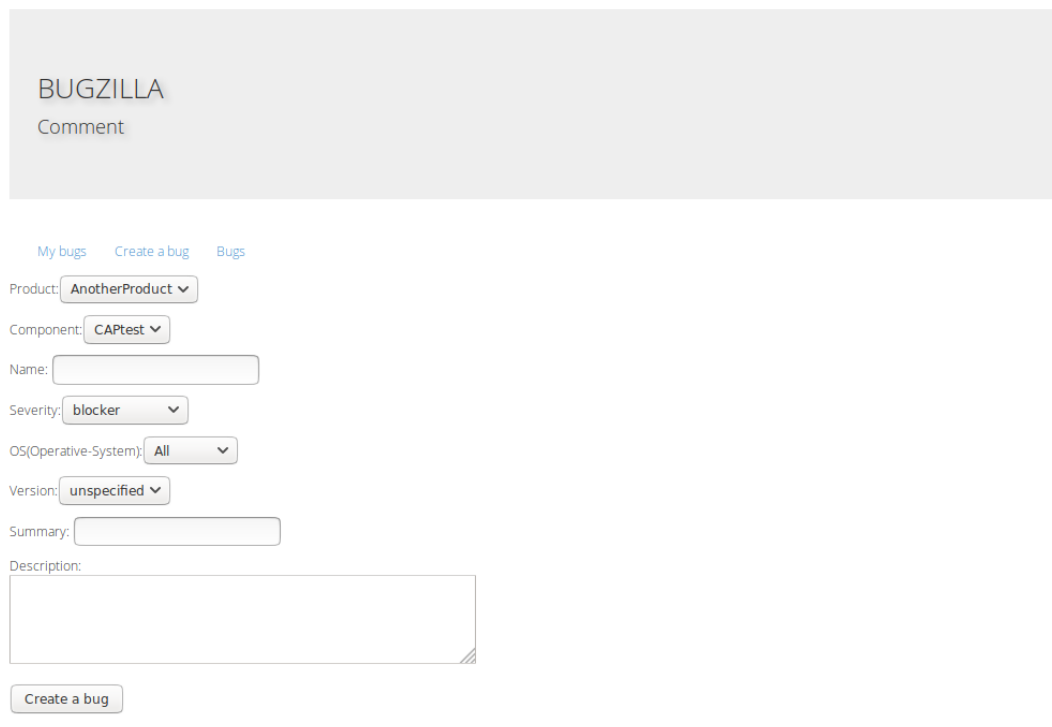


Figure 29: Option to create a new ticket in Bugzilla

4.3 Browse and look-up

4.3.1 Purpose/Description of the tool

The “browse and look-up” tool provides a graphical interface to the 5G EVE catalogues allowing the user to visualize the blueprints and descriptors for vertical services, contexts, experiments, NFV network services, VNFs and PNFs. For each element, the tool shows the most relevant information and provides a graphical representation of the given item. For example, the blueprints are represented as graphs that interconnect their atomic components according to the specified connectivity; NFV network services are shown as graphs with the VNFs interconnected through virtual links, etc. Figure 30 and Figure 31 show examples of NSD and VNFD representations.

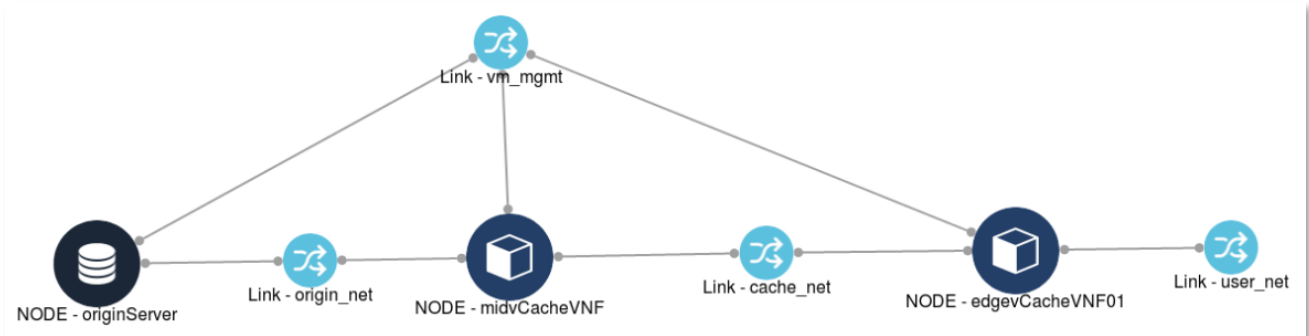


Figure 30: Graphical representation of an NSD

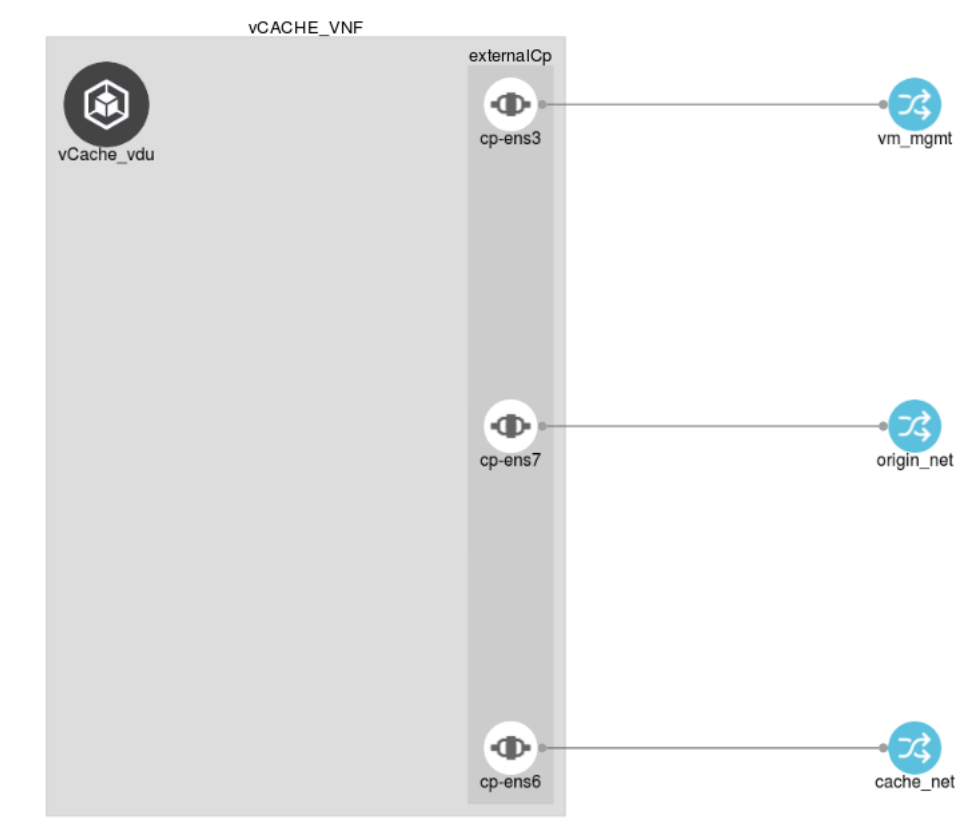


Figure 31: Graphical representation of a VNFD

The list of elements to be visualized can be filtered and classified based on different criteria, e.g. per site facility, per type, etc., tuning the kind of information available for different items and disclosed to different users. The browse and look-up tool provides also mechanisms for onboarding blueprints and descriptors, verifying their

compliance with the expected format and validating their content. In some cases, the user needs to onboard a set of correlated elements. For example, during the onboarding of a VSB, also the associated NSD and translation rules need to be provided. To facilitate the entire procedure, the tool provides an interface that guides the user in the different steps, indicating the fields to be filled in and the items to be onboarded.

Finally, the browse and look up tool implements Role Based Access Control (RBAC) functionalities, cooperating with the underlying catalogues and an external Identity and Access Management tool¹⁵ based on KeyCloak¹⁶. This approach allows to regulate the access to the different resources (and their associated actions) according to the role of the user, as discussed in section 2.4. Further RBAC rules can also be applied, according to the user profile and its group. For example, an experimenter may have access only to the ExpBs onboarded by an experiment developer who belongs to the same group.

4.3.2 Tool Design / Architecture

The browse and look up tool is implemented as a web application written in TypeScript and based on the Angular 8 framework, using the Angular Material toolkit for graphical layout. The web application runs on Node JS. It is released under the Apache 2.0 license. The Angular Material toolkit provides the common layout and a set of pre-defined icons used across all the web pages of the browse and look up tool. Moreover, the structure of the web pages is implemented through a modular set of Material components that represents specific items (top menu, left side menu, main contents) and can be configured and composed together to build the layouts of the different pages.

The interaction with the catalogues is based on REST APIs and it is handled through the Angular HttpClient-Module. Moreover, additional libraries, defined in Table 6, are used to implement specific features, like the interaction with the KeyCloak access management tool and the representation of network graphs.

Table 6: Libraries used in browse and look up tool

Library	Description and usage in the tool	License
cytoscape.js https://www.npmjs.com/package/cytoscape	Javascript library for building graphs. It is used for the graphical representation of NSDs, VNFDs and VSBs/ExpBs topologies.	MIT license
js-yaml https://www.npmjs.com/package/js-yaml	Javascript YAML parser and dumper. It is used to parse and dump the VNFDs and NSDs expressed in TOSCA format and YAML language.	MIT license
keycloak-js https://www.npmjs.com/package/keycloak-js	JavaScript Keycloak adapter. It is used to interact with the KeyCloak RBAC tool to manage authentication and authorization mechanisms.	Apache 2.0

4.3.3 Tool Implementation

The “browse and look up” tool gives access to the Portal Catalogue, allowing the experimenter to easily visualize blueprints and descriptors for the different elements managed by the 5G EVE platform. From the menu on the main page of the tool, shown in Figure 32, the experimenter can select the type of item to visualize.

¹⁵ The Identity and Access Management tool is out of scope for this deliverable and it will be documented in D4.2.

¹⁶ <https://www.keycloak.org/>

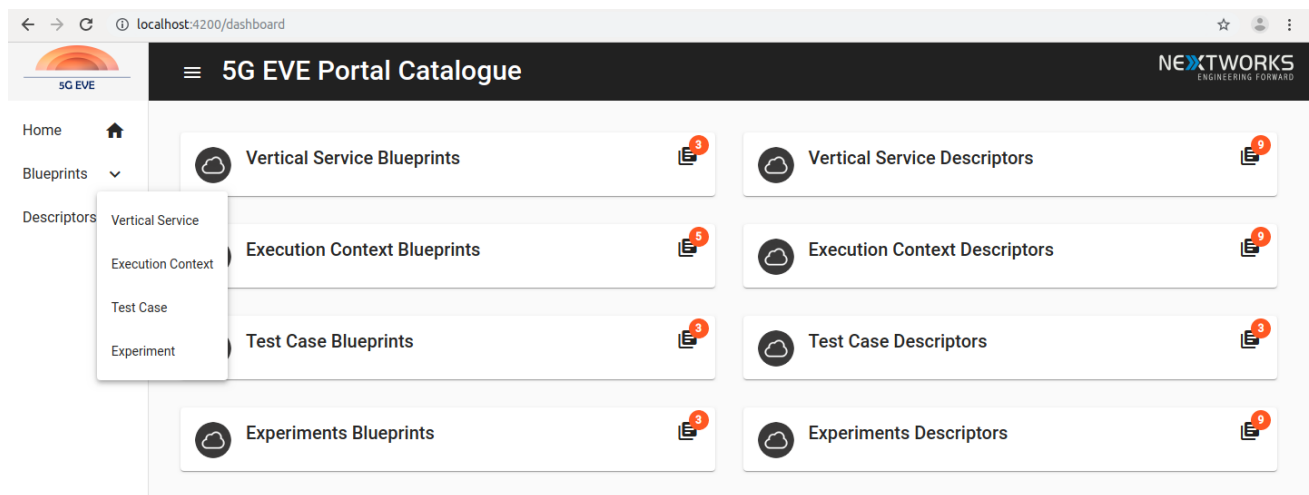


Figure 32: 5G EVE Portal Catalogue – main page

Based on the option selected by the experimenter, the system initially visualizes a list of blueprints (VSBs, CBs, TCBs or ExpBs) or experiment descriptors with their main information (see Figure 33), to provide an overview of the 5G EVE platform’s offer. The next version of the tool, fully integrated in the 5G EVE portal and planned for D4.2, will also include the support of filters to help the experimenter in the navigation of the services and experiments models offered by 5G EVE.

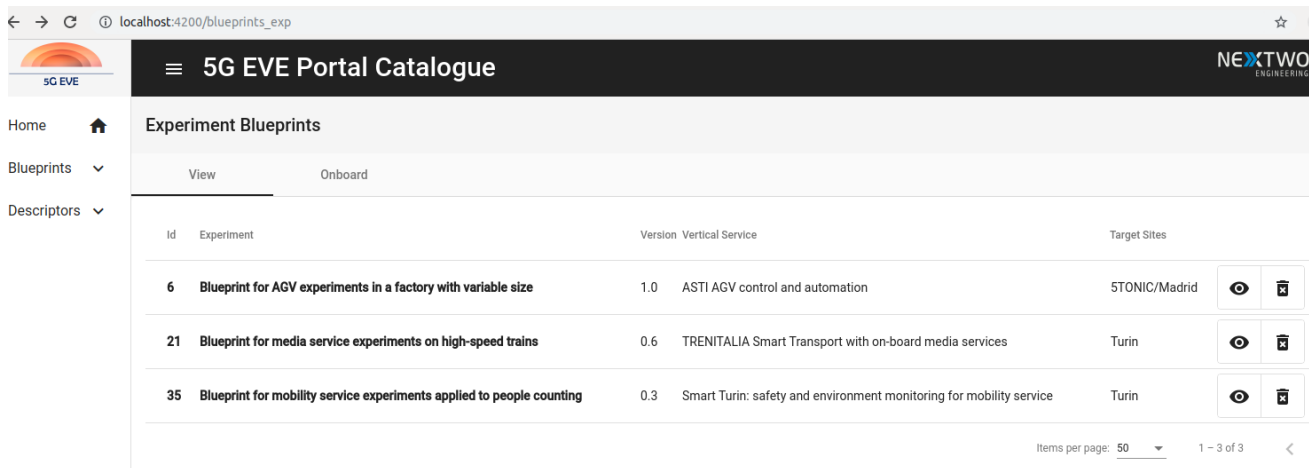
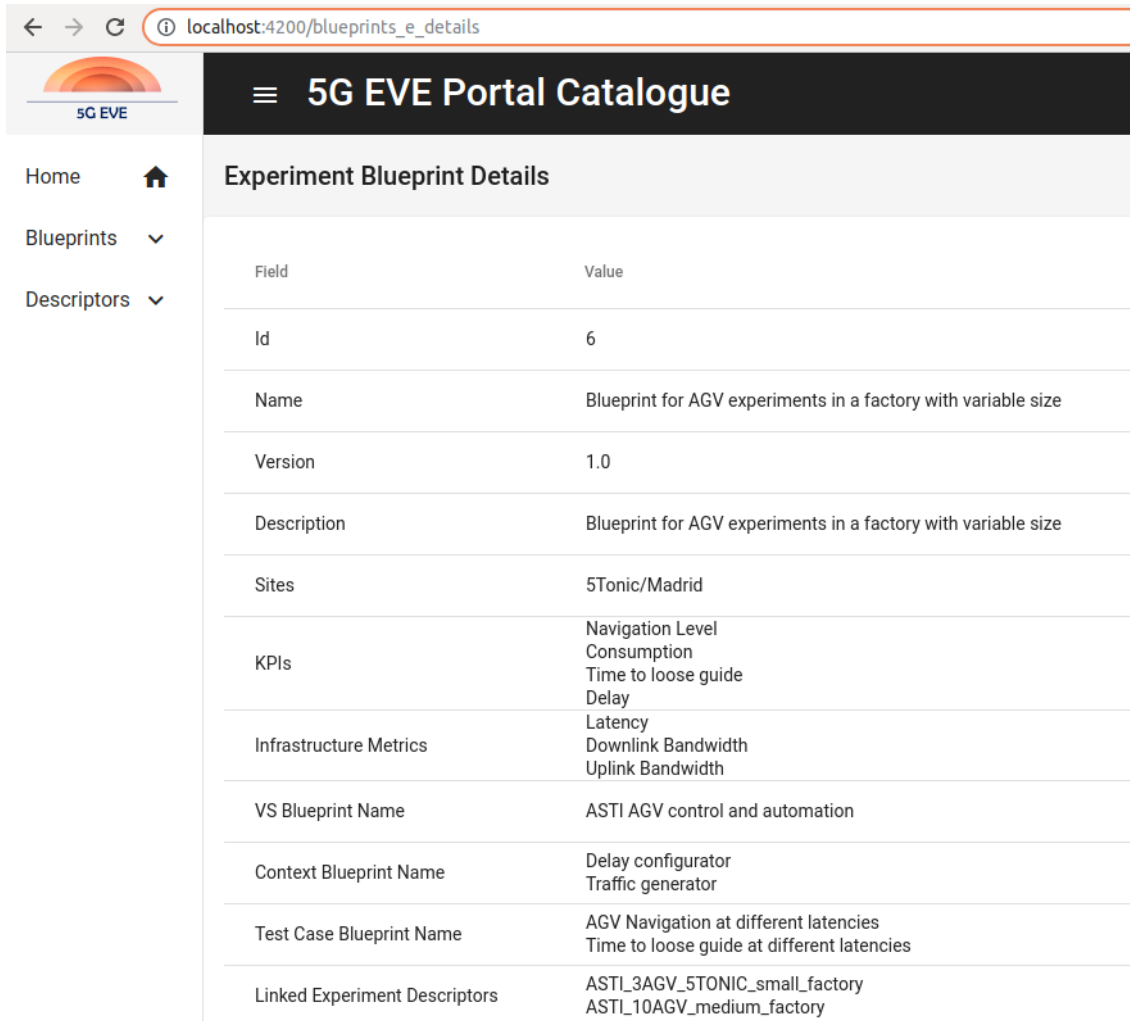


Figure 33: 5G EVE Portal Catalogue: List of experiment blueprints

Selecting a specific element from the initial list, the experimenter can visualize the details of the blueprint/descriptor of interest, including also direct links to the associated elements. For example, from the detailed view of an experiment blueprint shown in Figure 34, the experimenter can move to the pages of the associated vertical service blueprint, context blueprints or test case blueprints.



