5Genesis

5TH GENERATION END-TO-END NETWORK, EXPERIMENTATION, SYSTEM INTEGRATION, AND SHOWCASING

[H2020 - Grant Agreement No. 815178]

Deliverable D3.2

Management and Orchestration (Release B)

Editor E. Jimeno (ATOS)

Contributors ATOS (Atos Spain SA), UMA (Universidad de Malaga),NCSRD (National Center for Scientific Research "Demokritos"), FhG (Fraunhofer Gesellschaft Zur Foerderung derAngewandten Forschung E.V.), UPV (Universitat Politecnica de Valencia), UNIS (University ofSurrey), LMI (L.M. Ericsson Limited), AVA (Avanti Hylas 2 Cyprus Limited), TID (Telefónica I+D),INF (INFOLYSIS P.C.)

Version 1.0

Date March 30th, 2021

Distribution PUBLIC (PU)

List of Authors

Fraunhofer	FOKUS-Fraunhofer Gesellschaft e.V., Institute for Open Communication Systems
A. Prakash, S. K. Rajaguru, F. Eichhorn, M. Emmelmann, O. Keil	
INT	INTEL
V. Frascolla	
NCSRD	NATIONAL CENTER FOR SCIENTIFIC RESEARCH "DEMOKRITOS"
G. Xilouris, H. Koumaras,	
UMA	UNIVERSITY OF MALAGA
A. Díaz, I. González, B. García, P. Merino	
UPV	Universitat Politecnica de Valencia
A. Fornés, R. Onus	
ATOS	ATOS SPAIN SA
E. Jimeno, L. Gomez, J. Melian	

Disclaimer

The information, documentation and figures available in this deliverable are written by the 5GENESIS Consortium partners under EC co-financing (project H2020-ICT-815178) and do not necessarily reflect the view of the European Commission.

The information in this document is provided “as is”, and no guarantee or warranty is given that the information is fit for any particular purpose. The reader uses the information at his/her sole risk and liability.

Copyright

Version History

Rev. N	Description	Author	Date
1.0	Release of D3.2	ATOS	26/03/2021

LIST OF ACRONYMS

Acronym	Meaning
5G PPP	5G Infrastructure Public Private Partnership
API	Application Programmable Interface
BSS	Business Support System
CP	Connection Point
CSV	Comma Separated Values
ETL	Extract Transform Load (Data process)
ETSI	European Telecommunications Standards institute
HNF	Hybrid Network Function
HTTP	Hypertext Transfer Protocol
JSON	JavaScript Object Notation
KNF	Kubernetes Network Function
LW-UI	Light Weight User Interface
MANO	NFV Management and Organisation
MCS	Mission Critical Services
mMTC	Massive Machine Type Communications-5G Generic Service
MONROE	Measuring Mobile Broadband Networks in Europe.
MCPTT	Mission Critical Push To Talk
NFV	Network Function Virtualisation
NaaS	Network as a Service
NFVI	Network Function Virtualisation Infrastructure
NFVO	NFV Orchestrator
NG-UI	Next Gen User Interface
NS	Network Service
NSD	Network Service Descriptor
OSM	Open Source MANO
OSS	Operations Support System
PLA	Placement Optimization Module
PoP	Point of Presence
PNF	Physical Network Functions
REST	Representational State Transfer
SDK	Service Development Kit
SDN	Software Defined Networking
SNMP	Simple Network Management Protocol
SSH	Secure SHell
TCP	Transmission Control Protocol
UI	User Interface
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
VDU	Virtual Deployment Unit
VIM	Virtual Infrastructure Manager
VL	Virtual Link

Acronym	Meaning
VNF	Virtual Network Function
VNFD	Virtual Network Function Descriptor
WIM	WAN Infrastructure Manager
YAML	YAML Ain't Markup Language

Executive Summary

The aim of this document is to provide a final description of the design and implementation of the Management and Orchestration (MANO) building block of the 5GENESIS facility.

This document starts with a summary of the Release A of the MANO framework [1], i.e. the midterm version delivered on 5GENESIS project month 15 (M15) reported in Deliverable 3.1. The work has been evolving a lot from that implementation, several new features have been adopted during the project development, facilitated by discussions held in the regular work packages teleconferences and in project general assembly meetings.

During the second Release of the MANO building block, an updated architecture is described, including the components it is made of, that depicts all the internal interfaces and communication channels with the rest of the components of the 5GENESIS facility. Additionally, this document provides an exhaustive description of how this architecture has been implemented to fulfil the requirements of the deployments of the different 5GENESIS facilities, as well as all the specifications and workflow designed for the developed functionalities of the component.

Finally, the contribution to the open source community is described, with the implementation of the NS repository under the latest release of Open Source MANO (OSM) under version 8, from the 5GENESIS project.

This deliverable serves as the final version of the deliverable D3.1 Management and orchestration (Release A) [1] at M15, reporting the final implementation for the MANO enablers.

Table of Contents

1. INTRODUCTION

This deliverable presents the work done under the scope of Task 3.1 within Work Package (WP) 3, focusing on the Management and Orchestration (MANO) building block of the 5GENESIS platforms.

The outcome of that work is the description of a set of components, located in the MANO Layer, that are common to all the 5GENESIS platforms: Athens, Berlin, Limassol, Malaga and Surrey. Even if the implementation and the selection of the needed subcomponents is left open, depending on the technologies used in each platform, the input, output, and behaviour are common to all deployments.

The document is organized as follows:

- Section 1 (current) describes the scope of the document.
- Section 2 provides an overview of the current release, and a summary of what was provided during the first project cycle in Release A.
- Section 3 is the main part of this document and describes the updated MANO architecture and the specific requirements for 5GENESIS. In this section, the definition and workflow of the different features provided are described, as well as the manual describing all those features. Moreover, the contribution of 5GENESIS in the OSM framework is also presented.
- Section 4 presents the OpenTAP plugin developed for Network Service management in the experimentation framework.
- Finally, section 5 summarizes the work done for the MANO-related activities.

This deliverable focuses on the MANO layer, a component common to all the 5GENESIS platforms, which is instrumental to (i) have a common interface of the MANO technologies, abstracted, with a platform configuration and exposed though the use of Application Programmable Interface (APIs) to interact with them; (ii) create a common approach to the configuration and usage of the components that conform the MANO layer; (iii) offer a methodology for the repository operation; and finally (iv) describe the contribution to the open-source community from the project outputs.

Finally, this deliverable is useful to get a better understanding of what the MANO does and how it can facilitate the work for the automation of the Experiment deployments.

2. RELEASES INTRODUCTION

The implementation of the MANO component has followed the implementation roadmap of the other 5GENESIS components. This section reports the status of the building block at the moment of the Release A, delivered at M15 of the project. The main objective of the MANO module is to enable the management and operation of the NS services, to be provided as part of the experimentation validation of the slices in the infrastructure in 5G. This document presents the summary of the previous version of the Release A as well as the description and design of the final modules of the MANO component in the 5GENESIS architecture.

