VertIcal demos over Common large scale field Trials fOr Rail, energy and media Industries

D2.6 5G-VICTORI Infrastructure Operating System – Final Design Specification

This project has received funding from the European Union's Framework Programme Horizon 2020 for research, technological development and demonstration

5G PPP Research and Validation of critical technologies and systems

Project Start Date: 1st June 2019 **Duration**: 36 months

Call: H2020-ICT-2019 **Date of delivery**: 2022-01-31

Topic: ICT-19-2019 Version 1.0

Project co-funded by the European Commission

Under the H2020 programme

Dissemination Level: **Public**

Grant Agreement Number:	857201
Project Name:	VertIcal demos over Common large scale field Trials fOr Rail, energy and media Industries
Project Acronym:	5G-VICTORI
Document Number:	**D2.6**
Document Title:	5G-VICTORI Infrastructure Operating System – Final Design Specification
Version:	1.0
Delivery Date:	2021-11-30 (**2022-01-31**)
Responsible:	Digital Catapult (**DCAT**)
Editor(s):	Kostas Katsaros (**DCAT**)
Authors:	Kostas Katsaros (**DCAT**), Mark Rouse (**DCAT**), Satish Kumar (**DCAT**), Konstantinos Antonakoglou (**DCAT**), Christos Tranoris (**UoP**), Eric Troudt (**FhG**), Marc Emmelmann (**FhG**), Marius Iordache (**ORO**), Hojjat Baghban (**UNIVBRIS**), Anna Tzanakaki (**IASA**).
Reviewers:	Christos Tranoris (**UoP**) Eric Troudt (**FhG**) Marius Iordache (**ORO**) Anna Tzanakaki (**IASA**) Jesús Gutiérrez (**IHP**)
Keywords:	Service Orchestration, CI/CD, APIs, OSM
Status:	Final
Dissemination Level	**Public**
Project URL:	https://www.5g-victori-project.eu/

Revision History

Rev. N	Description	Author	Date
0.1	Draft Table of Contents (ToC)	Kostas Katsaros (**DCAT**)	2021-09-25
0.2	Initial contributions (architecture/components)	Kostas Katsaros, Satish Kumar, Konstantinos Antonakoglou, Mark Rouse (**DCAT**)	2021-10-10
0.3	Integration of Monitoring / Profiling	Hojjat Baghban (**UNIVBRIS**)	2021-10-15
0.4	Update workflow content, Architecture deployment	Kostas. Katsaros, Mark Rouse (**DCAT**)	2021-11-01
0.5	Inter-edge Connectivity Manager (ICM), Profiling updates	Kostas Katsaros, Konstantinos Antonakoglou (**DCAT**) Hojjat Baghban (**UNIVBRIS**)	2021-11-05
0.6	First Draft	Kostas Katsaros (**DCAT**)	2021-11-08
0.7	Addressing reviewers' comments, Updates in Mobility Manager	Kostas Katsaros, Mark Rouse, Satish Kumar (**DCAT**), Christos Tranoris (**UoP**), Marc Emmelmann, Eric Troudt (**FhG**), Marius Iordache (**ORO**), Hojjat Baghban (**UNIVBRIS**)	2021-11-24
0.8	Final Draft	Kostas Katsaros (**DCAT**)	2021-11-25
0.9	1st Technical Review	Anna Tzanakaki (**IASA**)	2021-11-30
0.10	Addressing review comments	Kostas Katsaros, Satish Kumar (**DCAT**), Hojjat Baghban (**UNIVBRIS**)	2021-12-16
0.11	2nd Technical Review	Anna Tzanakaki (**IASA**)	2021-12-20
0.12	Addressing review comments	Kostas Katsaros, Satish Kumar (**DCAT**)	2021-12-22
0.13	3rd Technical Review	Anna Tzanakaki (**IASA**), Jesús Gutiérrez (**IHP**)	2022-01-14
0.14	Addressing review comments	Kostas Katsaros, Satish Kumar (**DCAT**)	2021-01-20
0.15	Final Review	Anna Tzanakaki (**IASA**), Jesús Gutiérrez (**IHP**)	2022-01-14
1.0	Submission of the document	Jesús Gutiérrez (**IHP**)	2022-01-31

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Description
5G OS	5G Operation System
5G-VIOS	5G-VICTORI Operation System
AF	Application Function
AGA	API Gateway
ANN	Artificial Neural Network
API	Application Programmable Interface
AWS	Amazon Web Services
B2B	business to business
CI/CD	Continuous Integration and Continuous Delivery
CN	Core Network
CNF	Containerised Network Function
CPU	Central Processing Unit
DDoS	Distributed DoS
DE	Development Environment
DMVPN	Dynamic Multipoint Virtual Private Network
DoS	Denial of Service
DSE	Dynamic Slicing Engine
E2E	End to End
ELK	Elasticsearch, Logstash, Kibana
ETSI	European Telecommunications Standards Institute
ETSI NFV-IFA	ETSI NFV Interfaces and Architecture
ETSI NFV-SOL	ETSI NFV Solution
FCAPS	Fault, Configuration, Accounting, Performance & Security
GUI	Graphical User Interface
HNF	Hybrid Network Function
I/O	Input/Output
IAM	Identity Access Management
ICM	Inter-edge Connectivity Manager
iNS	Inter-domain Network Service
iNSD	iNS Descriptor
iPSEC	Internet Protocol security
KPI	Key Performance Indicator
L2	Layer-2 of OSI stack

L3	Layer-3 of OSI stack
LCM	Life-Cycle Management
MANO	Management and Orchestration
MEC	Multi-Access Edge Compute
MFA	Multi Factor Authentication
MIR	Maximum Input Rate
ML	Machine Learning
MON	Monitoring
NAT	Network Address Translation
NBI	NorthBound Interface
NEF	Network Exposure Function
NF	Network Function
NFV	Network Function Virtualization
NFVI	Network Function Virtualisation Infrastructure
NFVO	NFV Orchestrator
NHRP	Next Hop Resolution Protocol
NMS	Network Management System
NRF	Network Repository Function
NS	Network Service
NSB	Network Service Broker
NSC	Network Service Composer
NSD	NS Descriptor
NSI	Network Slice Instance
NSM	Network Service Manager
NSR	Network Service Request
NST	Network Service Template
NWDAF	Network Data Analytics Function
OAI	OpenAirInterface
ONAP	Open Network Automation Platform
OSM	Open Source MANO
OSPF	Open Shortest Path First
OSS/BSS	Operations Support Systems / Business Support Systems
PNF	Physical Network Function
POL	Policy
PoP	Point of Presence
QoE	Quality of Experience

QoS	Quality of Service
RAN	Radio Access Network
RBAC	Role Based Access Control
RO	Resource Orchestrator
SASE	Secure Access Service Edge
SBA	Service-Based Architecture
SC	Service Chaining
SDN	Software Defined Networking
SD-WAN	Software Defined Wide Area Network
SEPP	Security Edge Protection Proxy
SLA	Service Level Agreement
SME	Small and medium-sized enterprise
TLP	Telecom Layer Platform
URI	Uniform Resource Identifier
VAO	Vertical Application Orchestration
VCA	VNF Configuration and Abstraction
VIM	Virtualised Infrastructure Management
VNF	Virtual Network Function
VNFD	VNF Descriptor
VNF-FG	VNF Forwarding Graph
WAN	Wide Area Network

Executive Summary

This document presents the final design specification of the 5G-VICTORI Infrastructure Operating System (5G-VIOS), which extends the work reported in deliverable D2.5.

The document addresses end-to-end service provisioning, including end-to-end slicing between different domains and sites. It also provides an update of its architecture and how it is linked to that of 5G, particularly referring to network functions (NFs) from the 3GPP 5G Service-Based Architecture (SBA), including Security Edge Protection Proxy (SEPP), Network Repository Function (NRF), Network Exposure Function (NEF) and Network Data Analytics Function (NWDAF).

Following the cloud native implementation of 5G-VIOS, attention is given on the 5G-VIOS deployment options that can be adopted depending on the scenario, policies, and restrictions for each deployment. This includes the requirements for the integration of 5G-VIOS at each of the 5G-VICTORI facilities.

Initial integration testing was performed at the Bristol facility where the deployment framework was tested. This included the deployment of the Kubernetes cluster on the Bristol facility and on top of it deploy 5G-VIOS. We also tested instantiation of simple Network Services (NSs) through 5G-VIOS.

Furthermore, in this deliverable, component Application Programming Interfaces (APIs) and workflows have been extensively updated since the previous deliverable release.

Specifically, some of the highlights include:

- the Inter-edge Connectivity Manager (ICM) component that is responsible for the Dynamic Multipoint VPN (DMVPN) service that links 5G-VIOS with individual facilities,
- details for the Artificial Intelligence and Machine Learning (AI/ML) algorithms adopted by the Profiling component,
- updates on the portal that the vertical users and VNF providers would be using for the experiments, and
- step-by-step workflows for the key functionalities of 5G-VIOS.

1 Introduction

As 5G matures and deployments accelerate there is a growing desire for faster innovation and agility in network technology. Virtualisation and open and interoperable solutions foster the innovation and community-driven approach to solving some of the biggest challenges in 5G infrastructures, and they are gaining attention from operators deploying networks, as well as government policymakers supporting 5G deployments. With the wider adoption of 5G networks, new products and new business models in the business to business (B2B) field continuously emerge, aiming to meet the diverse and personalized requirements of industry users. The 5G platforms play an essential role in bringing technology players, vendors, operators, and vertical markets together, making necessary to deploy 5G solutions for vertical industries and to explore synergies of the common 5G infrastructure and services supporting a variety of vertical industries.

In this context, it is key to develop flexible architectures able to address a wide range of vertical applications. The infrastructures that adopt this architectural framework will be able to offer converged services across heterogeneous technology domains and have a unified software deployment. 5G-VIOS represents the cross-domain service orchestration platform within 5G-VICTORI that enables vertical users and Small and medium-sized enterprises (SMEs) to access a common infrastructure allowing cross-vertical collaborations and synergies in order to offer enhanced value propositions.

This deliverable extends the work carried out in the context of Task 2.4 on the modification and update of the 5G-VIOS component APIs and workflows, after successful integration activities and tests at the 5GUK (Bristol) facility. Additionally, an update of the 5G-VIOS high level architecture is provided and, specifically, the mapping to the 5G network architecture and network functions (NFs) from the 5G Service-Based Architecture (SBA) is described. Following the cloud native implementation of 5G-VIOS, recommendations are given on deployment options that can be adopted for 5G-VIOS, depending on the scenario, policies, and restrictions for each deployment. Examples of deployment targets are the 5G-VICTORI facilities.

1.1 Organisation of the document

This document comprises 5 sections. Following the Executive Summary and Introduction sections:

Section 2 identifies architectural principles that pervade throughout the 5G-VIOS and, hence, form the basis for the deployment options and most of the component description in subsequent section describes.

Section 3 provides details of individual 5G-VIOS components, their implementation and updated API specifications.

Section 4 illustrates the operations of 5G-VIOS with detailed workflows as they have been updated since deliverable D2.5.

Finally, Section 5 concludes this deliverable.

2 5G-VIOS Architecture

This section provides a detailed description of the 5G-VIOS architecture and its potential deployment over existing 5G network infrastructures. Examples of these are those deployed in the 5G-VICTORI vertical facilities. Specifically, we refer to deliverable D2.5 [1], to update the different components of 5G-VIOS, see the mapping and link of those to the 3GPP 5G SBA and, finally, to examine the deployment options for the 5G-VIOS platform depending on the scenario requirements and limitations.

2.1 Architecture description

To facilitate inter-domain orchestration and interconnection, a thin inter-domain orchestration brokering solution is developed, namely 5G-VIOS [1]. The 5G-VIOS is a common platform that enables management of slices, resources, and orchestration of services across different sites. 5G-VIOS provides network service (NS) deployment across different sites, dynamic layer-2 (L2) or layer-3 (L3) cross-site service interconnections, inter-site service composition and on-boarding, End-to-End (E2E) slice monitoring and management for the deployed E2E services. The design of 5G-VIOS considers the current status of the Management and Orchestration (MANO) platform at each facility and will work closely with each one to reflect the facility extensions to the common multi-site orchestration platform. This builds on top of the orchestration solutions of each facility and their 5G systems, to provide E2E services across the different sites (see Figure 2-1).

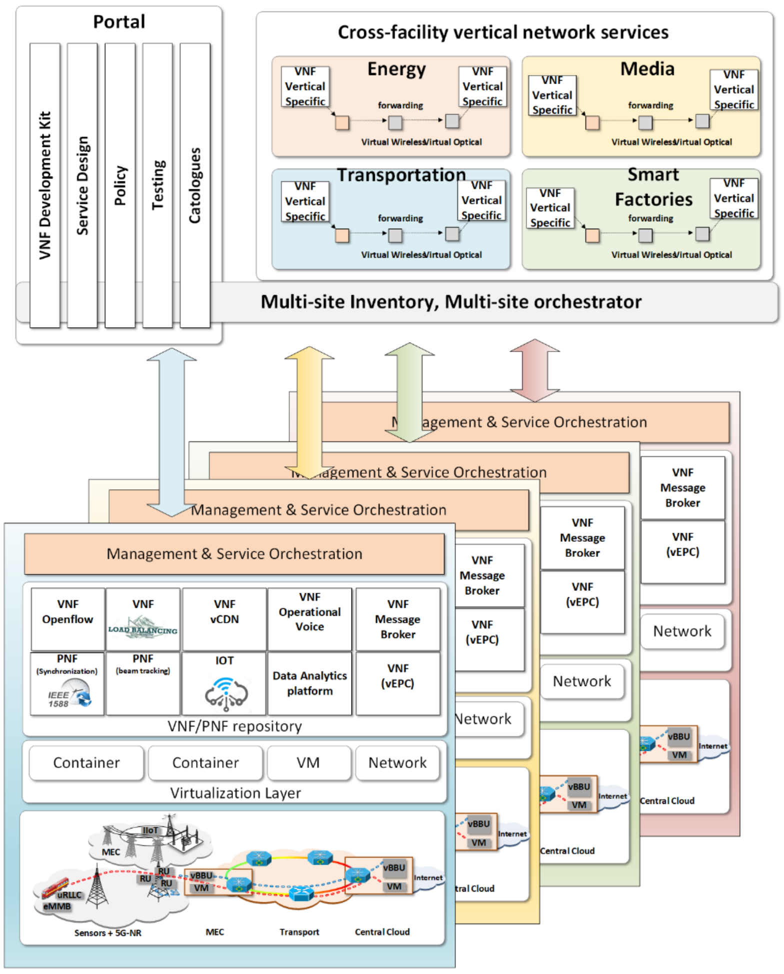


Figure 2-1 5G-VICTORI functional architecture overview

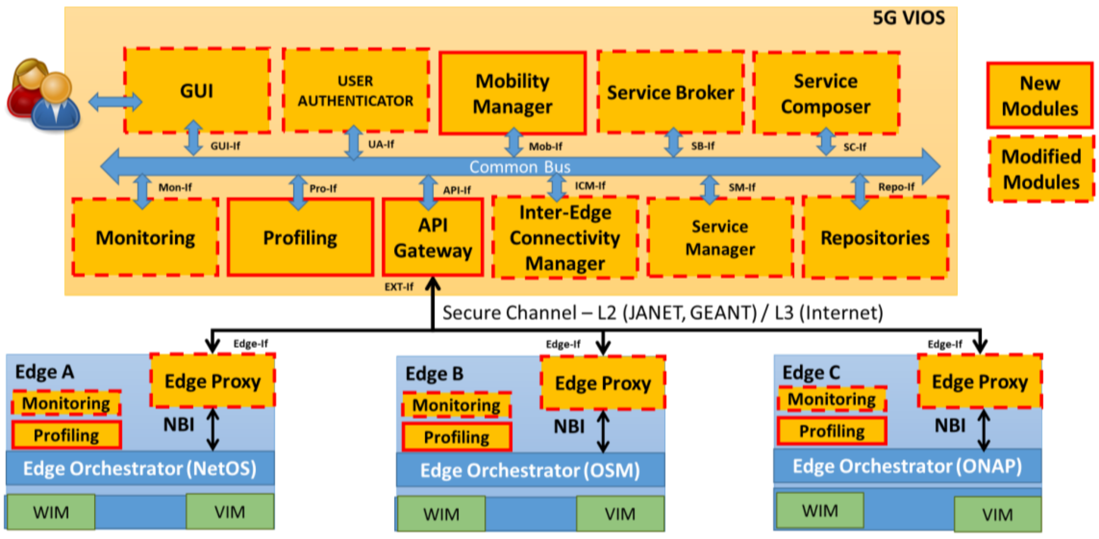


Figure 2-2 5G-VIOS high level architecture [1]

As introduced in deliverable D2.5 [1], 5G-VIOS has adopted a modular, cloud native architecture, following a SBA, allowing individual components to be developed and extended in parallel to each other, providing flexibility and adaptability. Individual components are developed as micro-services and are interconnected through a common bus over which they can communicate using RESTful APIs ensuring security through Role-based access control (RBAC) policies. Such architecture has several benefits including cost savings, better agility in managing the underlying infrastructure and increased speed with respect to system releases.

Figure 2-2 illustrates the high-level functional architecture of 5G-VIOS, whereby key components are identified as new or modified using as basis previous work in 5GUK Exchange, 5G-PICTURE and other previous activities. Individual edges could reflect different technology or administrative domains, each with their respective Network Function Virtualisation Orchestrator (NFVO), providing a host of 5G services and interfacing with 5G-VIOS through a proxy service. In 5G-VICTORI, there are two primary edge types: one that relates to a 5G-VICTORI facility, i.e. Patras, Berlin, Bristol or Alba Iulia; and a second that relates be a self-contained sub-set of such facility, particularly in Bristol facility with the nomadic node, or in Patras with the Autonomous Edge.

The 5G-VIOS architecture follows software-defined wide area networking (SD-WAN) and Secure Access Service Edge (SASE) principles exploiting Dynamic Multipoint Virtual Private Network (DMVPN), with a hub and spoke architecture. SASE is a network architecture evolution of SD-WAN and uses SD-WAN's principles and technological foundations, rolling SD-WAN and security into a cloud service that promises simplified WAN deployment, improved efficiency, and security, and to provide appropriate bandwidth per application. For example, SASE uses several connection types and virtualizes the network and security functions that control and secure a network. Differently from SD-WAN, with the SASE framework, the network and security functions run at a point of presence (PoP) to the user instead of running at traditional networking devices. The WAN side of SASE relies on capabilities supplied by entities including SD-WAN providers, carriers, content-delivery networks, network-as-a-service providers, bandwidth aggregators and networking equipment vendors. SASE supports zero-trust networking, which bases access on user, device and application rather than location and IP address.

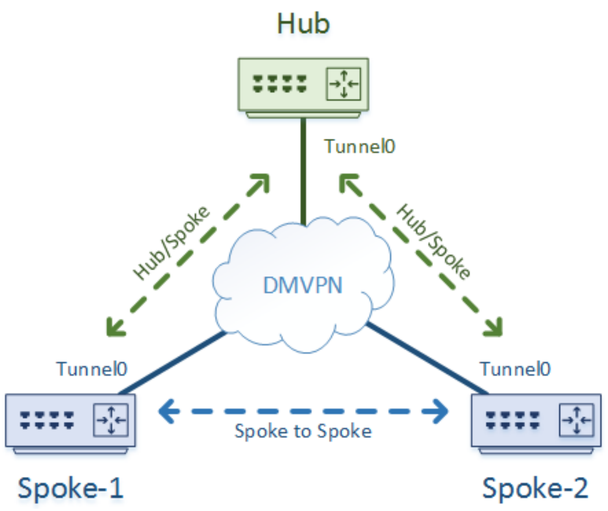


Figure 2-3 DMVPN Hub - Spoke model

This service is developed within the Inter-edge connectivity Manager (ICM). It makes use of Dynamic Multipoint VPN (DMVPN), a solution that enables the data to be transferred from one site/facility to another, without requiring the verification process of traffic. That used to be held at a main VPN server of the concerned organisation. This process helps the data moving from one site to another after the establishment of a secured network. The ICM is integrated with a unique software, which constructs Internet Protocol security (IPsec) and Generic Routing Encapsulation (GRE) tunnels in an unchallenged way. There are two main designs that are incorporated in this network: 1) the Hub-to-Spoke design, where the Hub instantiates individual tunnels with each Spoke in a one-to-one basis in a star architecture; and 2) the Spoke-to-Spoke design, where tunnels are dynamically generated between spokes on-demand in a mesh architecture. Details of the DMVPN implementation in 5G-VIOS are explained in Section 3.5.

2.2 Mapping of 3GPP components to 5G-VIOS

The 5G-VICTORI E2E reference architecture, depicted in Figure 2-4, looks alike that of the 5G SBA, where the 5G-VIOS components are interconnected similarly to 5G SBA NFs. This E2E reference architecture will be described in detail in deliverable D2.4 [2].

In principle, 5G-VIOS components extend certain 5G NFs, such as Security Edge Protection Proxy (SEPP), Network Exposure Function (NEF) and Network Repository Function (NRF), to support inter-domain connectivity and services. This means that certain 5G NFs functionalities are incorporated in the 5G-VIOS components.

Specifically, the 3GPP components are mapped to 5G-VIOS components as described in the following subsections.

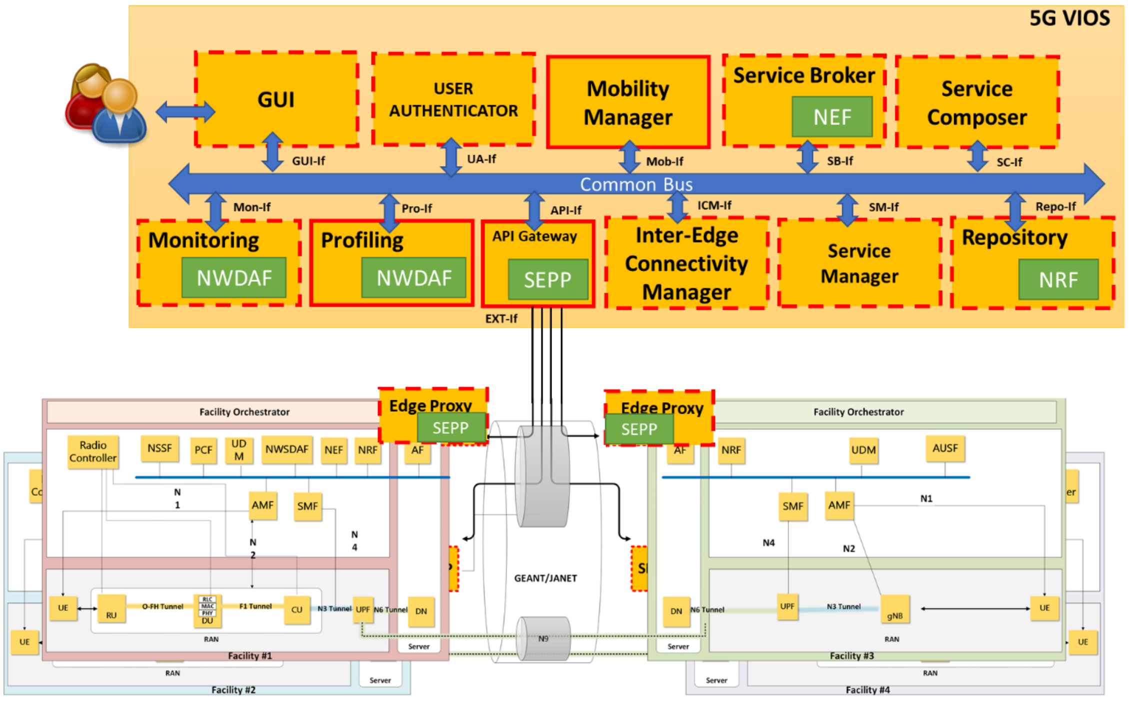


Figure 2-4 5G-VICTORI E2E Reference Architecture

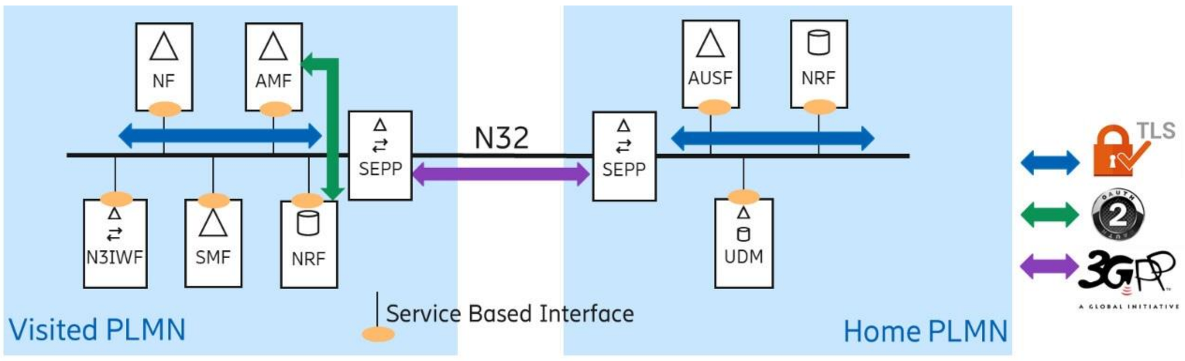


Figure 2-5 Simplified SBA for the 5G system in the roaming case (source: Ericsson)

2.2.1 Security Edge Protection Proxy (SEPP)

The SEPP is part of the roaming security architecture, as shown in Figure 2-5. As part of the 3GPP architecture, all signalling traffic across operator networks is expected to transit through these security proxies. Secondly, authentication between SEPPs is required. This enables effective filtering of traffic coming from the interconnect between these two 5G networks available in different facilities. Thirdly, a new application layer security solution on the N32 interface between the SEPPs was designed (3GPP TS 29.573) to provide protection of sensitive data attributes, while still allowing mediation services throughout the interconnect.

This functionality is being implemented through two 5G-VIOS components, namely the API Gateway and the Edge Proxy. These two components are responsible to enable secure connections between 5G-VIOS and individual edges and allow signalling between 5G-VIOS and individual edges. In principle, the 5G-VIOS Edge Proxy is an extended SEPP, implementing the basic SEPP functionality and extending it to communicate with the NFVO and other services running at each facility. It also works along with the ICM to establish the DMVPN network, thus enabling secure connections between facilities. A detailed explanation of these components is provided in sections 3.3 and 3.7, respectively.

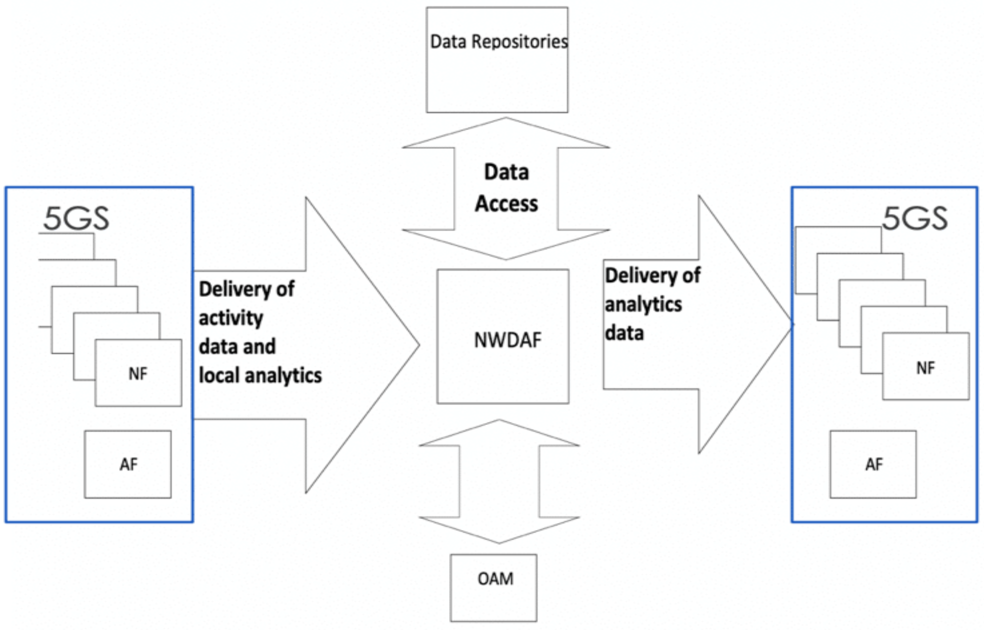

Data Repositories
Data Access
5GS
NF
AF
Delivery of activity data and local analytics
NWDAF
Delivery of analytics data
5GS
NF
AF
OAM

Figure 2-6 NWDAF framework (source: TMForum)

2.2.2 Network Data Analytics Function (NWDAF)

As described in 3GPP TS 23.501 V15.2.0 (2018-06), NWDAF represents operator managed network analytics logical function and is a key enabler for automation of 5G networks:

- NWDAF provides slice specific network data analytics to a NF.
- NWDAF provides network analytics information (i.e. load level information) to a NF on a network slice instance level and the NWDAF is not required to be aware of the current subscribers using the slice.
- NWDAF notifies slice specific network status analytic information to the NFs that are subscribed to it. NFs may collect directly slice specific network status analytic information from NWDAF.
- The NWDAF may serve use cases belonging to one or several domains, e.g. QoS, traffic steering, dimensioning, security.