Field	Value
Id	6
Name	Blueprint for AGV experiments in a factory with variable size
Version	1.0
Description	Blueprint for AGV experiments in a factory with variable size
Sites	5Tonic/Madrid
KPIs	Navigation Level Consumption Time to loose guide Delay
Infrastructure Metrics	Latency Downlink Bandwidth Uplink Bandwidth
VS Blueprint Name	ASTI AGV control and automation
Context Blueprint Name	Delay configurator Traffic generator
Test Case Blueprint Name	AGV Navigation at different latencies Time to loose guide at different latencies
Linked Experiment Descriptors	ASTL_3AGV_5TONIC_small_factory ASTL_10AGV_medium_factory

Figure 34: Detailed view of an experiment blueprint

From the experiment blueprint page, it is also possible to visualize the list of experiment descriptors (owned by the experimenter) that have been defined for that blueprint. Selecting a descriptor, the experimenter can visualize its details.

The software code of the browse and look-up tool is available in the following repository: <https://github.com/nextworks-it/slicer-catalogue/tree/5geve-release>

4.3.4 Interaction with other tools/layers

The “browse and look up” tool interacts with the following components of the 5G EVE platform, acting as client in all the cases:

- Identity and Access Management tool
- Portal catalogue
- I/W framework catalogue

The interaction with the Identity and Access Management tool (based on Keycloak) allows to secure the communication with the catalogues and it is based on the integration of the client-side “keycloak-js” adapter in the web application. Whenever the user accesses the browse and look up tool, he/she needs to login to Keycloak through the mediation of the web application login page. When the user is authenticated, the web application interacts with the REST services offered by the catalogues (also secured by Keycloak) including the bearer

token in the Authorization header of the HTTP messages. Tokens are automatically updated after their expiration time and they are used on the server side (i.e. at the catalogues) to verify the authorization of the user for accessing a given resource.

The interaction with portal and I/W framework catalogues is entirely based on the REST APIs exposed by their REST services to enable the CRUD actions on blueprints and descriptors. The browse and look up tool can access only a subset of the catalogues' REST APIs, as shown in Table 7. In fact, the creation, the update and the deletion of VNFDs and PNFDs is performed automatically through the synchronization of the I/W framework catalogue with the catalogues deployed in each 5G EVE facility.

Table 7: Browse and look up tool access to CRUD actions offered by 5G EVE catalogues

	Create	Read	Update	Delete
VSB	Y	Y	Y	Y
CB	Y	Y	Y	Y
ExpB	Y	Y	Y	Y
ExpD	Y	Y	Y	Y
VNFD		Y		
PNFD		Y		
NSD	Y	Y	Y	Y

4.4 Intent Based Interface

4.4.1 Purpose/Description of the tool

Intent-based Networking (IBN) provides a new approach to networking, where planning, design, implementation and changes to the network are made automatically by using special software. IBN's main goal is to simplify the creation, management and policy enforcement to the network with the use of artificial intelligence (AI), network orchestration and machine learning (ML).

As stated in [10] every intent-based networking system (IBNS) incorporates the following four aspects:

- Translation and validation: The system can translate a given command or business intent into actions that the software can perform. Additionally, it verifies that the intent can be executed successfully in the first place.
- Automated implementation: Once the intent or desired state is defined, the system will allocate network resources and enforce policies to meet the goal.
- State awareness: The system will continuously gather and monitor data to reflect the current state of the network.
- Assurance and dynamic optimization/remediation: Using machine learning, the system will implement and maintain the desired state of the network, applying automated corrective action if necessary. ML gives the network the ability to analyse, extract and learn from data dynamically.

In IBN humans express their "intent" to the network, without providing any configuration. Instead the network itself shall figure out what actions are necessary to be taken, to provide the expected network state (intent). Having stated that, it is not requested from the user/vertical to have any basic knowledge about networking.

Nowadays administrators must configure every device needed to accomplish a given task. With IBN, though, the configuration of the different devices is made automatically, without having engineers configuring each device separately.

As is also stated in [10] a few benefits of intent-based networking include:

- Reduces the complexity of the management and maintenance of network policies.
- Simplifies the deployment of additional network services.
- Reduces labour associated with traditional configuration of switches and routers.
- Strengthens network security capabilities.

- Improves agility of the entire network system.
- Eliminates repetitive or error-prone coding associated with manual inputs.

In 5G-EVE project the Intent-based Interface provides a way for the vertical experimenters to instantiate an experiment (experiment design and definition phase). The intention of the user will be collected and translated into predefined Vertical Service Blueprints (VSB) and consequently into Context and Experiment Blueprints as well, describing his/her intended service needs in networking terms, thus enabling the creation of an experiment descriptor (ExpD) assisted via natural language.

4.4.1.1 State of the art

In [11] is stated that IBN and Software Defined Networking (SDN) have the same goal of automating network configuration and management activities, which leads to many authors considering IBN as an SDN improvement, or the next evolutionary phase of SDN. SDN implies an organization scheme that tries to offer abstraction on individual configuration and management tasks, allowing you to think in terms of required services. On the other hand, IBN's main idea is that the level of abstraction makes us get even closer to the business, to what we want or need and not how to solve it. Therefore, we could have a network and communications platform ruled by an IBN solution, with a "low level" implementation both based or not on SDN.

So far there are several manufacturing companies that are making efforts to develop products based on IBN, as is also mentioned in [11]. Something that these companies have in common is that they have products in the market associated with SDN and SDN-WAN, which provide the base from which to develop the IBN concept. Some of the companies working on IBN are: Cisco, Apstra, Juniper, Veriflow and Iential, among others.

All the aforementioned companies focus on providing software on ML and AI in order to configure the network requirements, monitor the state of the network and predict network problems before they appear and solve them automatically, in order to maintain network's expected state.

As stated in [12] Intent-based networking captures and translates business intent into network policies that can be automated and applied consistently across the network. The end goal is for the network to continuously monitor and adjust network performance to assure the desired business outcome. For this to be achieved three functions are necessary:

- *Translation*: Capabilities that tell the network what to do to achieve the desired business outcome, based on a consistent and verified policy the network can act upon.
- *Activation*: Deployment of the expressed policies throughout the network infrastructure, by automating systemwide changes to all relevant network and security devices.
- *Assurance*: Continuous monitoring and verification that the desired intent has been applied and business outcome has been achieved. This can include remediation through recommended corrective actions and ongoing optimization through predictive analytics.

Lerner mentions in [13] that: "Intent-based networking is not a product, or a market. Instead, it is a piece of networking software that helps to plan, design and implement/operate networks that can improve network availability and agility. Another way to describe it would be lifecycle management software for networking infrastructure".

In 5G-EVE project we focus on extracting information about the desired state of the network, according to the input provided by the experimenters, and ensuring that the network can provide the requested service, based on the requirements of the service and the availability of the resources (available VNF, PNFs, interconnections, etc.). This way we cover the translation & validation part of an IBN. Additionally, by matching the intention of the users into VSBs, which provide some of the requirements necessary for providing a service, we cover the automated implementation part of the IBN.

4.4.1.2 Alignment with standardization (3GPP)

According to TR 28.812 listed in [14] 3GPP is looking into IBN, by focusing on a Study on scenarios for Intent driven management services for mobile networks. More specifically in the Introduction part of section 4 (4.1.1) it is mentioned that:

“An Intent Driven Management Service (IDMS) allows its consumer the ability to provide desired intent for managing the 5G network and service. An intent driven management service provider translates the intent to appropriate network deployment information and implements it.”

IDM reduces the management complexity of the underlying network infrastructure by expressing consumer desired intent without getting into the intricate detail of the underlying network infrastructure. IDM could contribute to efficient network management, especially in a multiple vendor scenario.

In 5G-EVE we try to reduce the management complexity both for the site managers and the system administrator by translating the intention of the experimenters into a suitable Blueprints (VSB, CB and ExpB), that will provide information about the network infrastructure needed for the deployment of the experiment/service. In that way the experimenters do not need to know about networking and how to establish a service.

4.4.2 Tool Design / Architecture

The architecture of the Intent-based tool consists of the following components:

- A Graphical User Interface (GUI), which provides a way to the experimenter to interact with the rest application. Through the GUI, the experimenters can provide their intention, using one of the two ways available (Free Text – Guided selection), which will be analysed in section 4.4.3.
- A Translation tool, which collects the information provided by the experimenter and creates a Blueprint schema, according to the experimenter’s need. The main functionality of this component is to recognize keywords in the intention provided and to translate them into specific actions/functionalities for the various blueprints.
- A database, where the confirmed experiments are stored. Through the database the application also checks the availability of the resources in each site/facility according to the scheduled experiments.
- An Apache Tomcat Server, which hosts the whole application and makes it available to the users to access the Intent-Based Interface.

4.4.3 Tool Implementation

As stated in section 3.1.2, for the experimenter to create an experiment, an ExpD is needed. The role of the intent-based interface in the 5G-EVE project is to find the most suitable VSB and to automatically fill the necessary information missing in the VSB in order to facilitate the process towards the full definition of the ExpD.

To define an experiment, the experimenters have two choices: i) through a free text format or ii) by providing the necessary information in the Guided Selection part of the GUI. The purpose of the Intent Based tool is to facilitate the experiment design by low-skilled experimenters (at least in terms of networking) by asking high level service-related information and translating them into specific experimental parameters. The final proposed experimental parameters defined by the Intent Based tool are shown in Figure 40, while the experimenter still has the opportunity to directly edit a proposed default value, should they want to fine-tune it (for more experienced experimenters, e.g. adjust a proposed KPI value). Some of the information the users could provide are:

- The service they want to run
- The sector of the service
- The country they are running the experiment from
- The country they want the experiment to be executed
- The date of the experiment
- The time of the experiment

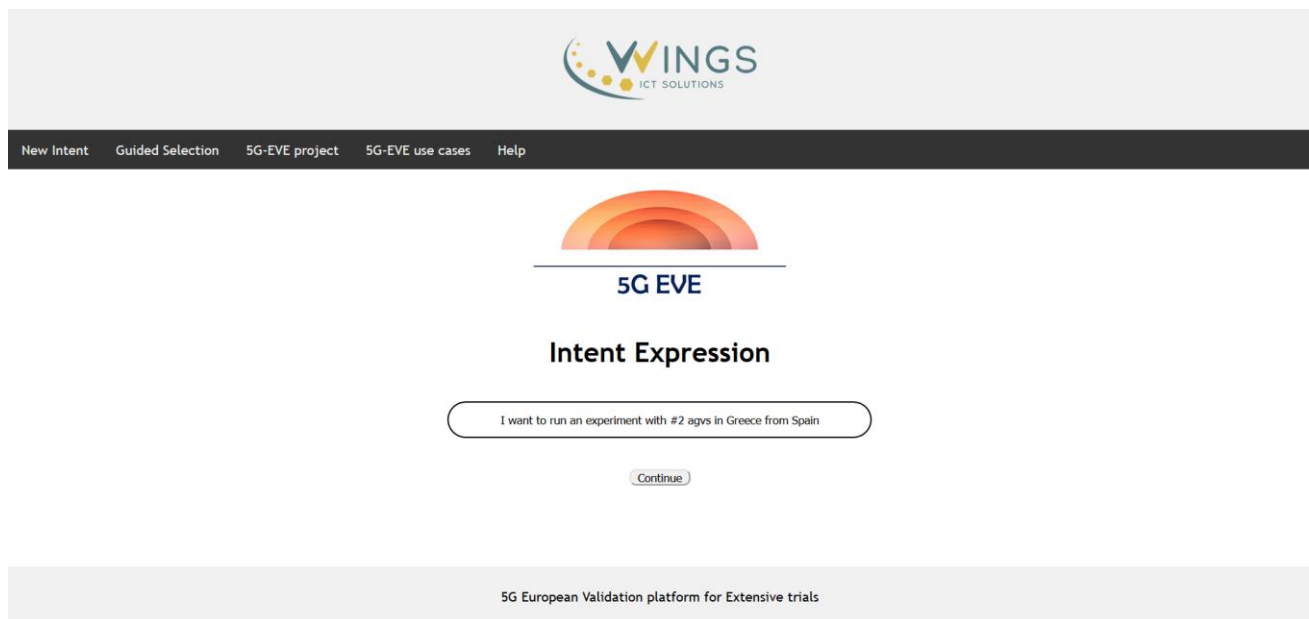
- Number of devices needed depending on the use case (e.g.: number of AGVs, number of smartphones, etc)

4.4.3.1 Free Text format

Through the Free Text format (Figure 35) the experimenters may define their intention by using natural language. After they have provided their intention the translation tool checks the intention and matches specific keywords to certain fields of the VSB. In case there are empty fields the users are prompted to provide the missing information (Figure 36). Also, in case the experimenters have provided invalid information, such as a passed date, or negative numbers, or a country that cannot support a specific service, a message is returned to them by stating the problem encountered.

After the intention has been given in free text format, the translation tool processes the text given in order to identify specific keywords and map them into specific actions. The keywords identified so far are the sectors, the services, and the following expressions in order to define where to run the experiment and the 5G EVE site facility the experimenter wants to run the experiment from: *in France/Greece/Italy/Spain, to the French/Greek/Italian/Spanish facility/site, from France/Greece/Italy/Spain*. Specific mapped symbols are used in order to define the necessary numbers in each case (e.g. number of AGVs, number of devices, data rate, latency, etc.). The application also recognizes dates in the form of dd/mm/yyyy and time in the form of hh:mm. In the last two cases checks take place in order to define if the values given are valid.

All services and keywords used in the Intent Based tool are based on the services, provided by each site facility, according to [8].



The screenshot displays the 'Intent Expression' interface of the 5G EVE platform. At the top, the 'WINGS ICT SOLUTIONS' logo is visible. Below it is a dark navigation bar with links for 'New Intent', 'Guided Selection', '5G-EVE project', '5G-EVE use cases', and 'Help'. The main content area features the '5G EVE' logo and the title 'Intent Expression'. A text input field contains the user's intent: 'I want to run an experiment with #2 agvs in Greece from Spain'. A 'Continue' button is positioned below the input field. At the bottom of the interface, the text '5G European Validation platform for Extensive trials' is displayed.

Figure 35: Intention through Free Text format

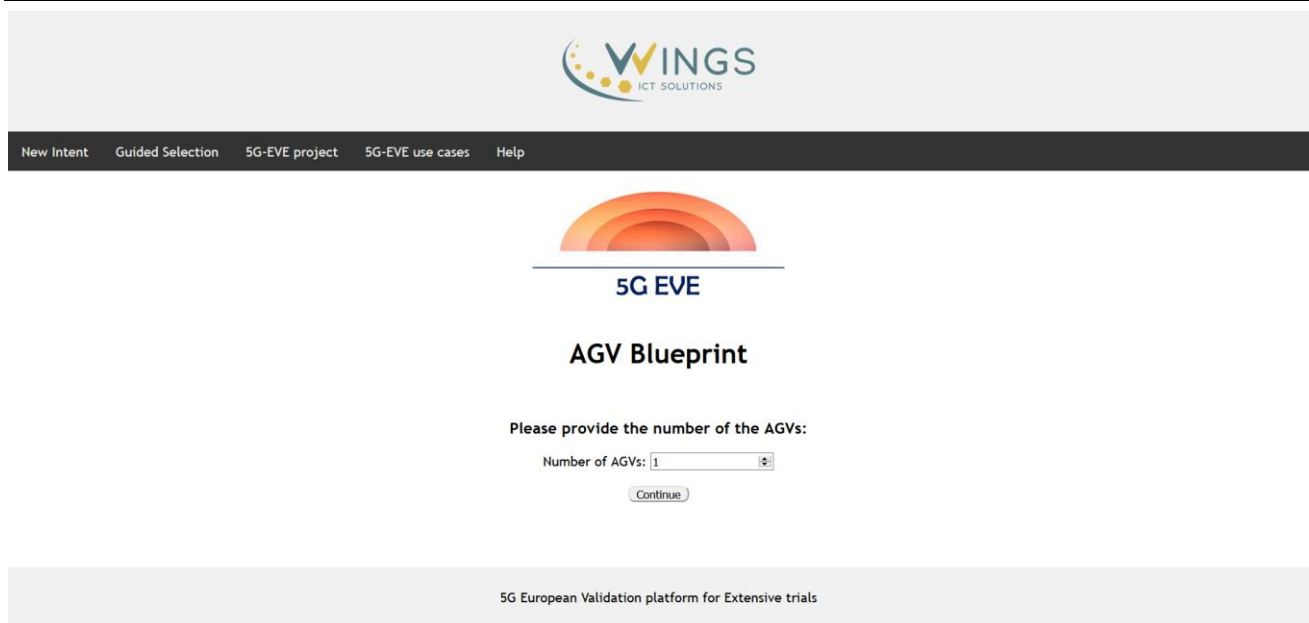


Figure 36: Users are prompted to provide missing fields

4.4.3.2 Guided Selection

In case of the Guided Selection the experimenters may select the service, they want to run, from a list of the available services (Figure 37). After they have selected, they will be redirected to a page providing information about the necessary fields to be filled (Figure 38). In case that only one choice is available, the field will be automatically filled, otherwise the experimenters will have to choose. Some values are also provided by default, but the users may change them according to their needs.



Figure 37: Guided Selection services

New Intent
Guided Selection
5G-EVE project
5G-EVE use cases
Help



5G EVE

AGV Guided Selection

Select the attributes you want your Blueprint to have

Select the country you want to run the experiment:
 Greece
 Spain

Select the time you want the experiment to be executed:
 Hour: Minutes:

Select the country you want to run the experiment from:
 France
 Greece
 Italy
 Spain

Select the date you want the experiment to be executed:
 Day: Month: Year:

Number of AGVs:

localhost:8080/Intent/GuidedSelection.jsp

Figure 38: Intention through Guided Selection

In both cases when the selection is finished a table will be presented, providing information about the expected functionality of the experiment as described through a completed VSB (Figure 39). There are also some fields automatically filled from the intent-based mechanism (Figure 40) to facilitate the experimenter. These fields contain information about the range of some KPIs needed to run the specific experiment, according to the vertical service providers as depicted in [8]. The default value is the minimum accepted to execute the service, according to the information provided by the Verticals in [8]. However, the experimenters may edit the proposed default values as they see fit in order to perfectly match their required service. The exact definitions of the various *Blueprint* fields utilized by the 5G EVE platform are given below:

- **E2E Latency:** Measures the duration between the transmission of a small data packet from the application layer at the source node and the successful reception at the application layer at the destination node plus equivalent time needed to carry the response back.
- **Data rate:** It is set as the minimum user experience data rate required for the user to get a quality experience of the targeted application/use case (it is also the required sustainable data rate).
- **Reliability:** The amount of sent packets successfully delivered to the destination within the time constraint required by the targeted service, divided by the total number of sent packets. NOTE: the reliability rate is evaluated only when the network is available.
- **Availability:** The network availability is characterized by its availability rate X, defined as follows: the network is available for the targeted communication in X% of the locations where the network is deployed and X% of the time. X% varies from 90% to 99.9999%.
- **Mobility:** Mobility refers to the system's ability to provide seamless service experience to users that are moving. In addition to mobile users, the identified 5G use cases show that 5G networks will have to support an increasingly large segment of static and nomadic users/devices.
- **Broadband connectivity:** High data rate provision during high traffic demand periods (It is also a measure of the peak data rate required).
- **Capacity:** Capacity is measured in bit/s/m² is defined as the total amount of traffic exchanged by all devices over the considered area. The KPI requirement on the minimum Traffic Volume Density/Area Capacity for a given use case is given by the product [required user experienced data rate]x[required connection density].