2.1. Release A summary

During the first project cycle, an early implementation was developed to fulfil the first evaluation of the platforms. Release A focused on defining a common architecture, interaction and interfaces with the neighbour components. The figure below depicts the defined architecture delivered for Release A with the first version of the component, in which minor modifications were made, the contribution to the release 8 of OSM was included, and the NS Repository was implemented.

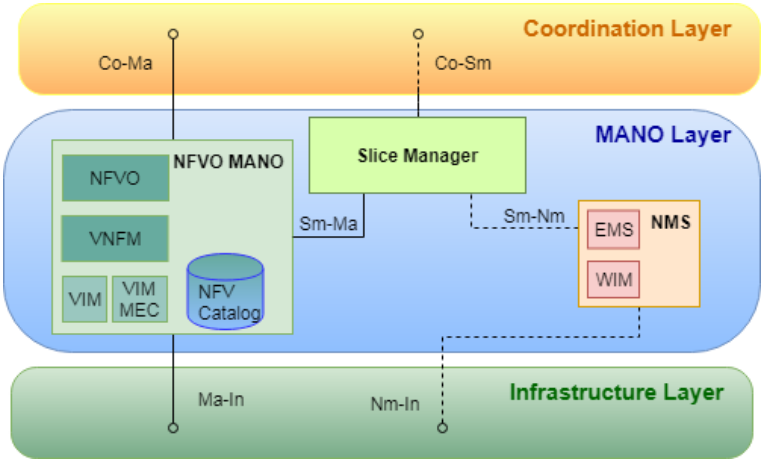

Figure 1 – Release A architecture of the MANO layer

The MANO building block has been upgraded and updated for Release B, in order to make the module compliant with the advance functionalities and security in the 5GENESIS facility. The main functionalities provided in the Release A that have been further developed in Release B are:

- Design and definition of the general architecture and interfaces between components.
- Implementation of Create/Read/Update/Delete (CRUD) operations for the management of NS, abstracting from the infrastructure in the testbed.
- Initial deployment and validation of MANO features.

A more detailed description of the design and implementation of the modules and enhanced features is presented along the different sections in the document.

3. MANO SOLUTION IN 5GENESIS

3.1. Introduction

The MANO component is a key part of the 5GENESIS facility that coordinates network resources of the virtualised functions and services, managing the lifecycle of multiple virtual networks on top of a common infrastructure.

The selected technology for the management and virtualization of the 5GENESIS facility was identified in the previous deliverable D3.1 'Management and orchestration (Release A)' [1], which, starting from an analysis of the state of the art, selected the open source project OSM [2] to deliver the orchestration framework aligned with the ETSI NFV Information models [22]. Moreover, the work of the 5GENESIS project using the framework resulted in inputs to the open community, e.g., in new features identified during the execution of the project, that are described in the following sections.

We also rely on one of the most used Virtual Infrastructure Manager (VIM) OpenStack to control the pool of compute, storage and networking resources to create service chains and deliver the network services to the 5GENESIS Experimenter. In the Edge computing platform to demonstrate the advantages of Multi-Access Edge Computing (MEC) capabilities, OpenNebula [3] was selected to manage the heterogeneous infrastructure, providing a more lightweight performance and scalability of process. All these components have been abstracted by the Dispatcher component, developed in the OPEN5GENESIS framework.

3.2. Architecture update

The entry point of the MANO building block is through the MANO Wrapper that manages all the requests in the coordinator layer and performs the needed operations before the NS can be deployed in the infrastructure. The following picture depicts the MANO components and the interaction with the several layers in the infrastructure.

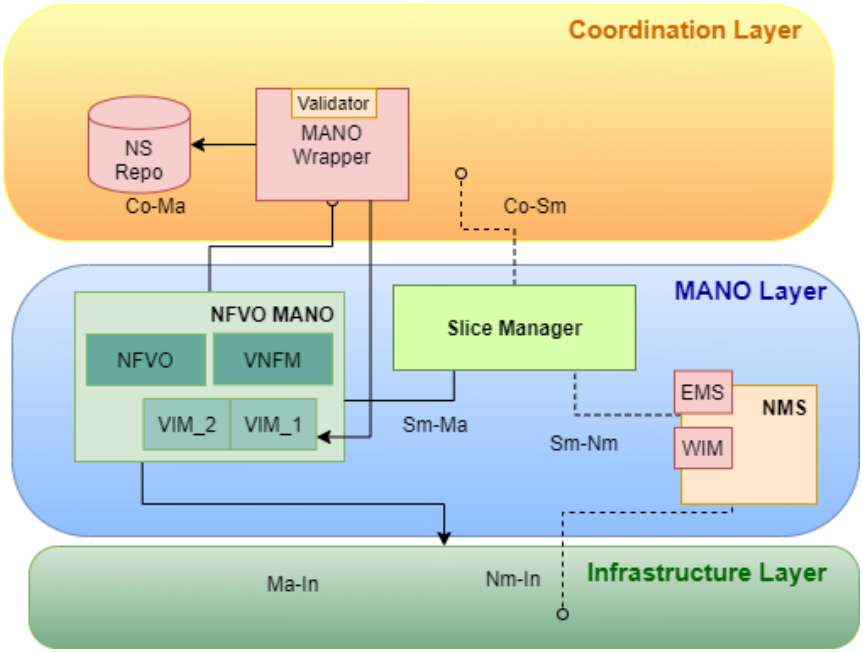

Figure 2 – MANO Layer architecture

3.3. Management Operations

With the introduction of Network Functions Virtualization (NFV) and Software-Defined Networking (SDN), service provider environments have become more dynamic and complex. The use of NFV by the network operators applied to the virtualisation of their physical appliances not only adds dynamicity to their networks, reducing the dependency of a physical location and dedicated hardware, but also reduces costs.

The switch from the manual process that traditionally implied the creation and deployment of services in an end-to-end environment to a fully automated one is possible thanks to the orchestrators, enabling the automation of the provisioning and coordination and management of physical and virtual resources; not only within the datacentre but also across the network infrastructure, composed of multiple technology layers and domains like cloud, metro, access, and core networks in fast and efficient manner. At the same time, the time taken to spin up a new network component is reduced from moths to a matter of hours or even minutes.