The input data of the NWDAF may come from multiple sources (see Figure 2-6), and the resulting actions undertaken by the consuming NF or Application Function (AF) may concern several domains, e.g. Mobility management, Session Management, QoS management, Application layer, Security management, NF life-cycle management (LCM).

Use case descriptions should include the following aspects:

1. General characteristics (domain: performance, QoS, resilience, security; time scale).
2. Nature of input data, e.g. logs, Key Performance Indicator (KPI), events.
3. Types of NF consuming the NWDAF output data, how data is conveyed and nature of consumed analytics.
4. Output data.
5. Possible examples of actions undertaken by the consuming NF or AF, resulting from these analytics.
6. Benefits, e.g. revenue, resource saving, QoE, service assurance, reputation.

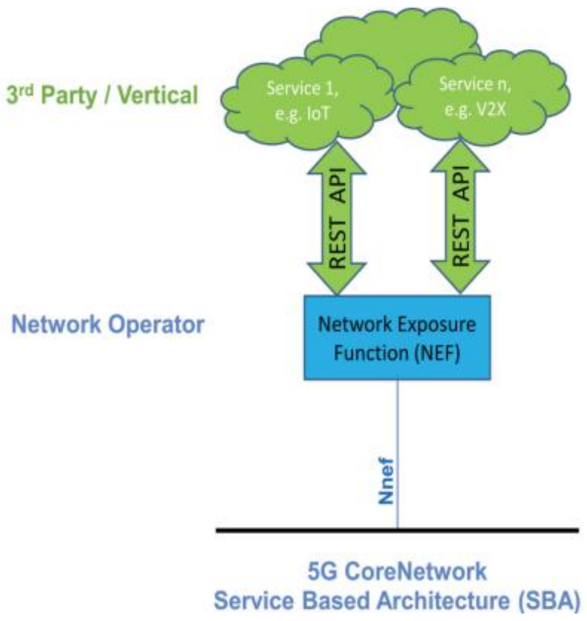


Figure 2-7 NEF framework [3]

This functionality is being extended through two 5G-VIOS components, the Monitoring and Profiling components (see Figure 2-4). These two components are responsible to enable data collection and analysis through AI/ML. Data from individual facilities and sub-systems, including the 5G Core (5GC), are collected and then aggregated per experiment or network service across different facilities. A detailed description of these components is presented in sections 3.9 and 3.10, respectively.

2.2.3 Network Exposure Function (NEF)

NEF facilitates a secure, robust, developer-friendly access to services and capabilities provided by 3GPP network functions (3GPP TS 29.522). This access is provided by a set of northbound RESTful APIs from the network domain to both internal (i.e. within the network operator's trust domain) and external applications (Figure 2-7).

This functionality is being implemented through the Service Broker (SBR) component[1] in 5G-VIOS. SBR enables different edges to expose their capabilities and services in a common infrastructure (5G-VIOS) that can be used to instantiate inter-domain services in collaboration with the service composer and other 5G-VIOS components. A detailed description of this component is provided in section 3.1.

2.2.4 Network Repository Function (NRF)

The NRF is a key element of the 5G SBA. It provides a single record of all NFs available in each PLMN, together with the profile of each and the services they support. The NRF supports the following functions:

- Maintains the profiles of the available NF instances and their supported services in the 5GC network.
- Allows consumer NF instances to discover other providers NF instances in the 5GC network.
- Allows NF instances to track the status of other NF instances.

[1] This was referred to as Network Service Broker in D2.5 – renamed to Service Broker thereafter.

The NRF interacts with every other element in the 5GC network. As such, it is an important element, as it allows other NFs – such as the User Plane Function (UPF), Access and Mobility Management Function (AMF), Session Management Function (SMF), Policy Control Function (PCF) and so on – to subscribe to, and get notified about, the registration in the NRF of new NF instances of a given type. In addition to maintaining profiles, it also supports service discovery functions, enabling other NFs to obtain information regarding available NFs that can support specific services (3GPP TS 29.510).

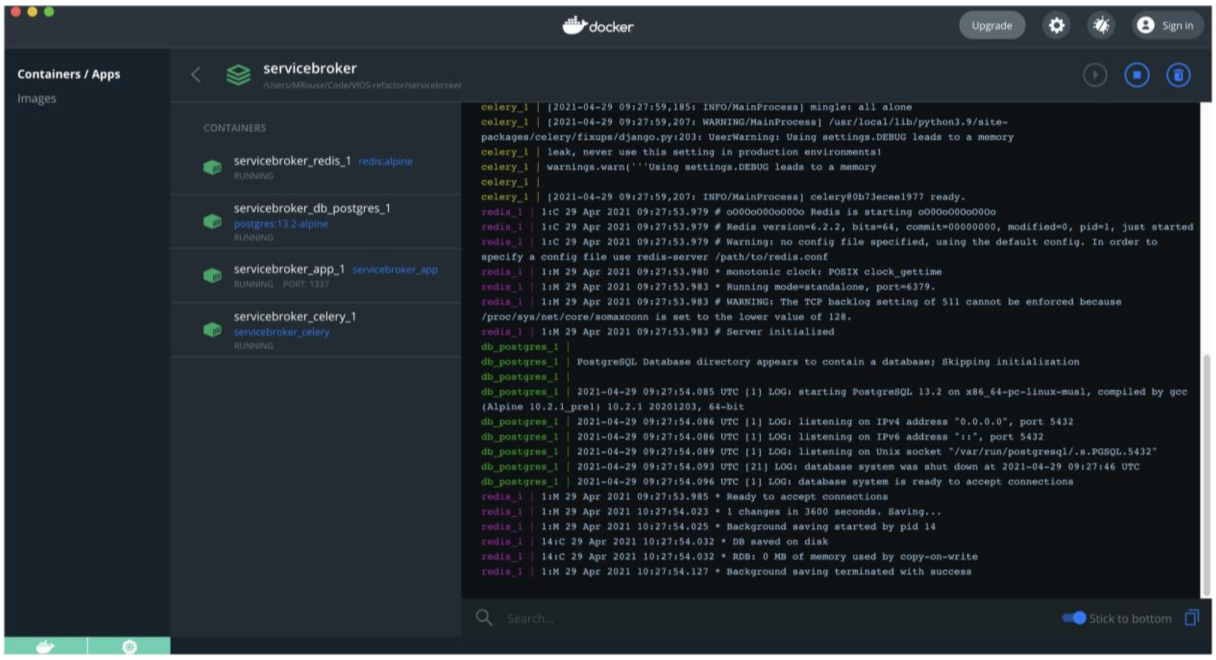
A common repository component is being implemented in 5G-VIOS, which would be able to interact with individual NRFs and thus facilitate inter-domain NSs. Detailed explanation of this component is presented in section 3.4.

2.3 Deployment

As described in Section 2.1, 5G-VIOS has adopted a cloud native architecture with each component being developed as a container. This section explains this and the tools used to automate the development and deployment of 5G-VIOS.

2.3.1 Docker containers

Each microservice application is maintained within its own Docker container, as shown in the example in Figure 2-8. This allows us to keep all the necessary dependencies and libraries needed together in one location, while making it easier to distribute the microservice. The Docker image for each microservice is stored as a package within its corresponding Git repository for centralised access when deploying 5G-VIOS micro-services to a container orchestrator like Kubernetes cluster.



```
celery_1 | [2021-04-29 09:27:59,185: INFO/MainProcess] mingle: all alone
celery_1 | [2021-04-29 09:27:59,207: WARNING/MainProcess] /usr/local/lib/python3.9/site-
packages/celery/fixups/django.py:203: UserWarning: Using settings.DEBUG leads to a memory
celery_1 | leak, never use this setting in production environments!
celery_1 | warnings.warn('''Using settings.DEBUG leads to a memory
celery_1 |
celery_1 | [2021-04-29 09:27:59,207: INFO/MainProcess] celery@0b73ecee1977 ready.
redis_1 | 1:C 29 Apr 2021 09:27:53.979 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
redis_1 | 1:C 29 Apr 2021 09:27:53.979 # Redis version=6.2.2, bits=64, commit=00000000, modified=0, pid=1, just started
redis_1 | 1:C 29 Apr 2021 09:27:53.979 # Warning: no config file specified, using the default config. In order to
specify a config file use redis-server /path/to/redis.conf
redis_1 | 1:M 29 Apr 2021 09:27:53.980 * monotonic clock: POSIX clock_gettime
redis_1 | 1:M 29 Apr 2021 09:27:53.983 * Running mode=standalone, port=6379.
redis_1 | 1:M 29 Apr 2021 09:27:53.983 # WARNING: The TCP backlog setting of 511 cannot be enforced because
/proc/sys/net/core/somaxconn is set to the lower value of 128.
redis_1 | 1:M 29 Apr 2021 09:27:53.983 # Server initialized
db_postgres_1 |
db_postgres_1 | PostgreSQL Database directory appears to contain a database; Skipping initialization
db_postgres_1 |
db_postgres_1 | 2021-04-29 09:27:54.085 UTC [1] LOG: starting PostgreSQL 13.2 on x86_64-pc-linux-musl, compiled by gcc
(Alpine 10.2.1_pre1) 10.2.1 20201203, 64-bit
db_postgres_1 | 2021-04-29 09:27:54.086 UTC [1] LOG: listening on IPv4 address "0.0.0.0", port 5432
db_postgres_1 | 2021-04-29 09:27:54.086 UTC [1] LOG: listening on IPv6 address "::", port 5432
db_postgres_1 | 2021-04-29 09:27:54.089 UTC [1] LOG: listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
db_postgres_1 | 2021-04-29 09:27:54.093 UTC [21] LOG: database system was shut down at 2021-04-29 09:27:46 UTC
db_postgres_1 | 2021-04-29 09:27:54.096 UTC [1] LOG: database system is ready to accept connections
redis_1 | 1:M 29 Apr 2021 09:27:53.985 * Ready to accept connections
redis_1 | 1:M 29 Apr 2021 10:27:54.023 * 1 changes in 3600 seconds. Saving...
redis_1 | 1:M 29 Apr 2021 10:27:54.025 * Background saving started by pid 14
redis_1 | 14:C 29 Apr 2021 10:27:54.032 * DB saved on disk
redis_1 | 14:C 29 Apr 2021 10:27:54.032 * RDB: 0 MB of memory used by copy-on-write
redis_1 | 1:M 29 Apr 2021 10:27:54.127 * Background saving terminated with success
```

Figure 2-8 5G-VIOS Docker container example

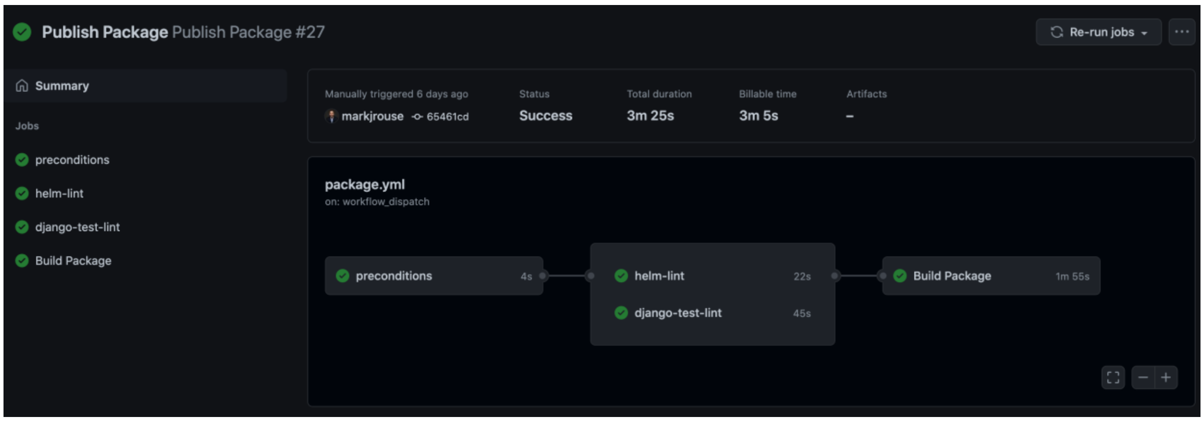
2.3.2 Continuous Integration (CI)

We use GitHub Actions to provide workflows, as shown in the example below in Figure 2-9, for what should happen when new code is posted to its respective repository. This gives us a Continuous Integration (CI) process that will automatically:

a) run any unit testing for a microservice,

b) perform code linting to ensure compliance to coding standards and errors are identified,
c) run any necessarily dependence checks, and
d) build a new Docker container to reflect the latest version of a microservice.

Each step in the workflow is dependent on the previous one, and a new Docker container cannot be built if any errors in the workflow are reported.



Publish Package Publish Package #27

Manually triggered 6 days ago markjrouse 65461cd
Status: Success
Total duration: 3m 25s
Billable time: 3m 5s
Artifacts: –

Jobs: preconditions, helm-lint, django-test-lint, Build Package

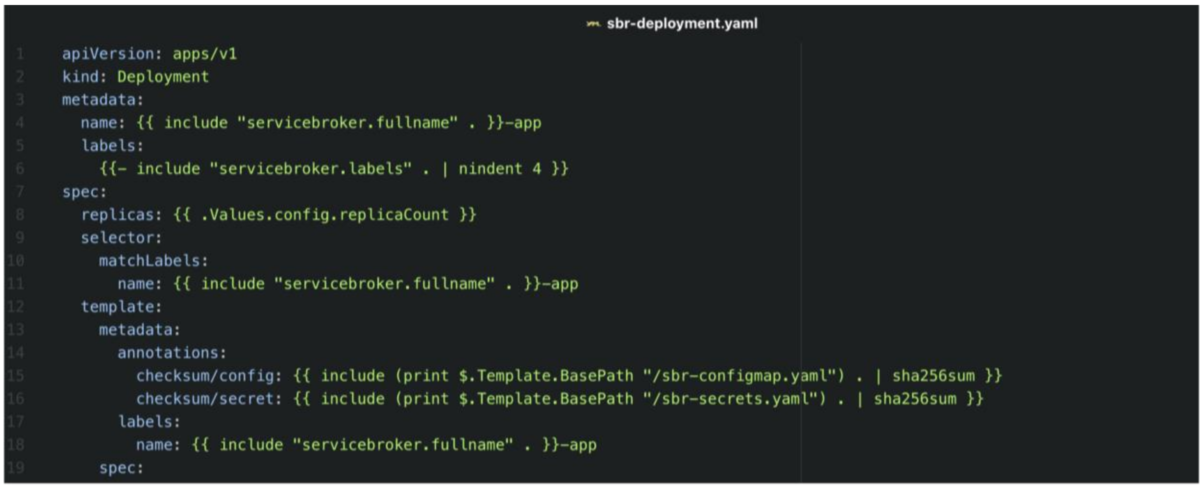
package.yml
on: workflow_dispatch

preconditions 4s → helm-lint 22s, django-test-lint 45s → Build Package 1m 55s

Figure 2-9 GitHub Actions example for 5G-VIOS deployment

2.3.3 Helm Charts

We use Helm Charts to provide a declarative way, as illustrated in Figure 2-10, to consistently install each microservice into a container orchestrator like Kubernetes. Each chart is maintained within its corresponding Git repository to allow us to version control a microservice's installation and configuration files.