- **Device Density:** Up to several hundred thousand simultaneous active connections per square kilometre shall be supported for massive sensor deployments. Here, active means the devices are exchanging data with the network.

AGV Blueprint Created

If the Blueprint satisfies your needs please press confirm, otherwise press the edit button to modify the blueprint

Blueprint Information	
Name	<i>AGV control and automation</i>
Description	<i>Blueprint for AGVs using 5G network for Automation and Control purposes</i>
Version	1.0
Identity	<i>BP Identity</i>
Sector	<i>Industry 4.0</i>
Launched from	<i>France</i>
Executed to	<i>Greece</i>
Date of Execution (dd/mm/yyyy)	1/7/2019
Time of Execution	17:00
Number of AGVs	1
Location of AGVs	<i>Greece</i>
Service	URLLC

Figure 39: Information about the Blueprint created

Latency	10ms
Data Rate	54Mbps
Reliability	99.9%
Availability	99.9%
Mobility	20Km/h
Broadband Connectivity	6480.0Mbps
Capacity	2.16Mbps/m2
Device Density	40000Devices/Km2
Network Slicing	Y
Security	N

Cancel Edit Confirm

Figure 40: Expected metric values of the experiment

In the application there is a Help page, which provides information to the experimenters on how to express an intention using the Free text format, and information as well on how to use the Guided selection tool (Figure 41).

New Intent Guided Selection 5G-EVE project 5G-EVE use cases Help

Help Page

In order to express an intent you have to define the service you want to run (e.g. recognition service) or the sector your service belongs to (e.g. smart transport).
 In order to define the country you wish to run the experiment from you have to use the word from followed by one of the available countries: France, Greece, Italy, Spain.

In order to define the country you wish to run the experiment to you have to use the word to followed by the country of your choice: France, Greece, Italy, Spain, or the words to the french/greek/italian/spanish site/facility). In case the service is provided only in one country the choice will be made automatically, otherwise you will receive a warning message to provide a country, where the service is available.

In case you want to provide some numbers (e.g. number of aqs, number of devices, etc) you have to use the symbols # and \$. Note that the numbers cannot be decimals or less than zero. In case of a provided number is out of range a warning message will be received to provide a valid value. Below are two list provided information about the cases each symbol is used for:

- #
 - Number of AGVs (AGVs scenario)
 - Number of devices (Recognition/Tracking/Uran mobility/Smart energy/Smart Turin/Smart Home scenarios)
 - Latency (AR Interaction/Business Augmented Booth/Connected Ambulance/Virtual visit scenarios)
 - Uplink (Immersive Media/On-Site Live Event Experience scenarios)
- \$
 - Data Rate (AGVs/Smart energy/Connected Ambulance scenarios)
 - Bandwidth (AR Interaction/Business Augmented Booth/Virtual visit/Immersive and Integrated Media/Ultra High-Fidelity Media/On-Site Live Event Experience scenarios)

In order to provide the date you want the experiment to be executed you have to use the following format: dd/mm/yyyy. If the date provided has passed you will receive a warning message. Also if the number of days provided to a specific month are wrong you will receive a warning message too.

In order to provide the time you want the experiment to be executed you have to use the following format: hh/mm, where hh varies from 0 to 23. In case you provide a wrong time the time will be filled automatically.

After you provide your intent, in case there are some necessary fields missing, you will be guided to select some values for the required fields.

When the selection is finished a table will be presented, providing information about the expected functionality of the experiment (Blueprint). You are provided with the choice to make changes to the Blueprint before finalizing your decision.

Guided Selection Helping notes

In case of the guided selection you have to select the service you want to run from one of the available services.
 After you made a selection you will be redirected to a page providing information about the necessary fields to be filled.
 In case that only one choice is available, the field will be automatically filled, otherwise you will have to choose. Some values are also provided by default, but you can change them according to your needs.
 When the selection is finished a table will be presented, providing information about the expected functionality of the experiment (Blueprint). You are provided with the choice to make changes to the Blueprint before finalizing your decision.

Information on the extra fields

Latency
 Measures the duration between the transmission of a small data packet from the application layer at the source node and the successful reception at the application layer at the destination node plus equivalent time needed to carry the response back.

Speed (data rate)
 It is set as the minimum user experienced data rate required for the user to get a quality experience of the targeted application/use case (it is also the required sustainable data rate).

Reliability
 The amount of sent packets successfully delivered to the destination within the time constraint required by the targeted service, divided by the total number of sent packets. NOTE: the reliability rate is evaluated only when the network is available.


Availability
 The network availability is characterized by its availability rate X_i , defined as follows: the network is available for the targeted communication in $X_i\%$ of the locations where the network is deployed and $X_i\%$ of the time. $X_i\%$ varies from 90% to 99.999%.

Mobility
 Mobility refers to the system's ability to provide seamless service experience to users that are moving. In addition to mobile users, the identified 5G use cases show that 5G networks will have to support an increasingly large segment of static and nomadic users/devices.

Broadband Connectivity
 High data rate provision during high traffic demand periods (it is also a measure of the peak data rate required).

Capacity
 Capacity is measured in bit/s/m2 is defined as the total amount of traffic exchanged by all devices over the considered area. The KPI requirement on the minimum Traffic Volume Density/Area Capacity for a given use case is given by the product [required user experienced data rate] x [required connection density].

Device Density
 Up to several hundred thousand simultaneous active connections per square kilometer shall be supported for massive sensor deployments. Here, active means the devices are exchanging data with the network.



5G European Validation platform for Extensive trials

Figure 41: Help Page

4.4.3.3 Programming language – Database

As mentioned in 4.4.2 the IBN tool consists of a GUI, a translator tool, an Apache Tomcat server and a database. Both the GUI and the translation tool are written in JavaSE and the database used is MySQL. The Apache Tomcat server (version 7) is also installed in the Java platform (Eclipse).

All the browsing pages (GUI) are .jsp files stored in Eclipse and are written in HTML and CSS. To communicate with the translator tool and pass the information provided by the users, these pages use a POST method to Java servlets. So, to pass information, RESTful services are used. There are separate servlets for each service, and they make checks about all the information needed to create a functional Blueprint. According to the intention of the users the servlets redirect them to the appropriate pages to provide the information missing.

After the necessary information is provided and the VSB is created, the Java tool communicates with MySQL to check the availability of the resources in a specific site facility at a specific date/time. The database consists only by one table, which saves all the parameters of a created VSB. In order to run the IBN app Tomcat server must get started. Tomcat is also installed in Eclipse.

The implementation of the Intent-Based Tool is available in the following github repository: <https://github.com/5GEVE/5G-EVE-WP4-intent-based-tool>

4.4.4 Interaction with other tools/layers

The Intent-Based Interface is part of the Portal and its services could be accessed through the 5G EVE experimenter platform GUI, for the experimenter to express an intention. After the intention is provided the IBN tool translates that intention into experiment requirements, as stated by the VSB and resulting ExpD. So, the main interaction of the IBN tool, with other tools, is the VSB instantiation during the experiment design and definition phase, by filling necessary fields of the information extracted from the experimenter's intention. After the Blueprint creation the IBN tool can be used again to express a new experiment.

Besides the VSB instantiation, the IBN tool interacts also with a database, which has information on the scheduled experiments to check the available resources in a specific site/facility at a requested date-time. If the experiment cannot be scheduled at that specific time, a suggestion is made to the users, by providing another time or site/facility (if the service can be supported to multiple countries) to schedule the experiment.

5 5G EVE catalogue software prototype

5.1 Catalogue components and functionalities

As introduced in section 2.4, the 5G EVE platform includes two main catalogues, placed at the portal and at the I/W framework respectively. The portal catalogue offers the centralized point of access to all the experiment-related information stored in the system, managing directly blueprints and descriptors for vertical services, contexts and experiments, and wrapping the access to the NFV descriptors (NSDs, VNFDs and PNFDs) and records handled at the I/W framework catalogue. Both the portal and the I/W framework catalogues have a multi-site scope and offer a complete view of the entities available in the entire 5G EVE platform, independently on the specific site where they are on-boarded or deployed. The I/W framework catalogue, in turn, operates on top of multiple per-site catalogues, each of them deployed in a specific site and providing a view restricted to the elements available on that site.

In a such distributed platform it is a challenge to maintain the synchronization between the per-site and the centralized catalogues. Moreover, most of the information elements stored in the different catalogues are associated to each other. For example, a VSBs is associated to an NSD that, in turn, is associated to several VNFDs and PNFDs, which may be distributed in different sites. An ExpB is associated to a VSB and to a list of CBs, where each of them is in turn associated to NSDs, and so on. This means that the onboarding of specific elements must be coordinated to guarantee the proper synchronization of all the involved repositories and the consistent storage of the different entities across the entire chain of catalogues, as also explained in the workflows defined in section 3.2. Additional workflows to coordinate and synchronized the 5G EVE I/W framework catalogue and the ones available in the different were reported in D3.2 [6], section 4.3.3.1.

The 5G EVE catalogues need to implement an extensive set of features to guarantee this kind of synchronization, going beyond the typical functionalities defined for standard NFV MANO catalogues. Table 8 highlights the peculiar features offered by the portal and I/W framework catalogues respectively. Moreover, both need to support suitable RBAC mechanisms to regulate the access of the users to the managed resources, as extensively discussed in section 2.4.

Table 8: Features of 5G EVE catalogues

5G EVE Catalogue	Features
Portal catalogue	Coordination between the onboarding of blueprints at the local catalogue and the onboarding of the associated NSDs at the I/W framework catalogue.
	Validation mechanisms to guarantee the consistency between VSDs/CDs/ExpDs and the related blueprints during their onboarding.
	Mechanisms to retrieve NSDs (stored at the I/W framework catalogue) associated to a given blueprints during the processing of query requests.
I/W framework catalogue	Mechanisms to guarantee the synchronization between the information available in the distributed catalogues at the different sites and the centralized I/W framework catalogue.
	Maintenance of the association between the local descriptors and the corresponding ones at the sites' catalogues.
	Subscriptions/notifications mechanisms to update the local repository when new VNF packages are onboarded on a site's catalogue.
	Automated onboarding of NSD into the catalogues at the target sites.
	Automated translation between the internal information model and the site-specific languages and information models used for the descriptors in each of the sites' catalogues.

It is also worth to mention that the 5G EVE catalogues are designed to maximize their interoperability in multi-domain environments. For this reason, where possible, the interfaces of the catalogues and their information models are compliant with the latest standards defined in major SDOs as ETSI NFV and ETSI MEC. Moreover, the procedural approach and the modelling related to service blueprints and descriptors are inspired by the 5G-TRANSFORMER solution, which provides a set of simplified mechanisms and interfaces to define and request vertical services. This solution is adopted also in other 5G PPP phase 2 and phase 3 projects, guaranteeing the compatibility and the easy interaction with other platforms. On the other hand, it should also be noted that the specific 5G EVE sites present a great variety of NFV MANO stacks, each of them providing its own NBI and characterised by its own information and data models. For this reason, the interaction between the I/W framework catalogue and the underlying sites requires a flexible approach to translate between different interfaces and information/data models.

5.2 Information models

One of the main goals of 5G EVE is to offer 5G services to vertical users in an easy way. We define high level information models for network services that are easily understandable by non-experienced users and partially customizable. Indeed, we also define the concept of blueprint, a template of a network service. The blueprint includes parameters to partially customize the service and allowing the vertical user to play with it and try it out under different conditions.

The concept of service blueprint is inherited from the 5G TRANSFORMER project [15]. In 5G EVE we extend it according to the specific requirements of the project. The main extensions are related to the fact that 5G EVE provides a platform for extensive trial and validation of 5G network services. Moreover, 5G EVE includes multiple site facilities and we worked to include multi-site support.

The information elements, with the relevant model, are the following:

- Vertical Service Blueprint (VSB): a high-level representation of a network service template. The service is designed by a Vertical user, who also provides software artifacts like virtual machines, application packages, etc. An Experiment Developer creates the VSB that represents the service of the Vertical.
- Context Blueprint (CB): a high-level representation of context network elements. We define context as the environmental conditions (from a network point-of-view) that the service can experience in a real-world deployment. For example, a CB can represent background traffic, undesired packet delay, undesired packet loss, etc. The representation of these elements as CBs enables their composition with VSB and their easy re-use with multiple VSBs.
- Experiment Blueprint (ExpB): a high-level representation of an experiment template. The ExpB is generated from the composition of a VSB with one or more CBs and it represent the network service under specific conditions of operation. The ExpB also includes information about tests to be run in the experiment, metrics to be monitored, and KPIs to be validated.
- Experiment Descriptor (ExpD): a high-level representation of a fully set-up experiment. An Experimenter configures an ExpB by assigning values to its parameters and by customizing tests to be performed. The procedure produces an ExpD, an information element containing all the relevant information to execute the experiment and collect the relevant information using the 5G EVE platform.

The high-level information models can be translated and associated to more detailed entities, called descriptors, as defined by the ETSI NFV Industrial Specification Group [16]. These descriptors, while containing all the information about the network service, are still technology agnostic and aim to be a generic standard for network orchestration. This makes the 5G EVE portal and interworking framework not tied to any specific Network Function Virtualization Orchestrator (NFVO) increasing their flexibility and applicability to all the trial sites participating in the project.

Descriptors used for network services are supported or extended by other descriptors. VNF and PNF descriptors define the building blocks of the network service. Experiment descriptors extend a network service by including all the information to run tests, collect monitoring data, perform KPI validation, etc.

For the implementation of all the information models, we followed ETSI guidelines and we used the YAML format. YAML is a serialization language that is easily understandable by both machines and humans. It can be

viewed as a superset of the more popular JSON format, as it offers a more complete information model. Migrating from JSON to YAML is easy, as every JSON file is also a valid YAML file.

As the main programming language for the 5G EVE portal is Java, we also provide a set of Plain Old Java Objects (POJOs) to map each of the blueprints presented in this section. Native and custom objects are easier to use by the developers instead of generic YAML. YAML files can be deserialized to POJOs, manipulated, and serialized back. This methodology also provides simple means to ensure the correctness of blueprints.

In the following section we are going to describe more in detail the scope and goal of each information model.

5.2.1 Vertical Service Blueprint

A Vertical Service Blueprint (VSB) is a high-level information model for a template of a network service. It includes a high-level description of the atomic functional components of the service and their interconnection. Furthermore, it contains a set of parameters that can be used to customize the service.

The VSB is used to describe in a formal and structured way the service that a Vertical wants to test on the 5G EVE platform. The VSB is written by the Experiment Developer, who interacts with the Vertical to gather information about their service and the kind of experiments they want to run on it.

As mentioned before, the VSB represents a *template* of the Vertical's service. Indeed, the VSB defines a set of parameters that allow the customization of the service in some of its parts. Usually, parameters are used to correctly size the service in terms of resources. We define Translation Rules that take one or more parameter values as input and accordingly select the appropriate NSD, DF and IL.

Apart from the textual representation, the VSB can also be used to produce a graphical rendering of the topology of the service. In fact, while the YAML format aims at being human friendly, it can be difficult to understand the connections between the functional components of the blueprint. The Experiment Developer can use the graphical rendering to easily check with the Vertical if the service representation as a blueprint is correct.

To represent the service, the VSB uses three main concepts: atomic functional components, end points, and connectivity services. Atomic functional components represent the building blocks of the service. Usually, each corresponds to a VNF or PNF. End points are the anchor points to connect the atomic components together. They are included in the description of the related atomic component but they are also described separately as they can have specific properties. For example, it is useful to distinguish between internal and external end points. Finally, connectivity services describe how endpoints, and thus atomic components) are connected to each other to create virtual links. Figure 42 shows a graphical representation of a VSB inspired by the ASTI use-case.

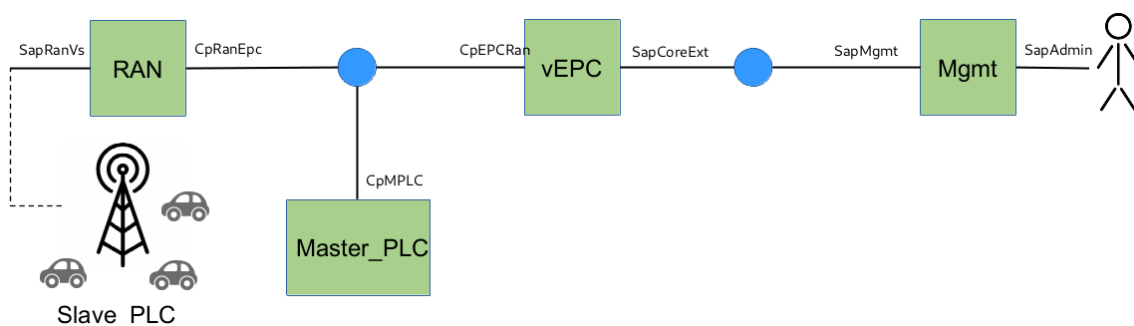


Figure 42: Graphical representation of a VSB example.

The green boxes are the atomic functional components, the black connectors are the endpoints, and finally the blue circles are the connectivity services. The service offers control and management functionalities for an autonomous guided vehicle use-case. The vehicles are shown in black on the left of the figure and they can be considered as PNFs. The Ctrl component is a VNF included in a Mobile Edge Computing server to provide low latency commands to the vehicles. The Mgmt component is a web interface with management functions to be operated by some administrator.

Table 9 reports the fields composing a VSB with their short description.

Table 9: Vertical Service Blueprint fields with relevant description

Field	Description
blueprintId	Unique Identifier for the VSB.
version	A version number.
name	Name for the VSB.
description	Short description of the VSB.
parameters	List of parameters. The list provides for each parameter its name, type, description, and the field of applicability
atomicComponents	List of atomic functional components (i.e., network functions and virtual applications in general) needed to implement the VSB.
endPoints	Specification of connection endpoints. They can be internal or external.
connectivityServices	List of virtual links and their relevant end points. Virtual links describe how the atomic functional components are connected.
serviceSequence	Description of how traffic flows among VNFs, supporting also multicast scenarios.
configurableParameters	Parameters that can be configured by the user for a specific instance of service derived from the given blueprint.
applicationMetrics	List of metrics that this service can provide. Metrics reported here are strictly application related. No network metrics are included.
compatibleSites	List of 5G EVE sites where the vertical service can be deployed.
compatibleContextBlueprint	List of IDs of the blueprints defining the experiment execution contexts that are compatible with the given vertical service.

A Vertical Service Blueprint YAML file example can be found in Annex B.

5.2.2 Context Blueprint (CB)

A Context Blueprint (CB) is like a VSB but it describes a template for network elements to define an operational context for the network service. CBs are used to define experimental conditions for the service described in the VSB, like artificial background traffic, artificial delay and similar. Like the VSB, the CB can include some parameters to customize its components.