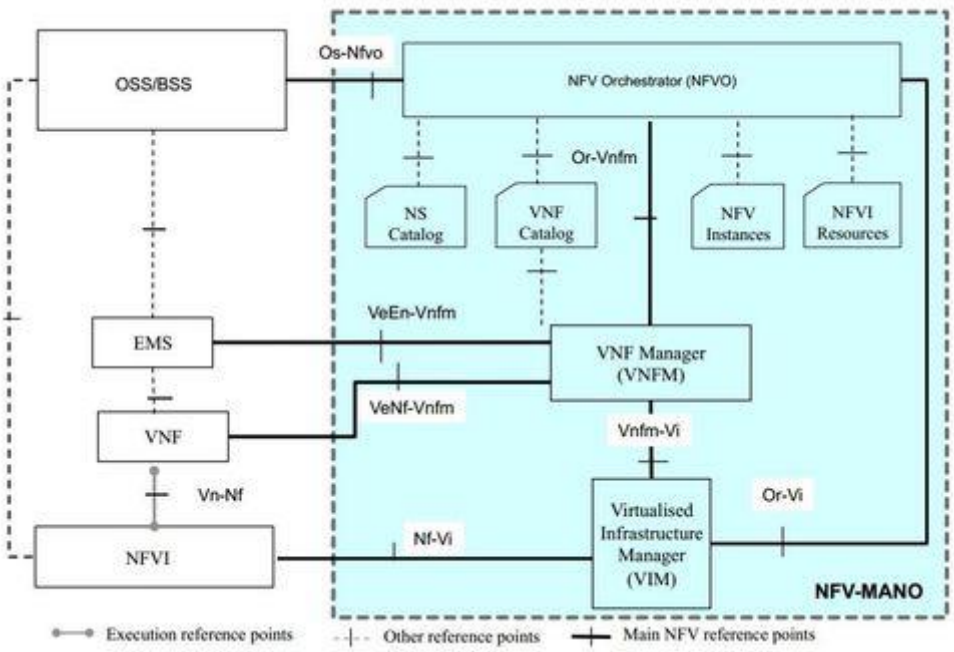

Figure 3 – MANO architecture [4]

The NFV MANO layer [Figure 3] is based on templates for the standard VNFs (descriptors). Users can request them based upon existing catalogues and choose from existing NFVI resources to deploy their platform or element.

3.3.1. OSM Release 8

According to the OSM release notes [5], OSM Release 8 brings a set of new features allowing to improve the orchestration of diverse virtualization environments, including Physical Network Functions (PNFs), a number of different VIMs for VNFs, and Kubernetes for Kubernetes-based Network Functions (KNFs). Among these new features the most notable ones are:

- The new VNF monitoring functionality, based on Prometheus [23] exporters, provides the flexibility of using traditional Simple Network Management Protocol (SNMP) as well as the ability to adopt newer technologies.
- An improved architecture, allowing to communicate with a Highly Available visualized clustering approach (VCA) cluster, which makes the whole system more robust against unexpected failures.
- Proxy charms on Kubernetes allow now Juju to manage the lifecycle of KNFs in a similar way to VNFs and improve the performance of VNF deployments.
- A brand new NextGen-UI offers an evolved user interface, based on the well-adopted Angular framework, that brings a fresh look to Release 8.
- The new Placement Optimization Module (PLA) adds for the first time in OSM an optimization engine, able to decide where to deploy the xNFs (VNFs or KNFs), based on technical criteria. This is an important aspect that the OSM project wants to progressively evolve in subsequent releases.
- Manageability has been improved, by implementing per tenant quotas in OSM (in addition to the quotas already provided by the VIM), and better integration with Operations Support System/Bussiness Support System (OSS/BSS) through a subscription mechanism.

- Support to a wider range of SDN Controllers has been added, as the options for deploying a data centre fabric have significantly increased in recent years, and OSM remains able to leverage the different options using the SDN Assist capability.
- VNF Repositories facilitate the distribution of VNFs by using a simple mechanism that allows users to find and download VNFs from vendors' repositories, thus conforming to the OSM guidelines.

3.3.2. Network Services specifications

The instantiation of services in the 5GENESIS facility relies on virtualized Network Services. Those services are specified in different descriptors that define the required specification and information modelling that a NS instance can adopt for the required automation of vertical services.

In 5GENESIS we have relied on the defined reference NFV architectural framework of the descriptors to manage and operate the services. The selected orchestrator, OSM [6] [7], stick to the NS definition of the ETSI framework, therefore the followed approach allows to describe our services and to save the work of having to adapt them to the 5GENESIS NFV orchestrator (NFVOs), providing full functionalities.

Network Services (NS) are composed by at least one VNF, which is defined by a set of images plus descriptor file that describe how the VIM should deploy and interconnect those images.

It might occur that not all parts of a NS are virtualised. Our models support this latter case, allowing the possibility of defining PNFs, i.e., functions that are physical appliances and cannot be virtualised, or even hybrid ones, e.g., Hybrid Network Functions (HNF).

Within the scope of T3.1, we have provided support for the virtualisation of the Mission Critical Push to Talk (MCPTT) and the Mission Critical Services (MCS). following specific steps when creating a NS, according to the OSM guide [8]:

- Decide what network function to deploy.
- Define the required specifications.
- Prepare a Descriptor to:
    - Generate default skeletons.
    - Modify default descriptors.
    - Validate descriptors.
    - Create the package for each descriptor.
- Onboard packages.
- Instantiate the NS Descriptors (NSD).

According to the specifications, first we need to create a basic diagram similar to the example in Figure 4, depicting how our VNF will look like. This will help us later to have a clear view of the aspects we need to define in the descriptor: interfaces, internal networks, visual display unit (VDUs), etc.

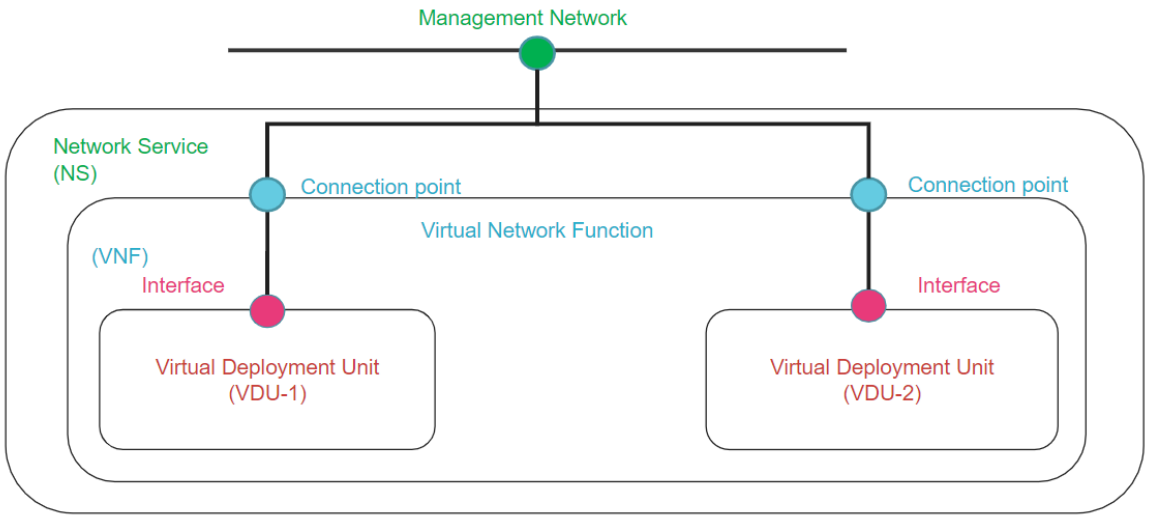

Figure 4 – Basic NS diagram

After that, we lean on basic VNF Descriptor (VNFD)s already created [9] to start describing our VNF by modifying an existing descriptor written in YAML to ease this task [Table 1].