sbr-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: {{ include "servicebroker.fullname" . }}-app
  labels:
    {{- include "servicebroker.labels" . | nindent 4 }}
spec:
  replicas: {{ .Values.config.replicaCount }}
  selector:
    matchLabels:
      name: {{ include "servicebroker.fullname" . }}-app
  template:
    metadata:
      annotations:
        checksum/config: {{ include (print $.Template.BasePath "/sbr-configmap.yaml") . | sha256sum }}
        checksum/secret: {{ include (print $.Template.BasePath "/sbr-secrets.yaml") . | sha256sum }}
      labels:
        name: {{ include "servicebroker.fullname" . }}-app
    spec:
```

Figure 2-10 Helm Chart example for declarative installation

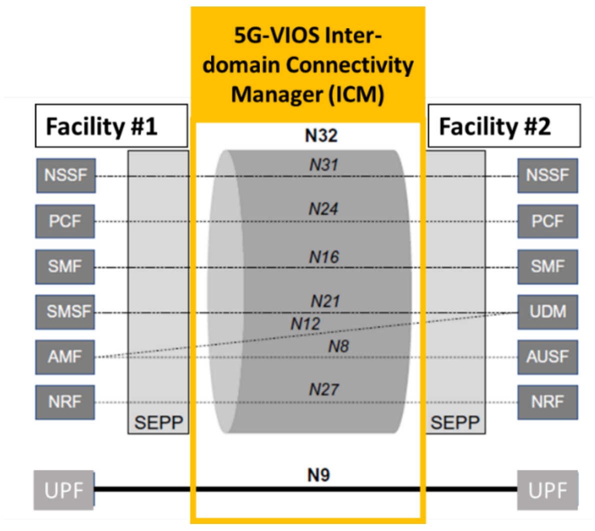


Figure 2-11 5G-VIOS cross facility connectivity

2.3.4 Deployment Options

A key requirement for 5G-VIOS was the support of cloud native principles and the Continuous Integration and Continuous Delivery (CI/CD) pipeline that is adopted in a containerised environment for this reason. This allows us to be flexible with the deployment of 5G-VIOS on any infrastructure depending on the service requirements and the restrictions imposed by operators.

Specifically, two options are considered in the 5G-VICTORI project: a) to deploy it on a hyper-scale cloud such as Amazon Web Services (AWS) or Google Cloud, or (b) to deploy it on-premise at the 5G-VICTORI facilities. Following the hub & spoke model architecture of DMVPN, as explained in section 2.1, the hub of this infrastructure will run 5G-VIOS, and each edge becomes a spoke. The requirement for this solution to work, is that the hub and spokes are accessible, either through public IPs or dedicated secure links. This would allow the Edge Proxy to communicate with the API Gateway in 5G-VIOS and enable the ICM to dynamically build the mesh network using DMVPN. The Hub-to-Spoke links would be used primarily for control, i.e. carry the N32 interface traffic, and the Spoke-to-Spoke would enable UPF to UPF connectivity over N9, as illustrated in Figure 2-11.

The first option would see that 5G-VIOS is deployed on a 3rd party infrastructure, such as AWS or Google cloud as illustrated in Figure 2-12. This makes 5G-VIOS easily available for all facilities as it is on a public domain, but introduces security concerns even though secure connections are used to communicate between 5G-VIOS and the edges. This can be seen as a 3rd party exchange service being offered to network operators in order to peer their services, like internet connectivity exchange services, with many to many relationships, as operators could get access to more than one peer.

The second option of deploying 5G-VIOS on-premise, such as at the Bristol 5G testbed (Figure 2-13), makes connectivity control easier as the facilities and services are operating on local networks. It can also potentially exploit the GÉANT infrastructure for the interconnectivity of the facilities, where it exists. For scenarios and use cases that are not involving other facilities, that option would be more sensible, as everything falls under one administrative domain. However, the principle design is not different from the first option, i.e. 5G-VIOS would still have to be deployed on a cloud infrastructure hosted by the facility. In addition, for the Bristol Facility there will be multiple edges across the facility where mobility scenarios are going to be tested. In the future, this can be seen as one network operator offering this service to other operators to facilitate inter-domain connectivity on a 1:1 service.

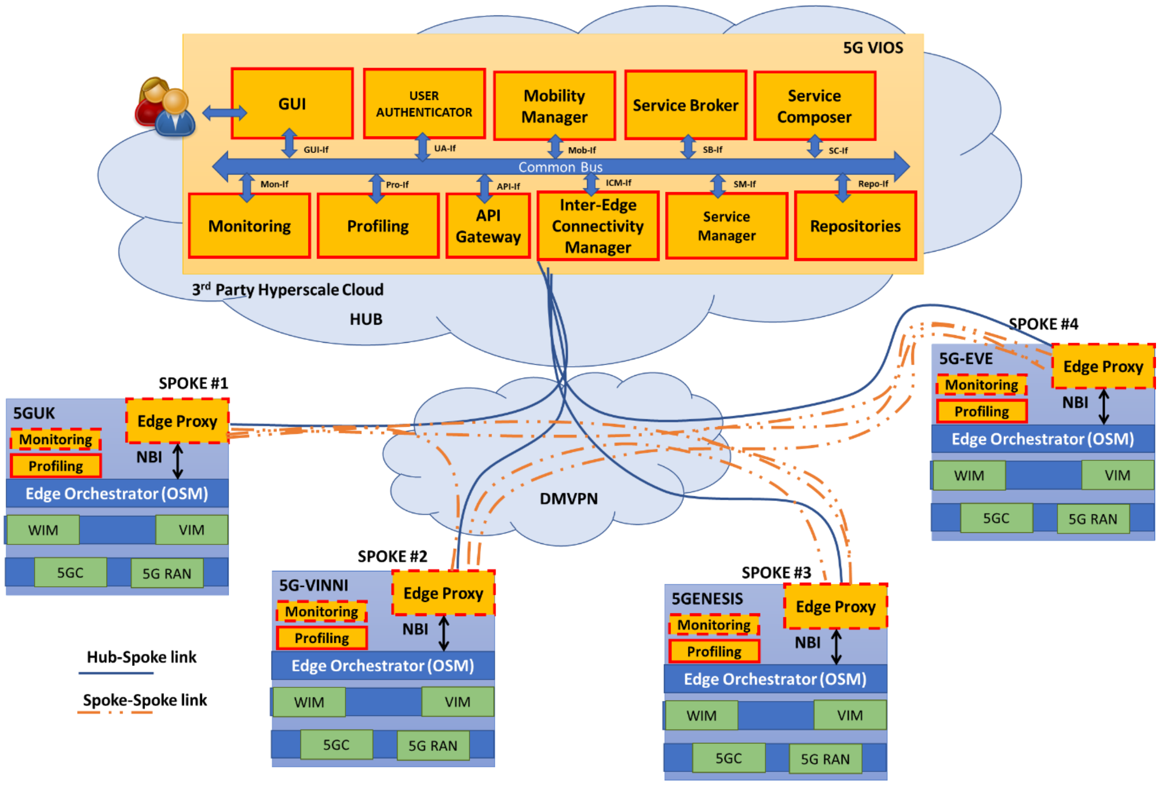


Figure 2-12 5G-VIOS deployment on a 3rd Party cloud infrastructure

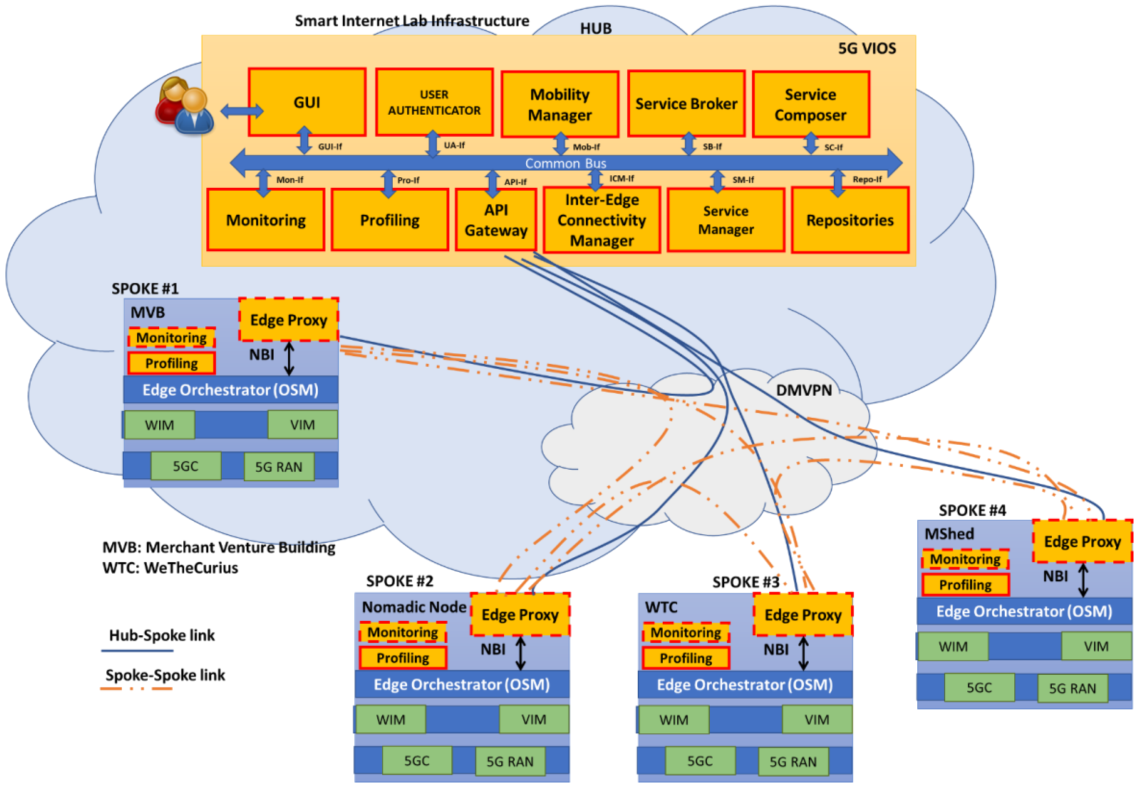


Figure 2-13 5G-VIOS on-premises deployment example

The integration of 5G-VIOS with each 5G VICTORI facility and edges within facilities is based on two principles. The first principle is the deployment of an Edge Proxy instance that interfaces southbound with the orchestrator's NorthBound Interface (NBI), the 5GC common bus where needed so the SEPP can link to NRF and the UPF. The second principle involves a routable IP address that is used from the Edge Proxy to establish secure DMVPN connections with the hub and the other spokes. Specifically, we have integrated with OSM Rel.9 NBI[2]. OSM is used in all four 5G-VICTORI facilities (Bristol, Patras, Berlin and Alba Iulia). Appropriate routing and security policies should be implemented to allow the 5GC common bus and N9 reference point for UPF to pass the traffic through the Edge Proxy and connect through the other Edge Proxy to a second 5GC.

[2] NOTE: Since OSM Release NINE, the information model has changed following SOL006 while the NBI is still the same (SOL005). More details on requirements are detailed in the Edge Proxy component (section 3.7).

3 5G-VIOS Components

This section describes each one of the 5G-VIOS components, their functionality and implementation, including the APIs and endpoints they offer. The components are available at the 5G-VICTORI GitHub repository [4].

3.1 Service Broker (SBR)

3.1.1 Description

The Service Broker (SBR) is at the heart of 5G-VIOS to broker NSs across multiple edges/domains. As explained in deliverable D2.5 [1] in section 2.3.2, the SBR acts as an intermediary between the Edge Proxy and rest of the 5G-VIOS components. It receives the Edge registration requests, which results in creating the required credentials and sending them back to the Edge Proxy.

The SBR aims to fulfil the functional requirements for the 5G-VIOS platform (see Table 3-1).

Table 3-1 SBR Functional Requirements

ID	Functional Requirement	Description
SBR1	Initiate a network service	Initiate a network service for the SMA
SBR2	Deploy a network service	Deploy network service for the SMA
SBR3	Terminate a network service	Terminate network service for the SMA
SBR4	Delete an edge	Delete an edge from 5G-VIOS
SBR5	Register an edge	Register a new edge with 5G-VIOS

3.1.2 Implementation

The SBR currently uses the components listed in Table 3-2 to deliver its endpoint functionality.

Table 3-2 SBR implementation technologies

Technology	Version	Description
Python	3.9.2-alpine	Base Docker image
Django	3.2.0	Base Python web framework
Django REST	3.12.4	Python REST API framework
Celery	5.1.2	Asynchronous distributed task queue
Redis	3.5.3	In-memory data structure store used for caching with Celery
PostgreSQL	13.2-alpine	Production grade high-performant relational database
Uvicorn	0.15.0	Python ASGI HTTP server
Aiohttp	3.7.4	Asynchronous HTTP Client for asyncio and Python

The SBR supports its operations with the use of two stateful tables in a PostgreSQL database.

3.1.2.1 Edge Registry

The Edge Registry table stores a record of all the edges registered within 5G-VIOS through the GUI component.

Table 3-3 SBR Edge Registry Data Model

Field Name	Data Type	Length	Description	Expected Value(s)	Allow Null
edge_id	UUID Field	36	Primary Key	A UUID	No
egde_name	Char Field	255	A unique name for the edge	A name	No
edge_ip_address	Char Field	255	The IP address of the edge	An IP address	No
edge_port	Char Field	255	The port number of the edge	A port number	No
edge_create_user	UUID Field	36	A link to user	A UUID	No
edge_con_id	UUID Field	36	A link to the Connection Registry table on the AGA	A UUID	No
edge_picons_id	UUID Field	36	A link to the PI Connection Registry table on the Connectivity Manager	A UUID	No
edge_nsd_id	UUID Field	36	A link to the NSD Catalog table on the REP	A UUID	No
edge_vnfd_id	UUID Field	36	A link to the VNFD Catalog table on the REP	A UUID	No
edge_cert_status	Char Field	255	Identifies if the edge's corresponding cert has been generated on the AGA	Pending, Generated, Failed	No
edge_create_time	Date Time Field	N/A	The datetime the record was created	A datetime stamp	No

3.1.2.2 NS Request Registry

The NS Request Registry table maintains a record of all the NS requests sent to the SBR by the SMA (see Table 3-4).

Table 3-4 SBR NS Request Registry Data Model

Field Name	Data Type	Length	Description	Expected Value(s)	Allow Null
nsr_id	UUID Field	36	Primary Key	A UUID	No
nsr_service_id	UUID Field	36	A link to the Service Registry table on the SMA	A UUID	No
nsr_experiment_id	UUID Field	36	Denotes the name id	A UUID	No
nsr_experiment_name	Char Field	255	Denotes the name of the experiment	A string	No
nsr_service_insd	JSON Field	N/A	The fully composed network service	JSON object	No
nsr_create_user	UUID Field	36	A link to user	A UUID	No
nsr_request_type	Char Field	255	Denotes the type of NSR	Standard NSR, Migrate NSR	No
nsr_current_status	Char Field	255	A control flag used by the SMA to instruct the SBR to perform several NSR actions	Pending, Initiating, Initiated, Deploying, Deployed,Terminating, Terminated	No
nsr_aga_status	Char Field	255	Identifies if the NSR has been registered with the AGA	Pending,Registered, Failed	Yes

nsr_epas_id	JSON Field	N/A	A link to the EP Activities Registry table on the AGA	JSON object	Yes
nsr_endpoints	JSON Field	N/A	Identifies what edges are to be used in the service	JSON object	Yes
nsr_migrate_stub	JSON Field	N/A	Identifies what migration data is to be applied to an existing nsr	JSON object	Yes
nsr_create_time	Date Time Field	N/A	The datetime the record was created	A datetime stamp	No

3.1.3 Operational Conditions

There are several conditions that apply to the SBR that impacts its functionality (Table 3-5).

Table 3-5 SBR operational conditions

Resource	Condition
Edge	An edge cannot be deleted if it has been used in a NSR
Edge	An edge must have a unique name
NSR	A NSR cannot be deleted if it has been initiated or deployed. It must first be terminated
NSR	A NSR must have only one [nsr_service_id] associated with it

3.1.4 APIs

The SBR provides the endpoints included in Table 3-6.

Table 3-6 SBR APIs

HTTP Method	DB CRUD Method	5G-VIOS Endpoint	Result
{GET}	READ	/api/sbr/edges	Get all edges
{POST}	CREATE	/api/sbr/edges	Register a new edge
{GET}	READ	/api/sbr/edges/{edge_id}	Get edge details for given edge_id
{PATCH}	PARTIAL UPDATE	/api/sbr/edges/{edge_id}	Updates [edge_cert_status] and [edge_con_id] fields
{DELETE}	DELETE	/api/sbr/edges/{edge_id}	Deletes edge if it has not been used in an active iNSD
{GET}	READ	/api/sbr/nsrs	Get all NSRs
{POST}	CREATE	/api/sbr/nsrs	Register a new NSR
{GET}	READ	/api/sbr/nsrs/{nsr_id}	Get NSR details for given nsr_id
{PATCH}	PARTIAL UPDATE	/api/sbr/nsrs/{nsr_id}	Updates the [nsr_current_status] field for given nsr_id
{DELETE}	DELETE	/api/sbr/nsrs/{nsr_id}	Deletes an NSR if that NSR has not been deployed
{PATCH}	PARTIAL UPDATE	/api/sbr/nsrs/{nsr_id}/initiate	Initiate new service for the SMA
{PATCH}	PARTIAL UPDATE	/api/sbr/nsrs/{nsr_id}/deploy	Deploy new service for the SMA
{PATCH}	PARTIAL UPDATE	/api/sbr/nsrs/{nsr_id}/terminate	Terminate service for the SMA
{GET}	READ	/api/sbr/healthcheck	Health Check endpoint for readiness and liveness probes

3.2 Service Manager (SMA)

3.2.1 Description

The Service Manager (SMA) is responsible for the LCMs of inter-edge NSs. It holds information about the capabilities of each testbed (i.e., the NF and NS catalogues). It also keeps information about the NSs that have been instantiated by an experimenter to allow control of their lifecycle. The Manager is also responsible for receiving the experimenter requests which contain information on the requested inter-edge NSs for an experiment and stores the requested NSs in a shared/unshared database and then forwards them to the SBR to deploy the NSs on respective edges which, in turn, the ICM will be utilised to interconnect the NSs on respective edges. During the inter-edge network service *running* phase, it invokes the monitoring component and does the LCM of the NS and update the profiling component if needed. Finally, it is responsible for terminating a running inter-edge NS. In specific circumstances, such as the migration, the Mobility Manager can interact with the Manager and then the ICM to migrate a NS from one edge to the other.

The SMA aims to fulfil the following functional requirements for the 5G-VIOS platform:

Table 3-7 SMA functional requirements

ID	Functional Requirement	Description
SMA1	Initiate a NS	Initiate a NS
SMA2	Deploy a NS	Deploy NS
SMA3	Terminate a NS	Terminate NS
SMA4	Update NS by creating and deleting connections	Migrates a service

3.2.2 Implementation

The SMA currently uses the components in Table 3-8 to deliver its endpoint functionality.

Table 3-8 SMA implementation technologies

Technology	Version	Description
Python	3.9.2-alpine	Base Docker image
Django	3.2.0	Base Python web framework
Django REST	3.12.4	Python REST API framework
Celery	5.1.2	Asynchronous distributed task queue
Redis	3.5.3	In-memory data structure store used for caching with Celery
PostgreSQL	13.2-alpine	Production grade high-performant relational database
Uvicorn	0.15.0	Python ASGI HTTP server
Aiohttp	3.7.4	Asynchronous HTTP Client for asyncio and Python

The SMA supports its operations with the use of two stateful tables in a PostgreSQL database.

3.2.2.1 Service Registry

The Service Registry table stores a record of all the active services registered within 5G-VIOS submitted by the Service Composer.

Table 3-9 SMA Service Registry Data Model

Field Name	Data Type	Length	Description	Expected Value(s)	Allow Null
service_id	UUID Field	36	Primary Key	A UUID	No
service_experiment_id	Char Field	36	A link to the INSD on the Service Composer	A UUID	No
service_insd	JSON Field	N/A	The fully composed network service	JSON object	No
service_create_user	UUID Field	36	A link to user	A UUID	No
service_request_type	Char Field	255	Denotes the type of service	standard nsr, migrate nsr	No
service_pi_id	UUID Field	36	A link to the PI Connection Registry on the Connectivity Manager	A UUID	No
service_si_id	UUID Field	36	A link to the SI Connection Registry on the Connectivity Manager	A UUID	No
service_nsr_id	UUID Field	36	A link to the NS Request Registry on the SBR	A UUID	Yes
service_sbr_status	Char Field	255	Denotes if the service has been registered with the SBR	Pending, Registered	Yes
service_initiate_status	Char Field	255	The initiate action status for a service	Pending, Initiating, Initiated	Yes
service_initiate_create_time	Date Time Field	N/A	The datetime the action occurred	A datetime stamp	Yes
service_deploy_status	Char Field	255	The deploy action status for a service	Pending, Deploying, Deployed	Yes
service_deploy_create_time	Date Time Field	N/A	The datetime the action occurred	A datetime stamp	Yes
service_terminate_status	Char Field	255	The terminate action status for a service	Pending, Terminating, Terminated	Yes
service_terminate_create_time	Date Time Field	N/A	The datetime the action occurred	A datetime stamp	Yes
service_control_status	Char Field	255	The control flag for running actions against a service	Pending, Initiate, Deploy, Terminate	Yes
service_create_time	Date Time Field	N/A	The datetime the record was created	A datetime stamp	No

3.2.2.2 Migration Registry

The Migration Registry table (see Table 3-10) maintains a record of additional details of the migrate nsr requests sent to the SMA by the Service Composer to show the links between the original and new service.

Table 3-10 SMA Migration Registry Data Model

Field Name	Data Type	Length	Description	Expected Value(s)	Allow Null
migrate_id	UUID Field	36	Primary Key	A UUID	No
migrate_original_service_id	UUID Field	36	The original [service_id] of the service to be migrated	A UUID	No
migrate_new_service_id	UUID Field	36	The new [service_id] of the service	A UUID	No
migrate_original_ns_id	UUID Field	36	A link to NS Registry on the Service Composer	A UUID	No
migrate_new_ns_id	UUID Field	36	A link to NS Registry on the Service Composer	A UUID	No
migrate_migrate_stub	JSON Field	N/A	The original and new edge_ids as part of the service migration	JSON Object	Yes
migrate_processed_status	Char Field	255	N/A	Pending, Processed	No
migrate_create_time	Date Time Field	N/A	The datetime the record was created	A datetime stamp	No

3.2.3 Operational Conditions

There are several conditions that apply to the SMA that impacts its functionality:

Table 3-11 SMA operational conditions

Resource	Condition
Service	A service cannot be deleted if it has been initiated or deployed. It must first be terminated
Service	A service must have only one [service_experiment_id] associated with it
Service	A service cannot be deployed if it’s not already in an initiated state
Service	A service cannot be terminated if it’s not already in a deployed state
Migrate	A [migrate_stub] cannot be sent to the SBR if the corresponding [service_id] does not have a [service_sbr_status] of Registered

3.2.4 APIs

The SMA provides the endpoints included in Table 3-12.

Table 3-12 SMA APIs

HTTP Method	DB CRUD Method	5G-VIOS Endpoint	Result
{GET}	READ	/api/sma/services	Get all services
{POST}	CREATE	/api/sma/services	Create a new service

{GET}	READ	/api/sma/services/{service_id}	Get service details for given service_id
{PATCH}	PARTIAL UPDATE	/api/sma/services/{service_id}	Used to update several control status fields
{DELETE}	DELETE	/api/sma/services/{service_id}	Deletes service if it has not part of an active deployment
{PATCH}	PARTIAL UPDATE	/api/sma/services/{service_id} /instantiate	Instructs the SBR to instantiate a service
{PATCH}	PARTIAL UPDATE	/api/sma/services/{service_id} /terminate	Instructs the SBR to terminate a service
{GET}	READ	/api/sma/migrations	Get all current migrate NSR requests
{POST}	CREATE	/api/sma/migrations	Create a new migrate NSR request
{GET}	READ	/api/sma/migrations/{migrate_ id}	Get migrate NSR details for given migrate_id
{PATCH}	PARTIAL UPDATE	/api/sma/migrations/{migrate_ id}	Updates [migrate_processed_status] field
{GET}	READ	/api/sma/healthcheck	Health Check endpoint for readiness and liveness probes

3.3 API Gateway (AGA)

3.3.1 Description

The API Gateway (AGA) component acts as the main entry point from the individual edge instances towards 5G-VIOS. It takes all API calls from clients, then routes them to the appropriate microservice with request routing, composition, and protocol translation. Typically, it handles a request by invoking multiple microservices and aggregating the results, to determine the best path. It can translate between protocols that are used internally. This is also known as a dispatcher.

It is exposing a Southbound API that talks with NBI of MANO and applications/services directly through the Edge Proxy, taking care of RBAC policies,

This component allows applications and services running within individual edge domains to interact with 5G-VIOS components or specific services and vice versa through the provided APIs. For instance, the applications can request the migration of one NF from one edge to the other edge utilising the provided APIs, or report KPIs to the monitoring component.

The AGA aims to fulfil the functional requirements for the 5G-VIOS platform from Table 3-13.

Table 3-13 AGA functional requirements

ID	Functional Requirement	Description
AGA1	Performs Edge Proxy heartbeat checks	Pings each Edge Proxy to ensure it's still contactable
AGA2	Manages Edge Proxy connections	Maintains a record of active connections
AGA3	Manages service activities with each Edge Proxy	Works to initiate, deploy and terminate services
AGA4	Manages SSL certificate generation	Generates SSL certificates for Edge Proxy communications

3.3.2 Implementation

The AGA currently uses the components in Table 3-14 to deliver its endpoint functionality:

Table 3-14 AGA implementation technologies

Technology	Version	Description
Python	3.9.2-alpine	Base Docker image
Django	3.2.0	Base Python web framework
Django REST	3.12.4	Python REST API framework
Celery	5.1.2	Asynchronous distributed task queue
Redis	3.5.3	In-memory data structure store used for caching with Celery
PostgreSQL	13.2-alpine	Production grade high-performant relational database
Uvicorn	0.15.0	Python ASGI HTTP server
Aiohttp	3.7.4	Asynchronous HTTP Client for asyncio and Python

The AGA supports its operations with the use of two stateful tables in a PostgreSQL database.

3.3.2.1 EP Activity Registry

The EP Activity Registry table stores a record of all the registered NS in 5G-VIOS at an Edge Proxy level. This means that a NS using more than one edge will have more than one record in the EP Activity Registry but linked via the nsr_id from the SBR.

Table 3-15 AGA EP Activity Registry Data Model

Field Name	Data Type	Length	Description	Expected Value(s)	Allow Null
epas_id	UUID Field	36	Primary Key	A UUID	No
epas_nsr_id	UUID Field	36	A link to the NS Requests Registry table on the SBR	A UUID	No
epas_edge_id	UUID Field	36	A link to the Edge Registry table on the SBR	A UUID	No
epas_request_type	Char Field	255	Denotes the type of NSR	Standard NSR, Migrate NSR	No
epas_create_user	UUID Field	36	A link to a user	A UUID	No
epas_switch_id	Integer Field	N/A	The id of the switch	An id number	No
epas_switch_port	Integer Field	N/A	The port number of the switch	A port number	No
epas_vlan_id	Integer Field	N/A	The vlan id	An id number	No
epas_epa_instance_id	UUID Field	36	A link to the NSR table on the Edge Proxy	A UUID	No
epas_epa_instance_status	Char Field	255	Identifies if the NSR has been registered with the Edge Proxy	Pending, Registered, Failed	No
epas_control_status	Char Field	255	A control flag used by the AGA to either initiate, deploy, or terminate a service	Initiate, Deploy, Terminate	No
epas_create_time	Date Time Field	N/A	The datetime the record was created	A datetime stamp	No

3.3.2.2 EP Connection Registry

The EP Connection Registry table stores a record of all the connection details of each edge registered in 5G-VIOS (see Table 3-16).

Table 3-16 AGA EP Connection Registry Data Model

Field Name	Data Type	Length	Description	Expected Value(s)	Allow Null
epcons_id	UUID Field	36	Primary Key	A UUID	No
epcons_edge_id	UUID Field	36	A link to the Edge Registry table on the SBR	A UUID	No
epcons_edge_name	Char Field	255	A unique name for the edge	A name	No
epcons_edge_ip_address	Char Field	255	The IP address of the edge	An IP address	No
epcons_edge_port	Integer Field	N/A	The port number of the edge	A port number	No
epcons_create_user	UUID Field	36	A link to a user	A UUID	No
epcons_epa_host_id	UUID Field	36	An id link to the Edge table on the Edge Proxy	A UUID	No
epcons_epa_host_status	Char Field	255	Identifies if the edge has been registered with the Edge Proxy	Pending, Registered, Failed	No
epcons_connection_status	Char Field	255	A control flag used by the AGA to determine the active state of an edge	Pending, Online, Offline	No
epcons_create_time	Date Time Field	N/A	The datetime the record was created	A datetime stamp	No

3.3.3 APIs

The AGA provides the endpoints in Table 3-17.

Table 3-17 AGA APIs

HTTP Method	DB CRUD Method	5G-VIOS Endpoint	Result
{GET}	READ	/api/aga/activites	Gets all service activity entries in the EPActivitiesRegistry
{POST}	CREATE	/api/aga/activites	Creates a new Edge Proxy activity entry when initiating a service
{GET}	READ	/api/aga/activites/{epas_id}	Gets specified Edge Proxy activity details for a given epas_id
{PATCH}	PARTIAL UPDATE	/api/aga/activites/{epas_id}	Updates [epas_epa_session_status] and [epas_epa_session_id] fields
{DELETE}	DELETE	/api/aga/activites/{epas_id}	Deletes a specified Edge Proxy activity for the given epas_id
{PATCH}	PARTIAL UPDATE	/api/aga/activites/{epas_id}/initiate	Initiate a service
{PATCH}	PARTIAL UPDATE	/api/aga/activites/{epas_id}/deploy	Deploy a service
{PATCH}	PARTIAL UPDATE	/api/aga/activites/{epas_id}/terminate	Terminate a service

{POST}	N/A	/api/aga/activites/{epas_id}/performance	Passes performance data from the Edge Proxy to the Profiler
{POST}	N/A	/api/aga/activites/{epas_id}/resources	Passes optimum resources data from the Edge Proxy to the Profiler
{POST}	N/A	/api/aga/activites/{epas_id}/metrics	Passes monitoring metrics data from the Edge Proxy to the Monitor
{POST}	N/A	/api/aga/activites/{epas_id}/migrate	Passes migrate requests from the Edge Proxy to the Mobility Manager
{GET}	READ	/api/aga/connections	Get all Edge Proxy connection records in the EPConnectionRegistry
{POST}	CREATE	/api/aga/connections	Register a new Edge Proxy connection
{GET}	READ	/api/aga/connections/{epcons_id}	Get Edge Proxy connection details for a given epcons_id
{PATCH}	PARTIAL UPDATE	/api/aga/connections/{epcons_id}	Updates the [epcons_epa_host_status] and [epcons_connection_status] fields for a given epcons_id
{DELETE}	DELETE	/api/aga/connections/{epcons_id}	Deletes an Edge Proxy connection entry for a given epcons_id
{GET}	READ	/api/aga/healthcheck	Health Check endpoint for readiness and liveness probes

3.4 Repository (REP)

3.4.1 Description

The repository is the key component of the 5G-VIOS framework which is an upgraded version of the NRF [5] that includes all the functionalities of NRF and the information needed to support the different facilities. More specifically, the repository keeps the following information:

1. NSs exposed by all the facilities: The 5G-VIOS repository act as a centralised NRF maintaining the information on the NFs available at the various edges. To enable this, the NRF at each facility (or edge) exposes the available NSs to 5G-VIOS by registering itself with the 5G-VIOS repository.
2. Registration of Edge Proxy: The 5G-VIOS Edge Proxy (an extended version of the SEPP – see section 3.7) available at each facility (or edge) registers itself with the 5G-VIOS repository.
3. Network service descriptor (NSD) and virtual network function descriptor (VNFD): The repository keeps the information on all the onboarded NS descriptors (NSDs) and VNFDs corresponding to each edge orchestrator.

Some NFs that are necessary for the operation of vertical and cross vertical industries (i.e. synchronization, positioning, signalling, voice, etc.), as well as the NS Catalogues, i.e., available Ns on each edge/cluster in the form of NSDs/VNFDs including the required images will be part of these repositories. Each Edge/cluster will expose a set of available vertical specific VNFs/PNFs that will be packaged and exposed through function Repositories. Then, the experimenter can invoke the Service Composer to request an inter-edge NS comprising of selected NFs belonging to different domains/site/edges provided in the Repositories. Also, if allowed by the edge policies, in case that an update occurs in the VNFs/ NSs in an Edge/Cluster, the Repositories shall also be updated.

The Repository (REP) aims to fulfil the functional requirements for the 5G-VIOS platform included in Table 3-18.

Table 3-18 REP functional requirements

ID	Functional Requirement	Description
REP1	Needs to store an NSD	Stores each registered edge's NSD catalogue
REP2	Needs to store an VNFD	Stores each registered edge's VNFD catalogue
REP3	Needs to support registration of edge proxy and NRF from each edge	Edge proxy and NRF from each edge (facility) register with the repository and store information on network functions available at each edge.

3.4.2 Implementation

The REP currently uses the components in Table 3-19 to deliver its endpoint functionality:

Table 3-19 REP implementation technologies

Technology	Version	Description
Python	3.9.2-alpine	Base Docker image
Django	3.2.0	Base Python web framework
Django REST	3.12.4	Python REST API framework
Celery	5.1.2	Asynchronous distributed task queue
Redis	3.5.3	In-memory data structure store used for caching with Celery