The 5G EVE platform will include some generic context elements that can be applied to the majority of network services. Experiment Developers can compile additional CBs specifically related to a single network service. For example, given a VSB that includes a cloud application, the Experiment Developer can create a specific traffic generator that emulates the behaviour of one or multiple users.

Figure 43 shows a graphical representation of a CB for generic background traffic.

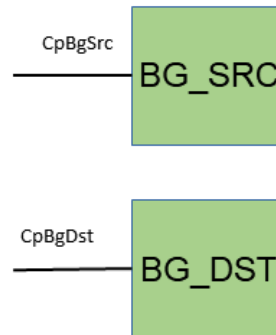


Figure 43: Graphical representation of a CB for artificial background traffic.

The CB is composed by just two atomic components and their endpoints. No connectivity services are provided. The CB must be combined with a VSB. It cannot be run or deployed on the 5G EVE platform by himself. Table 10 reports the fields composing a CB with their short description.

Table 10: Context Blueprint fields with relevant description

Field	Description
blueprintId	Unique Identifier for the CB.
version	A version number.
name	Name for the CB.
description	Short description of the CB.
parameters	List of parameters. The list provides for each parameter its name, type, description, and the field of applicability.
atomicComponents	List of atomic functional components (i.e., network functions and virtual applications in general) needed to implement the CB.
endPoints	Specification of connection endpoints. They can be internal or external.
connectivityServices	List of virtual links and their relevant end points. Virtual links describe how the atomic functional components are connected.
configurableParameters	Parameters that can be configured by the user for a specific instance of execution context derived from the given blueprint.
applicationMetrics	List of metrics that this execution context can provide. Metrics reported here are strictly application related. No network metrics are included.
compatibleSites	List of 5G EVE sites where the vertical service can be deployed.

5.2.3 Experiment Blueprint (ExpB)

An Experiment Blueprint (ExpB) describes a template for an experiment to be executed on the 5G EVE platform. The ExpB is created from the composition of a VSB and a custom set of CBs and it includes all of their parameters. Furthermore, the ExpB includes information about monitoring, KPIs, and tests to run.

The ExpB describes an experiment with a specific network topology that cannot be changed during its execution. The parameters allow for some, but limited customization of the experiment.

The ExpB is compiled by the Experiment Developer with the support of semi-automated tools. The tool will take as input a VSB and a selected set of CBs. Then, it is going to combine the selected elements respecting some constraints and it is going to propose various topologies to the Experiment Developer. Once selected the desired combination, the ExpB is persistently stored in the 5G EVE platform.

Figure 44 shows an ExpB example. The topology is the result of the combination of the components of the VSB in **Figure 42**, a CB for artificial background traffic (Figure 43), and a second CB to introduce artificial packet delay (DEL).

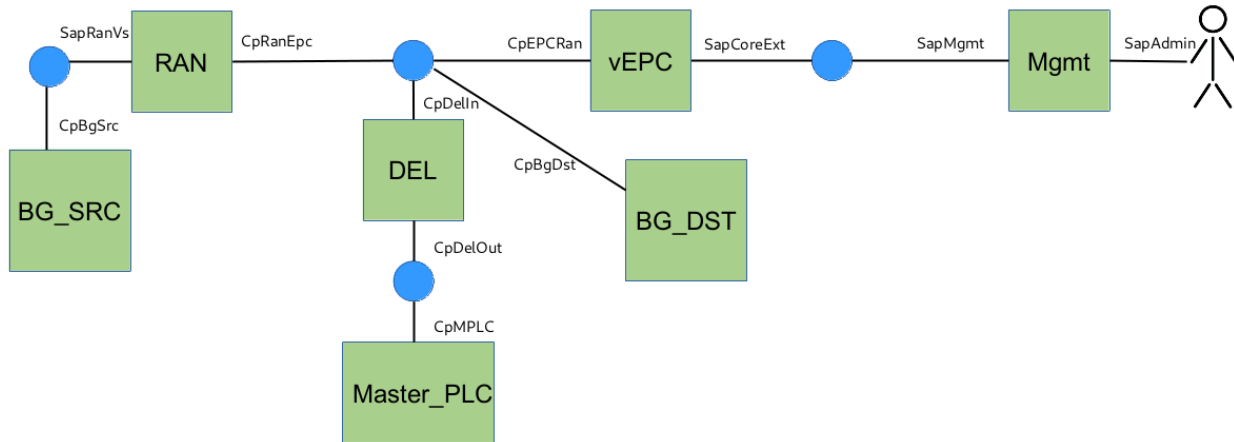


Figure 44: Graphical representation of an ExpB example.

Multiple combinations of the CB inside the topology of the blueprint are possible, as described in Section 3.2.1.2. Table 11 reports the fields composing an ExpB with their short description.

Table 11: Experiment Blueprint fields with relevant description

Field	Description
expBlueprintId	Unique Identifier for the ExpB.
version	A version number.
name	Name for the ExpB.
description	Short description of the ExpB.
vsBlueprintId	The identifier of the VSB this experiment was built from.
ctxBlueprintIds	List of the identifiers of the CBs this experiment was built from.
tcBlueprintIds	List of the identifiers of the TCBs this experiment includes. Test cases are parameter-based templates that describe the concrete actions to be executed during the experiment.
sites	The trial sites involved in this experiment. This determines the radio coverage for this experiment.
infrastructureMetrics	List of infrastructure metrics to be monitored for this experiment.
kpis	List of KPIs to be validated for this experiment. They can be both application and network KPIs.

5.2.4 Metrics

VSB, CB, and ExpB information models include a list of metrics. This field is used to advertise what kind of monitoring data the service, context, or experiment can provide once they are deployed. The formal representation of a metric is reported in Table 12. .

The three types of data collection for metrics are the following:

- CUMULATIVE: Increasing over time (e.g., CPU time)
- DELTA: Changing over time (e.g., bandwidth)
- GAUGE: Discrete items and fluctuating values (e.g., disk I/O, CPU utilization in %)

Metrics are partitioned in two sets: application metrics and infrastructure metrics. As the names suggest, the first set is composed of metrics related to the application or service domain, while the latest is composed of metrics available on the computing and network infrastructure. Metrics should be included in the blueprints as the approach to collect all the available metrics in the infrastructure is not sustainable in terms of data rates and data volumes.

Metrics are going to be discussed more thoroughly in D5.2 as WP5 is designing and developing monitoring and data collection tools.

Table 12: Metric fields with relevant description.

Field	Description
metricId	Unique Identifier for the metric.
name	Name for the CB.
metricCollectionType	Can be one of the following: CUMULATIVE, DELTA, GAUGE.
unit	The measurement unit of the metric (e.g., seconds).
interval	The sampling interval of this metric.
topic (application metric only)	Topic for publish/subscribe broker. Needed for collecting data.
iMetricType (infrastructure metric only)	The infrastructure metric type.

5.2.5 KPI

The ExpB includes a list of Key Performance Indicators (KPIs). A KPI is used to evaluate the performance of the experiment on one specific aspect. It can be constructed upon one simple metric or upon more metrics combined in a mathematical formula. An objective (or goal, or threshold) can be defined to validate the KPI. The latter is included in the Experiment Descriptor. Table 13 reports the fields composing an ExpB with their short description.

Table 13: KPI fields with relevant description

Field	Description
kpiId	Unique Identifier for the KPI.
kpiType	Enumerated identifying the type of KPI.
name	Name for the CB.
formula	Mathematical formula built on one or more metrics.
Unit	The measurement unit of the KPI (e.g., seconds).
Metrics	List of metrics involved in this KPI.
Interval	The sampling interval of this KPI.

KPI type and other implementation details are going to be discussed in D5.2 as WP5 is designing and developing KPI data collection and KPI validation.

5.2.6 Test Case Blueprint

The ExpB includes a list of Test Case Blueprints. A Test Case Blueprint is a customizable template that describes the actions to be performed during the execution of the experiment. Actions are collected in a script and user parameters and infrastructure variables can be used to customize the script execution. Table 14 reports the fields composing an ExpB with their short description.

Table 14: Test Case Blueprint fields with relevant description

Field	Description
scriptId	Unique Identifier for the script.
script	Source code of the script to be executed.
userParameters	Mapping of script variables to user parameters for the test customization.
infrastructureParameters	Mapping of script variables to infrastructure parameters for the dynamic test configuration based on the specific NS related to the experiment (e.g., IP addresses)

The script language is going to be described in detail in D5.2 as WP5 is taking care of the design and implementation of testing tools.

5.2.7 Experiment Descriptor (ExpD)

An Experiment Descriptor (ExpD) defines a specific experiment for the 5G EVE platform. All the parameters for the experiment are set and all the information to execute it are provided.

The ExpD is generated from an ExpB. Based on the desired VSB, the Experimenter can select one of the available ExpB for it. Then, the Experimenter is going to be asked to fill the parameters of the ExpB. The procedure is concluded with the generation and storage of the ExpD. Some metadata can be attached to the ExpD with information about the scheduling of the experiment. Table 15 reports the fields composing an ExpD with their short description.

Table 15: Experiment Descriptor fields with relevant description

Field	Description
expDescriptorId	Unique Identifier for the ExpD.
version	A version number.
name	Name for the ExpD.
description	Short description of the ExpD.
expBlueprintId	The identifier of the ExpB this experiment descriptor was built from.
vsDescriptorId	ID of the Vertical Service Descriptor describing the vertical service to be deployed to run this experiment. It provides the full specification of the parameters related to the vertical service, as defined in its originating blueprint.
ctxDescriptorIds	List of the IDs of the components associated to the experiment execution contexts (i.e. the Context Descriptors) to be deployed to run this experiment. They provide the full specification of the parameters related to the contexts, as defined in their originating blueprints.
testCaseDescriptorIds	List of the IDs of the test case descriptors defining the steps to run this experiment. They provide the full specification of the values of the variable parameters to be added in the experiment scripts, as defined in their originating blueprints.
kpiThresholds	List of thresholds for the KPIs defined in the originating experiment blueprint. They define the criteria to evaluate the results of the experiment.

5.2.8 VNFD, PNFD, and NSD

To describe network entities at a lower level of abstraction we use descriptors as defined by the ETSI NFV Industry Specification Group. The group has defined descriptors for all network entities in the NFV environment. The descriptors are intended to be used as information models in the architecture presented in MAN-001

deliverable [17]. A high level, technology agnostic description of the structure and composing fields of each entity is given in NFV-IFA (InterFaces and Architecture) deliverables.

VNFs and PNFs are considered as the atomic building blocks of network services. NFV-IFA 011 [18] provides a definition of VNF descriptors and specification about their packaging. A descriptor for a VNF (VNFD) includes information about the infrastructure resource requirements, constraints and dependencies, lifecycle events and operational behaviour. The VNF packaging and its standardization ensures a harmonized on-boarding process from providers.

NFV-IFA 014 [5] defines what information is included in a Network Service Descriptor (NSD). A network service (NS) is a composition of network functions with, optionally, some specifications on their connectivity. NSDs can be composed by means of nesting. Apart from information about composition and network topology, the NSD also includes information about its lifecycle management.

NFV-SOL (SOLutions) deliverables propose an implementation-ready solution for concepts defined in the NFV-IFA documents. The NFV-SOL 001 [2] deliverable defines a mapping of information elements in NFV-IFA 011 and NFV-IFA 014 to data types of TOSCA Simple Profile in YAML [19]. YAML files¹⁷ are provided to be imported into TOSCA processors and used to write custom NSD, VNFD, etc. To on-board an NSD on network orchestrators, the YAML files must be packaged with a proper format. NFV-SOL 007 [20] and NFV-SOL 004 [21] describe the NSD package and the VNF package, respectively. The documents include structure, format, contents and the naming conventions for the different files in each package. The format used for packaging is TOSCA CSAR.

Other deliverables describe the operations to manipulate the aforementioned descriptors. NFV-IFA 013 [22] defines the interfaces and exchanged information models (descriptors/entities/instances) between the Operations Support System (OSS)/Business Support System (BSS) and the Network Function Virtualisation Orchestrator (NFVO). The interface is called Os-Ma-nfvo. While the latter document specification is abstract, a RESTful protocol and data model implementation is given in NFV-SOL 005 [1], which implements all the operations defined in NFV-IFA 013. The API description is also provided in OpenAPI standard¹⁸. For more clarity, Table 16 with relevant correspondences between NFV-IFA and NFV-SOL deliverables.

Table 16: Mapping between NFV interfaces and architectures (NFV-IFA) and relevant solutions (NFV-SOL)

NFV-IFA	Short Description	NFV-SOL
IFA 011 (VNF Package, VNFD)	MANO; VNF Descriptor and Packaging Specification	SOL 001 SOL 004 (VNF Packaging)
IFA 013 (Os-Ma-nfvo)	MANO; Os-Ma-Nfvo reference point - Interface and Information Model Specification	SOL 005 (Os-Ma-nfvo)
IFA 014 (NSD)	MANO; Network Service Templates Specification	SOL 001 (VNFD, NSD) SOL 007

5.2.9 Blueprint Java library

We provide a Java library with a collection of Plain Old Java Objects that map all the blueprint information elements. The source code can be found in the 5G EVE Portal Catalogue project (see Annex E). Additionally, we also provide a command line tool for the validation of blueprint YAML files.

The following command

```
java -jar validator-VERSION.jar --type vsb ./vsb_asti_agv.yaml
```

¹⁷ <https://forge.etsi.org/gitlab/nfv/SOL001>

¹⁸ https://nfvwiki.etsi.org/index.php?title=API_specifications#OpenAPIs

Can be used to validate a VSB. Similarly, one can validate a CB or an ExpB.

5.3 Catalogue interfaces

The 5G EVE Catalogues expose north-bound interfaces based on the REST paradigm, which allows to manipulate the resources handled in the catalogues through Create – Read – Update – Delete (CRUD) actions. The REST API of the catalogues are implemented through the HTTP protocol, with the content of HTTP messages expressed in JSON or YAML languages (depending on the system configuration). The Swagger¹⁹ toolset have been used to document the REST API of the catalogues and the auto-generated documentation can be visualized on the browser at the following link:

http://<IP_ADDRESS>:<PORT>/swagger-ui.html

where IP_ADDRESS and PORT parameters refer to the IP address and the port where the catalogues are running. The following subsections 5.3.1 and 5.3.2 provides the details of the REST APIs exposed by the 5G EVE Portal Catalogue and the I/W framework Catalogue respectively.

5.3.1 5G EVE Portal Catalogue REST API

The REST APIs of the Portal Catalogue allows to execute CRUD actions on the following information elements:

- Vertical Service Blueprints;
- Context Blueprints;
- Test Case Blueprints²⁰;
- Experiment Blueprints and Experiment Descriptors.

The list of the Portal Catalogue REST APIs is defined in Table 17. The following tables specifies the single REST APIs.

Table 17: 5G-EVE Portal Catalogue REST API

HTTP method	URI	Description
GET	/portal/catalogue/ctxblueprint<?[filter_parameters]>	Get the list of the IDs of all the context blueprints matching the filter criteria provided in the URI query parameters. If the filter is not specified, the IDs of all the existing context blueprints are returned.
POST	/portal/catalogue/ctxblueprint	Creates and onboard a new context blueprint. The context blueprint is onboarded together with the associated NSD (if present in the request) and translation rules.
GET	/portal/catalogue/ctxblueprint/<id>	Get the details of the context blueprint with ID <id>.
DELETE	/portal/catalogue/ctxblueprint/<id>	Remove the context blueprint with ID <id>.
GET	/portal/catalogue/vsblueprint<?[filter_parameters]>	Get the list of the IDs of all the vertical service blueprints matching the filter criteria provided in the URI query parameters. If the filter is not specified, the IDs of all the existing vertical service blueprints are returned.
POST	/portal/catalogue/vsblueprint	Creates and onboard a new vertical service blueprint. The vertical service blueprint is

¹⁹ <https://swagger.io/>

²⁰ Test case blueprints are currently under definition in WP5 and they are not considered in this documentation.

		onboarded together with the associated NSD (if present in the request) and translation rules.
GET	/portal/catalogue/vsblueprint/<id>	Get the details of the vertical service blueprint with ID <id>.
DELETE	/portal/catalogue/vsblueprint/<id>	Remove the vertical service blueprint with ID <id>.
GET	/portal/catalogue/expblueprint<?[filter_parameters]>	Get the list of the IDs of all the experiment blueprints matching the filter criteria provided in the URI query parameters. If the filter is not specified, the IDs of all the existing experiment blueprints are returned.
POST	/portal/catalogue/expblueprint	Creates and onboard a new experiment blueprint. The experiment blueprint is onboarded together with the associated NSD (if present in the request) and translation rules.
GET	/portal/catalogue/expblueprint/<id>	Get the details of the experiment blueprint with ID <id>.
DELETE	/portal/catalogue/expblueprint/<id>	Remove the experiment blueprint with ID <id>.