Table 1 - Basic VNFD example

```
vnfd:
-   id: hackfest_basic-vnf
    name: hackfest_basic-vnf
    short-name: hackfest_basic-vnf
    version: '1.0'
    description: A basic VNF descriptor w/ one VDU
    logo: osm.png
    connection-point:
    -   name: vnf-cp0
    mgmt-interface:
        cp: vnf-cp0
    vdu:
    -   id: hackfest_basic-VM
        name: hackfest_basic-VM
        image: ubuntu16.04
        alternative-images:
        -   vim-type: aws
            image: ubuntu/images/hvm-ssd/ubuntu-artful-17.10-
amd64-server-20180509
        count: 1
        vm-flavor:
            vcpu-count: 1
            memory-mb: 1024
            storage-gb: 10
        interface:
        -   name: vdu-eth0
            type: EXTERNAL
            virtual-interface:
```

```
                type: PARAVIRT
            external-connection-point-ref: vnf-cp0
```

The last step will be to encapsulate all the VNFs within one NS, building our NSD. For that, we use again a basic model of NSD and adapt it to the specific needs.

Table 2 - Basic NSD example

```
nsd:
-   id: hackfest_basic-ns
    name: hackfest_basic-ns
    short-name: hackfest_basic-ns
    description: Simple NS with a single VNF and a single VL
    version: '1.0'
    logo: osm.png
    constituent-vnfd:
    -   vnfd-id-ref: hackfest_basic-vnf
        member-vnf-index: '1'
    vld:
    -   id: mgmtnet
        name: mgmtnet
        short-name: mgmtnet
        type: ELAN
        mgmt-network: true
        vnfd-connection-point-ref:
        -   vnfd-id-ref: hackfest_basic-vnf
            member-vnf-index-ref: '1'
            vnfd-connection-point-ref: vnf-cp0
```

MCPTT

The management of CRUD operation in the NS for the MCPTT use case, is simplified as all the service has been encapsulated as microservices within one single image for the VNFD (VNF_5GENESIS_NEMERGENT_generic) with specific resources request:

- memory-mb: 4096 MB
- storage-gb: 40 GB
- vcpu-count: 2 cores

The result is the VNFD described in Table 3.

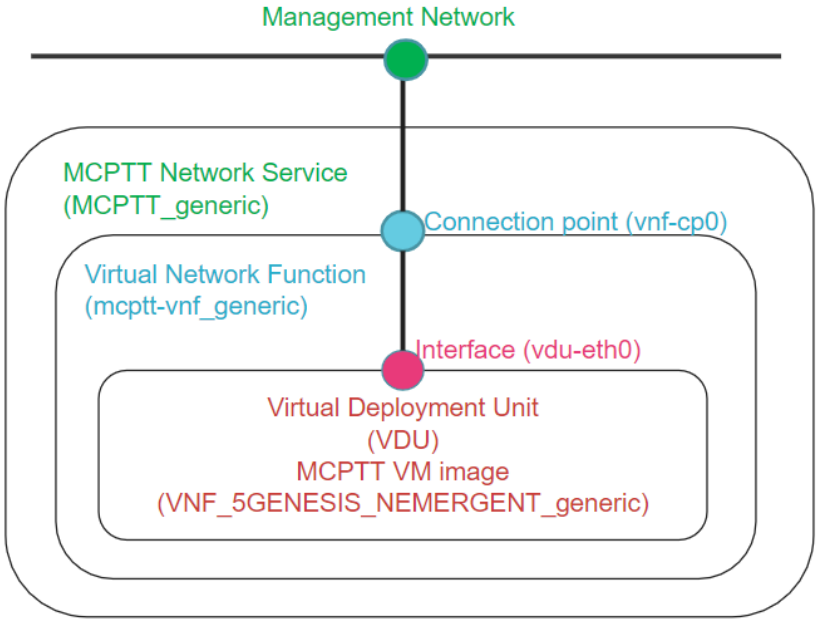

Figure 5 - MCPTT diagram

For the creation of the NSD, it is necessary to reference the previous VNFD by its ID (mcptt-vnf_generic) and connect the VNF connection points to the NS networks (vnf-cp0 → mgmtnet), as shown in Table 4.

Table 3 - MCPTT VNFD

```
vnfd:
- connection-point:
- name: vnf-cp0
  type: VPORT
description: Nemergent's MCPTT
id: mcptt-vnf_generic
mgmt-interface:
  cp: vnf-cp0
name: mcptt-vnf_generic
short-name: mcptt-vnf_generic
vdu:
- count: 1
  id: mcptt-VM
  image: VNF_5GENESIS_NEMERGENT_generic
  interface:
  - external-connection-point-ref: vnf-cp0
    name: vdu-eth0
    type: EXTERNAL
    virtual-interface:
      type: PARAVIRT
  name: mcptt-VM
  vm-flavor:
    memory-mb: 4096
```

```
    storage-gb: 40
    vcpu-count: 2
vendor: Nemergent
version: '1.0'
```

Table 4 - MCPTT NSD

```
nsd:
- constituent-vnfd:
- member-vnf-index: '1'
  vnfd-id-ref: mcptt-vnf_generic
description: MCPTT
id: MCPTT_generic
name: MCPTT_generic
short-name: MCPTT_generic
vendor: Nemergent
version: '1.0'
vld:
- id: mgmtnet
  mgmt-network: true
  name: mgmtnet
  short-name: mgmtnet
  type: ELAN
  vim-network-name: external_network_33
  vnfd-connection-point-ref:
  - member-vnf-index-ref: '1'
    vnfd-connection-point-ref: vnf-cp0
    vnfd-id-ref: mcptt-vnf_generic
```

3.3.3. MANO Wrapper

The *MANO Wrapper* is the component that interacts with the platform NFVO and VIMs to realise the functions assigned to those in a transparent way, that is, agnostic to the kind of NFVO or VIM used, simplifying the task to the external user. The Wrapper also bypasses the security inherent to the MANO components and filters the communication, exposing only features that are necessary for the user to run the system but not all of them, avoiding giving details of the underlying infrastructure to the user, who should not be aware of sensitive information but must use the functionalities of the applications.

The latest version of the MANO Wrapper includes also the management of the internal VNFD/NSD repository, that validates the packages and checks the consistency of the database before storing them.

The 5GENESIS NSD and VNFD repository exposes an API to handle the NSD and VNFD packages. This API has been integrated with the MANO wrapper component to keep backward compatibility with the previous version. In this way, all the repository processes are handled

internally, in a manner totally transparent to the user, who might think that packages are being onboarded directly in the NFVO, as happened in the previous version.

In this module we can find the following features:

- Index NS in the Repository (OSM 8 Compliant):
  - VNF.
  - NS.
- List all the VNFs and NSs available in the repository.
- Onboard in the NFVO the required artefacts of a NS.
- Delete a NS:
  - From the index.
  - From the NFVO catalogue.
- List all the VIMs available.
- Upload images to the VIM (OpenStack and OpenNebula).
- List all the available images.
- Provide logs of the MANO component.

The MANO Wrapper module requires the Auth component [10] to authorise and authenticate each request directed to itself. This external component is necessary for respecting the privacy of the artefacts, which might have a restrictive license over the artefact and cannot be used or viewed by all users. This also means that all requests described below must include user credentials.