PostgreSQL	13.2-alpine	Production grade high-performant relational database
Uvicorn	0.15.0	Python ASGI HTTP server
Aiohttp	3.7.4	Asynchronous HTTP Client for asyncio and Python

The REP currently supports its operations with the use of two stateful tables in a PostgreSQL database.

3.4.2.1 NSD Catalogue

The NSD Catalogue table stores a record of all NSDs for each edge registered in 5G-VIOS (see Table 3-20).

Table 3-20 REP NSD Catalogue Data Model

Field Name	Data Type	Length	Description	Expected Value(s)	Allow Null
nsd_catalogue_id	UUID Field	36	Primary Key	A UUID	No
nsd_catalogue_edge_id	UUID Field	36	An id link to the Edge Registry on the SBR	A UUID	No
nsd_catalogue_descriptor	JSON Field	N/A	The JSON content of the NSD	An NSD	No
nsd_catalogue_create_user	UUID Field	36	A link to a user	A UUID	No

nsd_catalogue_create_time	Date Time Field	N/A	The datetime the record was created	A datetime stamp	No

3.4.2.2 VNFD Catalogue

The VNFD Catalogue table stores a record of all VNFDs for each edge registered in 5G-VIOS.

Table 3-21 REP VNFD Catalogue Data Model

Field Name	Data Type	Length	Description	Expected Value(s)	Allow Null
vnfd_catalogue_id	UUID Field	36	Primary Key	A UUID	No
vnfd_catalogue_edge_id	UUID Field	36	An id link to the Edge Registry on the SBR	A UUID	No
vnfd_catalogue_descriptor	JSON Field	N/A	The JSON content of the VNFD	A VNFD	No
vnfd_catalogue_create_user	UUID Field	36	A link to a user	A UUID	No
vnfd_catalogue_create_time	Date Time Field	N/A	The datetime the record was created	A datetime stamp	No

3.4.3 APIs

The REP provides the endpoints included in Table 3-22.

Table 3-22 REP APIs

HTTP Method	DB CRUD Method	5G-VIOS Endpoint	Result
{GET}	READ	/api/rep/nsds	Get all NSD records
{POST}	CREATE	/api/rep/nsds	Create a new NSD record
{GET}	READ	/api/rep/nsds/{nsd_catalogue_id}	Get NSD details for given nsd_catalogue_id
{DELETE}	DELETE	/api/rep/nsds/{nsd_catalogue_id}	Deletes an NSD
{GET}	READ	/api/rep/vnfds	Get all VNFD records
{POST}	CREATE	/api/rep/vnfds	Create a new VNFD record
{GET}	READ	/api/rep/vnfds/{vnfd_catalogue_id}	Get VNFD details for given vnfd_catalogue_id
{DELETE}	DELETE	/api/rep/vnfds/{vnfd_catalogue_id}	Deletes a VNFD
{GET}	READ	/api/rep/heathcheck	Health Check endpoint for readiness and liveness probes

3.5 Inter-edge Connectivity Manager (ICM)

3.5.1 Description

The ICM component performs the dynamic interconnection among the domains/sites that is needed to enable (or disable) an E2E NS by configuring the interconnection infrastructure appropriately. It has two main responsibilities: (a) serving as a bootstrapping point by setting up the control plane of the edge and connecting it to the 5G-VIOS and secondly, (b) being responsible for creating the data path between the edges when a service is deployed. The ICM will interface with the edge proxy to form a secure link between 5G-VIOS and each edge using the underlying interconnection facility.

To enable the control plane between 5G-VIOS and all the edges and the data plane between distinct edges, we have used DMVPN in the 5G-VIOS framework as introduced in Section 2.1. DMVPN creates a hub-spoke mesh network topology in which each edge (i.e., spoke) is configured to connect with the connectivity manager (i.e., the hub) to provide access to the required resources. Additionally, each edge can communicate directly to the other edge, irrespective of their location and without going through the ICM components (hub). The DMVPN has been configured using the four key components:

1. Multipoint GRE (mGRE) tunnel interface: DMVPN with mGRE allows to add multiple destinations using only one tunnel interface on the router/edge. However, if an edge needs to tunnel traffic to another edge using mGRE or point-to-point may, it may not have information about which IP address to use as a destination. In DMVPN, the Next Hop Resolution Protocol (NHRP) is used for IP address resolution.
2. Next Hop Resolution Protocol (NHRP): An IP address has been assigned to each edge using NHRP. Each edge is connected to the ICM component which works as a central hub for all the edges. NHRP works as a client-server model in which ICM functions as a server while all the edges are clients. Each edge registers with the ICM with its public IP address. Through a process that involves registration and resolution requests from the client edges and resolution replies from the server ICM, traffic is enabled between different edges in the 5G-VIOS framework. NHRP works in three phases. In Phase 1, each edge registers with the ICM component and there is no direct communication between edges as all traffic goes through ICM. In Phase 2, all edge routers use mGRE to establish edge-to-edge connections. When an edge wants to establish a connection with the other edge, it sends an NHRP resolution request to ICM to find out the IP address of the other edge. Then, edge-to-edge tunnels are deployed on-demand without using any specific pre-made routes. Phase 3 builds on Phase 2 for improving scalability.
3. IPsec Tunnel: IPsec has been used to encrypt the traffic between edges.
4. Routing Protocol: Routing protocol enable the DMVPN to find the routes among network services deployed on different edges. Open Shortest Path First (OSPF) has been used as the interior routing protocol.

To configure the component of DMVPN, a virtual router has been deployed at ICM and at each edge. 5G-VIOS uses VyOS [6] as a virtual router that is an open-source network operating system based on Debian. It provides a free routing functionality for the bare metal, cloud and virtualised infrastructure.

The ICM aims to fulfil the functional requirements indicated in Table 3-23 for the 5G-VIOS platform.

Table 3-23 ICM functional requirements

ID	Functional Requirement	Description
ICM1	Set up physical interconnections for each edge	Stores switch_id, switch_port and vlan_pools for an edge
ICM2	Set up service interconnections for each NSR	Stores endpoints and calculated vlan_id for each edge in NSR

3.5.2 Implementation

The ICM currently uses the following components to deliver its endpoint functionality:

Table 3-24 ICM implementation technologies

Technology	Version	Description
Python	3.9.2-alpine	Base Docker image
Django	3.2.0	Base Python web framework
Django REST	3.12.4	Python REST API framework
Celery	5.1.2	Asynchronous distributed task queue
Redis	3.5.3	In-memory data structure store used for caching with Celery
PostgreSQL	13.2-alpine	Production grade high-performant relational database
Uvicorn	0.15.0	Python ASGI HTTP server
Aiohttp	3.7.4	Asynchronous HTTP Client for asyncio and Python
VyOS	1.3.0-epa1	Virtual router and VPN services for DMVPN

The ICM support its operations with the use of two stateful tables in a PostgreSQL database.

3.5.2.1 SI Connection Registry

The SI Connection Registry table stores a record of all the service interconnections in 5G-VIOS for each NSR (see Table 3-25).

Table 3-25 ICM SI Connection Registry Data Model

Field Name	Data Type	Length	Description	Expected Value(s)	Allow Null
sicons_id	UUID Field	36	Primary Key	A UUID	No
sicons_nsr_id	UUID Field	36	A link to the NS Requests Registry table on the SBR	A UUID	No
sicons_service_id	UUID Field	36	A link to the Service Registry table on the SMA	A UUID	No
sicons_endpoints	JSON Field	N/A	The endpoint(s) as part of an NSR	JSON containing switch_id, edge_id, switch_port , vlan_id	No
sicons_create_user	UUID Field	36	A link to a user	A UUID	No
sicons_sbr_endpoints_status	Char Field	255	Denotes if the endpoints have been registered with the SBR	Pending, Registered	Yes
sicons_create_time	Date Time Field	N/A	The datetime the record was created	A datetime stamp	No

3.5.2.2 PI Connection Registry

The PI Connection Registry table stores a record of all the physical interconnections in 5G-VIOS for each edge.

Table 3-26 ICM PI Connection Registry Data Model

Field Name	Data Type	Length	Description	Expected Value(s)	Allow Null
picons_id	UUID Field	36	Primary Key	A uuid	No
picons_edge_id	UUID Field	36	A link to the Edge Registry table on the SBR	A uuid	No
picons_create_user	UUID Field	36	A link to a user	A uuid	No
picons_switch_id	Integer Field	N/A	The id of the switch	An id number	No
picons_switch_port	Integer Field	N/A	The port number of the switch	A port number	No
picons_vlan_id	Integer Field	N/A	The vlan id	An id number	No
picons_vlan_pool_min	Integer Field	N/A	The minimum vlan pool range number	An integer	No
picons_vlan_pool_max	Integer Field	N/A	The maximum vlan pool range number	An integer	No
picons_create_time	Date Time Field	N/A	The datetime the record was created	A datetime stamp	No

3.5.3 APIs

The ICM provides the endpoints included in Table 3-27.

Table 3-27 ICM APIs

HTTP Method	DB CRUD Method	5G-VIOS Endpoint	Result
{GET}	READ	/api/icm/sicons	Gets all service interconnections
{POST}	CREATE	/api/icm/sicons	Creates a new service interconnection
{GET}	READ	/api/icm/sicons/{sicons_id}	Gets service interconnection details for a given id
{PATCH}	PARTIAL UPDATE	/api/icm/sicons/{sicons_id}	Updates the [sicons_sbr_endpoints_status] field
{DELETE}	DELETE	/api/icm/sicons/{sicons_id}	Deletes a service interconnection
{GET}	READ	/api/icm/picons	Gets all physical interconnections
{POST}	CREATE	/api/icm/picons	Creates a new physical interconnection
{GET}	READ	/api/icm/picons/{picons_id}	Gets physical interconnection details for a given id
{DELETE}	DELETE	/api/icm/picons/{picons_id}	Deletes a physical interconnection
{GET}	READ	/api/icm/healthcheck	Health Check endpoint for readiness and liveness probes

3.6 Service Composer (SCO)

3.6.1 Description

The objective of the service composer (SCO) is to create the Inter-domain Network Service Descriptor (INSD) by composing (or partially modifying) one or more NSDs. The SCO builds a template for the inter-domain NS that a user can choose to deploy over multiple selected edges. The SCO provides the northbound interface APIs to perform the CRUD operation for INSD. Users from the portal can select multiple NSDs from different edges and pass this information to the SCO to build INSD. After receiving the composed INSD response, users can instantiate the INSD by running experiments.

The SCO intends to fulfill the following functional requirement for the 5G-VIOS platform.

Table 3-28 Functional requirements by the Service Composer

ID	Functional Requirement	Description
SCO1	Create an INSD	Create an INSD based on request received from portal/user
SCO2	Retrieve INSD(s)	Get all INSDs or a INSD by ID
SCO3	Update INSD	Update the existing INSD by ID
SCO4	Delete INSD	Delete the existing INSD by ID

3.6.2 Implementation

The SCO is implemented as a containerized microservice using the tools and technologies included in Table 3-29. The SCO is bundled into the helm chart for deployment over the Kubernetes cluster.

Table 3-29 Technologies/tools used by the Service Composer

Technology	Version	Description
Python	3.9.2-alpine	Base programming language
Django	3.2.0	Base Python web framework
Django REST	3.12.4	Python REST API framework
Django Watchman	1.2.0	Provides a few health check endpoints
Redis	3.5.3	In-memory data structure store used for caching with Celery
PostgreSQL	13.2-alpine	Production grade high-performant relational database
Gunicorn	20.1.0	Python HTTP server
Aiohttp	3.7.4	Asynchronous HTTP Client for asyncio and Python

3.6.3 APIs

The SCO provides the north bound APIs to perform CRUD operation on INSD (see Table 3-30).

Table 3-30 Service Composer APIs

HTTP Method	CRUD Method	5G-VIOS Endpoint	Result
{GET}	READ	/api/sco/insds/	Get all INSDs
{GET}	READ	/api/sco/insds/{id}}/	Get a INSD by id
{POST}	POST	/api/sco/insds/	Create a new INSD
{DELETE}	DELETE	/api/sco/insds/{id}	Deletes INSD by id
{PUT}	UPDATE	/api/sco/insds/{id}	Updates INSD
{GET}	READ	/watchman/ping	Health Check endpoint for Kubernetes readiness and liveness probes

3.7 Edge Proxy (EPA)

3.7.1 Description

The Edge Proxy runs as a microservice at each edge (or facility). It provides a secure connection to exchange information between (i) the 5G-VIOS components and each edge, and (ii) the NFs available at two different edges. More specifically, edge proxy performs the following tasks:

1. Enable communication between 5G-VIOS and edge orchestrator: Edge proxy provides flexibility to 5G-VIOS components to communicate and manage the orchestration platform available at each edge/facility for the LCM of NS. It communicates with the 5G-VIOS components through the API gateway of 5G-VIOS over the IPSec secure link established through DMVPN.
2. Exposes North Bound Interfaces (NBIs): Edge proxy exposes northbound endpoint APIs to enable interaction between the 5G-VIOS components and the edge orchestrator (NFVO), the edge monitoring, the profiler and the facility specific services such as NetOS in Bristol. To achieve this, the edge proxy has defined connectors for the edge orchestrator (e.g., OSM), NetOS, and the edge monitoring and the profiler at the southbound interface. The edge proxy connector uses the northbound RESTful API of OSM, which follows the ESTI NFV SOL005 standard. Information on all the packages onboarded/available at the OSM is also exported to the repository component of 5G-VIOS during the edge registration process. Since release nine, the OSM information model has been fully aligned with ETSI NFV SOL006, based on the Yang model. Presently, edge proxy supports the deployment template based on SOL006. Thus, the edge proxy has assumed that the version of OSM available at the edge is nine or above. In the future, we will extend edge proxy to integrate with the previous version of OSM (mainly version 8) if that is required by specific facility.
3. Security Edge Protection Proxy: 5G-VIOS edge proxy is an extended version of 5G SEPP [7], which enables secure interconnection between 5G NFs available at distinct edges. It uses the N32 reference point interface between SEPPs available at two distinct edges. The edge proxy supports endpoints to all the functionality of SEPP, such as generating access token, bootstrapping, NF discovery, and NF management.

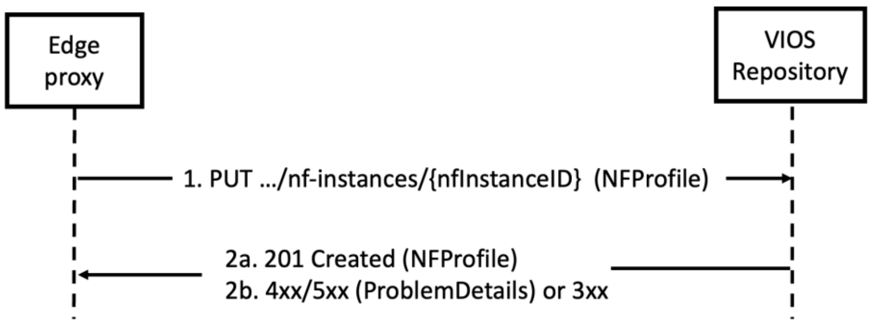

Edge proxy
VIOS Repository
1. PUT .../nf-instances/{nfInstanceID} (NFProfile)
2a. 201 Created (NFProfile)
2b. 4xx/5xx (ProblemDetails) or 3xx

Figure 3-1 Registration of Edge Proxy with VIOS Repository

Each edge register itself with the repository similarly to the SEPP registration with NRF in the 5G architecture. Figure 3-1 depicts the registration process of edge proxy with the 5G-VIOS repository. The edge proxy sends a PUT request to the resource Uniform Resource Identifier (URI) and on success 201 created shall be return. The payload body of the PUT response contains the representation of the created resource, and the location header contains the URI of the created resource. The repository returns 4xx/5xx on failure and 3xx for redirection.

Edge Proxy attempts to meet the functional requirements for the 5G-VIOS platform (Table 3-31).

Table 3-31 Functional Requirement of Edge Proxy

ID	Functional Requirement	Description
EPA1	Register/de-register an edge	register/de-register an edge with the edge proxy.
EPA2	Retrieve NSD	Retrieve all the NSDs onboarded on the edge orchestrator
EPA3	Retrieve VNFD	Get all the VNFDs onboarded on the edge orchestrator
EPA4	Instantiate network service	Instantiate network service to the edge
EPA5	Terminate network service	Terminate a deployed network service on the edge
EPA6	Support SEPP functionality	Access token, bootstrapping, NF discovery, and NF management

3.7.2 Implementation

The edge proxy microservice has been implemented using the tools/technologies included in Table 3-32.

Table 3-32 Technologies/tools used for the Edge Proxy

Technology	Version	Description
Python	3.9.2-alpine	Base programming language
fastapi	0.67.0	FastAPI framework, high performance, easy to learn, fast to code, ready for production
fastapi_utils	0.2.1	Fast api utility to reduce boilerplate and reuse common functionality.
asgiref	3.4.1	ASGI specs, helper code, and adapters
uvicorn	0.14.0	The lightning-fast ASGI server

tornado	6.1	Tornado is a Python web framework and asynchronous networking library, originally developed at FriendFeed
sqlalchemy	1.4.0	Python SQL toolkit and Object Relational Mapper
pytest	6.2.5	Framework to write small tests
flake8	3.8.3	The modular source code checker
aiohttp	3.7.4	Async http client/server framework
requests	2.24.0	Requests is ready for the demands of building robust and reliable HTTP–speaking applications
certifi	2020.12.5	Python package for providing Mozilla's CA Bundle
click	7.1.2	Composable command line interface toolkit
idna	2.10	Internationalized Domain Names in Applications (IDNA)
jinja2	2.11.3	Stand-alone template engine written in pure python
markupsafe	1.1.1	Safely add untrusted strings to HTML/XML markup
packaging	20.9	Core utilities for Python packages
prettytable	2.1.0	Python library for easily displaying tabular data in a visually appealing ASCII table format
pycurl	7.43.0.6	A Python Interface to The cURL library
pyparsing	2.4.7	Python parsing module
python-magic	0.4.22	Python interface to the libmagic file type identification library.
pyyaml	5.4.1	YAML parser and emitter for Python
urllib3	1.26.4	HTTP library with thread-safe connection pooling, file post, and more.
verboselogs	1.7	Verbose logging level for Python's logging module
wcwidth	0.2.5	Measures the displayed width of unicode strings in a terminal
Vyos	1.3.0-epa	Virtual router and VPN services for DMVPN

3.7.3 APIs

The edge proxy provides the northbound APIs (see Table 3-33) to perform registration/deregistration of edge, getting VNFDs, retrieving NSDs and instantiate/terminate NS. As discussed above, it also includes SEPP APIs.

Table 3-33 Edge Proxy APIs

HTTP Method	CRUD Method	5G-VIOS Endpoint	Result
{GET}	READ	/epa/v1/edges/	Get the registered Edge
{POST}	CREATE	/epa/v1/edges/	Register the edge with edge Proxy
{GET}	READ	/epa/v1/edges/{id}/	Get the registered edge by ID with edge proxy
{DELETE}	DELETE	/epa/v1/edges/{id}/	De-register the edge by ID with edge Proxy
{GET}	READ	/epa/v1/osm/vnf_packages/	Get all VNFD onboarded on edge orchestrator
{GET}	READ	/epa/v1/ osm/vnf_packages/{id}	Get VNFD by id onboarded on edge orchestrator
{DELETE}	DELETE	/epa/v1/ osm/vnf_packages/{id}	Delete VNFD by id from the edge orchestrator
{GET}	READ	/epa/v1/ osm/nsd/	Get all NSDs onboarded on the edge orchestrator

{GET}	READ	/epa/v1/ osm/nsd/{id}/	Get NSD by id onboarded on the edge orchestrator
{DELETE}	DELETE	/epa/v1/ osm/nsd/{id}/	Delete NSD by id from the edge orchestrator
{GET}	READ	/epa/v1/ osm/ns_instances/	Get all instantiated network service to the edge
{POST}	CREATE	/epa/v1/ osm/ns_instances/	Instantiate a network service to the edge
{GET}	READ	/epa/v1/ osm/ns_instances/{id}/	Get instantiated network service from the edge
{DELETE}	DELETE	/epa/v1/ osm/ns_instances/{id}/	Terminate the network service from the edge
{GET}	READ	/epa/v1/vims/	Get all vim registered to the edge orchestrator
{GET}	READ	/watchman/ping/	Check liveliness of the edge proxy

3.8 Portal (GUI)

3.8.1 Description

This component allows end-users to interact with the 5G-VICTORI infrastructure and services in a form of a web-portal. For instance, the experiment platform users can first register with the GUI and then access the GUI using the required credentials to login and utilise the services exposed by the 5G-VIOS. On the other hand, the edges/clusters can be registered to the 5G-VIOS through the GUI and their available NSs (Catalogues, in the form of VNFD/NSDs) and the images can be onboarded to the repositories. The users can also see the list of registered edges and compose the services for an inter-edge NS. Afterwards, the users can instantiate and later deploy this inter-edge NS.

The GUI is an interface for the creation, management and monitoring of experiments and communicates with the required 5G-VIOS components to relay user input and trigger the 5G-VIOS workflows to fulfil the user requests (see Figure 3-2). The process is explained in deliverable D4.1 [8] as a common methodology adopted by each facility to 5G-VICTORI experimentation procedures.

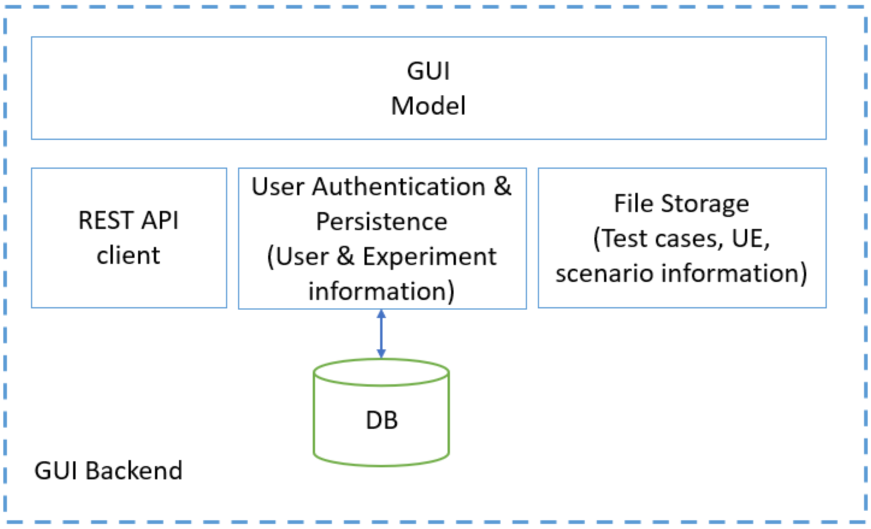

GUI
Model
REST API client
User Authentication & Persistence (User & Experiment information)
File Storage (Test cases, UE, scenario information)
DB
GUI Backend

Figure 3-2 GUI component architecture

3.8.2 Implementation

As discussed in deliverable D2.5 [1], one of the implementation options was to adopt existing portal services from ICT-17 project, and particularly 5GENESIS. We have opted for such a solution, adopting it to fit the 5G-VICTORI architecture. The GUI is composed of a Flask-based Portal web server, another Flask-based web server for data storage (for test case, UE and network scenario information) and the client-side JavaScript and jQuery scripts running on the user's browser that allow communication with the Portal web server. The Portal web server includes an HTTP client implementation for communicating with other components of VIOS.

Current Portal capabilities include:

- User registration and login with authentication.
- Edge registration, connectivity monitoring and removal.
- Experiment instantiation, termination and removal.
- Test case, UE and Scenario file upload and storage.
- User access level management and user removal.

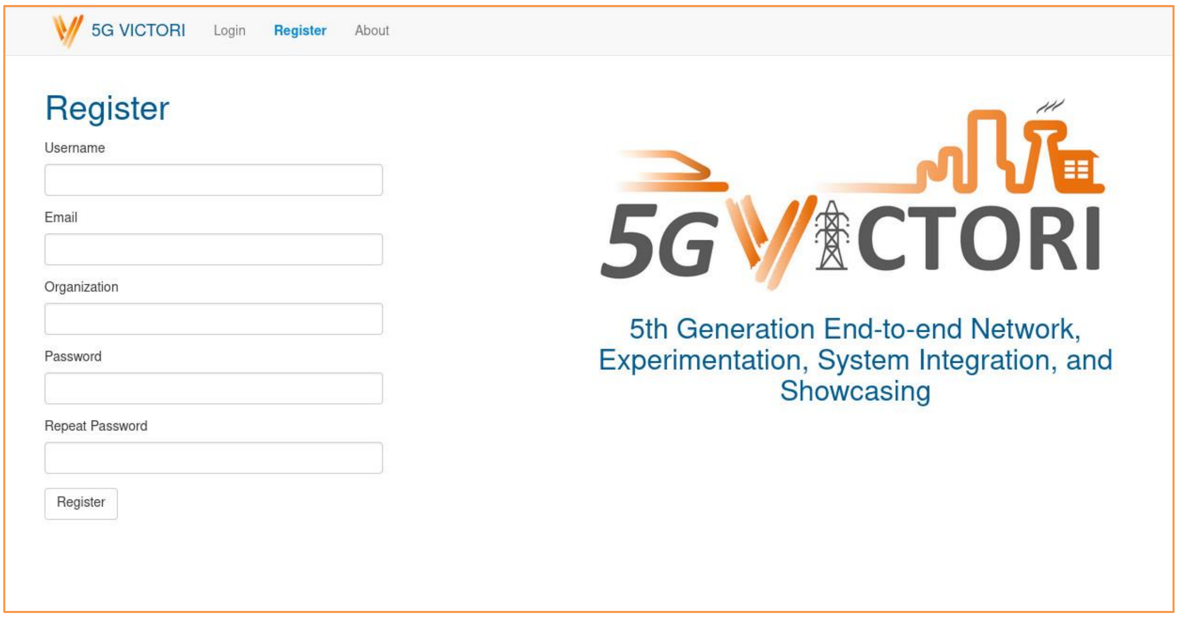
Figure 3-2 displays the GUI component's architecture:

- GUI Model: contains the user and experiment models as well as the web-portal endpoints for user requests from the browser. The GUI Model also handles the authentication of the user information which is stored in the relevant database. Users can also upload testcase, UE and scenario information through the GUI. This information is stored in a separate database.
- REST API client: The web-portal endpoints use REST API interfaces in order to communicate with the appropriate 5G-VIOS components.
- Databases: The user and experiment information are saved in a SQLite database of the web-portal. The test case, UE and scenario information are stored on a separate SQLite database controlled by a separate persistence service.

In the following screenshots the different workflows and web-portal features are illustrated. Figure 3-3 shows the registration page where only a new user can be registered in the web-portal database. In Figure 3-4, the sign in page is shown after successful registration. The user needs to be activated first by the web-portal administrator.

Figure 3-5 and Figure 3-6 show the edge registration page and edge monitoring and management page respectively. In Figure 3-7, the user can create an experiment by filling in the experiment creation form. The information model related to the experiment data is defined in deliverable D4.1 [8]. After a successful experiment creation, the user is directed to the main page, which is the experiment dashboard. In Figure 3-8, the experiment dashboard allows the user to instantiate an experiment. After a successful deployment, the user can also terminate the experiment. Before instantiation and after termination the user can also delete the experiment from the dashboard.

Figure 3-9 shows the page for uploading and managing the test cases as defined in deliverable D4.1 [8]. Figure 3-10 shows how the administrator can manage users, e.g., to change the permission level and activate or deactivate the user. Finally, Figure 3-11 shows the user profile page where the user is also able to change password.


5G VICTORI
Login
Register
About

Register

Username

Email

Organization

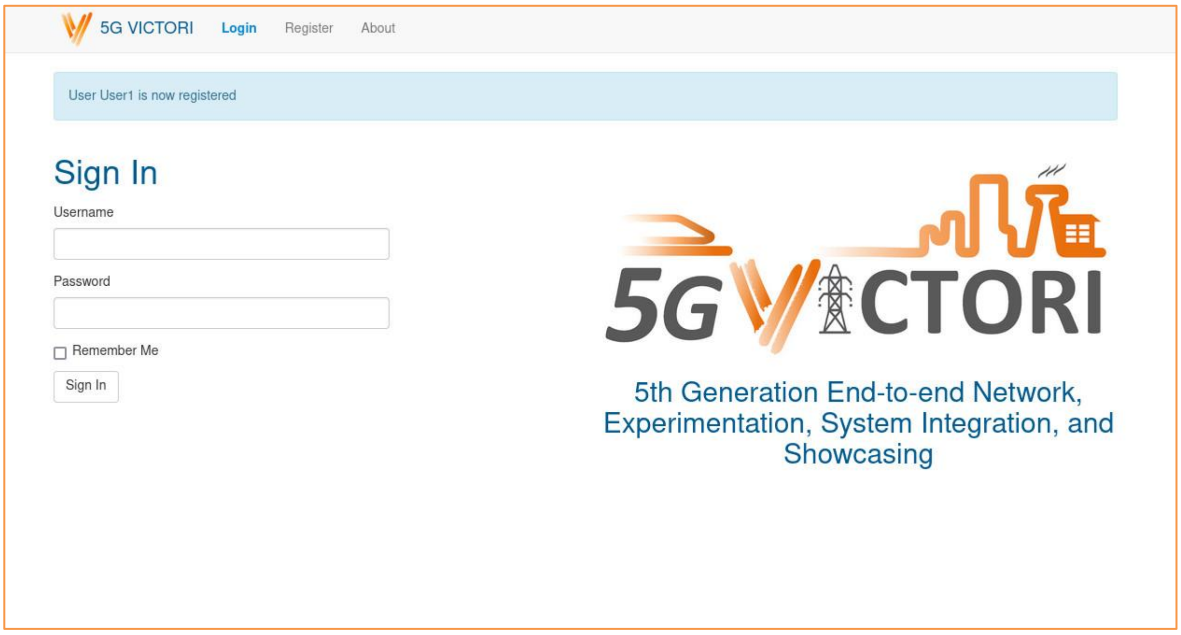
Password

Repeat Password

Register

5G VICTORI
5th Generation End-to-end Network, Experimentation, System Integration, and Showcasing

Figure 3-3 User Registration page


5G VICTORI
Login
Register
About

User User1 is now registered

Sign In

Username

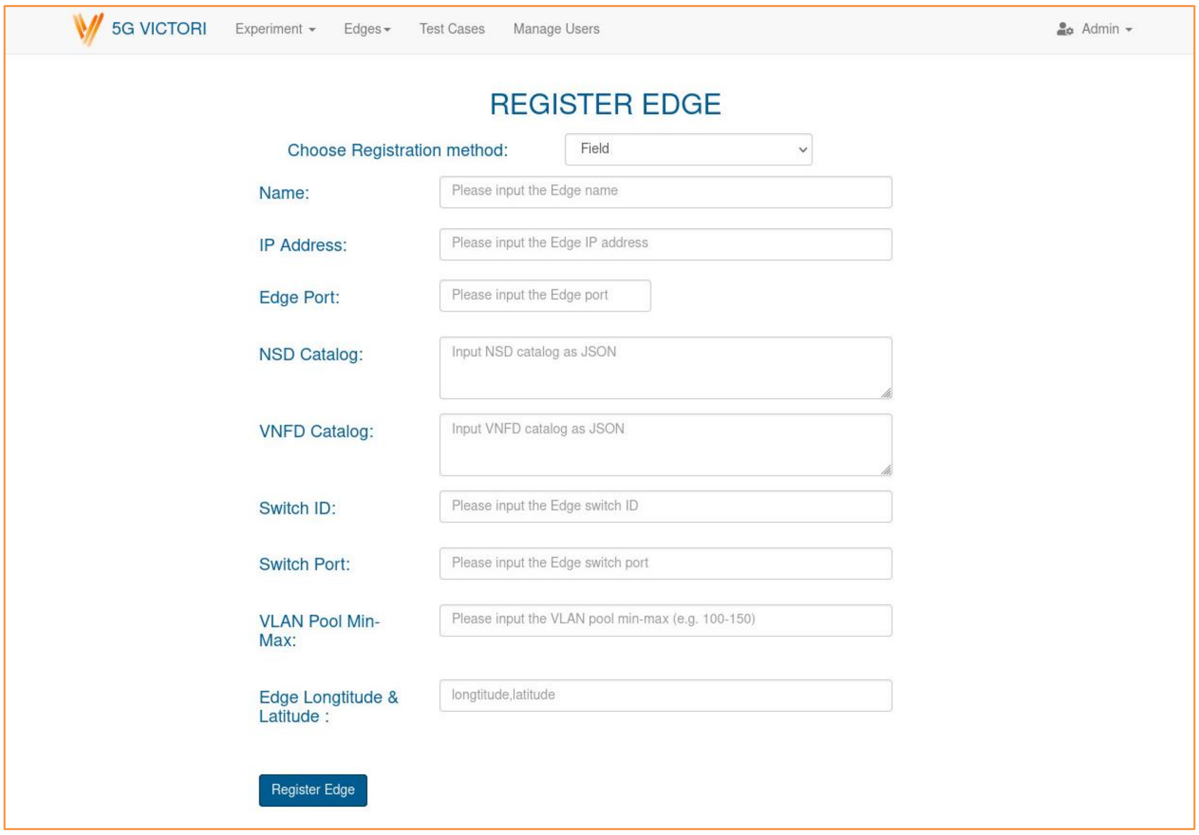
Password

☐ Remember Me

Sign In

5G VICTORI
5th Generation End-to-end Network, Experimentation, System Integration, and Showcasing

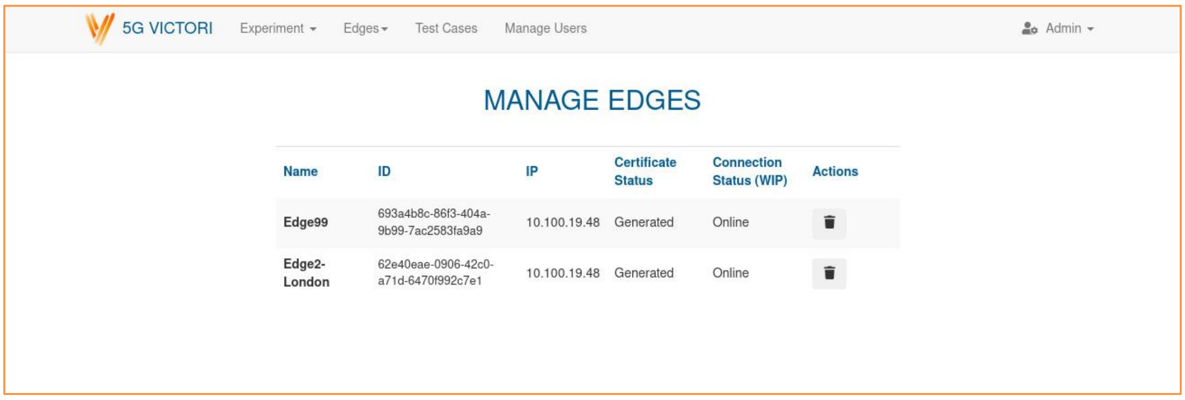
Figure 3-4 Sign in page after successful registration



REGISTER EDGE

Choose Registration method:	Field
Name:	Please input the Edge name
IP Address:	Please input the Edge IP address
Edge Port:	Please input the Edge port
NSD Catalog:	Input NSD catalog as JSON
VNFD Catalog:	Input VNFD catalog as JSON
Switch ID:	Please input the Edge switch ID
Switch Port:	Please input the Edge switch port
VLAN Pool Min-Max:	Please input the VLAN pool min-max (e.g. 100-150)
Edge Longtitude & Latitude :	longtitude,latitude

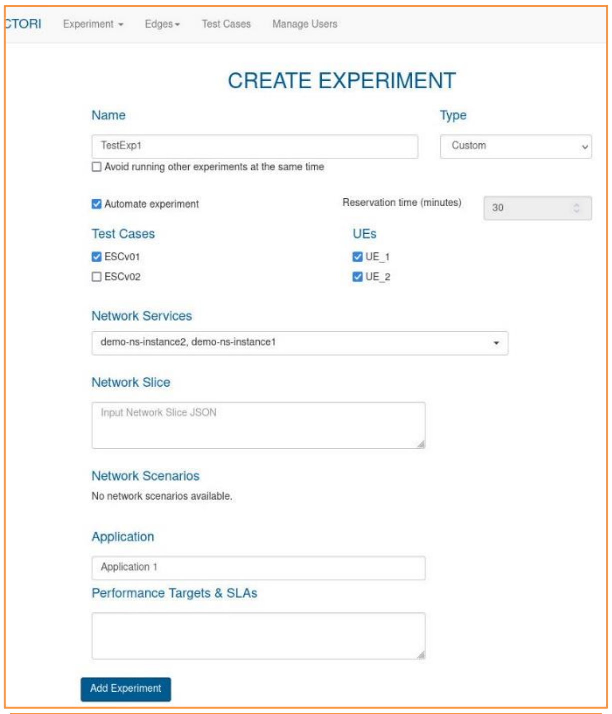
Figure 3-5 Edge registration page



MANAGE EDGES

Name	ID	IP	Certificate Status	Connection Status (WIP)	Actions
Edge99	693a4b8c-86f3-404a-9b99-7ac2583fa9a9	10.100.19.48	Generated	Online	🗑
Edge2-London	62e40eae-0906-42c0-a71d-6470f992c7e1	10.100.19.48	Generated	Online	🗑

Figure 3-6 Edge monitoring and management dashboard



CREATE EXPERIMENT

Name: TestExp1
Type: Custom
☐ Avoid running other experiments at the same time
☑ Automate experiment
Reservation time (minutes): 30

Test Cases
☑ ESCv01
☐ ESCv02

UEs
☑ UE_1
☑ UE_2

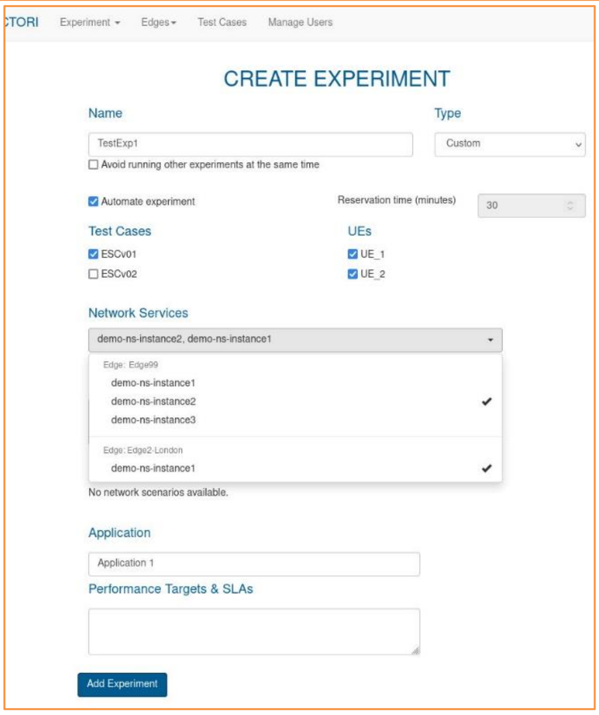
Network Services: demo-ns-instance2, demo-ns-instance1

Network Slice: Input Network Slice JSON

Network Scenarios
No network scenarios available.

Application: Application 1

Performance Targets & SLAs



CREATE EXPERIMENT

Name: TestExp1
Type: Custom
☐ Avoid running other experiments at the same time
☑ Automate experiment
Reservation time (minutes): 30

Test Cases
☑ ESCv01
☐ ESCv02

UEs
☑ UE_1
☑ UE_2

Network Services: demo-ns-instance2, demo-ns-instance1

Edge: Edge99
- demo-ns-instance1
- demo-ns-instance2 ✓
- demo-ns-instance3

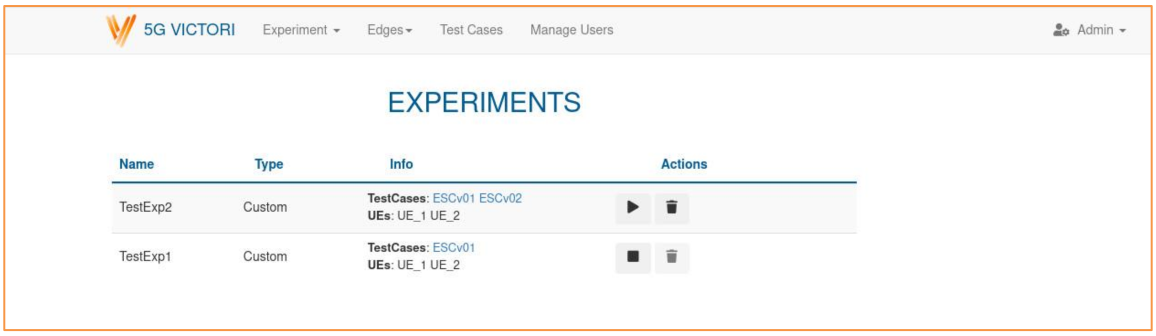
Edge: Edge2-London
- demo-ns-instance1 ✓

No network scenarios available.

Application: Application 1

Performance Targets & SLAs

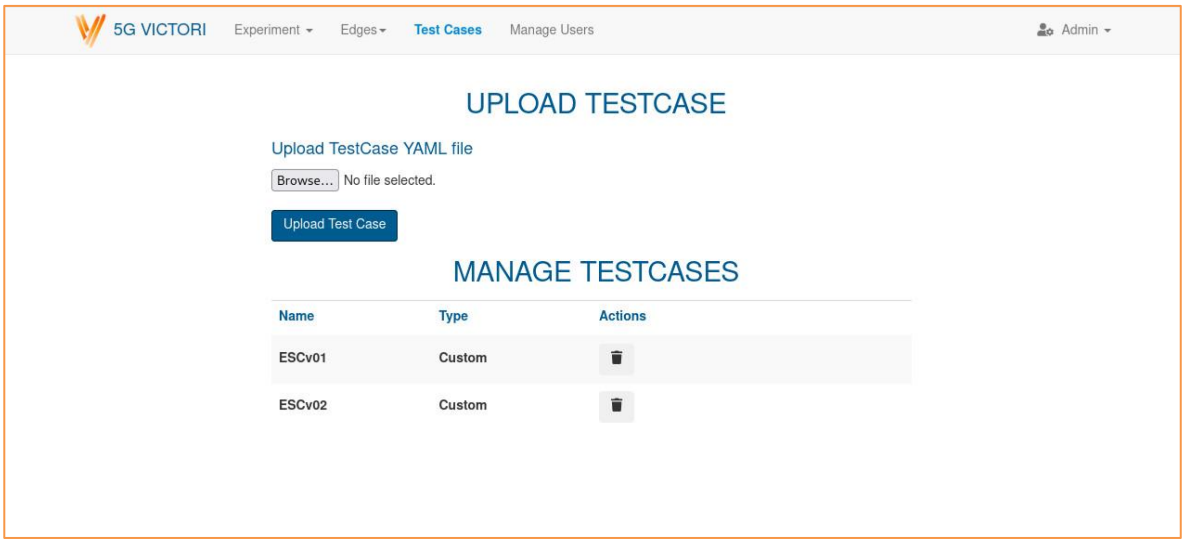
Figure 3-7 Experiment creation page (right image is demonstrating network service selection per edge) as defined in [8]



EXPERIMENTS

Name	Type	Info	Actions
TestExp2	Custom	**TestCases**: ESCv01 ESCv02 **UEs**: UE_1 UE_2	▶ 🗑
TestExp1	Custom	**TestCases**: ESCv01 **UEs**: UE_1 UE_2	■ 🗑

Figure 3-8 Experiment dashboard



UPLOAD TESTCASE

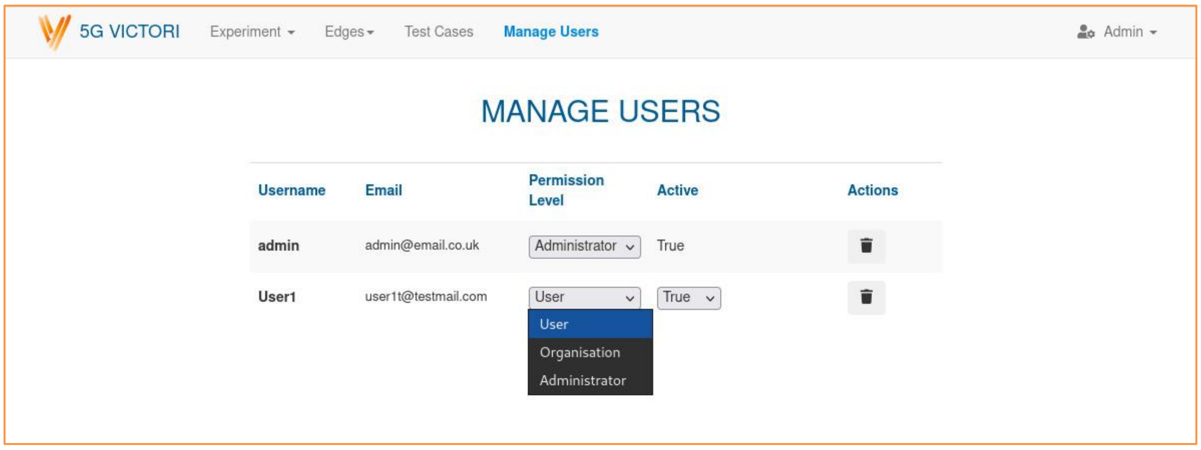
Upload TestCase YAML file

No file selected.

MANAGE TESTCASES

Name	Type	Actions
ESCv01	Custom	🗑
ESCv02	Custom	🗑

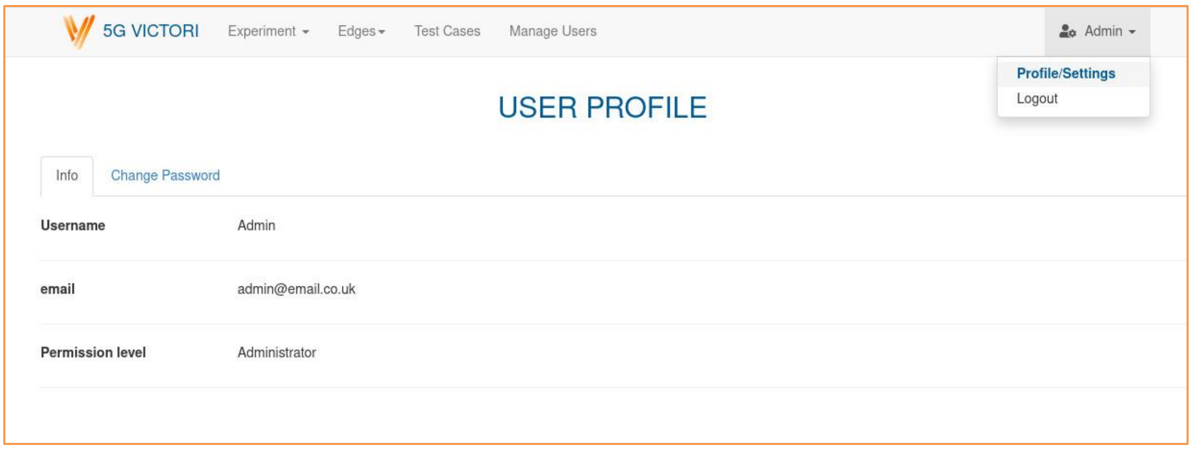
Figure 3-9 Test case upload and management page



MANAGE USERS

Username	Email	Permission Level	Active	Actions
admin	admin@email.co.uk	Administrator	True	🗑
User1	user1t@testmail.com	User	True	🗑

Figure 3-10 User management page



USER PROFILE

Username	Admin
email	admin@email.co.uk
Permission level	Administrator

Figure 3-11 User profile page

3.9 Profiling (PROF)

The NWDAF, which is one of the novel NFs in 5G networks, provides automatic data analytics in the underlying network. Also, it is supposed to provide an efficient decision making over NF LCM.

The VNF Profiling component in 5G-VIOS uses monitoring information to create mathematical or computational models for the performance of VNFs known as profiles. Given the functionality of NWDAF, the 5G-VIOS profiling's role is to uncover this relationship between the resource configuration, the service requirements and its performance targets. The 5G-VIOS profiling utilises ML-based techniques to estimate the absolute number of resources (including RAM, CPU cores, and link capacity) required to meet the required performance targets in the specific environment.