Table 18: REST API – GET /portal/catalogue/ctxblueprint<?[filter_parameters]>

GET /portal/catalogue/ctxblueprint<?[filter_parameters]>			
URI Query Parameters	ContextType	String	Type of context execution
	Site	String	5G EVE site where the context execution if available
	VsbId	String	ID of the VSB which the queried context blueprints are compatible to.
Response body	ContextID	List<String>	List of IDs of the contexts matching the filter provided in the URI Query parameters.
Successful response HTTP code: 200 OK			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 500 INTERNAL ERROR			

Table 19: REST API – POST /portal/catalogue/ctxblueprint

POST /portal/catalogue/ctxblueprint			
Request body	ctxBlueprint	ContextBlueprint (see format in sect. 5.2.2)	Content of the context blueprint to be onboarded.
	nsds	List<Nsd>	NSD of the Network Services required to deploy the context execution elements.
	translationRules	List<TranslationRule>	Rules regulating the translation between a context descriptor based on the given blueprint and the associated network service.
Response body	n/a	n/a	n/a
Successful response HTTP code: 201 CREATED			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 409 CONFLICT, 500 INTERNAL ERROR			

Table 20: REST API – GET /portal/catalogue/ctxblueprint/<id>

GET /portal/catalogue/ctxblueprint/<id>			
URI Query Parameters	id	String	Unique ID of the queried context blueprint.
Response body	blueprintId	String	Unique ID of the context blueprint.
	version	String	Version of the context blueprint.
	name	String	Name of the context blueprint.
	ctxBlueprint	ContextBlueprint (see format in sect. 5.2.2)	Content of the context blueprint.
	onBoardedNsdInfoId	List<String>	IDs of the onboarded NSD associated to the given context blueprint.
	activeCtxdId	List<String>	IDs of the context descriptors associated to the given blueprint.
Successful response HTTP code: 200 OK			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 404 NOT FOUND, 500 INTERNAL ERROR			

Table 21: REST API – DELETE /portal/catalogue/ctxblueprint/<id>

DELETE /portal/catalogue/ctxblueprint/<id>			
URI Query Parameters	id	String	Unique ID of the context blueprint to be deleted.
Response body	n/a	n/a	n/a
Successful response HTTP code: 204 NO CONTENT			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 404 NOT FOUND, 500 INTERNAL ERROR			

Table 22: REST API – GET /portal/catalogue/vsblueprint<?[filter_parameters]>

GET /portal/catalogue/vsblueprint<?[filter_parameters]>			
URI Query Parameters	Site	String	5G EVE site where the vertical service can be deployed
Response body	VsbID	List<String>	List of IDs of the vertical service blueprints matching the filter provided in the URI Query parameters.
Successful response HTTP code: 200 OK			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 500 INTERNAL ERROR			

Table 23: REST API – POST /portal/catalogue/vsblueprint

POST /portal/catalogue/vsblueprint			
Request body	vsBlueprint	VerticalServiceBlueprint (see format in sect. 5.2.1)	Content of the vertical service blueprint to be onboarded.
	nsds	List<Nsd>	NSD of the Network Services required to deploy the vertical service elements.
	translationRules	List<TranslationRule>	Rules regulating the translation between a vertical service descriptor based on the given blueprint and the associated network service.
Response body	n/a	n/a	n/a
Successful response HTTP code: 201 CREATED			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 409 CONFLICT, 500 INTERNAL ERROR			

Table 24: REST API – GET /portal/catalogue/vsblueprint/<id>

GET /portal/catalogue/vsblueprint/<id>			
URI Query Parameters	id	String	Unique ID of the queried vertical service blueprint.
Response body	blueprintId	String	Unique ID of the vertical service blueprint.
	version	String	Version of the vertical service blueprint.
	name	String	Name of the vertical service blueprint.
	vsBlueprint	VerticalServiceBlueprint (see format in sect. 5.2.1)	Content of the vertical service blueprint.
	onBoardedNsdInfoId	List<String>	IDs of the onboarded NSDs associated to the given vertical service blueprint.
	activeVsdId	List<String>	IDs of the vertical service descriptors associated to the given blueprint.
Successful response HTTP code: 200 OK			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 404 NOT FOUND, 500 INTERNAL ERROR			

Table 25: REST API – DELETE /portal/catalogue/vsblueprint/<id>

DELETE /portal/catalogue/vsblueprint/<id>			
URI Query Parameters	id	String	Unique ID of the vertical service blueprint to be deleted.
Response body	n/a	n/a	n/a
Successful response HTTP code: 204 NO CONTENT			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 404 NOT FOUND, 500 INTERNAL ERROR			

Table 26: REST API – GET /portal/catalogue/expblueprint<?[filter_parameters]>

GET /portal/catalogue/expblueprint<?[filter_parameters]>			
URI Query Parameters	Site	String	5G EVE site where the experiment can be deployed
	VsbId	String	ID of the VSB which the queried experiments blueprints are associated to.
Response body	ExpbID	List<String>	List of IDs of the experiment blueprints matching the filter provided in the URI Query parameters.
Successful response HTTP code: 200 OK			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 500 INTERNAL ERROR			

Table 27: REST API – POST /portal/catalogue/expblueprint

POST /portal/catalogue/expblueprint			
Request body	experimentBlueprint	ExperimentBlueprint (see format in sect. 5.2.3)	Content of the experiment blueprint to be onboarded.
	nsds	List<Nsd>	NSD of the Network Services required to deploy the experiment elements.
	translationRules	List<TranslationRule>	Rules regulating the translation between an experiment descriptor based on the given blueprint and the associated network service.
Response body	n/a	n/a	n/a
Successful response HTTP code: 201 CREATED			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 409 CONFLICT, 500 INTERNAL ERROR			

Table 28: REST API – GET /portal/catalogue/expblueprint/<id>

GET /portal/catalogue/expblueprint/<id>			
URI Query Parameters	id	String	Unique ID of the queried experiment blueprint.
Response body	blueprintId	String	Unique ID of the experiment blueprint.
	version	String	Version of the experiment blueprint.
	name	String	Name of the experiment blueprint.
	experimentBlueprint	ExperimentBlueprint (see format in sect. 5.2.3)	Content of the experiment blueprint.
	onBoardedNsdInfoId	List<String>	IDs of the onboarded NSDs associated to the given experiment blueprint.
	activeExpdId	List<String>	IDs of the experiment descriptors associated to the given blueprint.

Successful response HTTP code: 200 OK
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 404 NOT FOUND, 500 INTERNAL ERROR

Table 29: REST API – DELETE /portal/catalogue/expblueprint/<id>

DELETE /portal/catalogue/expblueprint/<id>			
URI Query Parameters	id	String	Unique ID of the experiment blueprint to be deleted.
Response body	n/a	n/a	n/a
Successful response HTTP code: 204 NO CONTENT			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 404 NOT FOUND, 500 INTERNAL ERROR			

Table 30: REST API – GET /portal/catalogue/expdescriptor<?[filter_parameters]>

GET /portal/catalogue/expdescriptor<?[filter_parameters]>			
URI Query Parameters	Site	String	5G EVE site where the experiment can be deployed
	ExpbId	String	ID of the ExpB which the queried experiment descriptors are associated to.
Response body	expdID	List<String>	List of IDs of the experiment descriptors matching the filter provided in the URI Query parameters.
Successful response HTTP code: 200 OK			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 500 INTERNAL ERROR			

Table 31: REST API – POST /portal/catalogue/expdescriptor

POST /portal/catalogue/expdescriptor			
Request body	experimentDescriptor	ExperimentDescriptor (see format in sect. 5.2.4)	Content of the experiment descriptor to be onboarded.
Response body	n/a	n/a	n/a
Successful response HTTP code: 201 CREATED			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 409 CONFLICT, 500 INTERNAL ERROR			

Table 32: REST API – GET /portal/catalogue/expdescriptor/<id>

GET /portal/catalogue/expdescriptor/<id>			
URI Query Parameters	id	String	Unique ID of the queried experiment blueprint.
Response body	expDescriptor	ExperimentDescriptor (see format in sect. 5.2.4)	Content of the experiment descriptor.
Successful response HTTP code: 200 OK			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 404 NOT FOUND, 500 INTERNAL ERROR			

Table 33: REST API – DELETE /portal/catalogue/expdescriptor/<id>

DELETE /portal/catalogue/expdescriptor/<id>			
URI Query Parameters	id	String	Unique ID of the experiment descriptor to be deleted.
Response body	n/a	n/a	n/a
Successful response HTTP code: 204 NO CONTENT			
Error response HTTP code: 400 BAD REQUEST, 403 FORBIDDEN, 404 NOT FOUND, 500 INTERNAL ERROR			

5.3.2 5G EVE I/W framework Catalogue REST API

The I/W framework Catalogue offers a REST API that is fully compliant with the ETSI NFV SOL 005 [1] specification and, in particular, implements a subset of its NSD Management Interface (for NSDs and PNFs) and VNF Package Management Interface (for VNFs). NSDs, PNFs and VNFs are expressed in TOSCA language, following the format defined in ETSI NFV SOL 001 [2].

The list of APIs supported by the I/W framework catalogue is reported in Table 34. A full definition of protocol messages and HTTP response codes is available in the standard [1].

Table 34: I/W framework Catalogue REST APIs

ETSI NFV SOL 005 interface	Resource	URI	HTTP Method	Description
NSD Management	NS Descriptors	/ns_descriptors	GET	Query of NSDs.
			POST	Creation of a new NSD.
	Individual NS Descriptor	/ns_descriptors/<nsdInfoId>	GET	Query information about an NSD.
			PATCH	Modify the status of an NSD.
			DELETE	Delete an NSD.
	NSD Content	/ns_descriptors/<nsInfoId>/nsd_content	GET	Retrieve the content of an NSD.
			PUT	Upload the content of an NSD.
	PNF Descriptors	/pnf_descriptors	GET	Query of PNFs.
	Individual PNF Descriptor	/pnf_descriptors/<pnfdInfoId>	GET	Query information about a PNF.
	PNFD content	/pnf_descriptors/<pnfdInfoId>/pnfd_content	GET	Retrieve the content of a PNF.
	Subscriptions	/subscriptions	POST	Subscribe to receive notifications about PNF and NSD onboarding or changes.
			GET	Query of subscriptions for NSDs and PNFs.
	Individual subscriptions	/subscriptions/<subscriptionId>	GET	Retrieve information about a subscription for NSDs or PNFs.
DELETE			Remove a subscription for NSDs or PNFs.	
Notification endpoint	/notifications	POST	Receive notifications from 5G EVE site facilities catalogues about	

				onboarding or changes of NSDs or PNFDs.
VNF Package Management	VNF Packages	/vnf_packages	GET	Query VNF Packages information.
	Individual VNF Package	/vnf_packages/<vnfPkgId>	GET	Retrieve information about a VNF package.
	VNFD	/vnf_packages/<vnfPkgId>/vnfd	GET	Retrieve a VNFD within a VNF package.
	VNF Package content	/vnf_packages/<vnfPkgId>/package_content	GET	Retrieve a VNF package.
	Subscription	/subscriptions	POST	Subscribe to receive notifications about VNF Packages onboarding or changes.
			GET	Query of subscriptions for VNF Packages.
	Individual Subscription	/subscriptions/<subscriptionId>	GET	Retrieve information about a subscription for VNF Packages.
			DELETE	Remove a subscription for VNF Packages.
Notification endpoint	/notifications	POST	Receive notifications from 5G EVE site facilities catalogues about onboarding or changes of VNF Packages.	

5.4 Software design

The 5G EVE Portal and I/W framework catalogues are both developed in Java, as Maven projects, and adopt PostgreSQL as backend database. The 5G EVE Portal catalogue is developed as an extension of the Vertical Service Blueprint catalogue embedded in the 5G-TRANSFORMER Vertical Slicer²¹ [15], while the 5G EVE I/W framework catalogue is developed as an extension of the 5G Apps and Services Catalogue²² developed in the context of the 5G-MEDIA project [23].

The high-level software design of the **5G EVE Portal catalogue** is represented in Figure 45, highlighting in red the extensions needed to evolve the 5G-TRANSFORMER Vertical Slicer catalogue for supporting the additional information models and functionalities required in 5G EVE. The core of the catalogue is implemented in the *Catalogue Service* component, a singleton service that interacts with the database to handle all the requests for queries, creation, deletion and modification of blueprints and descriptors. The interaction with the database is mediated through the Java Persistency API (JPA), acting on a number of *JPA repositories*. These repositories model the tables on the PostgreSQL backend database, where the elements of the 5G EVE blueprints and descriptors are stored. The Catalogue Service implements a Java interface (the Catalogue Service Interface in the figure) that provides all the CRUD methods to operate on blueprints and descriptors, jointly with the related NFV elements (e.g. NSDs, VNF packages, PNFDs). All these entities are defined through dedicated libraries that implement the associated *Information Models*, where for NFV elements the standard ETSI NFV IFA models are used ([18], [5]). In 5G EVE, the Information Model libraries are extended with the classes defining experiment and context blueprints and descriptors.

On the north-bound of the Catalogue Service, the *REST controller* component implements the REST APIs exposed to the user, usually through the mediation of the graphical interface. In 5G EVE portal, the GUI of the catalogue is implemented through the *browse and look-up tool*, which offers a complete view of service, context

²¹ <https://github.com/5g-transformer/5gt-vs>

²² <https://github.com/nextworks-it/5g-catalogue>

and experiment entities. Moreover, additional mechanisms for authentication and authorization enable to regulate the access to the different types of information based on the profile associated to the 5G EVE users.

Internally, the Catalogue Service implements the logic to coordinate the interaction with the I/W framework catalogue and keep the synchronization between the blueprints and descriptors handled at the portal and I/W framework associated NFV descriptors handled at the I/W framework. Moreover, the workflows regulating the interactions with between Portal and I/W framework catalogues can be regulated through configurable policies, handled at the *Policy Management* component. The actual communication with the I/W framework catalogue is managed through a dedicated driver, which implements the client side of the NFV catalogue services for a standard ETSI NFVO. In 5G EVE a new driver is introduced to translate between the ETSI NFV IFA models and interfaces ([18], [5]), adopted internally at the Portal Catalogue, and the ETSI NFV SOL models and interfaces [2] adopted at the I/W framework and thus implemented in the I/W framework Catalogue.

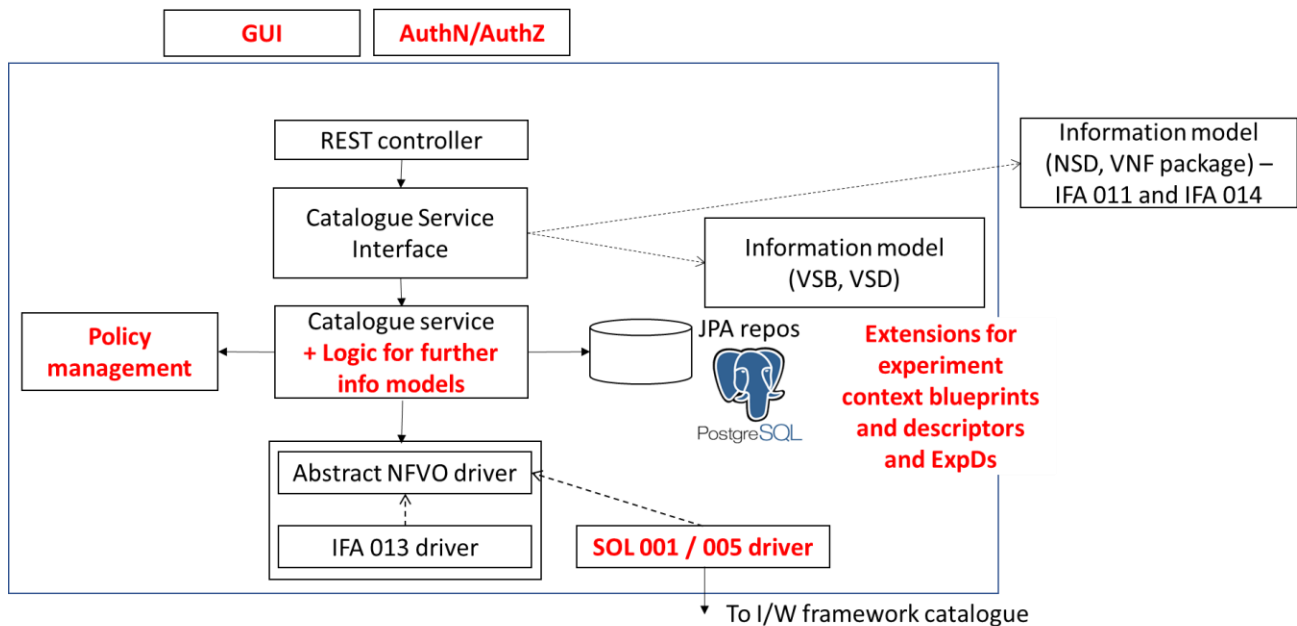


Figure 45: High-level software design of 5G EVE Portal catalogue

The structure of the 5G EVE Portal Catalogue software is represented in Figure 46. The software is composed by three different Java projects, all of them using Apache Maven as building automation tool: (i) the Catalogue Lib with the information model, (ii) the Catalogue Service Interface with the messages and primitives exposed at the interfaces of the catalogue, and (iii) the Catalogue Service with the internal implementation of the catalogue functionalities. An additional Web GUI implements the browse and look-up tool and interacts with the core of the system via REST API. The rationale behind this split is to completely decouple the development of the information models, the definition of the operations provided by the system, the logic to implement such operations and the GUI.

The definition of the Information Models and the definition of the operations supported and the messages used to interact with the 5G EVE Portal Catalogue are implemented in the Catalogue Libs and in the Catalogue Service Interface projects respectively. These two projects are packaged as JAR files and they can be included as dependencies by any other software component that need to operate on the information model elements (e.g. the Experiment Lifecycle Manager, introduced in section 3.2.2, needs to process ExpD elements) or to interact with the catalogue itself, guaranteeing a smooth interoperability.

The Catalogue Lib project contains java classes representing the information models for Vertical Service Blueprints and Vertical Service Descriptors, Context Blueprints and Context Descriptors, Experiment Blueprints and

Experiment Descriptors. External libraries²³ are also integrated to include the definition of ETSI NFV information models for NSDs, VNF packages and PNFs, according to the ETSI NFV IFA and SOL definition. The Catalogue Service Interface defines the messages and the operations defined on the 5G EVE blueprints and descriptors, thus importing the Catalogue Lib as dependency. The VSBs/VSDs are mostly based on the ones developed in 5G-TRANSFORMER, with minor extensions. On the other hand, CBs/CDs and ExpBs/ExpDs are original contributions from the 5G EVE project. The Catalogue Service project implements the software application of the 5G EVE Portal Catalogue, and it is based on the Spring boot framework. In detail, it implements the core of the catalogue logic with the interaction towards the I/W framework, the JPA repositories (based on the information model classes of the Catalogue Lib project) and the REST controller offering the REST APIs towards the Web GUI. As shown in the picture, the REST Controller interacts with an external KeyCloak²⁴ server to handle authentication and authorization. Every REST request arriving to the controller is authenticated and authorized using the Open Id Connect protocol and KeyCloak at the backend.