In the following subsections we describe the different operations that are enabled trough the Wrapper. All of them depict the involved roles in the facility and the workflow describing the interaction between the modules comprised in the operation.

Index VNFs

The VNF is represented by a VNFD plus a set of image files. Once the images are up (process described below), we need to index the VNF package in the platform repository. The VNF artefact must go through a syntax validation and dependency check before being indexed in the repository to ensure its consistency.

In the request, the user must specify not only the VNFD package but also the visibility it will be registered with. The VNF could be either public or private, the latter meaning that it is visible only to the user that registered it. In the repository, the package is stored and mapped together with the user owner information to verify the NS access in the repository, this will be verified also by the Auth component.

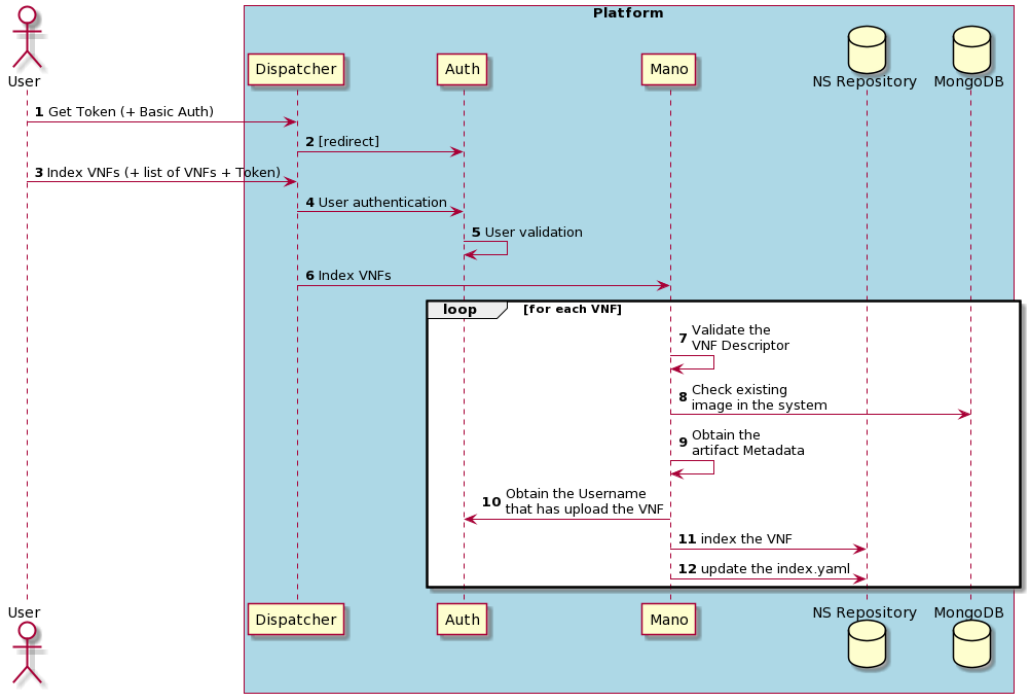

Figure 6 – Index VNF flow

Through the following request of the Open APIs, the MANO Wrapper Enabler will index the VNFD in the repository:

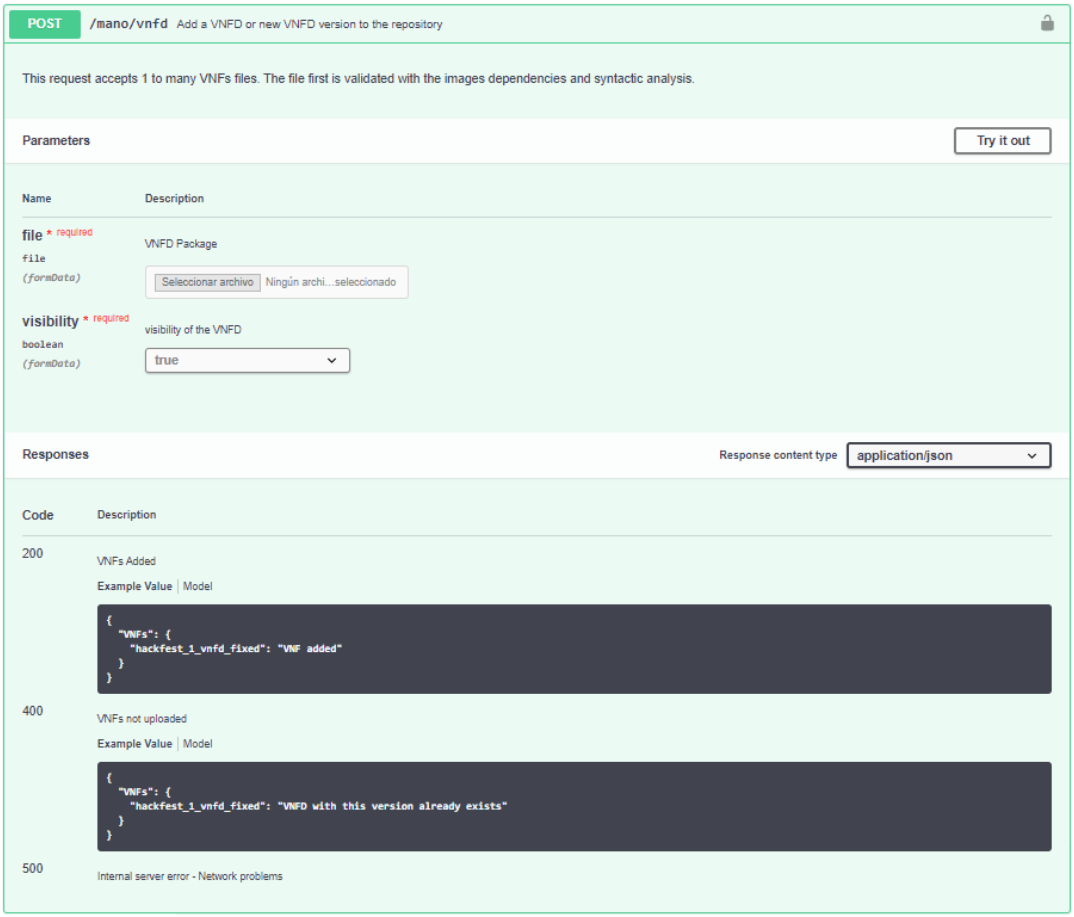

Figure 7 – Index VNFD

Index NS

A NS is represented by an NSD. Before indexing an NSD, the VNFDs referenced in it must be already registered to satisfy the dependency requirements. The NS artefact must go through a syntax validation and dependency check before being indexed in the repository to ensure its consistency.

In the request, the user must specify not only the NSD package, but also the visibility that will be registered with. The NS could be public or private, that similar to the VNFD is visible only to the user that registered it.