3.9.1 Description

The proposed method in profiling components is integrated in the **orchestrator** (i.e., at each edge nodes) allowing to perform real-time updates on the models obtained offline, hence achieving dynamic profiling. Actually, the result of each edge node's profiling will be updated into the 5G-VIOS profiling component. Basically, the profiling component is required for NFV management and orchestration (NFV MANO) to promote localised changes in the graph topology of the VNF chain during runtime to respond to demands' variations or any QoS degradation.

The 5G-VIOS profiling system design concentrates on single VNF profiling. The proposed profiling implementation distinguishes every atomic part of a VNF chain and identifies their contribution to the overall NS. The idea of profiling each VNF individually paves the way for efficient service chain composition in such a way that a service provider can evaluate the individual VNF profiles to select the most appropriate VNF among the ones in the same functionalities [9]. However, it may re-use already profiled VNFs to compose new chains, where new profiling rounds can be completely avoided. It is assumed that each VNF is profiled across the whole range of possible input workload values and all available resource configurations [10]. This may lead to an ample space of multiple parameters and many combinations to test, which will result in an expensive and overextended measurement period [11]. Also, given the huge resource configuration space of a VNF and the fact that its re-deployment or reconfiguration takes a considerable amount of time, executing profiling measurements over the complete configuration space is infeasible [11]. To overcome this issue, we decided to select and profile only a subset of all possible resource configurations.

As it is illustrated in Figure 3-12, through the predictor manager component, the profiler is capable to predict certain quantities after past measurements. It works through complex ML solutions, e.g., MIMO Artificial Neural Network (ANN) models.

The first role of the predictor manager is to predict the Optimum Maximum Input Rate (MIR) for previous untested resource configurations which meet the given KPIs. The second role is to calculate the absolute required resources for satisfying both the given KPIs and the Optimum MIR in the target environment. Both mentioned roles can be performed in parallel by utilising the ML-based techniques.

3.9.2 Implementation

Figure 3-12 illustrates the high-level architecture of the 5G-VIOS profiling component. It assumes that each VNF is associated with a list of performance targets (i.e., SLA requirements), and relevant profiling parameters. The VNFD refers to the VNF catalogue, which is a set of templates that describe the deployment and operational characteristics of available VNFs as well as their images. The VNFDs could also include resource requirements such as CPU, RAM, and Disk space. Therefore, the VNFD and the Profiling parameters specify which resources should be tested throughout the profiling process, the minimum and maximum values per resource, and the performance metrics to be collected and analysed. The performance targets are evaluated via KPIs that decide if the VNFs are successfully handling the workload or not. The Profiler module resides as part of the profiling platform and has three main building blocks: the *Weighted Resource Configuration Selector*; the *Analyser and Post Processor*; and the *Predictor Manager*.

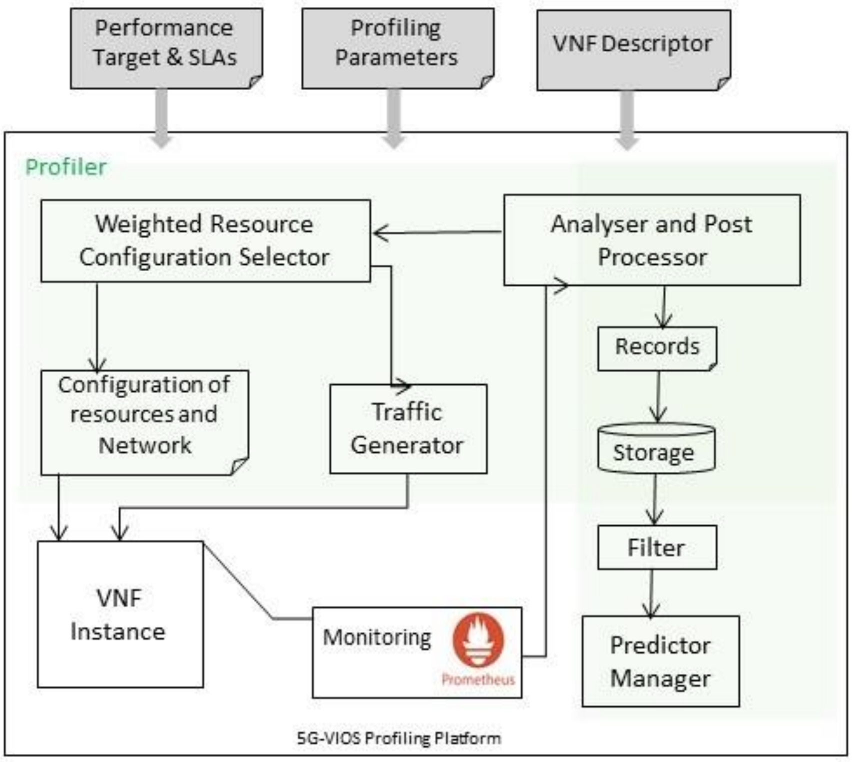

Performance Target & SLAs
Profiling Parameters
VNF Descriptor
Profiler
Weighted Resource Configuration Selector
Analyser and Post Processor
Records
Configuration of resources and Network
Traffic Generator
Storage
Filter
VNF Instance
Monitoring
Prometheus
Predictor Manager
5G-VIOS Profiling Platform

Figure 3-12 5G-VIOS Profiler system architecture [9]

The ***Weighted Resource Configuration Selector*** gets the VNF descriptor, profiling parameters and the list of performance targets. It selects a set of resource configurations and assigns them to the VNF, and then runs it with these allocated resources. Afterwards, it requests for traffic to be generated and sent by the *Traffic Generator* to the *VNF instance*. Alongside the help of the ***Analyser and Post Processor***, it queries the monitoring data from the Prometheus tool to find and record the *Optimum MIR* that the *VNF instance* can handle to meet all performance targets. The outcome of this step is referred to as the "VNF Performance Dataset".

Basically, the ***Predictor Manager*** is designed to create and train prediction models that can predict certain quantities based on past measurements. It has two roles:

- First, it creates a model to accurately predict the *Optimum MIR* for previously *untested* resource configurations in which the given KPIs can be met.
- Second, it calculates the absolute amounts of resources required to meet both the given KPIs and the *Optimum MIR* in the target environment.

The mentioned roles can be achieved simultaneously by utilising ML-based techniques. We consider the offline techniques which can satisfy the needs of static network conditions and, hence, be deployed to such environments with good profiling performance expectations. Likewise, performance expectations can be also met for short term period deployments. These trained models can be used to bootstrap profiling, which can be further continued with other models such as online trained ones.

3.9.2.1 The 5G-VIOS profiling method

The weighted selection of the resource configurations as well as the process of computing the Optimum MIR are explained and illustrated in Figure 3-13. Basically, the profiling workflow comprises five steps.

In step (1) the profiler sets the minimum configuration for all the resource types and runs the VNF with these allocated resources. Actually, the preferred configuration of resources allocated to the VNF instance will be saved for further decision. For example, if the VNF can have 1-16 CPU cores, 128-2048 MB Memory, and 128-4096 Mbps link capacity, the minimum configuration for CPU, Memory, and Link capacity is equal to 1, 128, and 128, respectively. Then, the weighted resource configuration will be run to find the Optimum MIR that VNF instance can handle with minimum configuration of resources while meeting the performance targets.

In a similar way, step (2) sets the maximum configuration for all the resources. Based on the example given, it will assign 16, 2048, and 4096 to CPU, memory, and link capacity, respectively, as the maximum resource configuration. Then it will run the VNF and will find the Optimum MIR for the assigned resource types' configuration.

In step (3) the profiler allocates the average configuration of resource types to the VNF and will run it. Following our example, it will be equal to 8, 1088, and 2112 associated to CPU, memory, and link capacity, respectively. Then, it will find the Optimum MIR that the VNF can handle with average configuration of resources; We refer to this traffic rate as the MIR_{Avg}.

In step (4) the profiler computes the weight of each resource types (WR) by checking its impact on the Optimum MIR. To check the impact of each resource types (R), it selects the maximum configuration of that resource and the average for other resources. Then, it will run the VNF with these allocated resources and will find the Optimum MIR that the VNF instance can handle.

We refer to this traffic rate as the MIR_R. Afterwards, the profiler will compute the weight of that resource types (WR) defined in (1). For instance, based on the previous example, to find the MIR_{CPU} and then the W_{CPU}, the VNF will be executed with 16 CPU cores, 1088 MB Memory, and 2112 Mbps Link capacity. It is worth saying that before computing the weights, the related input rates are normalised by utilising feature scaling, and the process of finding each weight is repeated ten times to improve the accuracy of the weights.

$$W_R = abs(MIR_R - MIR_{Avg})/MIR_{avg} \tag{1}$$

Finally, through step (5), the selection of resources will be prioritized by considering their weights (W). According to the computed weights of resources, each resource types may have a different probability for being selected. So, the profiler randomly selects a resource type R from the list of Resources according to the relative sequence of their weights. Then, it will select a random configuration (i.e., a random number between the minimum and maximum amount) for that resource type, R. It is noted that all the other resource types' of value will be configured in average. Afterwards, the Optimum MIR for this configuration will be computed, and the respective record will be stored. For instance, if W_{CPU} is greater than the other resource types', the CPU has a higher probability to be selected. Assuming the profiler has selected the CPU, following our example, the selected configuration will be equal to a, which is a random number between 1-16 for the number of CPU cores, 1088 units of memory, and 2112 link capacity unit. The profiler will continue this randomly weighted selection of resource types and will add the records to the "VNF Performance Dataset" until the end of profiling time. The data set are being stored in MongoDB. Meanwhile, the profiling time is specified in the profiling parameters.

3.9.2.2 **Optimum MIR calculation**

The *Optimum MIR,* as a performance metric, shows the maximum load in terms of the traffic rate a *VNF instance can handle* while meeting the performance targets *regarding* an allocated configuration of resources. To compute this metric, the five-step approach illustrated in Figure 3-13 "*Find Overloaded input rate (IR)"* is being utilised.

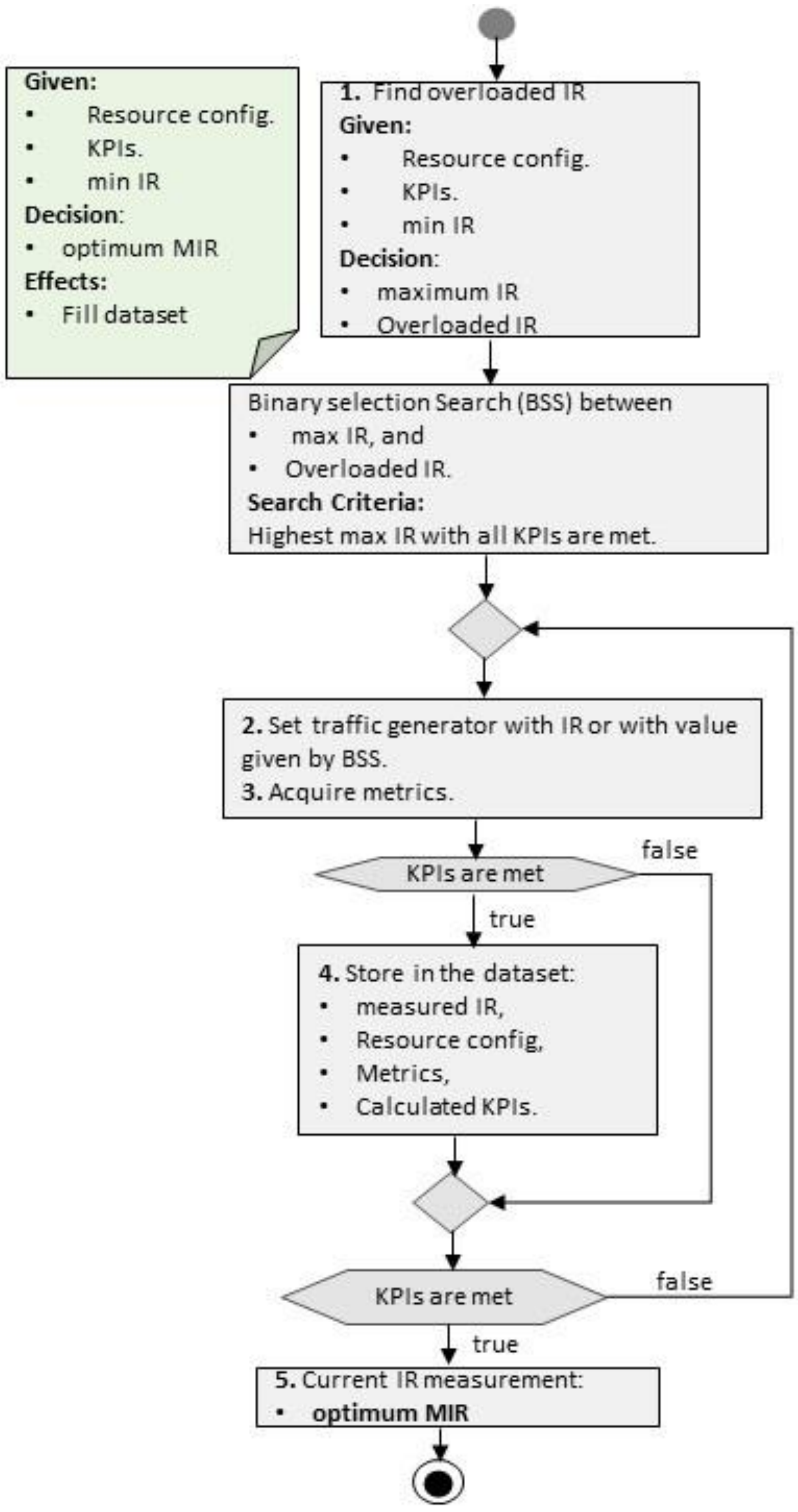


Figure 3-13 Algorithm for Finding the Optimum MIR. Internally it calls the Algorithm in Figure 3-14 [9]

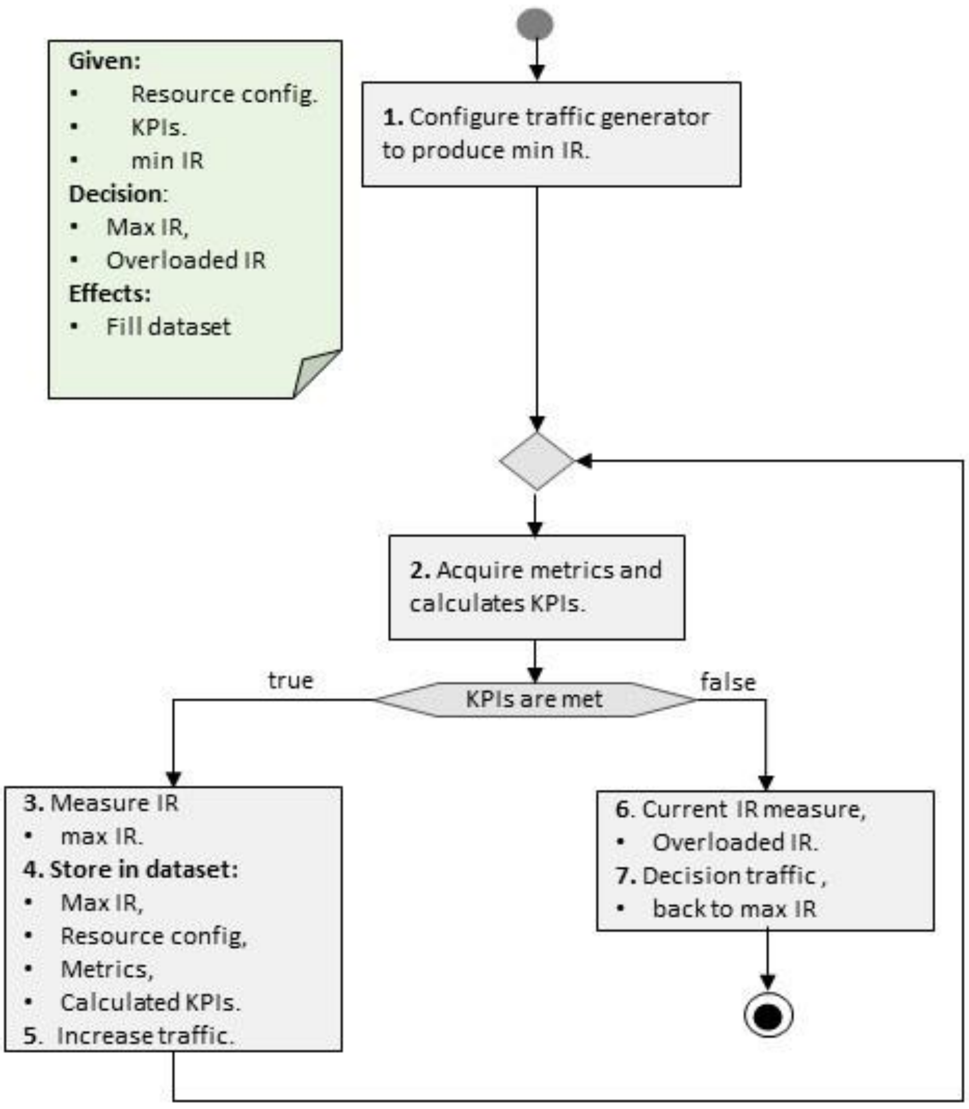


Figure 3-14 Algorithm for finding the overloaded input rate (IR)

- In the first step, the *Weighted Resource Configuration Selector* component sets the Traffic Rate to given minimum IR and requests the Traffic Generator component (i.e., here we use Iperf3) to send UDP packets to the VNF instance component with the selected rate. In 5G-VIOS profiling method, the *Traffic Generator* component is a "pluggable" component; for the time being, the implementation is based on iPerf3, which can generate UDP and TCP packets. So, in case the injection of the TCP packets is needed, we can generate TCP packets, monitor and profile the performance of VNFs. In the experiments, UDP was preferred since it results in more precise throughput measurements.
- In the next step, the *Analyser and Post Processor* component retrieves the monitoring data through Prometheus monitoring tool and computes the performance metrics such as CPU utilisation, memory utilisation and latency. While the measured performance metrics meet the required KPIs, it will record the traffic rate, Preference resource types configurations, the metrics captured from the Prometheus and the measured performance corresponding to the required KPIs. In addition, it will feedback the *Weighted Resource Configuration Selector* component to increase the amount of traffic rate and send the increased amount to the *VNF instance*. When any of the measured

performance metrics stops meeting the performance targets, it will return two traffic rates. Meanwhile, we consider the traffic rate could be handled by the system referred to as the "*Max IR*", while the increased traffic that is not satisfying the KPIs is called "Overloaded IR". It is assumed *Min IR* can always be handled.

- In the next stage, as it can be seen in in Figure 3-14, the binary search BSS method [12] will be called recursively to find and return the Optimum MIR for the VNF instance with the allocated configuration of resources while allowing all the given KPIs to meet at the same time.
- Finally, the *Analyser and Post Processor* component will post this record to the "VNF Performance Dataset", that in MongoDB.

3.9.3 APIs

The Profiling component considers set of APIs at both edge profiling as well as 5G-VIOS profiling to provide services to other mentioned 5G-VIOS and edge nodes. Table 3-34 and Table 3-35 illustrate the required API endpoints.

Table 3-34 5G-VIOS Profiling APIs

HTTP Method	DB CRUS Method	5G-VIOS Endpoint	Result
{GET}	READ	api/Pro/ns/{id}/Exp-desc/	Getting an Exp-Descriptor related to specific NS_{id}.
{POST}	CREATE	api/Pro/ns/{id}/Exp-desc/	Posting an Exp-Descriptor related to NS_{id} to the corresponding Edge-Proxies
{GET}	READ	api/Pro/ns/{id}/performance_Profiles_Dataset	Reading the Performance_Profiles_Dataset related to NS_{id}.
{POST}	CREATE	api/Pro/ns/{id}/performance_Profiles_Dataset	Exporting the performance profiles_dataset (from Database) related to a NS to Grafana
{POST}	CREATE	api/Pro/iNS/{id}/performance_Profiles_Dataset	export the performance profiles_dataset (from Database) related to an iNS to Grafana.
{POST}	CREATE	api/Pro/ns/{id}/performance_Profiles_Dataset/NN	Train the neural network (NN) with the Performance_Profiles dataset related to NS_{id}.
{GET}	READ	api/Pro/ns/{id}/performance_Profiles_Dataset/NN	Testing the neural network (NN) with the Performance targets related to NS_{id}.
{POST}	CREATE	api/Pro/ns/{id}/optimum_resources	Posting the Optimum_resources related to NS_{id} to SCO.
{READ}	GET	api/Pro/ns/{id}/Exp_outcome	Reading the Exp_Outcome related to NS_{id} from the Edge Proxy.
{POST}	CREATE	api/Pro/ns/{id}/Exp_outcome	posting the Exp_Outcome related to NS_{id} to SM/SB/GUI.
{READ}	GET	api/Pro/ns/{id}/metrics	Reading the metrics related to the NS_{id} from 5G-VIOS monitoring.
{READ}	GET	api/Pro/iNS/{id}/metrics	Reading the metrics related to the iNS from 5G-VIOS monitoring.

Table 3-35 Edge Profiling APIs

HTTP Method	DB CRUS Method	5G-VIOS Endpoint	Result
{POST}	CREATE	api/edge_Pro/ns/{id}/Profiling_files	Upload Prometheus yaml file and node_exporter's,w.r.t. VNF type (Snort), to the VNF under test
{POST}	CREATE	api/edge_Pro/ns/{id}/install	Install Prometheus, node_exporter, iperf3_server on the VNF under test, install iperf3_client on the corresponding vm,
{POST}	CREATE	api/edge_Pro/traffic_generator	Asks the traffic_generator to send an amount of traffic
{DELETE}	DELETE	api/edge_Pro/traffic_generator	Asks the traffic_generator to delete an amount of traffic
{POST}	CREATE	api/edge_Pro/ns/{id}ping	Pings the vnf under test
{GET}	READ	api/edge_Pro/ns/{id}/metrics	Reads the metrics (including the Ping metrics and the Node_Exporter (Prometheus) metrics) from the edge monitoring component The metrics can also include the monitoring metrics exposed by the APPs (including the exp_outcome)
{POST]	CREATE	Api/edge_pro/ns	Create a new network service through the Edge osm
{DELETE}	DELETE	Api/edge_pro/ns/{id}	Delete a network service for a given id
{POST}	ADD	Api/edge_Pro/ns/{id}/Performance_Profiles	Store/add the record (Performance_Profiles) in Elastic search (add the record to the Edge Profiling database (Performance_Profiles_Dataset) Later, it can expose the record to the 5G-VIOS profiling
{POST}	CREATE	Api/edge_Pro/ns/{id}/Performance_Profiles_Dataset	POST Performance_Profiles_dataset to the Edge Proxy
{POST}	CREATE	Api/edge_Pro/ns/{id}/Exp_outcome	Post exp_outcome to the edge proxy

3.10 Monitoring (MON)

5G-VIOS monitoring have been designed to monitor the service status and make sure it is functioning properly. Basically, 5G-VIOS pulls monitoring information from the underlying NFVO and applications directly, which would be offered through dashboards to the users and consumed internally for generation of performance profiles. The role of 5G-VIOS monitoring is to monitor the KPIs attained for the requested service. It supports traffic monitoring specific to each network slice.

The monitoring components at each edge node will exploit Prometheus tools to monitor the running NS on edge nodes. During the slice life cycle execution, the monitoring is supposed to work regarding given SLA requirements into KPIs. Basically, the 5G-VIOS generates user and service performance profiles based on gathered monitoring KPIs. A set of APIs have been designed for 5G-VIOS monitoring. They are being developed to integrate with 5G-VIOS components as well as edge nodes. Then the network operators can check the status of the deployed NSs as well as available computing and network resources utilisation.

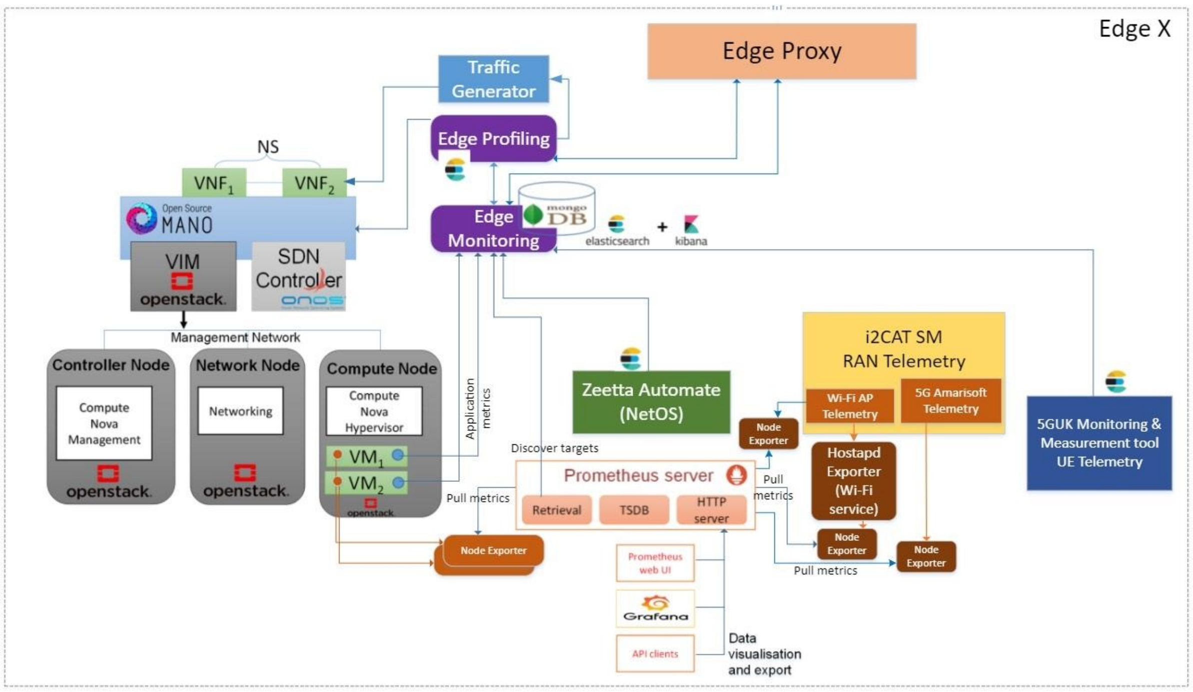


Figure 3-15 The edge profiling component and its connection with OSM orchestrator, Zeetta Automate, deployed application on VMs, etc.

3.10.1 Description

The 5G-VIOS monitoring is formed by two sides: the Edge Monitoring, and the central 5G-VIOS monitoring.

Edge Monitoring

The Edge Monitoring will be deployed on each edge as well as nomadic node. Basically, the Edge Monitoring can reach the monitoring metrics associated to each deployed applications (i.e., App1 and App2) then store them into the MongoDB. As it can be seen in the Figure 3-15, the Edge Monitoring needs to read the monitoring metrics from the deployed resources as well as i2CAT slice manager. Also, the monitoring metrics from the 5G Amarisoft is needed to be read and store in MongoDB. Finally, the Edge Monitoring need to post the obtained metrics to the Edge Proxy component. In other words, the Edge Proxy can get the required monitoring metrics from the Edge Monitoring-specific MongoDB through API.

5G-VIOS Monitoring

Basically, the 5G-VIOS monitoring reads the monitoring metrics from the Edge Proxy components deployed at each Edge Node. It stores the obtained monitoring data into the MongoDB. The 5G-VIOS Monitoring integrates the iNS monitoring metrics from the Edge Proxies. Then it can post the iNS monitoring metrics to GUI, SMA (SM), and 5G-VIOS profiling through the designed APIs.

3.10.2 Implementation

We are planning to employ polling status indicators through appropriate APIs or by subscription to notification events. These messages are collected by the edge proxy and passed to the Monitoring component inside 5G-VIOS. Each edge node is equipped with monitoring component. At the orchestrator of edge nodes (i.e., OSM), the OSM monitoring feature (‘i.e., ’mon-collector’’ module) will be utilised to collect NFVI and VNF metrics whenever specified at the descriptor level.

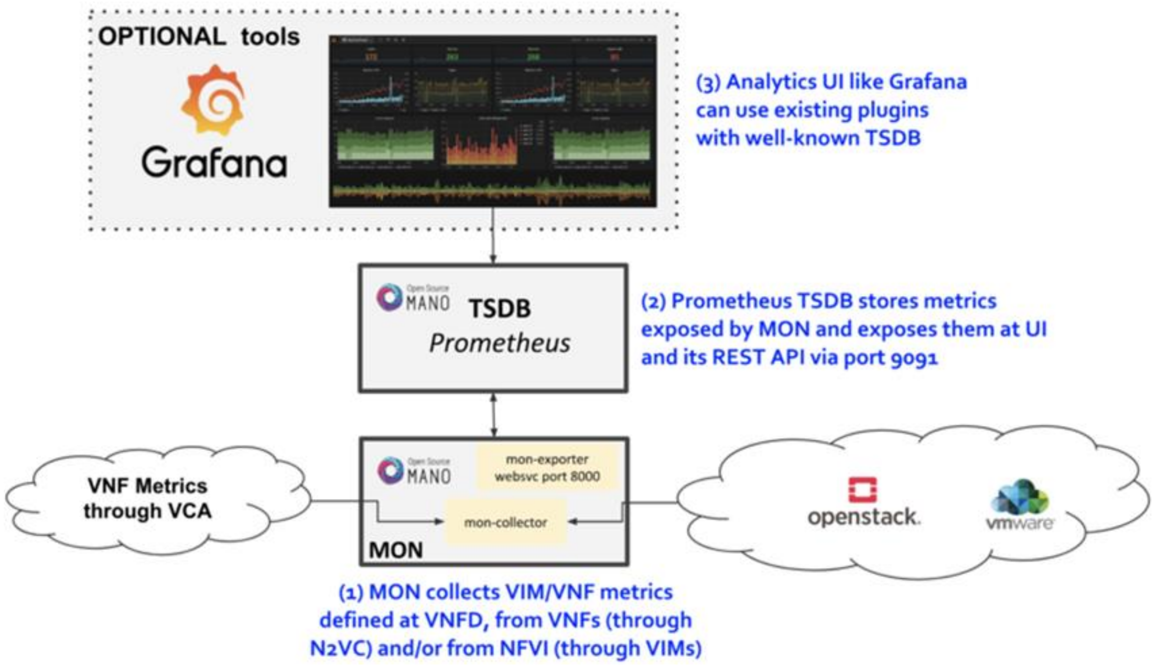

OPTIONAL tools
Grafana
(3) Analytics UI like Grafana can use existing plugins with well-known TSDB
MANO TSDB
Prometheus
(2) Prometheus TSDB stores metrics exposed by MON and exposes them at UI and its REST API via port 9091
VNF Metrics through VCA
MANO
mon-exporter websvc port 8000
mon-collector
MON
openstack.
vmware
(1) MON collects VIM/VNF metrics defined at VNFD, from VNFs (through N2VC) and/or from NFVI (through VIMs)

Figure 3-16 performance management and Monitoring diagram in OSM (release 5+)

3.10.2.1 Monitoring System Architecture

Figure 3-16 shows the OSM Monitoring diagram[3]. In step (1), the VIM/VNF metrics are collected from the VIM (OpenStack or VMWare) and the running VNFs. As can be seen in step (2), the Prometheus stores these metrics in its data base (TSDB) and the metrics can be retrieved through its REST APIs. Also, if needed to visualise the data, tools such as Grafana can be integrated with Prometheus (as shown in step (3).

3.10.3 APIs

We designed the set of APIs for edge monitoring as well as the 5G-VIOS monitoring to provide services to other mentioned 5G-VIOS components. Table 3-36 and Table 3-37 illustrate the required API endpoints designed for edge monitoring as 5G-VIOS monitoring, respectively.

Table 3-36 Edge Monitoring Endpoint APIs

HTTP Method	DB CRUD Method	Endpoint	Result
{GET}	READ	api/edge_Mon/ns/{id}/ping	Reads the rtt_avg related to the ping
{POST}	ADD	api/edge_Mon/ns/{id}/Metrics	Add/integrate the monitored metric to the “Metrics” Elasticsearch DB
{GET}	READ	api/edge_Mon/ns/{id}/App	Reads the metrics exposed by the Application APIs
{GET}	READ	api/edge_Mon/ns/{id}/Zeetta	Reads the metrics exposed by the Zeetta APIs
{GET}	READ	api/edge_Mon/ns/{id}/5GUK	Reads the metrics exposed by the 5GUK monitoring ElasticSearch

[3] https://osm.etsi.org/wikipub/index.php/OSM_Performance_Management

{GET}	READ	api/edge_Mon/ns/{id}/Prometheus	Reads the metrics exposed by the Prometheus
{GET}	READ	api/edge_Mon/ns/{id}/Amarisoft	Reads the metrics exposed by the Zeetta APIs
{POST}	CREATE	api/edge_Mon/ns/{id}/Metrics	Post the “Metrics” to Edge Proxy

Table 3-37 5G-VIOS Monitoring Endpoint APIs

HTTP Method	DB CRUD Method	Endpoints	Result
{GET}	READ	api/Mon/ns/{id}/Metrics api/Mon/edge/{id}/ns/{id}/Metrics	Read the monitored metrics “Metrics” from the corresponding Edge Proxies
{POST}	ADD	api/Mon/iNS/{id}/Metrics	Add the monitored metrics “Metrics” from the corresponding Edge Proxies and integrate them to the “Metrics_iNS”
{POST}	CREATE	api/Mon/ns/{id}/Metrics	Post the monitored metrics “Metrics” to the GUI or the SM or the 5G-VIOS profiling
{POST}	CREATE	api/Mon/iNS/{id}/Metrics	Post the monitored metrics “Metrics_iNS” to the GUI or the SM or the 5G-VIOS Profiling

3.11 Mobility Manager (MOB)

3.11.1 Description

One of the key innovations that 5G-VIOS aspires to achieve is to introduce mobility management within the LCM of a service, so that it would be able to offer seamless service continuity to vertical users while they move. Connectivity continuity is a well investigated subject and there are solutions that offer horizontal and vertical handover coupled with multi-homing protocols allowing sessions to continue. However, service continuity is challenging and depends if on whether the application is stateful or stateless, as the application needs to follow the user to a different edge instance. This is an open challenge as several issues must be tackled such as the change of the IP address of the user, the communication between edge hosts when using Network Address Translation (NAT) and more. The combined smooth integration of network and application mechanisms is needed to guarantee session and service continuity during inter-edge handover.

The Mobility Manager (MOB) will interact with the SMA that takes care of LCM, requesting the service migration. Initially that would be initiated by individual applications, which monitor the mobility of the users, sending a request through the AGA. However, this could be further extended to automated migration using the monitoring and profiling modules whereby the migration would be triggered by performance related events.

The MOB aims to fulfil the functional requirements for the 5G-VIOS platform included in Table 3-38.

Table 3-38 MOB functional requirements

ID	Functional Requirement	Description
MOB1	Check availability of NSDs in target edge	During the service migration process checks if NSDs are available, and if not start the onboarding process
MOB2	Request new INSD from the SCO and send to the SMA	Sends new data to the SCO so that a new INSD can be composed and sent to the SMA
MOB3	Send a migration stub to the SMA	To begin the migration process, the MOB will send a migration stub to the SMA

3.11.2 Implementation

The MOB uses the components in Table 3-39 to deliver its endpoint functionality. The MOB supports its operations with the use of a stateful table in a PostgreSQL database.

Table 3-39 MOB implementation technologies

Technology	Version	Description
Python	3.9.7-slim	Base Docker image
fastapi	0.70.0	High performant API micro-framework
asyncpg	0.24.0	Asynchronous database interface library designed specifically for PostgreSQL
ormar	0.10.22	Asynchronous mini ORM for Python, with support for Postgres
PostgreSQL	13.2-alpine	Production grade high-performant relational database
Uvicorn	0.15.0	Python ASGI HTTP server
Gunicorn	20.1.0	Python WSGI HTTP Server (Provides process management)

3.11.2.1 Migration Stubs Registry

The Migration Stubs Registry table stores a record of all the service migration requests in 5G-VIOS for each NSR.

Table 3-40 MOB Migration Stubs Registry Data Model

Field Name	Data Type	Length	Description	Expected Value(s)	Allow Null
migration_id	UUID Field	36	Primary Key	A UUID	No
new_edge_id	UUID Field	36	The edge_id of where the service is to be migrated to	A UUID	No
current_edge_id	UUID Field	36	The edge_id of current edge	A UUID	No
current_nsr_id	UUID Field	36	A link to the current NSR_ID	A UUID	No
current_service_id	UUID Field	36	A link to the current Service_ID	A UUID	No
current_experiment_id	UUID Field	36	A link to the current Experiment_ID	A UUID	No
new_service_id	UUID Field	36	A link to the new Service_ID	A UUID	No
new_insd	JSON Field	N/A	The new composed ISND	A JSON object	No

create_user	UUID Field	36	A link to a system user	00000000-0000-0000-0000-000000000000	No
create_datetime	Date Time Field	N/A	The datetime the record was created	A datetime stamp	No

3.11.3 APIs

The MOB provides the endpoints included in Table 3-41.

Table 3-41 MOB APIs

HTTP Method	DB CRUD Method	5G-VIOS Endpoint	Result
{GET}	READ	/api/mob/migrations	Gets all migrations
{POST}	CREATE	/api/mob/migrations	Creates a new migration
{GET}	READ	/api/mob/migrations/{migrations_id}	Gets migration details for a given id
{PATCH}	PARTIAL UPDATE	/api/mob/migrations/{migrations_id }	Updates fields for a given id
{DELETE}	DELETE	/api/mob/migrations/{migrations_id }	Deletes a migration for a given id
{GET}	READ	/api/mob/healthchecks/readiness	Health Check endpoint for readiness and liveness probes

4 Workflows

5G-VIOS is an inter-edge management and orchestration platform which performs the LCM of inter-edge NS (or experiments). More specifically, 5G-VIOS can perform various workflows such as initiate and instantiation of NSs, deletion, and termination of the running NSs, modification (e.g., mobility management) in the deployed NSs. Additionally, it also manages the registration and deletion of edge proxy and stores the required information in the repository to communicate with different edge facilities during and after the deployment of NSs. The next subsections provide the detailed description and step-by-step process of each workflow that is supported by the 5G-VIOS architecture.

4.1 Registering a new Edge

The LCM of an inter-edge NS starts with the registration of the different edge facilities that it would require to run, if it is not already connected with 5G-VIOS. 5G-VIOS components need to communicate with the various applications/technologies available at each edge facility. The communication between the 5G-VIOS and edge proxy is enabled by registering the edge with the 5G-VIOS via GUI portal, through the process described as “Edge Registration. Figure 4-1 depicts the step-by-step process of registering an edge with 5G-VIOS platform.

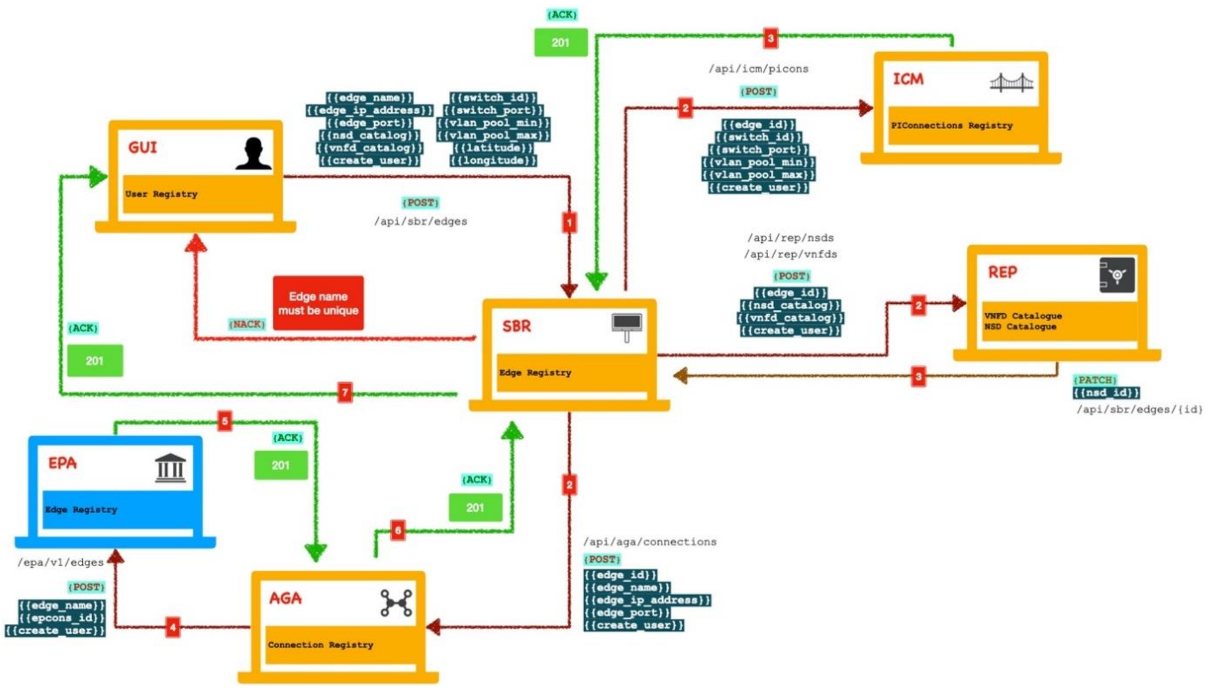


Figure 4-1 Edge Registration Workflow Overview

4.1.1 Step 1

From the GUI, an end-user will complete an Edge Registration form to upload specific details that are needed to onboard a new Edge within 5G-VIOS. When the end-user clicks submit, these details are sent to the SBR in a `{POST}` request. The SBR will validate the incoming `{POST}` request. If the `{POST}` request is invalid, the SBR will send back a 400 `{NACK}` response, exampled in Figure 4-2.