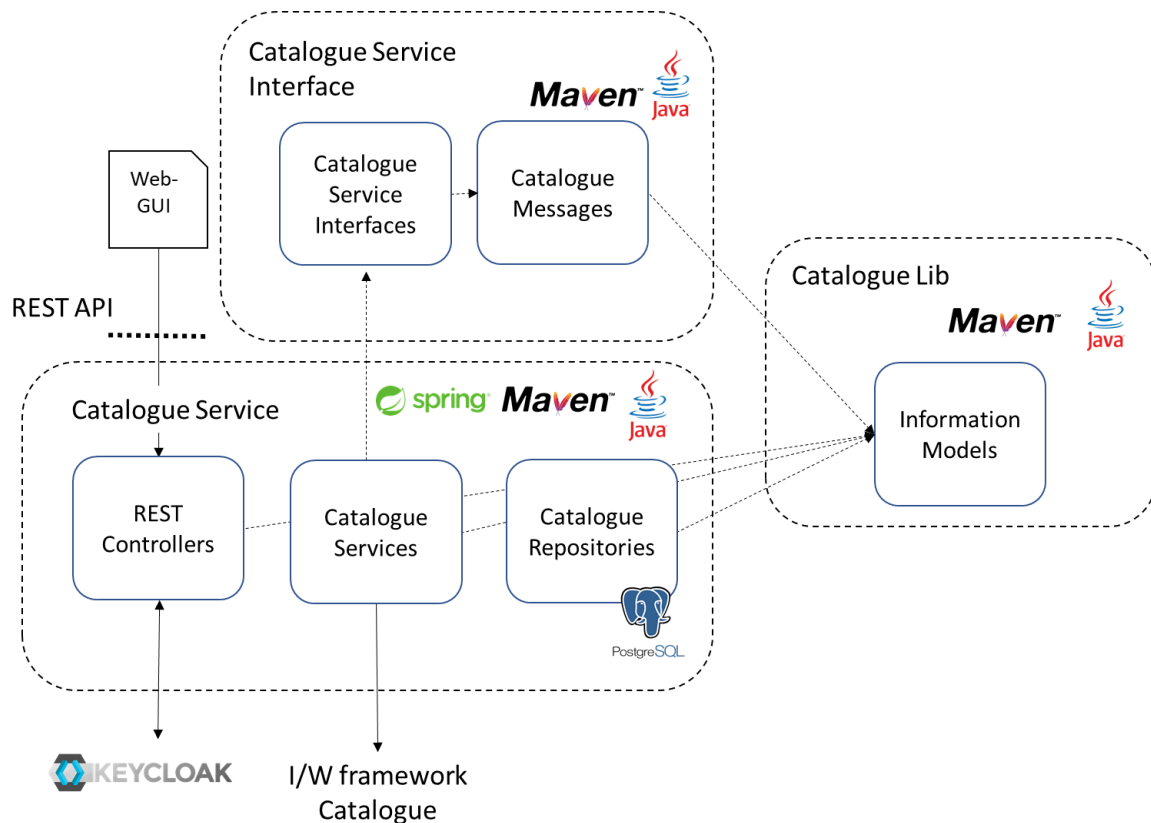


Figure 46: 5G EVE Portal catalogue: software structure

The high-level software design of the **5G EVE I/W framework catalogue** is represented in Figure 47, where again the extensions from the original 5G Apps and Services Catalogue are highlighted in red. The overall principles of the 5G EVE I/W framework catalogue are the same of the Portal catalogue, with information models for ETSI NFV descriptor defined in a dedicated library, the core of the system implementing the logic of the CRUD operations supported by the catalogue and modelled through a Java interface and a REST controller to provide the REST APIs at the north-bound interface. A Web GUI offers the possibility to easily interact with the system via web browser, while authentication and authorization mechanisms guarantee the required access control based on user profiles. The information model of NSDs, VNF packages and PNFs are based on the TOSCA language [19] and they are compliant with the ETSI NFV SOL 001 specification [2], with minor extensions related to 5G EVE requirements. The REST API of the I/W framework catalogue is based on the

²³ <https://github.com/nextworks-it/nfv-ifa-libs> , <https://github.com/nextworks-it/nfv-sol-libs>

²⁴ <https://www.keycloak.org/>

ETSI NFV SOL 005 [1] Open API, implementing the subset of functionalities related to the management of the descriptors. It should be noted that the I/W framework catalogue adopts not only a backend database (also based on PostgreSQL), but it also stores the files included in NSD and VNF packages in the filesystem, adopting an indexing structure that facilitates the retrieval of the information.

The management of the interaction with the multiple NFVOs deployed at the 5G EVE sites requires an additional logic, which has been implemented in the Catalogue Service component to handle the synchronization of the descriptors onboarding procedures with the distributed 5G EVE facilities. The communication with the different NFVOs is handled through a number of site-dependent *NFVO drivers*, customized according to the specific interfaces and information models exposed by each site. Due to the asynchronous nature of the interactions between the I/W framework catalogue and the site-specific NFVOs, a Kafka²⁵-based message bus is adopted to regulate the exchange of concurrent messages between the Catalogue Service component and the different drivers. This approach allows to identify the expected consumers of the messages delivered through the bus adopting an efficient subscription-notification paradigm.

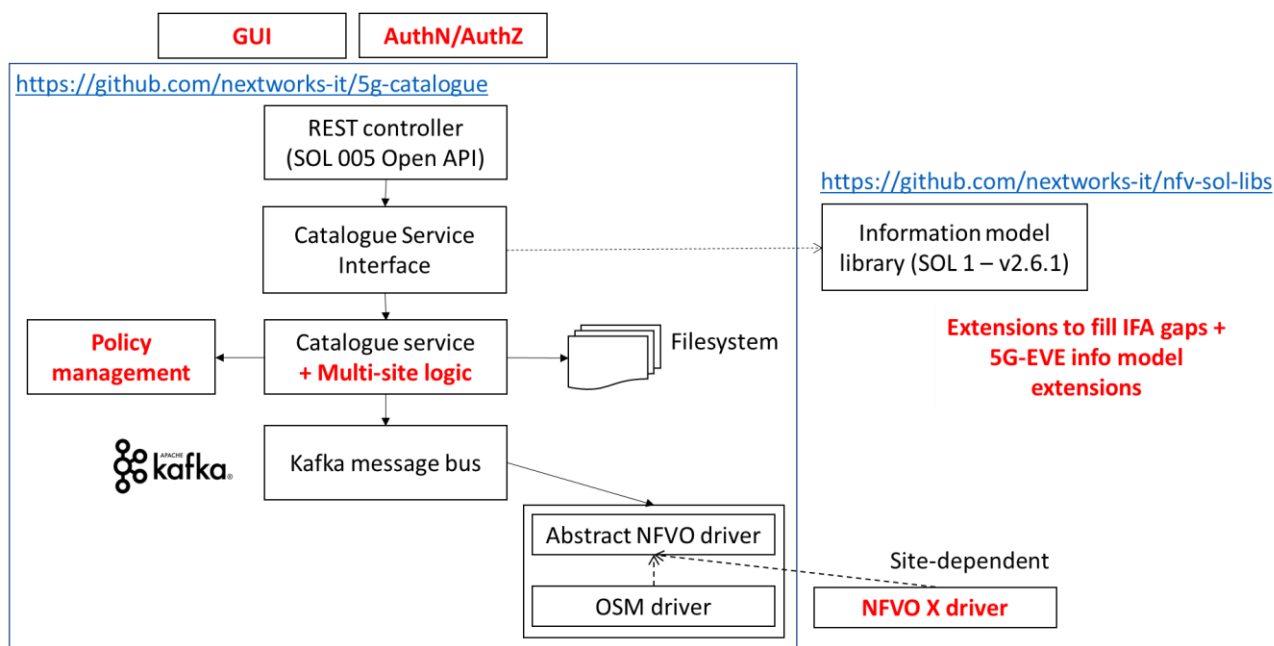


Figure 47: High-level software design of 5G EVE I/W framework catalogue

The structure of the 5G EVE I/W framework Catalogue software is represented in Figure 48. As for the portal catalogue, the software is composed by different Java projects, adopting Apache Maven as building automation tool, in order to decouple the definition of information models and interfaces from the internal implementation of the catalogue logic. In this case both the information model and the catalogue interfaces from ETSI NFV SOL specification are integrated in a single Java project, since they are not foreseen as needed as disjoint dependencies for other software components. As for the Portal Catalogue, the Web GUI is implemented as an external entity that interacts with the core system via REST APIs. The user profile-based authorization is again managed using KeyCloak as backend server. The usage of the same Identity and Access Management system for both catalogues allows to manage the configuration of users and their roles and groups in a single entity.

²⁵ <https://kafka.apache.org/>

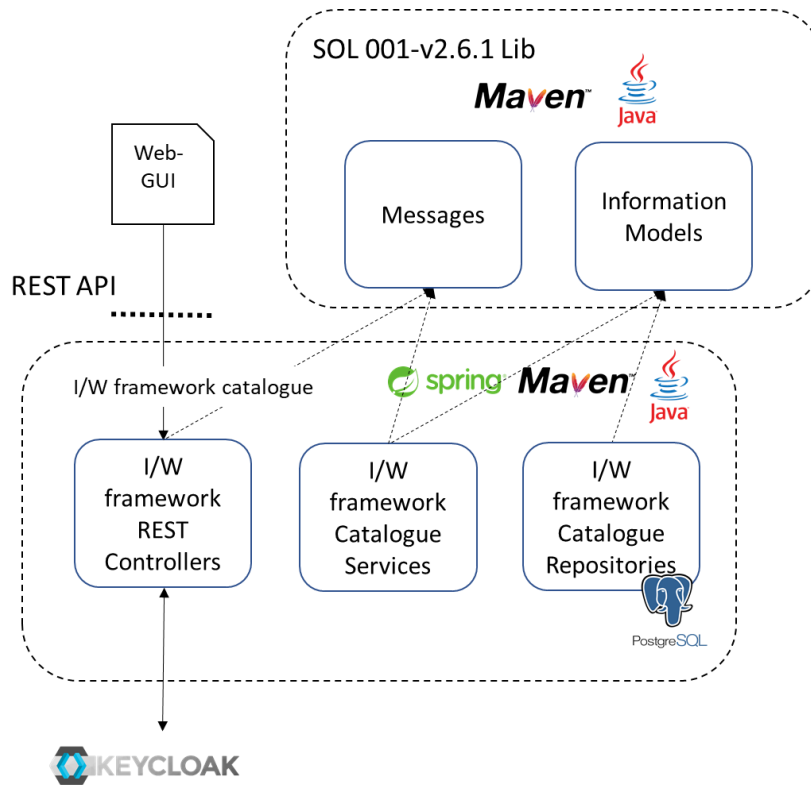


Figure 48: 5G EVE I/W framework catalogue: software structure

5.4.1 Installation guide

The 5G EVE Portal Catalogue and the I/W framework Catalogue software have been tested over machines with Ubuntu Server 16.04 as operative system. A number of software dependencies are required, as specified in Table 35 along with the minimum version, in order to install and execute this software. For the installation of these software bundles we remand to the respective official web pages.

Table 35: 5G EVE Portal and I/W framework Catalogue software dependencies

Name	Recommended version
Java JDK	1.8
Git	2.7.4
Apache2	2.4.18
Apache Maven	3.3.9
Kafka	2.3.0
PostgreSQL	10.9

It is important to highlight that both the I/W framework and the Portal Catalogue use databases that should be created inside the PostgreSQL server together with the correspondent users.

The Portal Catalogue and the I/W framework Catalogue also depend on the NFV MANO IFA and the NFV MANO SOL libraries, implementing the information models for VNFs, PNFs and NSDs in ETSI NFV IFA and SOL format respectively. These two open source libraries (released with Apache 2.0 license) are available as Maven projects from two different repositories as indicated in Table 36 and they can be downloaded using the git tool through the command:

```
git clone <repository link>
```


Once downloaded, the two libraries must be installed in the system running the install commands specified in the table from their main folder.

Table 36: 5G EVE Portal and I/W framework Catalogue library dependencies

Name	Repository	Install commands
NFV MANO IFA	https://github.com/next-works-it/nfv-ifa-libs/tree/feat-librefactor	mvn clean install
NFV MANO SOL	https://github.com/next-works-it/nfv-sol-libs	./install_nfv_sol_libs.sh

Once all the software pre-requisites and the dependencies have been installed, the 5G EVE Portal and I/W framework Catalogues can be downloaded and installed in the system. The 5G EVE Portal Catalogue and the I/W framework Catalogue are available at the git repositories shown in Table 37. Each repository includes the details of the installation and execution steps required for each component.

It is important to stress that each of these bundles should be configured using a Java properties file based on the `application.properties` files available in the `/src/main/resources` folder of each project. These files contain, among others, the parameters related for the PostgreSQL database configuration mentioned before.

Table 37: 5G EVE Portal and I/W framework Catalogue repositories

Name	Repository
5G Eve Portal Catalogue	https://github.com/next-works-it/slicer-catalogue/tree/5geve-release
I/W framework Catalogue	https://github.com/next-works-it/5g-catalogue

5.4.2 Licenses and dependencies

The 5G EVE Portal and I/W framework catalogues are both released as open source software, under the Apache 2.0 license. They are written in Java, using the Spring framework, and they adopt external open source libraries and software tools to implement functionalities like persistency, exchange of asynchronous messages and RBAC mechanisms. They use PostgreSQL as backend SQL database, Kafka as message bus and KeyCloak as RBAC tool.

Table 38 reports the licenses of the major external software libraries and tools that have been integrated in the 5G EVE Portal and I/W framework catalogues.

Table 38: Software licenses of 5G EVE catalogues

5G EVE catalogues implementation		
Component	Description	Software License
Portal catalogue GUI	Web GUI of the 5G EVE Portal Catalogue, implementing the browse and lookup tools to navigate vertical service, context and experiment blueprints and descriptors.	Apache 2.0 license
Portal catalogue backend	Core implementation of the 5G EVE Portal Catalogue.	Apache 2.0 license
I/W framework catalogue GUI	Web GUI of the 5G EVE I/W framework Catalogue, implementing the browse and lookup tools	Apache 2.0 license

	to navigate NFV descriptors (NSDs, VNFDs and PNFDs).	
I/W framework catalogue backend	Core implementation of the 5G EVE I/W framework Catalogue.	Apache 2.0 license
5G EVE catalogues dependencies		
Component	Description	Software License
Spring framework	Application framework used for the development of the 5G EVE catalogues core	Apache 2.0 license
PostgreSQL	Backend SQL database used for the persistency of 5G EVE catalogues data	PostgreSQL license
Apache Kafka	Distributed streaming platform adopted as message bus to exchange asynchronous messages among the internal modules of the I/W framework catalogue.	Apache 2.0 license
Apache HTTP Server	HTTP server running the 5G EVE catalogues web GUI.	Apache 2.0 license
KeyCloak	Identity and Access Management system for handling authentication and authorization of 5G EVE users when accessing 5G EVE catalogues.	Apache 2.0 license

6 Conclusions

This deliverable has introduced the concept of 5G EVE experiments, providing details about their structure, their lifecycle and the role of the different categories of 5G EVE actors in the different phases of an experiment. As such, it provides an initial guideline about how verticals can interact with the 5G EVE platform through the 5G EVE Portal and how they can validate their own vertical services in customizable 5G environments.

This deliverable has also provided the first release of a set of 5G EVE experimentation tools that compose the 5G EVE Portal and helps the 5G EVE users in the process of defining, customizing and executing their experiments over the 5G EVE platform. The following modules have been made available and documented in this report:

- The experiment monitoring tools, for the collection, storage and visualization of experiments' metrics, KPIs and results;
- The browse and look-up tool, to help experimenters in the navigation of experiment blueprints and descriptors available in the 5G EVE library, using a web-based graphical interface;
- The intent-based tool, to facilitate experimenters in the definition and customization of their experiments, using the natural language or guided procedures;
- The ticketing tool, to facilitate the interaction between 5G EVE users and the 5G EVE support teams.
- The 5G EVE Portal Catalogue, for the organized storage and controlled access of all the different kinds of blueprints and descriptors associated to the definition of 5G EVE experiments.

All these components will be further refined during the rest of the project and they will be properly integrated in the overall 5G EVE Portal in the context of task T4.2. Moreover, they will be continuously evolved and improved, considering the feedbacks that will be received from the 5G EVE users in terms of usability when the system will become publicly available.

Acknowledgment

This project has received funding from the EU H2020 research and innovation programme under Grant Agreement No. 815074.

Annex A – Service Blueprints Design and Implementation Manual

In this annex we present the implementation manual for experiment developers to design a service Blueprint by using the ASTI use case as a working example.

Step I: Design of the service sequence

The first step before designing a service blueprint is to come up with the service sequence with all the connection endpoints well-defined. The service sequence describes how all the service network functions interact with each other and the order in which they interact. The service sequence can be in form of a VNF Forwarding Graph (VNFFG) or VNFFG in textual notation. For example, the service sequence for the ASTI AGV use case is provided below.

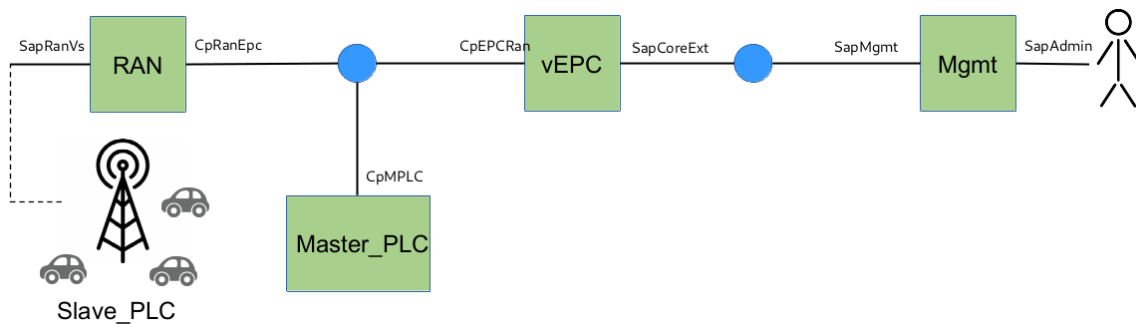


Figure 49: Service sequence for the ASTI AGV use case

Step II: Generate the Service Blueprint in YAML format

- i) Create an empty YAML file and save it with your preferred name of choice and an extension .yaml
- ii) Next, start populating the new YAML file with the service blueprint parameters. These parameters are presented in Table I below (5G-TRANSFORMER, D3.1);

Table 39: Example of service blueprint parameters

Name	Type	Required	Description	Example (ASTI Use case)
blueprintId	string	true	A unique identifier for the blueprint	blueprintId: '513e154e-4bee-45f1-9a11-40716dbbb684'
version	string	true	Identifies the version of the blueprint	version: '1.0'
name	string	true	Name of the blueprint	name: 'ASTI AGV control and automation'
description	string	true	Short description of the blueprint	description: 'Blueprint for AGVs using 5G network for Automation and Control purposes'
parameters	array	true	List of parameters to be used in the vertical service. For each parameter; provide the id, name,	parameters: - parameterId: 'number_of_AGVs'

			type, description and the field in the blueprint where it will be used	<pre> parameterName: 'Number of AGVs' parameterType: 'number' parameterDescription: 'Number of AGVs (Mandatory)' applicabilityField: 'autonomous_vehicle' - parameterId: 'location_of_AGVs' parameterName: 'Location of the AGVs' parameterType: 'geolocation' parameterDescription: 'Location of AGVs (Mandatory)' applicabilityField: 'autonomous_vehicle' ... </pre>
atomicComponents	array	true	List of atomic functional components (i.e., network functions and virtual applications in general) needed to implement the vertical service. For each component, provide the dimensioning in terms of number of users/ session capacity, scaling aspects, bandwidth and parameters to be monitored.	<p>Using the figure above, the atomic components are; RAN, Master_PLC, vEPC and the management (Mgmt) node. In YAML these components can be represented as:</p> <pre> atomicComponents: - componentId: 'RAN' serversNumber: 1 imagesUrls: [] endPointsIds: ['SapRANVs', 'CpRANEpc'] lifecycleOperations: {} - componentId: 'Master_PLC' serversNumber: 1 imagesUrls: ['master_plc_image'] endPointsIds: ['CpMPLC'] lifecycleOperations: instantiate: 'scripts/instantiate' terminate: 'scripts/terminate' update: 'scripts/update' heal: 'scripts/heal' ... </pre>
endPoints	array	true	Specifies the service connection endpoints. The	Once again using the figure above:

			<p>endpoints can be internal or external. For each endpoint, specify its id and whether it's external, management and/or a ran connection endpoint.</p>	<ul style="list-style-type: none"> The external connection points are: ['SapRANVs', 'SapCoreExt', Sap_Mgmt] While the internal connections points are: ['CpRANEpc', 'CpEPCran', 'cpMPLC', 'cpEPCran'] <p>Definition of all the service connection endpoints in the blueprint following the schema below: endPoints:</p> <ul style="list-style-type: none"> - endPointId: 'SapRANVs' external: true management: false ranConnection: true - endPointId: 'CpRANEpc' external: false management: false ranConnection: false <p>...</p>
connectivityServices	array	true	<p>Specification of the type of connectivity service to be used for the vertical service virtual links (e.g., L2VPN, L3VPN etc.) and properties (QoS and/ flow management, protection, restoration etc.) that can be necessary for service orchestration and instantiation. For each connectivity service, provide the endpoint ids, whether it's external, and connectivity properties if any</p>	<p>connectivityServices:</p> <ul style="list-style-type: none"> - endPointIds: ['CpRanEpc', 'CpMPLC', 'CpEpcRan'] external: false - endPointIds: ['SapCoreExt', 'SapMgmt'] external: false
serviceSequence	array	true	<p>Description of how the atomic functional components should interact. This can be in form of VNFFG or VNFFG in textual notation. For each component, provide the component id and the endpoint id.</p>	<p>From the figure above, you can see that each atomic functional component has two end points i.e. entry and exit points. In the blueprint, the service sequence can be represented as:</p> <p>serviceSequence:</p> <ul style="list-style-type: none"> # sensor data flow - hopEndPoints: - vsComponentId: 'RAN'

				endPointId: 'SapRANVs' - vsComponentId: 'RAN' endPointId: 'CpRANEpc' - vsComponentId: 'Master_PLC' endPointId: 'CpMPLC' ...
configurableParameters	array	False	Parameters that can be configured by the user for a specific instance of service derived from the given blueprint.	
applicationMetrics	array	true	List of application metrics to be monitored e.g. received requests, fulfilled requests, number of connected clients	applicationMetrics: - metricId: 'navigation_level' name: 'Measurement of the quality of the AGV navigation.' metricCollectionType: 'GAUGE' unit: " interval: '1s' topic: '/app/navigation_level'
compatibleSites	array	true	The 5GEVE sites where this blueprint had radio coverage	compatibleSites: ['spain']
compatibleContextBlueprint	Array	true	List of IDs of the blueprints defining the experiment execution contexts that are compatible with the given vertical service.	compatibleContextBlueprints: ['ctx_delay', 'ctx_bg_traffic']

Step III: Preparation and generation of Network Service Descriptors (NSDs)

After generating the vertical service blueprint, the vertical has to generate the NSDs that correspond to the network service deployment flavours (DFs) defined in the blueprint. In 5G EVE, NSDs are designed following ETSI GS NFV-SOL 001 standard [2].

Annex B - Vertical Service Blueprint YAML file example

```
# vsb_asti_agv.yaml
---
blueprintId: 'vsb_asti_agv'
version: '1.0'
name: 'ASTI AGV control and automation'
description: 'Blueprint for AGVs using 5G network for Automation and Control
purposes'
parameters:
  - parameterId: 'number_of_AGVs'
    parameterName: 'Number of AGVs'
    parameterType: 'number'
    parameterDescription: 'Number of AGVs (Mandatory)'
    applicabilityField: 'autonomous_vehicle'
atomicComponents:
  - componentId: 'RAN'
    serversNumber: 1
    # componentName: 'Radio Access Network'
    imagesUrls: []
    endPointsIds: ['SapRANVs', 'CpRANEpc']
    lifecycleOperations: {}
    # scalingRules: []
  - componentId: 'Master_PLC'
    # componentName: 'Robot Controller Server'
    serversNumber: 1
    imagesUrls: ['master_plc_image']
    endPointsIds: ['CpMPLC']
    lifecycleOperations:
      instantiate: 'scripts/instantiate'
      terminate: 'scripts/terminate'
      update: 'scripts/update'
      heal: 'scripts/heal'
    # scaling_rules: ['scale out if load > 60% for 2 mins']
  - componentId: 'vEPC'
    # componentName: 'Virtualized EPC'
    serversNumber: 1
    imagesUrls: []
    endPointsIds: ['cpEPCRan', 'SapCoreExt']
    lifecycleOperations: null
    # scaling_rules: []
  - componentId: 'Mgmt'
    # componentName: 'Robot Management Server'
    serversNumber: 1
    imagesUrls: ['mgmt_image']
    endPointsIds: ['SapMgmt', 'SapAdmin']
    lifecycleOperations:
      instantiate: 'scripts/instantiate'
      terminate: 'scripts/terminate'
      update: 'scripts/update'
      heal: 'scripts/heal'
    # scaling_rules: ['scale up if load > 90% for 2 mins']
endPoints:
  - endPointId: 'SapRANVs'
    external: true
    management: false
    ranConnection: true
  - endPointId: 'CpRANEpc'
    external: false
    management: false
    ranConnection: false
```

```
- endPointId: 'CpMPLC'
  external: false
  management: false
  ranConnection: false
- endPointId: 'CpEPCRan'
  external: false
  management: false
  ranConnection: false
- endPointId: 'SapCoreExt'
  external: true
  management: false
  ranConnection: false
- endPointId: 'SapMgmt'
  external: true
  management: false
  ranConnection: false
- endPointId: 'SapAdmin'
  external: true
  management: false
  ranConnection: false
connectivityServices:
- endPointIds: ['CpRanEpc', 'CpMPLC', 'CpEpcRan']
  external: false
- endPointIds: ['SapCoreExt', 'SapMgmt']
  external: false
serviceSequence:
# sensor data flow
- hopEndPoints:
- vsComponentId: 'RAN'
  endPointId: 'SapRANVs'
- vsComponentId: 'RAN'
  endPointId: 'CpRANepc'
- vsComponentId: 'Master_PLC'
  endPointId: 'CpMPLC'
compatibleContextBlueprint: ['ctx_delay', 'ctx_bg_traffic']
applicationMetrics:
- metricId: 'navigation_level'
  name: 'Measurement of the quality of the AGV navigation.'
  metricCollectionType: 'GAUGE'
  unit: ''
  interval: '1s'
  topic: '/app/navigation_level'
- metricId: 'power_consumption'
  name: 'Measurement of the power consumption from the AGV.'
  metricCollectionType: 'GAUGE'
  unit: 'watt'
  interval: '1s'
  topic: '/app/power_consumption'
- metricId: 'time_to_lose_guide'
  name: 'Amount after which the AGV lost the guide.'
  metricCollectionType: 'CUMULATIVE'
  unit: 's'
  interval: '1s'
  topic: '/app/time_to_lose_guide'
- metricId: 'connected_robots'
  name: 'Number of currently connected robots'
  metricCollectionType: 'GAUGE'
  unit: ''
  interval: '5s'
  topic: '/app/connected-robots'
```






Annex C – Data collection and experimentation monitoring tools survey and selection

In this section we explain the methodology used to survey and select the data collection and experimentation monitoring tools. We adopted a methodology in four steps:

1. **State of the art:** we collected information on tools supported by active communities paying particular attention to their available documentation, support, availability, maintenance, open-source, usability;
2. **Evaluation:** we identified a set of characteristics and features needed to satisfy the requirements of experiment monitoring and maintenance & results collection;
3. **Rank:** we evaluated each tool on the base of the evaluation process defined in the previous step, obtaining a ranked list of prioritized tools which best satisfy the requirements;
4. **Select:** we selected the best toolset taking into account the ranked list of the previous step.

In the first step, we operated a survey collecting 16 tools related to the data collection (including data aggregation and pre-processing) and 8 tools related to the experiment monitoring (including data indexing, storing and visualization). It is worth to mention that 3 tools appear in both the lists (ELK, Telegraf and SciPy) since their capabilities cover both the data collection and experiment monitoring aspects. In particular, the following tables summarize the two sets of tools:

Table 40: State of the art on data collection

Tool	Brief description
	Prometheus is an open-source systems monitoring and alerting toolkit. Prometheus works well for recording any purely numeric time series. It fits both machine-centric monitoring as well as monitoring of highly dynamic service-oriented architectures. In a world of microservices, its support for multi-dimensional data collection and querying is a particular strength.
	Graphite is a free open-source software tool that monitors and graphs numeric time-series data such as the performance of computer systems. It focuses on being a passive time series database with a query language and graphing features. Graphite collects, stores, and displays time-series data in real time.
 Telegraf	Telegraf is a plugin-driven server agent for collecting and reporting metrics. Telegraf has integrations to source a variety of metrics, events, and logs directly from the containers and systems it's running on, pull metrics from third-party APIs, or even listen for metrics via a StatsD and Kafka consumer services. It also has output plugins to send metrics to other datastores, services, and message queues, including InfluxDB, Graphite, OpenTSDB, Datadog, Librato, Kafka, MQTT, NSQ, and many others
	Scipy is a Python-based ecosystem of open-source software for mathematics, science, and engineering. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering. SciPy builds on the NumPy array object and is part of the NumPy stack. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab.
	InfluxDB is an open-source time series database with a commercial option for scaling and clustering. It is written in Go and optimized for fast, high-availability storage and retrieval of time series data. It has no external dependencies and provides a SQL-like language with built-in time-centric functions for querying a data structure composed of measurements, series, and points.

 <p>OPENTSDDB</p>	<p>OpenTSDB is a scalable time series database built on top of Hadoop and HBase. It simplifies the process of storing and analyzing large amounts of time-series data from sources like as server operations and load metrics, or sensors measuring environmental data.</p>
 <p>elasticsearch</p>	<p>The ELK Stack is a collection of three open-source products: Elasticsearch, Logstash, and Kibana. Elasticsearch is a NoSQL database that is based on the Lucene search engine. Logstash is a log pipeline tool that accepts inputs from various sources, executes different transformations, and exports the data to various targets. Kibana is a visualization layer that works on top of Elasticsearch. Elastic Stack is a complete end-to-end log analysis solution which helps in deep searching, analysing and visualizing the log generated from different machines.</p>
	<p>Amazon CloudWatch is a component of Amazon Web Services (AWS) that provides monitoring for AWS resources and the customer applications running on the Amazon infrastructure. The application automatically provides metrics for CPU utilization, latency, and request counts; users can also stipulate additional metrics to be monitored, such as such as memory usage, transaction volumes or error rates.</p>
	<p>Datadog is designed to help app developers, IT professionals, and other app operations specialists to monitor the performance of their apps, tools, and services, gather data from scattered sources, and turn them into detailed and reliable insights. From there, users can produce sound strategies for their apps and tools, create better applications, and improve their services.</p>
	<p>Librato is a cloud-based monitoring platform for development and operations teams who want the flexibility to monitor the metrics and events important to their application deployment, while leaving storage, analysis and alerting to a service that can scale with their operation.</p>
	<p>Apache Kafka is an open-source stream-processing software platform written in Scala and Java. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. Its storage layer is essentially a massively scalable pub/sub message queue designed as a distributed transaction log, making it highly valuable for enterprise infrastructures to process streaming data.</p>
	<p>MongoDB is a free and open-source cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents.</p>
	<p>Apache Spark is a general-purpose & lightning fast cluster computing system. It provides high-level API and it is a parallel data processing framework. Spark is written in Scala but provides rich APIs in Scala, Java, Python, and R. It can be integrated with Hadoop and can process existing Hadoop HDFS data.</p>
	<p>R is a programming language and a development' s environment focused on statistical analysis of data. It's a free software and it is available for different operating systems. It provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification, clustering, ...) and graphical techniques, and is highly extensible.</p>
	<p>MQTT stands for MQ Telemetry Transport. It is a publish/subscribe, extremely simple and lightweight messaging protocol, designed for constrained devices and low-bandwidth, high-latency or unreliable networks. The design principles are to minimise network bandwidth and device resource requirements whilst also attempting to ensure reliability and some degree of assurance of delivery.</p>


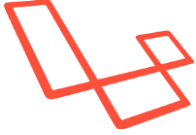
	<p>NSQ is a realtime distributed messaging platform designed to operate at scale, handling billions of messages per day. It promotes distributed and decentralized topologies without single points of failure, enabling fault tolerance and high availability coupled with a reliable message delivery guarantee.</p>
---	--

Table 41: State of the art on experimentation monitoring tools

Tool	Brief description
 Link	<p>OML is an instrumentation tool to support monitoring systems in testbeds. It allows application writers to define customizable Measurement Points (MPs) inside new or pre-existing applications. The MPs define the metrics that will be monitored. Data are collected for each measurement point in OML stream format. Experimenters running the applications can then direct the measurement streams from these MPs to remote collection points, for storage in measurement databases.</p>
 Link	<p>Tableau Public is a free service that lets anyone publish interactive data visualizations to the web. Visualizations that have been published to Tableau Public can be embedded into web pages and blogs, they can be shared via social media or email, and they can be made available for download to other users. Visualizations are created in the accompanying app Tableau Desktop Public Edition. No programming skills are required.</p>
 Link	<p>ELK is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana. Elasticsearch is a distributed, RESTful search and analytics engine capable of solving a growing number of use cases. It is the heart of the Elastic Stack; Logstash is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to Elasticsearch; Kibana is in charge of visualizing Elasticsearch data and navigating the Elastic Stack.</p>
 Telegraf	<p>Telegraf is a plugin-driven server agent for collecting and reporting metrics. Telegraf has integrations to source a variety of metrics, events, and logs directly from the containers and systems it's running on, pull metrics from third-party APIs, or even listen for metrics via a StatsD and Kafka consumer services. It also has output plugins to send metrics to other datastores, services, and message queues, including InfluxDB, Graphite, OpenTSDB, Datadog, Librato, Kafka, MQTT, NSQ, and many others</p>
 Grafana	<p>Grafana is a data visualization and monitoring tool with support for many different storage backends for your time series data (Data Source). Each Data Source has a specific Query Editor that is customized for the features and capabilities that the particular Data Source exposes. The following datasources are officially supported: Graphite, InfluxDB, OpenTSDB, Prometheus, Elasticsearch, CloudWatch.</p>
 Wizzie	<p>The Wizzie Data Platform is a fast, highly scalable, end-to-end solution for real-time processing, storage and visualization of any data type and format, which helps you in your decision-making process, giving you the tools and resources to create value out of your data and act upon it. In particular, the Wizzie ecosystem is built upon two main components: Prozzie and the Wizzie Data Platform. This ecosystem allows you to collect, process and index any type of data, from any source you can imagine.</p>
 SciPy	<p>SciPy is a Python-based ecosystem of open-source software for mathematics, science, and engineering. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering. SciPy builds on the NumPy array object and is part of the NumPy stack. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab.</p>

 Laravel	<p>Laravel is a web application framework based on PHP and widely used to rapidly deploy web-based application and related interfaces to easily visualize data and manage processes. It follows a model-view-controller design pattern. Laravel reuses the existing components of different frameworks which helps in creating a web application. The web application thus designed is more structured and pragmatic.</p>
--	---

In the second step, we evaluated the state-of-the-art tools on the base of a selected set of features, listed in the following tables, able to match with several requirements identified for the data collection and experimentation monitoring tools:

- tool must provide an intuitive GUI;
- tool must be able to monitor the progress of the experiment (live KPI monitoring);
- Tool must allow partial interaction with it in an online fashion:
 - Set thresholds for some parameters (like QoS or QoE);
 - Start/pause/stop of the experiment;
 - Experiment re-configuration options;
- tool should support multiple real-time statistics of selected KPI will be displayed in various forms (digital displays, running graphs, etc.), providing live insights during the experiment:
 - depending on the vertical’s request, additional data analytics and post-processing management tools may be also deployed in dedicated SFs to compute vertical-specific KPIs or trigger automated actions at the orchestrator;
 - performance improvement suggestions from the diagnostics mechanism should be provided too;
- tool must provide connectivity to heterogenous data sources;
- tool should provide publish/subscribe functionalities;
- tool should provide (basic) data search/filtering functionalities;
- tool should provide a turnkey solution for easy deployment, installation and configuration in a virtual environment.

Table 42: evaluated features of data collection tools

Name	Description
Data Analytics	capacity to examine data to find statistical information
Data Visualization	capacity to display data in a visual context in order to facilitate the meaning of information (intuitive GUI)
Data Storage	capacity to retain digital data
Open Source	type of software whose source code is released under a license in which the copyright holder grants users the rights to study, change, and distribute the software to anyone
SQL Compliant	software that is accordant to SQL query language
Alert System	it’s part of the monitoring tool and it refers to the capacity to send alert messages and notifications
Community	group of Internet users who exchange messages and participate in discussion forums on topics of common interest

Distributed	capacity to memorize data in storage devices distributed in a computer network, rather than read or stored centrally on a local device, and then made available through a client-server mechanism between remote devices
-------------	--

Table 43: evaluated features of experimentation monitoring tools

Name	Description
GUI	tool provides a Graphic User Interface (GUI)
Multi-tenant dashboard support	tool allows the definition of dashboards to display multiple results from multiple sources
Interaction	tool allows the interaction with the user and not only the display of results
Integration capacity	tool can be easily integrated in a general GUI
Open API	tool defines an open API (e.g. REST API), used for connecting the monitoring tool with other tools
Integrated tool	tool provides many functionalities integrated in one single product
Community	Tool has a wide community support in case of troubleshooting
Maturity	general concept, but related to the presence of successful use cases, the easiness to work with the tool, the use of well-known programming languages or supporting tools

In the third step, we operated a simple Crisp-Set Qualitative Comparative Analysis (csQCA), assigning, for each tool and for each feature, a boolean value (✓ = feature fully supported, X= feature not fully supported). Summing each boolean value, we calculated a score for each tool. This allowed us to rank the evaluated tools, as shown in Figure 50 and Figure 51.