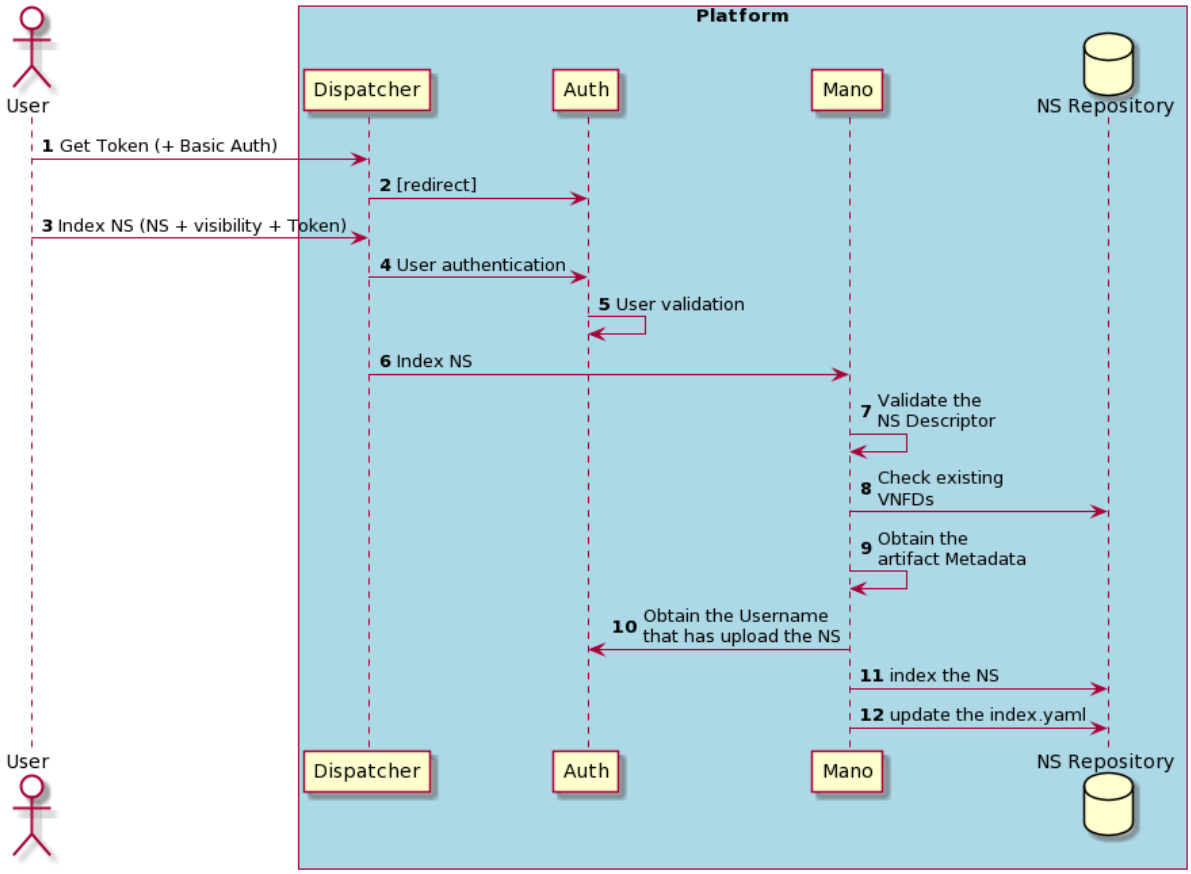

Figure 8 – Index NS flow

Through the following request of the Open APIs, the MANO Wrapper Enabler will index the NSD in the repository:

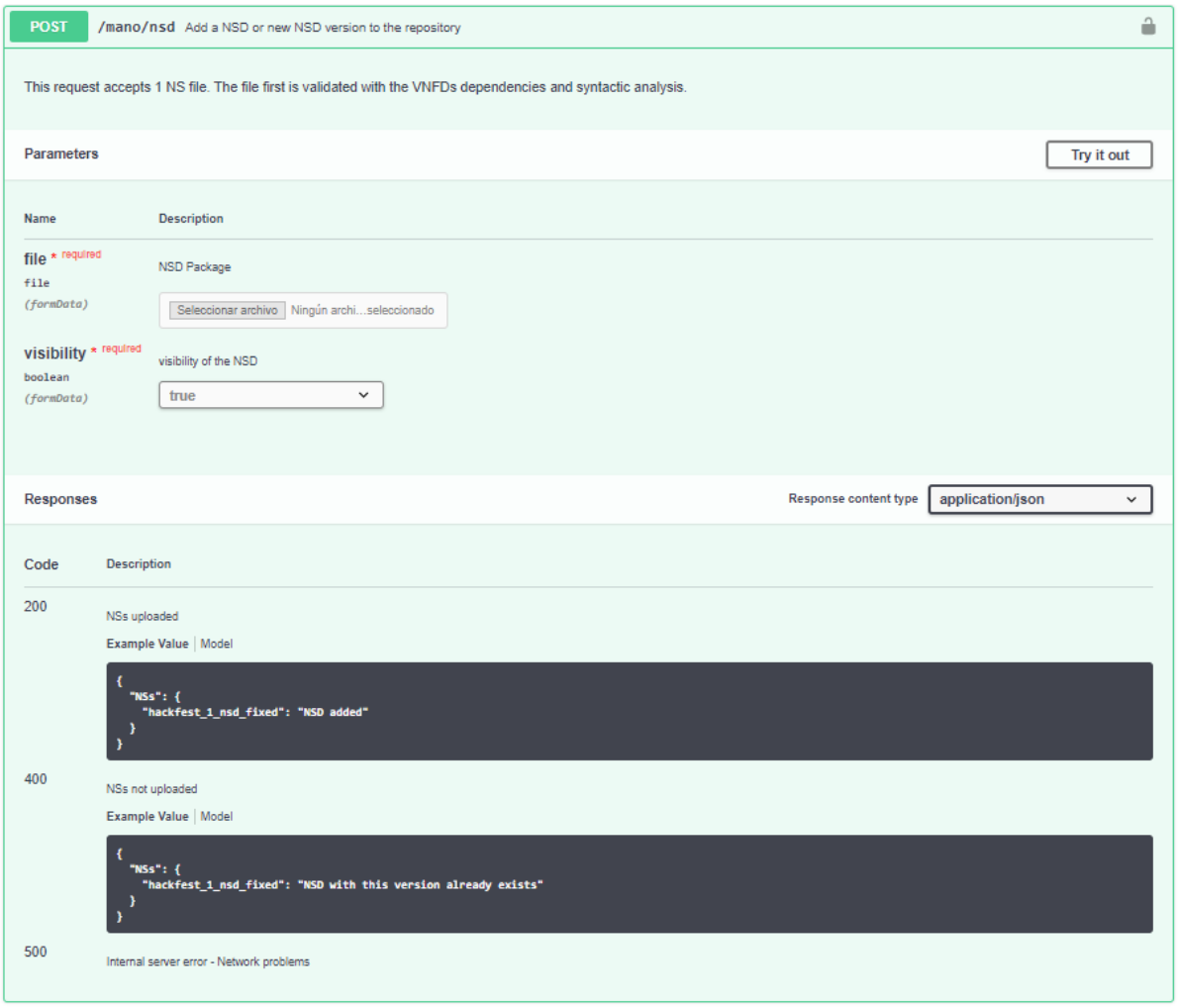

Figure 9 – Index NSD

The request will answer with the confirmation of the index NSD in the repository.