```
{
  "NACK": {
    "origin": "service broker",
    "message": "unable to create edge",
    "reason": "edge_name already exists"
  }
}
```

Figure 4-2 Edge Registration Step 1 – 400 NACK Response

4.1.2 Step 2

If validation is successful, the SBR sends specific data using `{POST}` requests to other components within 5G-VIOS. The SBR sends physical interconnection details to the ICM, NSD/VNFDs to the REP, and Edge identity details to the AGA.

4.1.3 Step 3

The ICM, and REP components will respond to the SBR, if successful, with appropriate 201 `{ACK}` responses to denote those new entries have been created in their respective registries.

4.1.4 Step 4

The AGA will take the `{{edge_id}}` from the incoming `{POST}` request, and auto generate a new SSL certificate for that Edge to be used to secure communications. The AGA will also send a `{POST}` request to the EPA to register the new Edge.

4.1.5 Step 5

If the `{POST}` request to the EPA is successful, the EPA will respond to the AGA with a 201 `{ACK}` response.

4.1.6 Step 6

The AGA will then respond back to the SBR with a 201 `{ACK}` response to indicate that the edge has been fully registered, and the SSL certificate has been generated.

4.1.7 Step 7

The SBR will finally respond back to the GUI with a 201 `{ACK}` response to inform the end-user the edge has been registered successfully, including the new `{{edge_id}}`, as exampled in Figure 4-3.