Tool	Features								Score
	Data Analytics	Data Visualization	Data Storage	Open Source	SQL compliant	Alerting System	Community	Distributed File System	
Prometheus	✓	✓	✓	✓	✓	✓	✓	X	7
Kafka	✓	✓	✓	✓	X	✓	✓	✓	7
Elasticsearch	✓	✓	✓	✓	✓	✓	✓	X	7
Spark	✓	✓	X	✓	✓	X	✓	✓	6
Telegraf	X	✓	✓	✓	✓	✓	✓	X	6
Scipy	✓	✓	X	✓	✓	X	✓	✓	6
R	✓	✓	X	✓	✓	X	✓	X	5
MongoDB	✓	✓	✓	✓	X	X	✓	X	5
Influxdb	X	✓	✓	✓	X	✓	✓	X	5
OpenTSDB	X	✓	✓	✓	X	✓	✓	X	5
CloudWatch	✓	✓	X	✓	X	✓	✓	X	5
Datadog	✓	✓	X	X	✓	✓	✓	X	5
Librato	✓	✓	X	✓	X	✓	✓	X	5
Graphite	X	✓	✓	✓	X	✓	X	X	4
NSQ	X	X	X	✓	X	✓	✓	✓	4
MQTT	X	X	X	✓	X	✓	✓	X	3

Figure 50: csQCA for data collection tools

Tool	Features								Score
	GUI	Dashboard	Interaction	Integration capacity	Open API	Integrated tool	Community	Maturity	
OML	✓	✓	X	✓	X	X	X	X	3
Tableau Public	✓	✓	X	✓	X	X	✓	X	4
ELK	✓	✓	✓	✓	✓	✓	✓	✓	8
Telegraf	X	X	X	✓	✓	X	✓	✓	4
Laravel	✓	X	✓	✓	✓	X	✓	✓	6
Scipy	X	X	X	✓	✓	X	✓	✓	4
Grafana	✓	✓	✓	✓	✓	X	✓	✓	7
Wizzie	✓	✓	✓	✓	✓	✓	✓	X	7

Figure 51: csQCA for experimentation monitoring tools

In the last step, we performed a joint analysis to select the toolset that best matches our requirements. The results show that ELK and Grafana would be the most appropriate tools. However, Grafana needs tools for data collection and processing (like Prometheus). Wizzie could be an alternative to ELK, knowing that it is also an integrated tool that includes analytics capabilities too. However, the maturity level is lower than ELK.

In conclusion, the applied methodology allowed us to select ELK (Logstash + Elasticsearch + Kibana) as the prominent toolset to be used for data collection and experiment monitoring.

Annex D – Video Load generator for video KPIs monitoring

This appendix provides information about a possible approach to measure video-related KPIs using video probes to emulate different load conditions.

Regarding active instrumental video probes, the monitoring system should be able to interact with them in some asynchronous fashion. The idea is that there is some mediator process in the middle which polls the probe for status, progress and results availability. Individual testing of Video KPIs can be done by emulating video traffic in the background at the same time that single video test is executed in parallel. For the individual video tests, we propose a video server which is able to provide the system with the delay added to the system to serve individual segments (HLS) for the case of adaptive streaming scenarios. Several video statistics will be available as a result of the execution. Figure 52 provides snapshots of one professional system which could potentially be used for our purpose.

AdaptiveStreaming Summary	
AdaptiveStreaming Total Attempted Sessions	20
AdaptiveStreaming Total Successful Sessions	20
AdaptiveStreaming Total Unsuccessful Sessions	0
AdaptiveStreaming Total Aborted Sessions	0
AdaptiveStreaming Total Attempted Channels	20
AdaptiveStreaming Total Successful Channels	20
AdaptiveStreaming Total Unsuccessful Channels	0
AdaptiveStreaming Total Aborted Channels	0
AdaptiveStreaming Total Attempted Http Requests	36042
AdaptiveStreaming Total Successful Http Requests	35043
AdaptiveStreaming Total Unsuccessful Http Requests	999
AdaptiveStreaming Total Abort Http Requests	0
AdaptiveStreaming Total Attempted Manifest Requests	18022
AdaptiveStreaming Total Successful Manifest Requests	17023
AdaptiveStreaming Total Unsuccessful Manifest Requests	999
AdaptiveStreaming Total Abort Manifest Requests	0
AdaptiveStreaming Total Manifest Requests Timeout Expired	0
AdaptiveStreaming Total Attempted Fragment Requests	18020
AdaptiveStreaming Total Successful Fragment Requests	18020
AdaptiveStreaming Total Unsuccessful Fragment Requests	0
AdaptiveStreaming Total Abort Fragment Requests	0
AdaptiveStreaming Total Attempted Key File Requests	0
AdaptiveStreaming Total Successful Key File Requests	0
AdaptiveStreaming Total Unsuccessful Key File Requests	0
AdaptiveStreaming Total Abort Key File Requests	0
AdaptiveStreaming Total Buffer Underrun Fragment	20
AdaptiveStreaming Total Pre-Cached Fragment	18000
AdaptiveStreaming Minimum HTTP Response Time (Millisecond)	15
AdaptiveStreaming Average HTTP Response Time (Millisecond)	46
AdaptiveStreaming Maximum HTTP Response Time (Millisecond)	178

AdaptiveStreaming Minimum Audio Adaptivity Score	0
AdaptiveStreaming Average Audio Adaptivity Score	0
AdaptiveStreaming Maximum Audio Adaptivity Score	0
AdaptiveStreaming Minimum Video Adaptivity Score	100
AdaptiveStreaming Average Video Adaptivity Score	100
AdaptiveStreaming Maximum Video Adaptivity Score	100
AdaptiveStreaming Total Upshifts	0
AdaptiveStreaming Total Downshifts	0
AdaptiveStreaming Total Rate Maintaining	18020
AdaptiveStreaming Minimum Buffering Wait Time (Millisecond)	0
AdaptiveStreaming Average Buffering Wait Time (Millisecond)	0
AdaptiveStreaming Maximum Buffering Wait Time (Millisecond)	102
AdaptiveStreaming Minimum Stream Setup Time (Millisecond)	413
AdaptiveStreaming Average Stream Setup Time (Millisecond)	424
AdaptiveStreaming Maximum Stream Setup Time (Millisecond)	457
AdaptiveStreaming Minimum Channel Duration	5400000
AdaptiveStreaming Average Channel Duration	5400000
AdaptiveStreaming Maximum Channel Duration	5400000
AdaptiveStreaming Average Response + Download Time for Fragments < 200 kbps	0
AdaptiveStreaming Average Response + Download Time for Fragments 200 - 499 kbps	0
AdaptiveStreaming Average Response + Download Time for Fragments 500 - 999 kbps	0
AdaptiveStreaming Average Response + Download Time for Fragments 1000 - 1999kbps	0
AdaptiveStreaming Average Response + Download Time for Fragments >= 2000 kbps	66
AdaptiveStreaming 101: Switching Protocols	0
AdaptiveStreaming 1XX-199: Informational Errors	0
AdaptiveStreaming 200: OK	35043
AdaptiveStreaming 201: Created	0
AdaptiveStreaming 204: No Content	0
AdaptiveStreaming 2XX-299: Successful 2XX	0

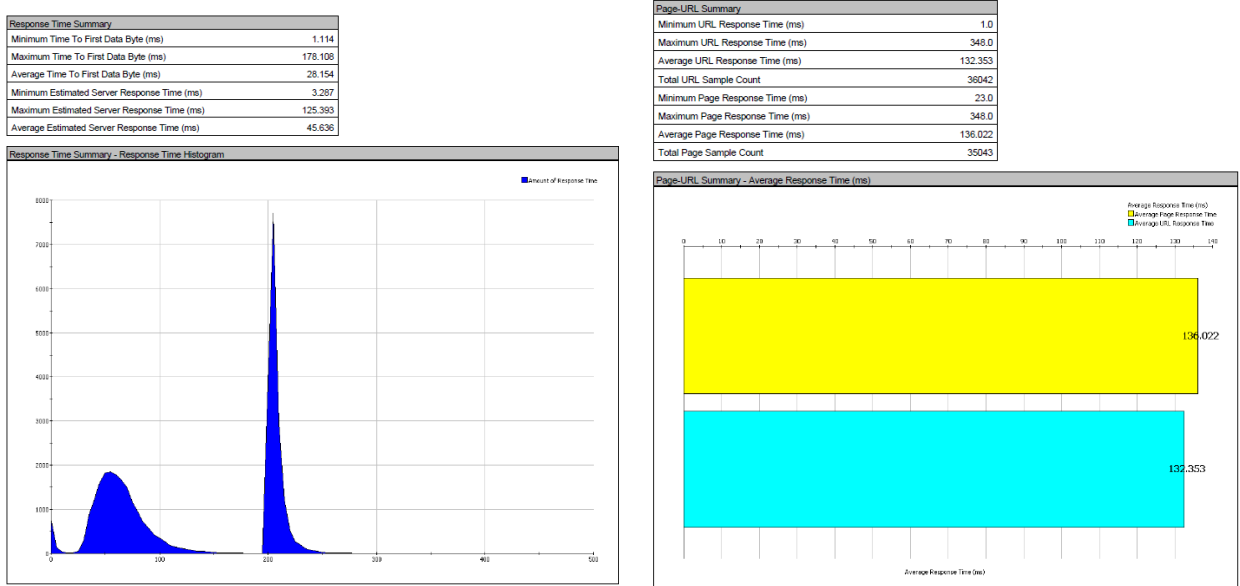


Figure 52: Video KPIs

Other KPIs such as instant bandwidth can also be evaluated both from the server perspective (e.g. segment size divided by time required to serve the segment) or from an E2E perspective requesting some reference segments from and android application which computes this bandwidth allocated to video. In this case such application needs to provide the system with a simple interface with our monitoring tool so that it allows to call it using simple REST interfaces and using common data schemas such as CSV or JSON.

As previously explained in section 2.3.3 there will be couple of VNFs:

Monitoring VNF (VNF_TYPE_1): Video Server controller for Video MEC – sys-monitoring tools and video server instant KPIs

Workload VNF (VNF_TYPE_2): Manager of the performance load computational node

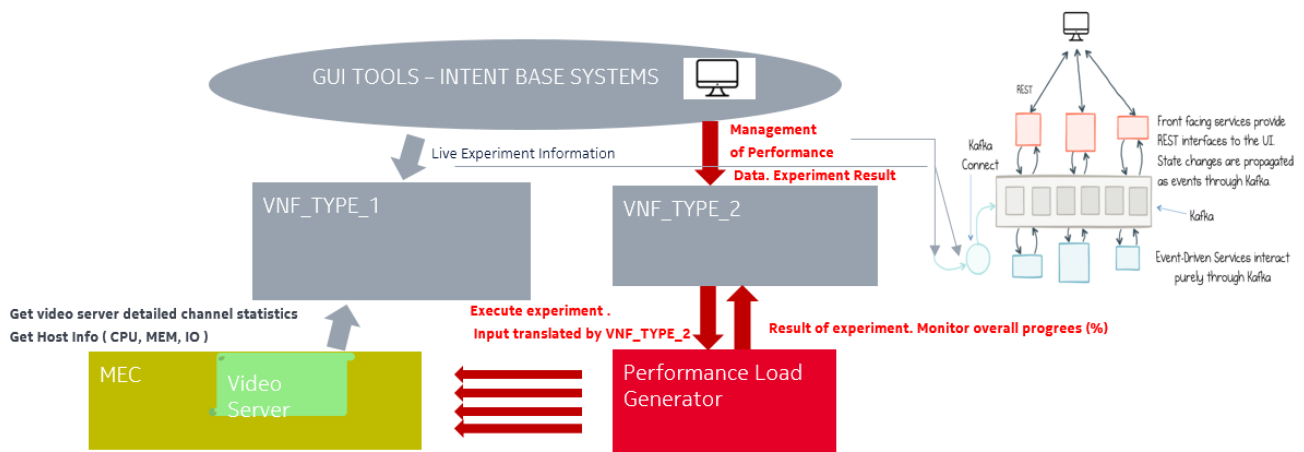


Figure 53: Monitoring system for video-related KPIs

Collection and mediation from other 3PT performance video simulators can be made by retrieving the test result and transforming it into the common format provided by the system for all the experiments. Should that process be too complex at least raw data in simple plain text formats will be available.

Performance node in charge of high load simulation uses typically simple formats as input and [CSV/JSON/PDF] formats as output depending on whether a final printable report or some structure data one is required. This may lead to the necessity for transforming the first into some readable plain files describing the background performance load of the experiment. The same applies to output, focusing mainly in the summary part.

VNF2 will be in charge of interfacing from the south bound perspective with the load simulator host, and from the Northbound perspective will report this CSV/JSON data which can be ingested directly by ELK by means of:

- HTTP REST API
- Logstash pipelines
 - CSV or UDP pipelines
- Posting the information into the KAFKA bus.

(*) Example of CSV pipelines specification in dockerized ELK deployment

```
logstash:
  image: docker.elastic.co/logstash/logstash-oss:6.2.4
  volumes:
    - ../logstash/pipeline:/usr/share/logstash/pipeline
    - ../logstash/config:/usr/share/logstash/config
    - ../logstash/csv:/csv
    - ../logstash/templates:/templates
```

In the video traffic use case, we will collect measurements following a strategy similar to this: Thousands of clients simulating traffic with different latencies. Every client in the performance tool will be launched with different latency. There will be several groups of users with different latencies (Group1 : 30 ms, Group 2: 50ms, Group 3: 100ms). This will emulate not only the huge bandwidth being used by the system but also the different locations and power received by real end user devices in a mobile network,

Stress performance test requires an appropriately dimensioned MEC instance for executing the simulation, being this dimensioning another factor to be taken into consideration (or even provisioned) while configuring the experiment.

LINUX PERFORMANCE TOOLS

This set of tools may be hosted in what we called VNF2 tasking into consideration, that no additional side host is required in this case for the performance measurements.

We will focus on three standard tools which can be used for our purposes and how can we interact with the rest of the experiment infrastructure.

The 3 tools are IPERF3, QPERF and PING.

These tools are ideally wrapped under bash shell scripts which will provide the system with a parameter based simple API which is coherent with the experiments, allowing the flexibility to specify stuff such as the experiment ID, the user ID executing the experiment for example or any other metadata. Output of this tools will result in simple txt, CSV or JSON file which will contain the tools measures, date, duration of the experiment and external metadata which is not known but the tool itself.

Additionally, these scripts may have a placeholder upon some trigger condition another external system can be called to notify about the measure existence. Our proposal is that this can be done by placing the output in some folder which is exposed via simple web server for retrieving it/them from external modules or actively calling such external modules if they are providing some web interface via *wget* or *curl* simple Linux tools. If we

encounter the case in which such external system requires https connectivity, we need to provide the system hosting these scripts with the appropriate certificates or just executing these calls using the `-nocertificates` option only in case we are under a trusty configuration controlled under 5G-EVE labs premises.

It is remarkable the fact that both `iperf3` and `qperf3` requires another instance running on the remote side. Due to that it is a requirement that such tools are pre-installed before running the experiment. Our scripts should check for availability of these tools and provide a null result status of the experiment in the case it is not present.

As running this `iperf3/qperf` server instance continuously may lead to noise to some other experiments, this server instance should be executed only during the experiment execution.

Leading this execution process from the client is a convenient way for achieving this goal. This can be done via ssh as long as keys between client and server are interchanged in advance.

In the case of this tools and their wrappers, both local plain text data files containing the measurements and active calls pushing messages into Kafka bus can be combined. We can also offer this data in a convenient way just transforming it into JSON files being offered by a simple web server hosted in the same box.

Annex E – Software repositories

Table 44 reports the repositories with the software and the tools developed or deployed for the 5G EVE portal components reported in D4.1.

Table 44: Software repositories for D4.1 portal components.

Tool Name	Repository
5G EVE Monitoring and visualization tools	https://github.com/5GEVE/5geve-wp4-monitoring-dockerized-env https://github.com/5GEVE/5geve-wp4-visualization-tool-transport-vertical
5G EVE Ticketing system	http://vm-images.netcom.it.uc3m.es/5GEVE
Browse and look-up tool	https://github.com/nextworks-it/slicer-catalogue/tree/5geve-release
Intent-based tool	https://github.com/5GEVE/5G-EVE-WP4-intent-based-tool
5G EVE Portal Catalogue	https://github.com/nextworks-it/slicer-catalogue/tree/5geve-release
I/W framework Catalogue	https://github.com/nextworks-it/5g-catalogue
Blueprint Validator	https://github.com/TheWall89/blueprint-validator
Blueprint yaml examples	https://github.com/TheWall89/blueprint-yaml

References

- [1] ETSI GS NFV-SOL 005, “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; RESTful protocols specification for the Os-Ma-nfvo Reference Point”, v2.6.1, April 2019
- [2] ETSI GS NFV-SOL 001, “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification”, v2.6.1, May 2019
- [3] ETSI GS NFV-SOL 006, “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on YANG Specification”, v2.6.1, June 2019
- [4] 5G EVE D4.2, “First version of the experimental portal and service handbook”, December 2019
- [5] ETSI GS NFV-IFA 014, “Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Network Service Templates Specification”, v3.1.1, August 2018
- [6] 5G EVE D3.2, “Interworking Reference Model”, June 2019
- [7] 5G EVE D3.3, “First implementation of the interworking reference model”, October 2019
- [8] 5G EVE D1.1, “Requirements definition and analysis from participant vertical industries”, October 2018
- [9] 5G EVE D2.1, “Initial detailed architectural and functional site facilities description”, September 2018
- [10] Rouse, M. Retrieved from WhatIs.com: <https://whatis.techtarget.com/definition/intent-based-networking-IBN>
- [11] Cano, L. (2018, April 18). Intent Based Networking. Retrieved from PandoraFMS: <https://pandorafms.com/blog/intent-based-networking/>
- [12] Cisco Solutions for Intent-Based Networking (IBN) Solution Overview. (2019, June 10). Retrieved from Cisco: <https://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/digital-network-architecture/nb-06-ibn-sol-overview-cte-en.html>
- [13] Lerner, A. (2017, February 7). Intent-based Networking. Retrieved from Gartner: <https://blogs.gartner.com/andrew-lerner/2017/02/07/intent-based-networking/>
- [14] 3GPP Specs. (n.d.). Retrieved from 3GPP A Global Initiative: <https://www.3gpp.org/DynaReport/28-series.htm>
- [15] 5G-TRANSFORMER D3.1, “Definition of vertical service descriptors and SO NBI”. Available: http://5g-transformer.eu/wp-content/uploads/2018/03/D3.1_Definition_of_vertical_service_descriptors_and_SO_NBI.pdf
- [16] “ETSI NFV ISG Portal,” [Online]. Available: <https://portal.etsi.org/tb.aspx?tbid=789&SubTB=789,832,831,801,798,799,848,802,828#/>
- [17] ETSI GS NFV-MAN 001, “Network Functions Virtualisation (NFV); Management and Orchestration”, v1.1.1, December 2014
- [18] ETSI GS NFV-IFA 011, “Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; VNF Descriptor and Packaging Specification”, v3.1.1, August 2018
- [19] OASIS, “TOSCA Simple Profile in YAML Version 1.2”, August 2017
- [20] ETSI GS NFV-SOL 007, “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; Network Service Descriptor File Structure Specification”, v2.6.1, March 2019
- [21] ETSI GS NFV-SOL 004, “Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package Specification”, v2.6.1, April 2019
- [22] ETSI GS NFV-IFA 013, “Network Functions Virtualisation (NFV) Release 3; Management and Orchestration; Os-Ma-Nfvo reference point – Interface and Information Model Specification”, v3.1.1, August 2018
- [23] 5G-MEDIA D4.1, “5G-MEDIA Catalogue APIs and Network Apps”, http://www.5gmedia.eu/cms/wp-content/uploads/2018/09/5G-MEDIA-D4.1-5G-MEDIA-Catalogue-APIs-and-Network-Apps_v1.0.pdf