Flow for VNF/NS indexing process

The MANO microservice is in charge of validating all the artefacts for the NS onboarding (Images, VNFDs, and NSDs). The following flow (Figure 10) explains how this component analyses the current available resources and validate the new ingestion of each artefact. It also describes all the process of indexing a NS, taking into account the artefacts' dependencies for each component in a Network Service: (V/H/P)NFs and images.

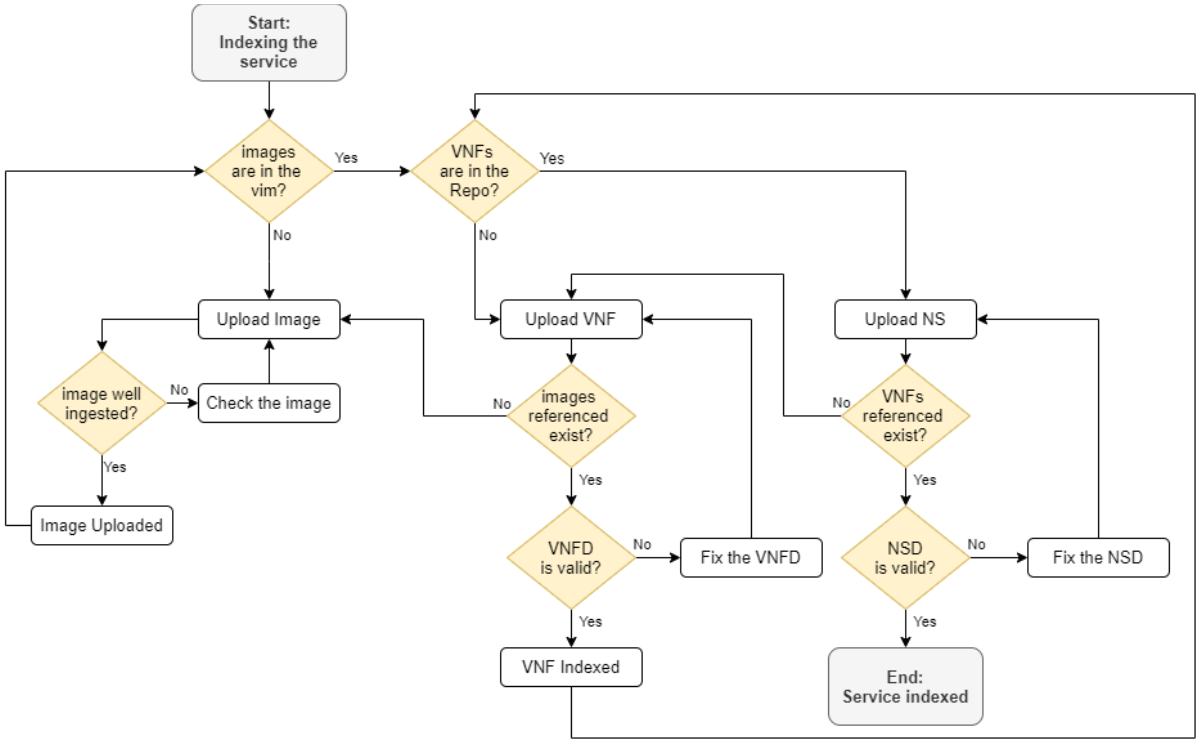

Figure 10 - Complete flow for indexing network services

List VNFs

List the VNFs stored in the repository. The list will contain the public VNFDs and the private ones that were uploaded by the same user. The user can be obtained from the basic auth or from the token (by asking the username of this token internally to the Auth Module).

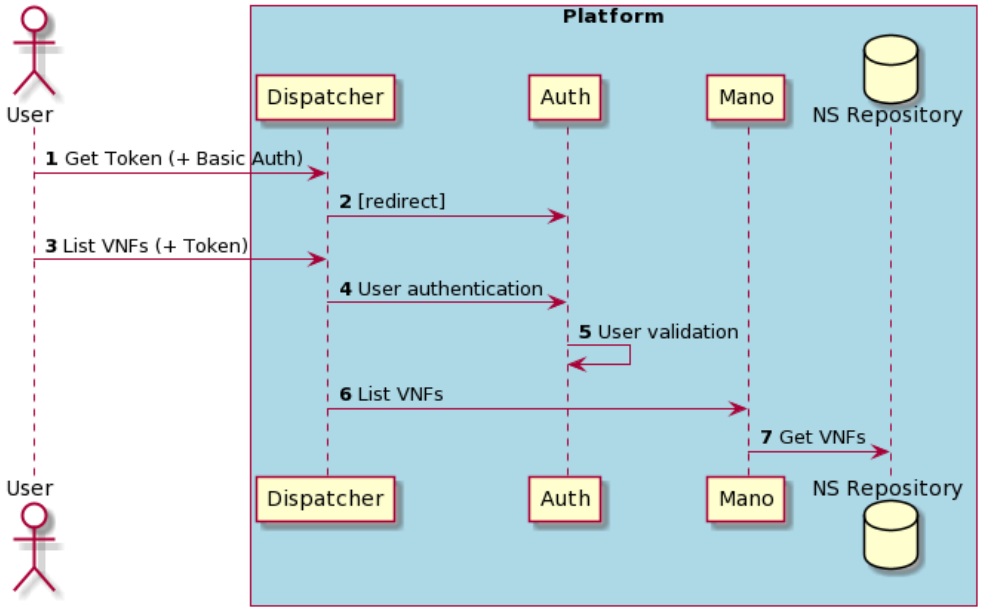

Figure 11 – List VNF flow

Through the following request, the MANO Wrapper lists the VNFDs in the repository. It is possible also to apply the verbose field in the request to retrieve more information about the VNFs packages that are indexed in the repository, e.g., its description or version.

Only the available VNFs for the user, both public and private ones, will be retrieved.