```
{
  "ACK": {
    "origin": "service broker",
    "message": "edge registered successfully",
    "id": "7cbfa963-a05e-4f7f-b173-e658b4bfe4c1"
  }
}
```

Figure 4-3 Edge Registration Step 7 – 201 ACK Response

If steps 4, 5, and 6 fail, the SBR will send back either of the following `{NACK}` responses, depending on the nature of the failure. Error responses are captured with a `{NACK}` response exampled in Figure 4-4.

```
{
  "NACK": {
    "origin": "service broker",
    "message": "edge registration error",
    "reason": "edge registered in vios, but not edgeproxy"
  }
}
```

Figure 4-4 Edge Registration Step 7 – NACK Response

If no response is received from the destination, then a 408 **{NACK}** is returned, as exampled in Figure 4-5.

```
{
  "NACK": {
    "origin": "service broker",
    "message": "api gateway connection error",
    "reason": "no response from api gateway"
  }
}
```

Figure 4-5 Edge Registration Step 7 – 408 NACK Response

4.2 Deleting an Edge

When an edge facility is no longer required, the 5G-VIOS administrator can delete the relevant information about the edge from 5G-VIOS, and thus disconnecting that facility from 5G-VIOS. Figure 4-6 shows the step-by-step process of deleting an edge from 5G-VIOS.

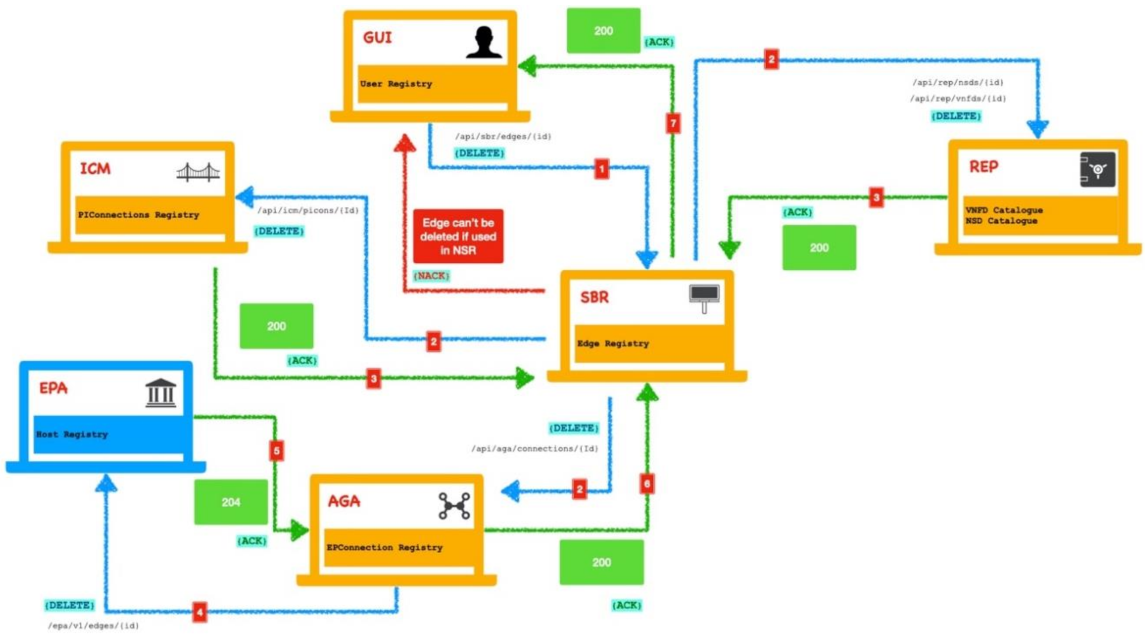


Figure 4-6 Edge Deletion Workflow Overview

4.2.1 Step 1

From the GUI, when the end-user clicks on the delete button, a **{DELETE}** request is sent to the SBR. The SBR validates the incoming **{DELETE}** request. The SBR will check if the edge can be deleted. If the Edge has already been used in an active NSR, then the SBR will send back to the GUI a 400 **{NACK}** response, as exampled in Figure 4-7.

```
{
  "NACK": {
    "origin": "service broker",
    "message": "unable to delete edge",
    "reason": "edge is used in a registered NSR"
  }
}
```

Figure 4-7 Edge Deletion Step 1 – 400 NACK Response

4.2.2 Step 2

If the SBR's delete check is valid, then `{DELETE}` requests are sent to other components in 5G-VIOS to delete entries from respective registries.

4.2.3 Step 3

The ICM, and REP components respond back to the SBR with 200 `{ACK}` responses.

4.2.4 Step 4

The AGA sends a `{DELETE}` request to the EPA.

4.2.5 Step 5

If the delete is successful, then the EPA will send back to the AGA a 204 `{ACK}` response.

4.2.6 Step 6

The AGA will then respond back to the SBR with a 200 `{ACK}` Response.

4.2.7 Step 7

The SBR will finally respond back to the GUI with a 200 `{ACK}` response to inform the user the edge has been deleted, as exampled in Figure 4-8.

```
{
  "ACK": {
    "origin": "service broker",
    "message": "edge has been deleted",
    "id": "None"
  }
}
```

Figure 4-8 Edge Deletion Step 7 – 200 ACK Response

If steps 4, 5, and 6 fail, the SBR will send back either of the following `{NACK}` responses, depending on the nature of the failure. Error responses are captured with a {NACK} response exampled in Figure 4-9.

```
{
  "NACK": {
    "origin": "service broker",
    "message": "edge deletion error",
    "reason": "unable to delete edge from api gateway"
  }
}
```

Figure 4-9 Edge Deletion Step 7 – NACK Response

If no response is received from the destination, then a 408 {NACK} is returned, as exampled in Figure 4-10.

```
{
  "NACK": {
    "origin": "service broker",
    "message": "api gateway connection error",
    "reason": "no response from api gateway"
  }
}
```

Figure 4-10 Edge Deletion Step 7 – 408 NACK Response

4.3 Create a new inter-domain network service (Experiment)

A user can create an experiment by selecting the NS packages available at different facilities through the 5G-VIOS portal. The experiment can utilise a single or multiple network domains / facilities. Once the user submits the request, the service composer performs the composition of selected NSs in the experiments and creates the deployment template (inter-edge network service descriptor). Figure 4-11 portrays the step-by-step process for the inter-domain NS creation workflow.

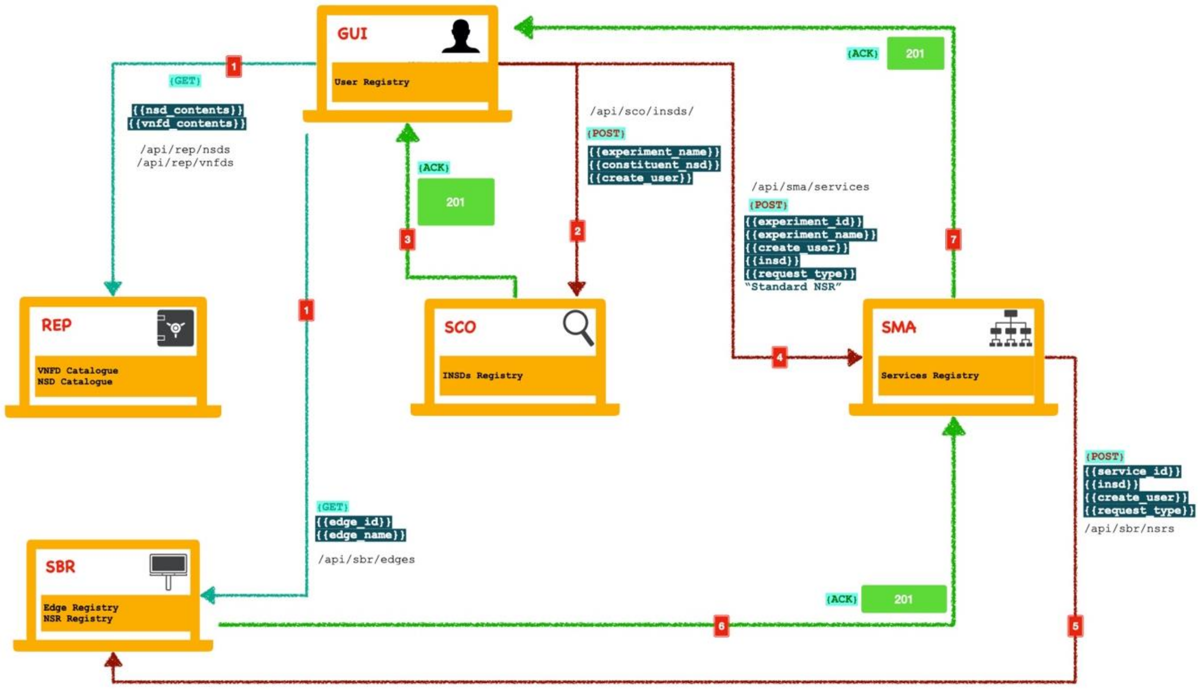


Figure 4-11 Inter-domain network service Creation Workflow Overview

4.3.1 Step 1

The GUI uses `{GET}` requests to collect the Edges from the SBR, and their corresponding NSDs/VNFDs from the REP so that a combination of resources can be selected at the end-user's discretion.

4.3.2 Step 2

The selected Edges and NSDs are sent as a `{POST}` request to the SCO to create a fully composed iNSD.

4.3.3 Step 3

If successful, the SCO will respond to the GUI with a 201 `{ACK}` response, including the fully composed iNSD.

4.3.4 Step 4

The fully composed `{{iNSD}}`, `{{experiment_name}}`, `{{experiment_id}}`, and the `{{create_user}}` are sent as a `{POST}` request to the SMA to register a new Service.

4.3.5 Step 5

The SMA sends a `{POST}` request to the SBR to register a Network Service Request (NSR) relating to the Service just created.

4.3.6 Step 6

If successful, the SBR will respond to the SMA with a 201 `{ACK}` response

4.3.7 Step 7

The SMA will then respond back to the GUI with a 201 `{ACK}` response, exampled in Figure 4-12, to inform the end-user that a new service has been created.

```
{
  "ACK": {
    "origin": "service manager",
    "message": "service registration successful",
    "id": "48e1cfa0-1820-4a5d-a04e-6f15eec7155e"
  }
}
```

Figure 4-12 Service Creation Step 7 – 201 ACK Response

If step 6 fails, the SMA will send back either of the following `{NACK}` responses, depending on the nature of the failure. Error responses are captured with a `{NACK}` response exampled in Figure 4-13.

```
{
  "NACK": {
    "origin": "service manager",
    "message": "service registration error",
    "reason": "unable to register new service"
  }
}
```

Figure 4-13 Service Creation Step 7 – NACK Response

If no response is received from the destination, then a 408 `{NACK}` is returned, as exampled in Figure 4-14.

```
{
  "NACK": {
    "origin": "service manager",
    "message": "service broker connection error",
    "reason": "no response from the service broker"
  }
}
```

Figure 4-14 Service Creation Step 7 – 408 NACK Response

4.4 Instantiate a Service/Experiment (Initiate)

Once the inter-domain NS has been created through the previous workflow, the user can deploy the experiment through the GUI portal. The deployment of inter-domain NS is done in two phases. The first one is the initiate process in which the 5G-VIOS performs a check for the resources (e.g., NSD is onboarded at edge OSM) available at each edge that is needed for the deployment of NS. The second phase is the actual deployment / instantiation of the NS that is described in the following sub-section (4.5). The step-by-step process for the initiate workflow is depicted in Figure 4-15, and further, the description of each step is provided in the next subsections.

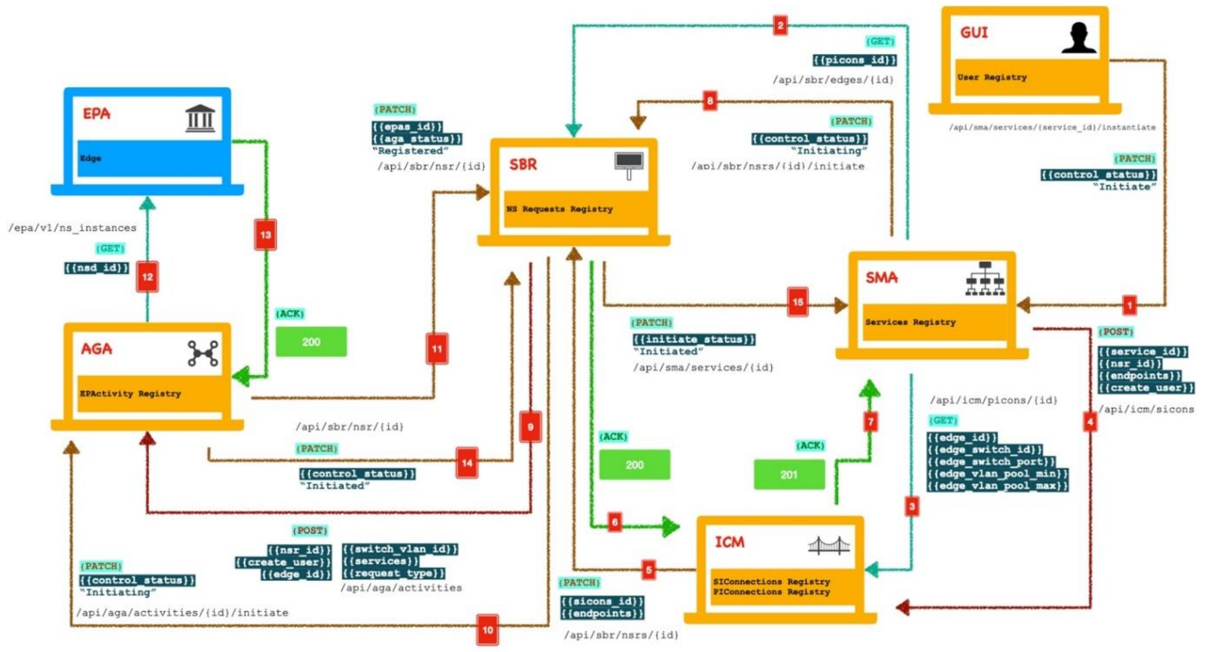


Figure 4-15 Service Instantiate (Initiate) Workflow Overview

4.4.1 Step 1

In the GUI, the end-user clicks on the Instantiate button. This sends a **{PATCH}** request to the SMA to change the `{{control_status}}` to “Initiate”, which instructs the SMA to begin the instantiate process for the selected Service.

4.4.2 Step 2

The SMA sends a **{GET}** request for each Edge in the iNSD to obtain that Edge’s `{{picons_id}}`.

4.4.3 Step 3

The SMA uses each `{{picons_id}}` for each Edge and sends a **{GET}** request to the ICM to obtain physical interconnection details for each Edge.

4.4.4 Step 4

The SMA combines the physical interconnect details for each Edge into an `{{endpoints}}` JSON object, and along with `{{service_id}}`, `{{nsr_id}}`, and `{{create_user}}`, sends a **{POST}** request to the ICM to setup the service interconnection record for the given service.

4.4.5 Step 5

The ICM will then send the `{{endpoints}}` and `{{sicons_id}}` as a **{PATCH}** request to the SBR to update the related NSR.

4.4.6 Step 6

If successful, the SBR will respond to the ICM with a 200 **{ACK}** response.

4.4.7 Step 7

The ICM will respond back to the SMA with a 201 **{ACK}** response.

4.4.8 Step 8

The SMA will automatically send a `{PATCH}` request to the SBR to change the `{{control_status}}` of the NSR to "Initiating". The SMA will also respond back to the GUI with a 200 `{ACK}` response, exampled in Figure 4-16, to inform the end-user that a instantiate request has been sent for that service.

```
{
  "ACK": {
    "origin": "service manager",
    "message": "instantiate request sent",
    "id": "48e1cfa0-1820-4a5d-a04e-6f15eec7155e"
  }
}
```

Figure 4-16 Service Instantiate Step 8 – 200 ACK Response

If steps 6 and 7 fails, the SMA will send back to the GUI either of the following `{NACK}` responses, depending on the nature of the failure. Error responses are captured with a `{NACK}` response exampled in Figure 4-17.

```
{
  "NACK": {
    "origin": "service manager",
    "message": "service instantiate error ",
    "reason": "unable to instantiate service "
  }
}
```

Figure 4-17 Service Instantiate Step 8 – NACK Response

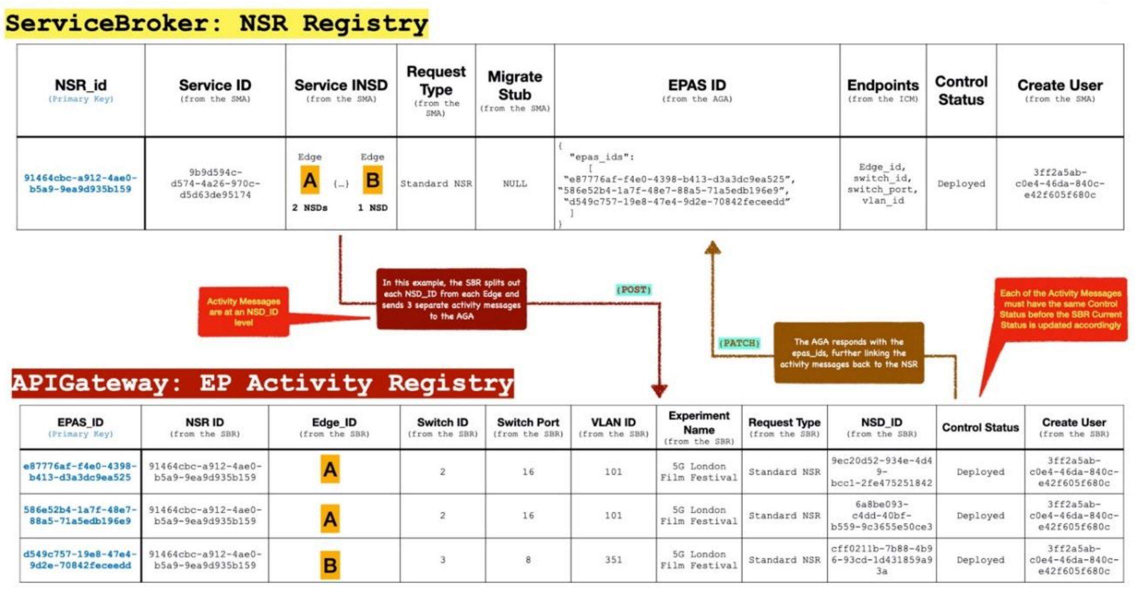
If no response is received from the destination, then a 408 {NACK} is returned, as exampled in Figure 4-18.

```
{
  "NACK": {
    "origin": "service manager",
    "message": "connectivity manager connection error ",
    "reason": "no response from the connectivity manager "
  }
}
```

Figure 4-18 Service Instantiate Step 8 – 408 NACK Response

4.4.9 Step 9

The SBR will extract the NSD_IDs from each Edge in the `{{iNSD}}` and sends separate `{POST}` requests to the AGA to create their corresponding activity messages, as exampled in Figure 4-19.



ServiceBroker: NSR Registry

NSR_id (Primary Key)	Service ID (from the SMA)	Service INSD (from the SMA)	Request Type (from the SMA)	Migrate Stub (from the SMA)	EPAS ID (from the AGA)	Endpoints (from the ICM)	Control Status	Create User (from the SMA)
91464cbc-a912-4ae0-b5a9-9ea9d935b159	9b9d594c-d574-4a26-970c-d5d63de95174	Edge A 2 NSDs {…} Edge B 1 NSD	Standard NSR	NULL	{ "epas_ids": ["e87776af-f4e0-4398-b413-d3a3dc9ea525", "586e52b4-1a7f-48e7-88a5-71a5edb196e9", "d549c757-19e8-47e4-9d2e-70842feceedd"] }	Edge_id, switch_id, switch_port, vlan_id	Deployed	3ff2a5ab-c0e4-46da-840c-e42f605f680c

APIGateway: EP Activity Registry

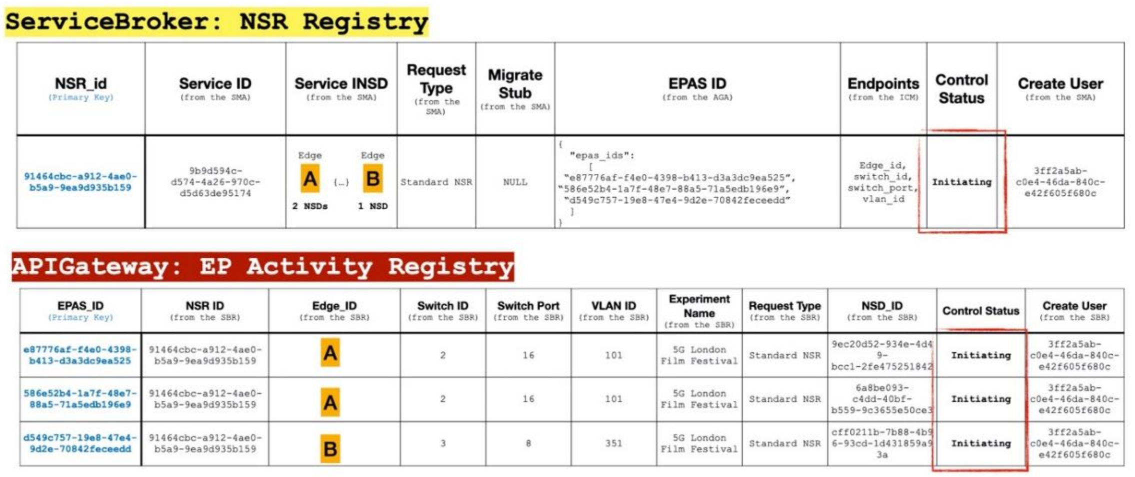
EPAS_ID (Primary Key)	NSR ID (from the SBR)	Edge_ID (from the SBR)	Switch ID (from the SBR)	Switch Port (from the SBR)	VLAN ID (from the SBR)	Experiment Name (from the SBR)	Request Type (from the SBR)	NSD_ID (from the SBR)	Control Status	Create User (from the SBR)
e87776af-f4e0-4398-b413-d3a3dc9ea525	91464cbc-a912-4ae0-b5a9-9ea9d935b159	A	2	16	101	5G London Film Festival	Standard NSR	9ec20d52-934e-4d49-bcc1-2fe475251842	Deployed	3ff2a5ab-c0e4-46da-840c-e42f605f680c
586e52b4-1a7f-48e7-88a5-71a5edb196e9	91464cbc-a912-4ae0-b5a9-9ea9d935b159	A	2	16	101	5G London Film Festival	Standard NSR	6a8be093-c4dd-40bf-b559-9c3655e50ce3	Deployed	3ff2a5ab-c0e4-46da-840c-e42f605f680c
d549c757-19e8-47e4-9d2e-70842feceedd	91464cbc-a912-4ae0-b5a9-9ea9d935b159	B	3	8	351	5G London Film Festival	Standard NSR	cff0211b-7b88-4b96-93cd-1d431859a93a	Deployed	3ff2a5ab-c0e4-46da-840c-e42f605f680c

Activity Messages are at an NSD_ID level
In this example, the SBR splits out each NSD_ID from each Edge and sends 3 separate activity messages to the AGA
{POST}
{PATCH}
The AGA responds with the epas_ids, further linking the activity messages back to the NSR
Each of the Activity Messages must have the same Control Status before the SBR Current Status is updated accordingly

Figure 4-19 Service Instantiate Step 10 – SBR to AGA Data Example 1

4.4.10 Step 10

The SBR will then send corresponding **{PATCH}** requests to the AGA for each of the activity messages relating to the Service and NSR to be instantiated to update the `{{control_status}}` to “Initiating”, as exampled in Figure 4-20.



ServiceBroker: NSR Registry

NSR_id (Primary Key)	Service ID (from the SMA)	Service INSD (from the SMA)	Request Type (from the SMA)	Migrate Stub (from the SMA)	EPAS ID (from the AGA)	Endpoints (from the ICM)	Control Status	Create User (from the SMA)
91464cbc-a912-4ae0-b5a9-9ea9d935b159	9b9d594c-d574-4a26-970c-d5d63de95174	Edge A 2 NSDs {…} Edge B 1 NSD	Standard NSR	NULL	{ "epas_ids": ["e87776af-f4e0-4398-b413-d3a3dc9ea525", "586e52b4-1a7f-48e7-88a5-71a5edb196e9", "d549c757-19e8-47e4-9d2e-70842feceedd"] }	Edge_id, switch_id, switch_port, vlan_id	Initiating	3ff2a5ab-c0e4-46da-840c-e42f605f680c

APIGateway: EP Activity Registry

EPAS_ID (Primary Key)	NSR ID (from the SBR)	Edge_ID (from the SBR)	Switch ID (from the SBR)	Switch Port (from the SBR)	VLAN ID (from the SBR)	Experiment Name (from the SBR)	Request Type (from the SBR)	NSD_ID (from the SBR)	Control Status	Create User (from the SBR)
e87776af-f4e0-4398-b413-d3a3dc9ea525	91464cbc-a912-4ae0-b5a9-9ea9d935b159	A	2	16	101	5G London Film Festival	Standard NSR	9ec20d52-934e-4d49-bcc1-2fe475251842	Initiating	3ff2a5ab-c0e4-46da-840c-e42f605f680c
586e52b4-1a7f-48e7-88a5-71a5edb196e9	91464cbc-a912-4ae0-b5a9-9ea9d935b159	A	2	16	101	5G London Film Festival	Standard NSR	6a8be093-c4dd-40bf-b559-9c3655e50ce3	Initiating	3ff2a5ab-c0e4-46da-840c-e42f605f680c
d549c757-19e8-47e4-9d2e-70842feceedd	91464cbc-a912-4ae0-b5a9-9ea9d935b159	B	3	8	351	5G London Film Festival	Standard NSR	cff0211b-7b88-4b96-93cd-1d431859a93a	Initiating	3ff2a5ab-c0e4-46da-840c-e42f605f680c

Figure 4-20 Service Instantiate Step 10 – SBR to AGA Data Example 2

4.4.11 Step 11

The AGA will respond with **{PATCH}** requests to the SBR to update the corresponding NSR with the associated `{{epas_ids}}` pertaining to the NSR in question.

4.4.12 Step 12

For each activity message group, the AGA will send **{GET}** requests to the EPA to determine if each NSD_ID exists.

4.4.13 Step 13

The EPA will respond with 200 **{ACK}** responses, and the AGA will update the `{{control_status}}` with “Initiated” flags for each activity message that was successful.

4.4.14 Step 14

The AGA will check the `{{control_status}}` of each activity message in the group, and if all statuses are "Initiated", the AGA will then send a **`{PATCH}`** request to the SBR to update the `{{control_status}}` to "Initiated".

4.4.15 Step 15

The SBR will then automatically send a **`{PATCH}`** request to the SMA to update the `{{control_status}}` to "Initiated".

4.5 Instantiate a Service/Experiment (Deploy)

The actual deployment of each NS belonging to an inter-domain NS to its corresponding edge is performed once the initiate workflow is successful. For instantiation of each NS, the 5G-VIOS components send the request to edge-proxy via the API gateway. Then, the edge proxy communicates to the OSM via its NBI to instantiate the NS on the corresponding infrastructure. Figure 4-21 depicts a step-by-step process of NS instantiation to the edge facilities.

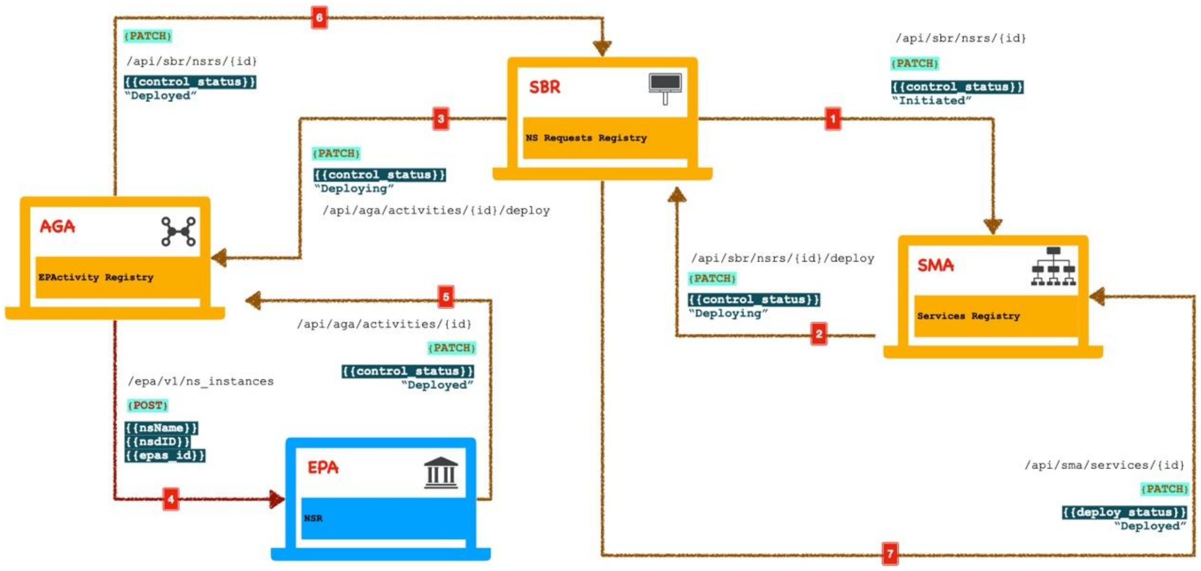


Figure 4-21 Network service Instantiate (Deploy) Workflow Overview

4.5.1 Step 1

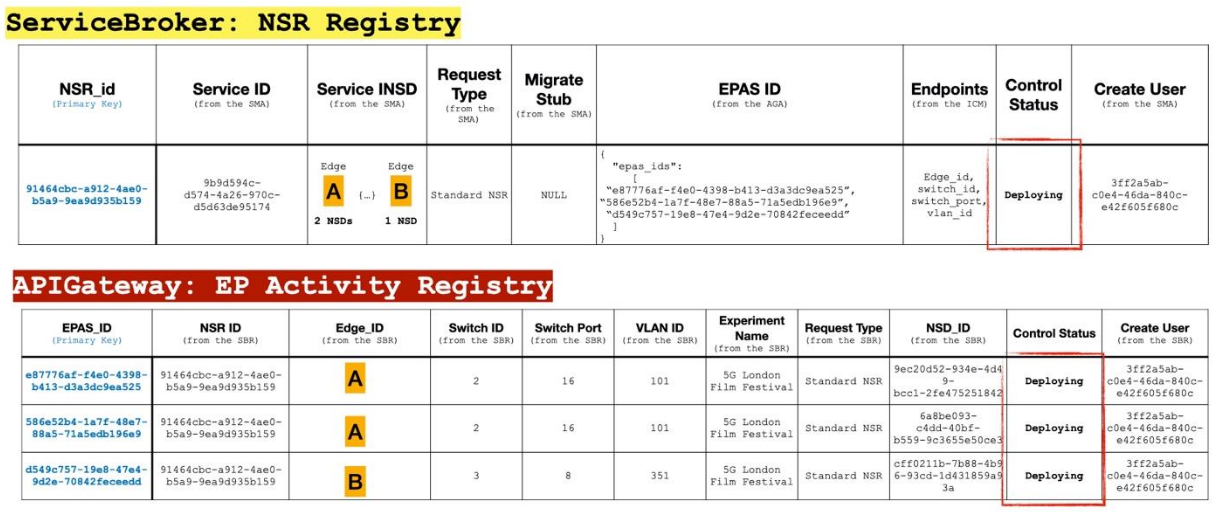
When the SBR sends the **`{PATCH}`** request to the SMA to update the `{{control_status}}` for the given service in question to "Initiated", this automatically starts the deploy process.

4.5.2 Step 2

The SMA will send a **`{PATCH}`** request to the SBR to change the `{{control_status}}` to "Deploying".

4.5.3 Step 3

The SBR sends a **`{PATCH}`** request to the AGA to update the `{{control_status}}` to "Deploying" for each of the activity messages in the group, as exampled in Figure 4-22.



ServiceBroker: NSR Registry

NSR_id (Primary Key)	Service ID (from the SMA)	Service INSD (from the SMA)	Request Type (from the SMA)	Migrate Stub (from the SMA)	EPAS ID (from the AGA)	Endpoints (from the ICM)	Control Status	Create User (from the SMA)
91464cbc-a912-4ae0-b5a9-9ea9d935b159	9b9d594c-d574-4a26-970c-d5d63de95174	Edge A 2 NSDs {…} Edge B 1 NSD	Standard NSR	NULL	{ "epas_ids": ["e87776af-f4e0-4398-b413-d3a3dc9ea525", "586e52b4-1a7f-48e7-88a5-71a5edb196e9", "d549c757-19e8-47e4-9d2e-70842feceedd"] }	Edge_id, switch_id, switch_port, vlan_id	Deploying	3ff2a5ab-c0e4-46da-840c-e42f605f680c

APIGateway: EP Activity Registry

EPAS_ID (Primary Key)	NSR ID (from the SBR)	Edge_ID (from the SBR)	Switch ID (from the SBR)	Switch Port (from the SBR)	VLAN ID (from the SBR)	Experiment Name (from the SBR)	Request Type (from the SBR)	NSD_ID (from the SBR)	Control Status	Create User (from the SBR)
e87776af-f4e0-4398-b413-d3a3dc9ea525	91464cbc-a912-4ae0-b5a9-9ea9d935b159	A	2	16	101	5G London Film Festival	Standard NSR	9ec20d52-934e-4d49-bcc1-2fe475251842	Deploying	3ff2a5ab-c0e4-46da-840c-e42f605f680c
586e52b4-1a7f-48e7-88a5-71a5edb196e9	91464cbc-a912-4ae0-b5a9-9ea9d935b159	A	2	16	101	5G London Film Festival	Standard NSR	6a8be093-c4dd-40bf-b559-9c3655e50ce3	Deploying	3ff2a5ab-c0e4-46da-840c-e42f605f680c
d549c757-19e8-47e4-9d2e-70842feceedd	91464cbc-a912-4ae0-b5a9-9ea9d935b159	B	3	8	351	5G London Film Festival	Standard NSR	cff0211b-7b88-4b96-93cd-1d431859a93a	Deploying	3ff2a5ab-c0e4-46da-840c-e42f605f680c

Figure 4-22 Service Instantiate Step 10 – SBR to AGA Data Example 3

4.5.4 Step 4

For each activity message in the group, the AGA will send a **{POST}** request to the EPA with `{{nsName}}`, `{{nsdID}}`, and `{{epas_id}}` to instruct the EPA to deploy on the relevant Edge.

4.5.5 Step 5

If successful, the EPA will respond with a **{PATCH}** request to the AGA to update the `{{control_status}}` for that specific activity message to "Deployed".

4.5.6 Step 6

The AGA will check the `{{control_status}}` of each activity message in the group, and if all statuses are "Deployed", the AGA will then send a **{PATCH}** request to the SBR to update the `{{control_status}}` to "Deploy".

4.5.7 Step 7

The SBR will then automatically send a **{PATCH}** request to the SMA to update the `{{deploy_status}}` to "Deployed".

4.6 Migrate a Service/Experiment

Migration of a NS belonging to an experiment (inter-edge NS) is performed for the mobile end-user and triggered when the user moves from the coverage area of one edge to another edge. This relates to the NS migration, not to connectivity handover. At this stage the migration workflow is initiated by the end-user application which track the position of the user. Figure 4-23depicts all the steps required to perform the migration workflow.

31. Jan. 2022

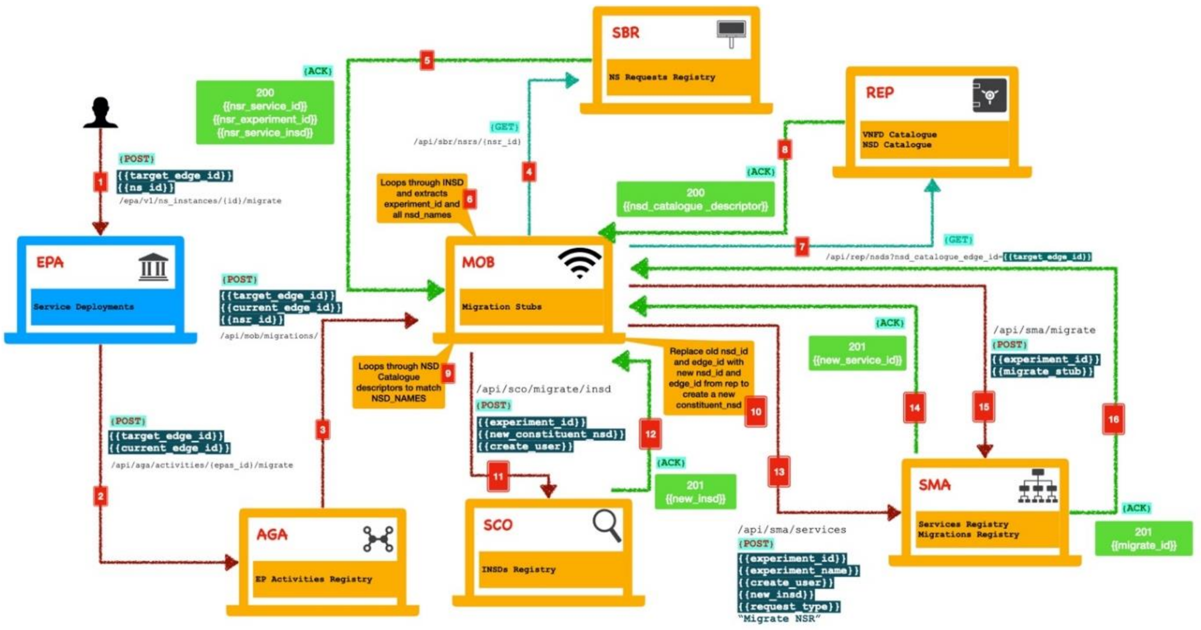


Figure 4-23 Migrate Service Workflow Overview

4.6.1 Step 1

The end-user Application will send the `{{target_edge_id}}` and the `{{ns_id}}` in a **`{POST}`** request to the EPA.

4.6.2 Step 2

The EPA will then send a **`{POST}`** request with the `{{target_edge_id}}` and `{{current_edge_id}}` to the AGA.

4.6.3 Step 3

The AGA will send a **`{POST}`** request with the `{{target_edge_id}}`, `{{current_edge_id}}` and `{{nsr_id}}` to the MOB to create a new migration stub.

4.6.4 Step 4

The MOB uses the `{{nsr_id}}` to perform a **`{GET}`** request on the SBR NS Requests Registry.

4.6.5 Step 5

The SBR will return the whole NSR, including the `{{nsr_service_id}}`, `{{nsr_experiment_id}}`, and `{{nsr_service_insd}}` back to the MOB.

4.6.6 Step 6

The MOB will loop through the `{{nsr_service_insd}}` to extract all the `{{nsdNames}}` for each edge.

4.6.7 Step 7

The MOB will then send a **`{GET}`** request to the REP for all the NSD records for the `{{target_edge_id}}`.

4.6.8 Step 8

The REP will respond with all NSD records that match the `{{target_edge_id}}`.

4.6.9 Step 9

The MOB loops through each NSD record to perform a `{{nsdNames}}` match to see if all the NSDs from the current edge are also present on the new edge.

4.6.10 Step 10

The MOB will create a new `{{constituent_nsd}}` object, replacing the current edge with the target edge.

4.6.11 Step 11

The MOB will send the new `{{constituent_nsd}}` object as a **`{POST}`** request to the SCO.

4.6.12 Step 12

The SCO responds with a new INSD.

4.6.13 Step 13

The MOB sends a **`{POST}`** request to the SMA to create a new Service with the new INSD.

4.6.14 Step 14

The SMA responds back to the MOB with the `{{new_service_id}}`.

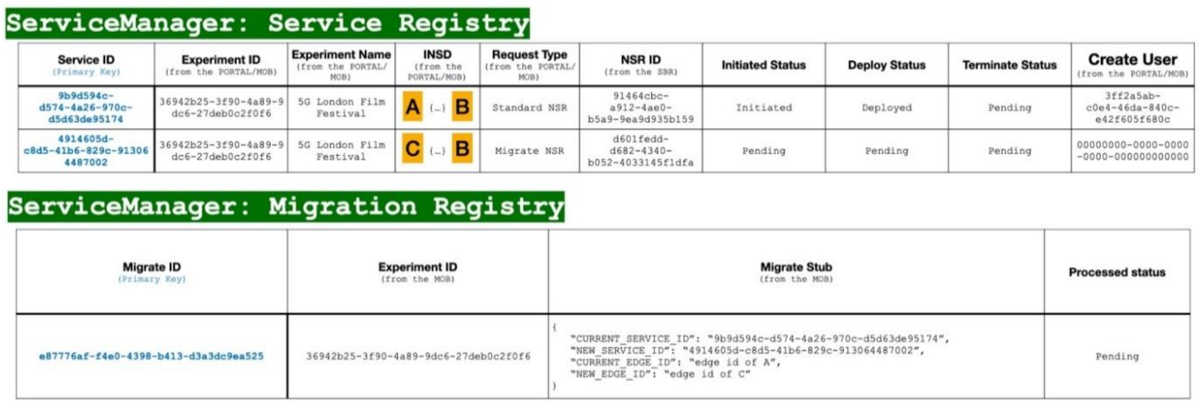
4.6.15 Step 15

The `{{new_service_id}}` is incorporated into the migration stub, and the migration stub is then sent as a **`{POST}`** request to the SMA. The migration stub is exampled in Figure 4-24.

```
{
  "current_edge_id": "a2312327-1ea8-487a-8c68-c6b1d6a737aa",
  "new_edge_id": "55b1fac2-c239-4800-a7df-72b29c4a2dbb ",
  "current_service_id": "0fc7f20f-b999-4611-a6cb-028c03b44979",
  "new_service_id": "710a6221-5953-4eeb-b691-4b1e1afe215b "
}
```

Figure 4-24 Service Migration Step 15 - Example Migration Stub

Once step 15 is complete, both the Service and Migration Registries would have data as exampled in Figure 4-25.



ServiceManager: Service Registry

Service ID (Primary Key)	Experiment ID (from the PORTAL/MOB)	Experiment Name (from the PORTAL/MOB)	INSD (from the PORTAL/MOB)	Request Type (from the PORTAL/MOB)	NSR ID (from the SBR)	Initiated Status	Deploy Status	Terminate Status	Create User (from the PORTAL/MOB)
9b9d594c-d574-4a26-970c-d5d63de95174	36942b25-3f90-4a89-9dc6-27deb0c2f0f6	5G London Film Festival	A {…} B	Standard NSR	91464cbc-a912-4ae0-b5a9-9ea9d935b159	Initiated	Deployed	Pending	3ff2a5ab-c0e4-46da-840c-e42f605f680c
4914605d-c8d5-41b6-829c-913064487002	36942b25-3f90-4a89-9dc6-27deb0c2f0f6	5G London Film Festival	C {…} B	Migrate NSR	d601fedd-d682-4340-b052-4033145f1dfa	Pending	Pending	Pending	00000000-0000-0000-0000-000000000000

ServiceManager: Migration Registry

Migrate ID (Primary Key)	Experiment ID (from the MOB)	Migrate Stub (from the MOB)	Processed status
e87776af-f4e0-4398-b413-d3a3dc9ea525	36942b25-3f90-4a89-9dc6-27deb0c2f0f6	{ "CURRENT_SERVICE_ID": "9b9d594c-d574-4a26-970c-d5d63de95174", "NEW_SERVICE_ID": "4914605d-c8d5-41b6-829c-913064487002", "CURRENT_EDGE_ID": "edge id of A", "NEW_EDGE_ID": "edge id of C" }	Pending

Figure 4-25 Service Migration Step 15 – SMA Data example

4.6.16 Step 16

The SMA will respond back to the MOB with the new `{{migrate_id}}`.

The MOB will then send the instantiate request to the SMA to begin the initiate and deploy process of the new service.

4.7 Terminate a Service/Experiment

Finally, once an experiment has completed its run, it is time to terminate the NSs. The termination of a NS removes all network resources associated with this NS from each facility. A termination of a NS is triggered in two scenarios: (i) when a user manually terminates the network service from the portal and (ii) when a NS migrates from one edge to another edge. The termination request is send from either the portal or the mobility manager components to edge proxy via the API gateway, and then, edge proxy sends a termination request to the OSM platform available at the edge facility. A step-by-step workflow is shown in Figure 4-26.

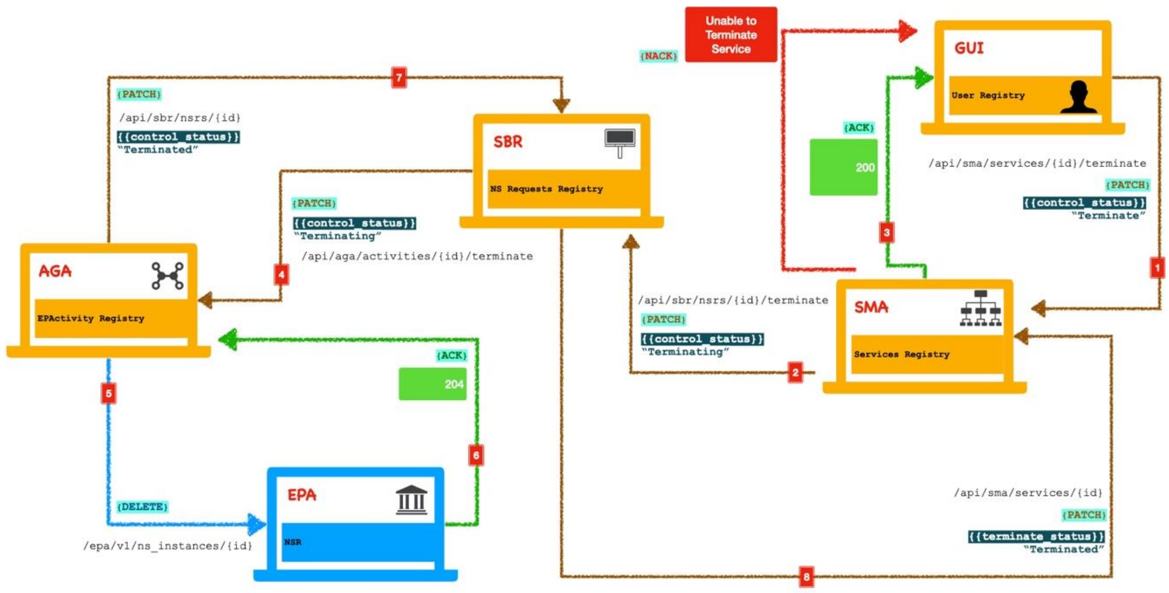


Figure 4-26 Service Terminate Workflow Overview

4.7.1 Step 1

Through the GUI, the end-user clicks on the terminate button. A **{PATCH}** request is sent to the SMA to update the `{{control_status}}` to "Terminate" for the corresponding Service. The SMA will check if the Service can be terminated at this stage. If the termination check fails, the SMA will respond back to the GUI with a 400 **{NACK}** response, as exampled in Figure 4-27.

```
{
  "NACK": {
    "origin": "service manager",
    "message": "unable to terminate",
    "reason": "the service must be first deployed"
  }
}
```

Figure 4-27 Service Terminate Step 1 – 400 NACK Response

4.7.2 Step 2

If the terminate check passes, the SMA sends a **{PATCH}** request to the SBR to update the `{{control_status}}` to "Terminating".

4.7.3 Step 3

The SMA will respond back to the GUI with a 200 {ACK} response to inform the end-user that the terminate request has been sent, as exampled in Figure 4-28.

```
{
  "ACK": {
    "origin": "service manager",
    "message": "terminate request sent",
    "id": "48e1cfa0-1820-4a5d-a04e-6f15eec7155e"
  }
}
```

Figure 4-28 Service Terminate Step 3 – 200 ACK Response

4.7.4 Step 4

The SBR will then send a {PATCH} request to the AGA for each of the activity messages in the group to update the {{control_status}} to "Terminating".

4.7.5 Step 5

For each activity message in the group, the AGA will send {DELETE} requests to the respective EPAs to delete the Service from each Edge's orchestrator.

4.7.6 Step 6

Each EPA will send a 204 {ACK} response to denote that the Service has been removed from the Edge's orchestrator. The AGA will update the {{control_status}} of each activity message in the group to "Terminated".

4.7.7 Step 7

The AGA will check the {{control_status}} of each activity message in the group, and if all statuses are "Terminated", the AGA will then send a {PATCH} request to the SBR to update the {{control_status}} to "Terminated".

4.7.8 Step 8

The SBR will then automatically send a {PATCH} request to the SMA to update the {{terminate_status}} to "Deployed".

4.8 Delete a Service/Experiment

After the termination of inter-edge NS, all the resources corresponding to each NS are removed from the edge infrastructure. However, information related to the NS is not removed completely from the 5G-VIOS, mainly for future reference, e.g. redeployment of that NS. The delete workflow removes all the information related to the inter-domain NS from all the components databases. A detailed description of the deletion workflow is depicted in Figure 4-29.

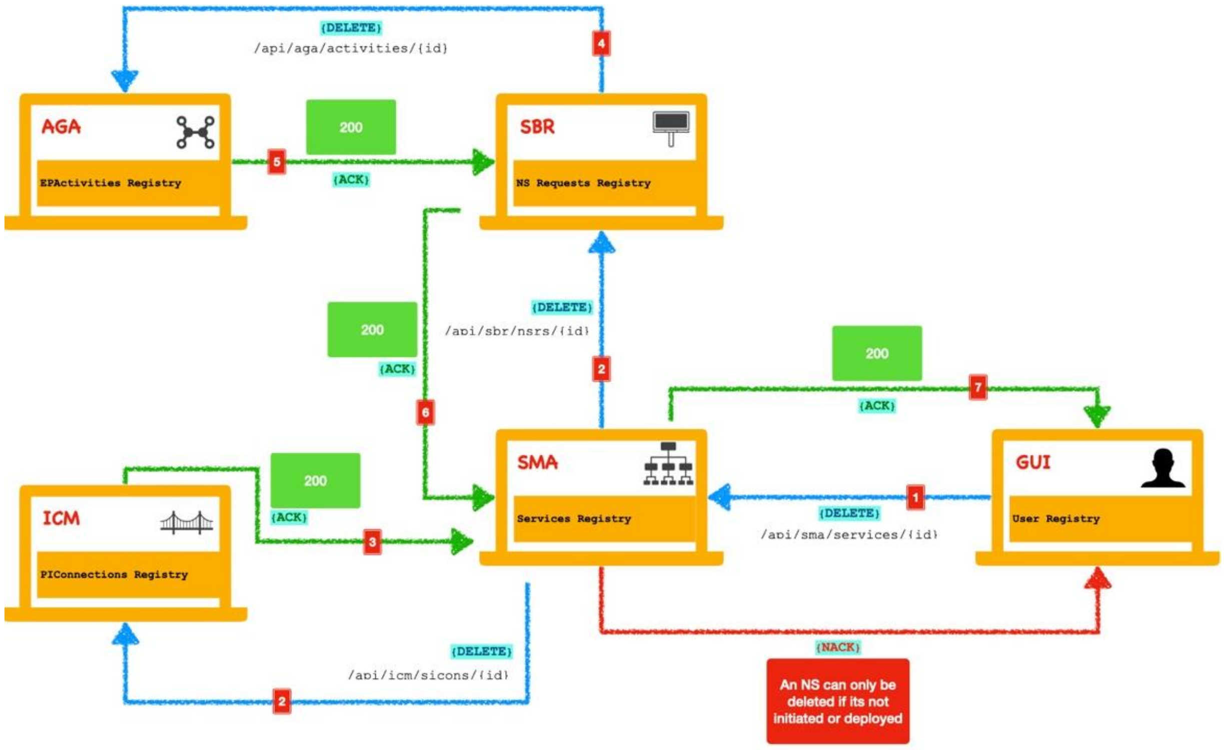


Figure 4-29 Service Delete Workflow Overview

4.8.1 Step 1

Through the GUI, the end-user clicks on the delete button. The SMA will check if the service can be deleted at this stage. If the deletion check fails, the SMA will respond back to the GUI with a 400 `{NACK}` response, as exampled in Figure 4-30.

```
{
  "NACK": {
    "origin": "service manager",
    "message": "unable to delete service",
    "reason": "terminate request must be sent first"
  }
}
```

Figure 4-30 Service Deletion Step 1 – 400 NACK Response

4.8.2 Step 2

If the delete check passes, the SMA sends `{DELETE}` requests to the SBR, and ICM.

4.8.3 Step 3

If successful, the ICM will respond with a 200 `{ACK}` response.

4.8.4 Step 4

The SBR then sends a `{DELETE}` request to the AGA for each activity message that relates to the corresponding NSR.

4.8.5 Step 5

If successful, the AGA will respond with a 200 `{ACK}` response.

4.8.6 Step 6

The SBR responds to the SMA with a 200 `{ACK}` response to denote the corresponding NSR has been deleted.

Step 7

The SMA will then respond back to the GUI with a 200 `{ACK}` response, exampled in Figure 4-31, to inform the end-user that a Service has been deleted.

```
{
  "ACK": {
    "origin": "service manager",
    "message": "service deletion successful",
    "id": "None"
  }
}
```

Figure 4-31 Service Delete Step 7 – 200 ACK Response

5 Conclusions

The 5G-VICTORI project builds upon the architectural concept of a single E2E platform that spans across multiple facilities providing interconnection and internetworking among them, thus creating a common infrastructure of integrated network and compute/storage resources.

To facilitate this, a multi-domain orchestration platform is being developed, the 5G-VIOS. 5G-VIOS is a holistic platform that can automate the deployment of a network slice as an experiment across multiple facilities and configure the various underlying technologies without the myriad of minute information. The vertical user is interacting with the underlying facilities through a graphical interface for easier access, simply selecting the network services it needs and where they should be instantiated. 5G-VIOS is then responsive to automate the deployment of those network services across the different domains that the user has selected. Further, 5G-VIOS provides an AI/ML based data collection and analytic functionality, implemented within the Profiling component, for optimising the deployment of network services on different facilities.

This deliverable presents the specifications of the 5G-VIOS implementation and the integration to individual facilities. The microservice-based design of the 5G-VIOS framework provides flexibility such as scalability, resilience, faster deployment cycle, quick debugging, easy maintenance, etc., during integration. The deployment of 5G-VIOS follows a cloud native approach in which all its containerized components can be deployed on any commercial and non-commercial Kubernetes cluster as infrastructure-as-a-code. The 5G-VIOS components extend certain 5G NFs such as NRF, NEF, SEPP to support inter-domain orchestration, connectivity, and services as mentioned below.

Key components for the delivery of 5G-VIOS functionality include the following:

- Inter-edge Connectivity Manager (ICM) component follows software-defined wide area networking (SD-WAN) and Secure Access Service Edge (SASE) principles exploiting Dynamic Multipoint Virtual Private Network (DMVPN), with a hub and spoke architecture. Once network services are deployed at the various facility, ICM, with the help of DMVPN, creates a transport network/slice. ICM uses multipoint GRE to create a tunnel with multiple destinations, NHRP to resolve the IP address of 5G-VIOS and each facility, IPSec for creating secure connection by encryption, and OSPF for connecting two network functions deployed at distinct facilities.
- The repository is an upgraded version of the Network Repository Function (NRF) that includes all the functionalities of NRF. The information stored and managed by the repository consists of registration information of all the facilities/edges, exposed network functions from different facilities, and the details about Virtual Network Function Descriptor (VNFD) and Network Service Descriptor (NSD) onboarded to the edge orchestrators.
- The Edge Proxy component is responsible for integrating individual facilities with 5G-VIOS. It is an extended version of 5G SEPP which enables connection between network functions available at the distinct facility. It exposes the northbound API endpoints to enable communication between the 5G-VIOS component and different technologies available at different facilities such as network orchestrator (NFVO), edge monitoring and profiling, NetOS (specific to Bristol facility), etc. The edge proxy defines a connector at the southbound interface for each technology available at the facility.

- Profiling component that is using advanced AI/ML algorithms to optimise the placement of inter-domain network services. The VNF Profiling component in 5G-VIOS uses monitoring information to create mathematical or computational models for the performance of VNFs known as profiles. In the light of the NWDAF, the 5G-VIOS profiling's role is to uncover this relationship between the resource configuration, service demand, and performance targets.
- Portal provides an interface for the user to interact with 5G-VICTORI services and facilities. Using the portal interface, the user can create, manage, and monitor experiments and communicate with the required 5G-VIOS components to relay user input and trigger the 5G-VIOS workflows to fulfil the user requests. Portal has been implemented using 5GENESIS, an ICT-17 project, and composed of a Flask-based Portal web server, another Flask-based web server for data storage (for the test case, UE, and network scenario information), and the client-side JavaScript and jQuery scripts.

The updated step-by-step workflows for the key functionalities of 5G-VIOS have enabled to visualise the operation of 5G-VIOS.

Next steps for the development of 5G-VIOS include the integration and testing with the rest of the 5G-VICTORI facilities, tweaking where needed the Edge Proxy to facilitate the specific implementation and orchestrators that are available, such as the potential variations in OSM releases available on each facility.

6 References

[1] 5GVICTORI, “Deliverable D2.5 - 5G VICTORI Infrastructure Operating System - Initial Design Specification,” 2020.

[2] 5G-VICTORI, “Deliverable D2.4 5G-VICTORI end-to-end reference architecture,” to be published.

[3] G. Mayer, “RESTful APIs for the 5G Service Based Architecture,” *Journal of ICT Standardization,* vol. 6, pp. 101-116, 2018.

[4] 5G-VICTORI, “5G-VIOS Github Repositoty,” [Online]. Available: https://github.com/5G-VICTORI-project.

[5] 5G, Network function repository services, document 3GPP TS 29.510, Version 16.5.0, Release 16, 3GPP 2020-11.

[6] VyOS an opensource linux-based operating system for routers and firewalls, [online] Available: https://vyos.io/.

[7] 5G, Public Land Mobile Network (PLMN) Interconnection, document 29.573, Version 17.2.0, Release 17, 3GPP 2021-09.

[8] 5G-VICTORI, “Deliverable D4.1 Field trials methodology and guidelines,” 2020.

[9] S. Moazzeni, P. Jaisudthi, A. Bravalheri, N. Uniyal, X. Vasilakos, R. Nejabati and D. Simeonidou, “A novel autonomous profilling method for next generation NFV orchestrators,” *IEEE Transactions on Network and Service Management,* vol. 18, pp. 642-655, 2021.

[10] M. Peuster and H. Karl, “Understand your chains: Towards perfornance profile-based network service management,” in *5th European Workshop Software Defined Network*, 2016.

[11] M. Peuster and H. Karl, “Understand your chains and keep your deadlines: Introducing time-constrained profiling for NFV,” in *14th International Conference on Network Service Management*, 2018.

[12] Y.-C. Chang, “N-Dimension golden selection searcg - Its variants and limitations,” in *2nd International Conf. Biomedical Engineering and Informatics*, Tianjin, China, 2009.

[13] N. Uniyal, A. S. Muqaddas, D. Gkounis, A. Bravalheri, S. Moazzeni, F. Sardis, M. Dohler, R. Nejabati and D. Simeonidou, “5GUK Exchange: Towards Sustainable End-to-End Multi-Domain Orchestration of Softwarized 5G Networks,” *Computer Networks,* vol. 178, 2020.