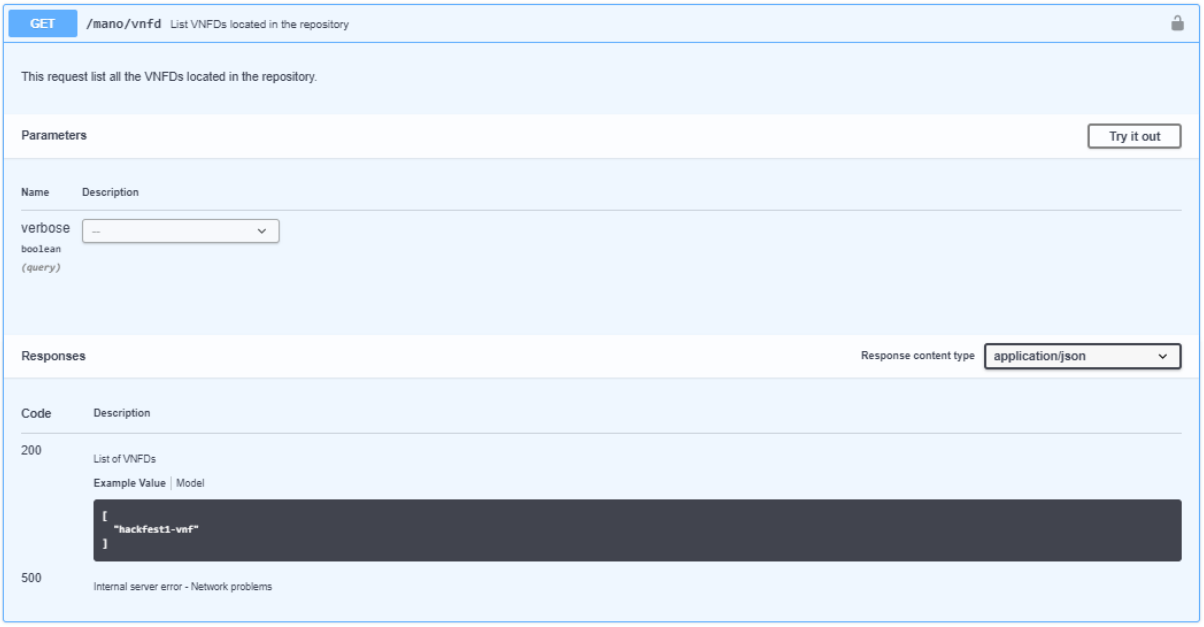

Figure 12: Get VNFs interface

The request returns the list of indexed VNFDs in the repository.

List NSs

List the NSs stored in the repository. The list contains the public NSDs and the private ones that were uploaded by the same user. The user can be obtained from the basic auth or from the token (by asking the username of this token internally to the Auth Module). These NS will be used in the creation of the Experiment Descriptor, identified by its ID.

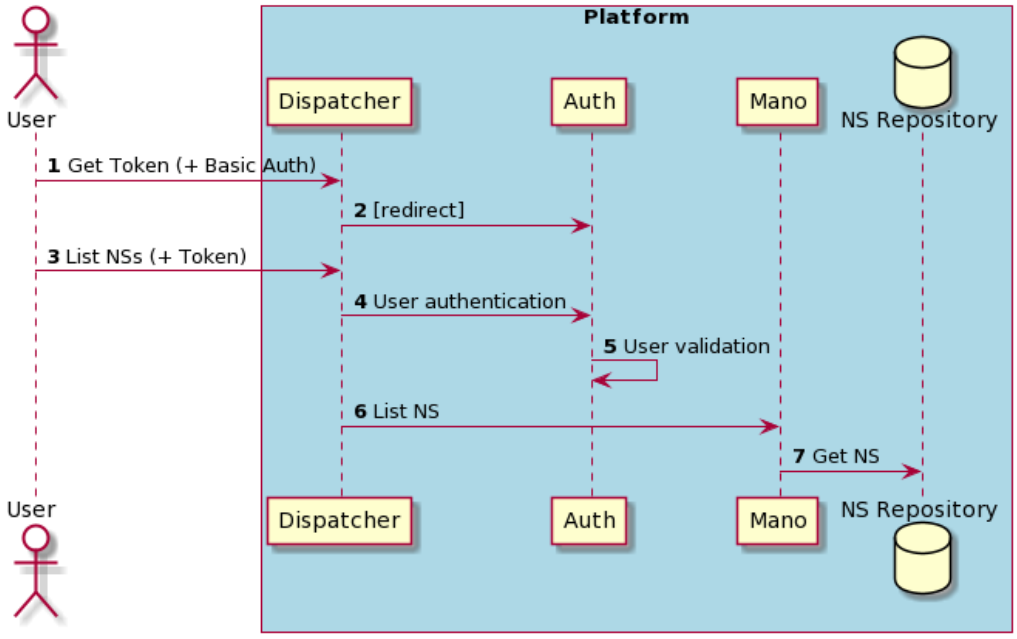

Figure 13 – List NS flow

Through the following request, the MANO Wrapper lists the NSDs in the repository. It is possible also to apply the verbose field in the request to retrieve more information about the NSs packages that are indexed in the repository, e.g., its description or version.

Only the available VNFs for the user will be retrieved, both the public and the private ones.

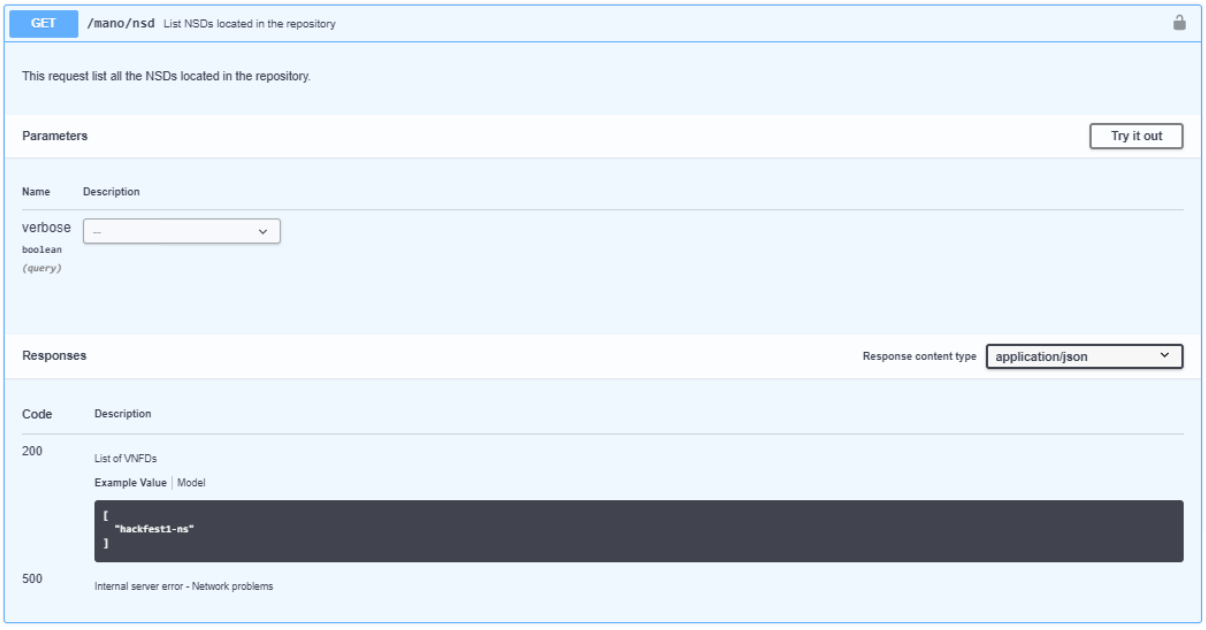

Figure 14: Get NSs interface

The request returns the list of indexed NSDs in the repository.

Onboard NS

The onboarding process consists in uploading all the packages required from the repository to the NFVO. The packages to be onboarded are the requested NS package with all the VNFs dependencies. The onboarding process is expected to be requested internally before an experiment is executed. However, this request is also available for generic purposes such us testing NS performance without executing a complete experiment.

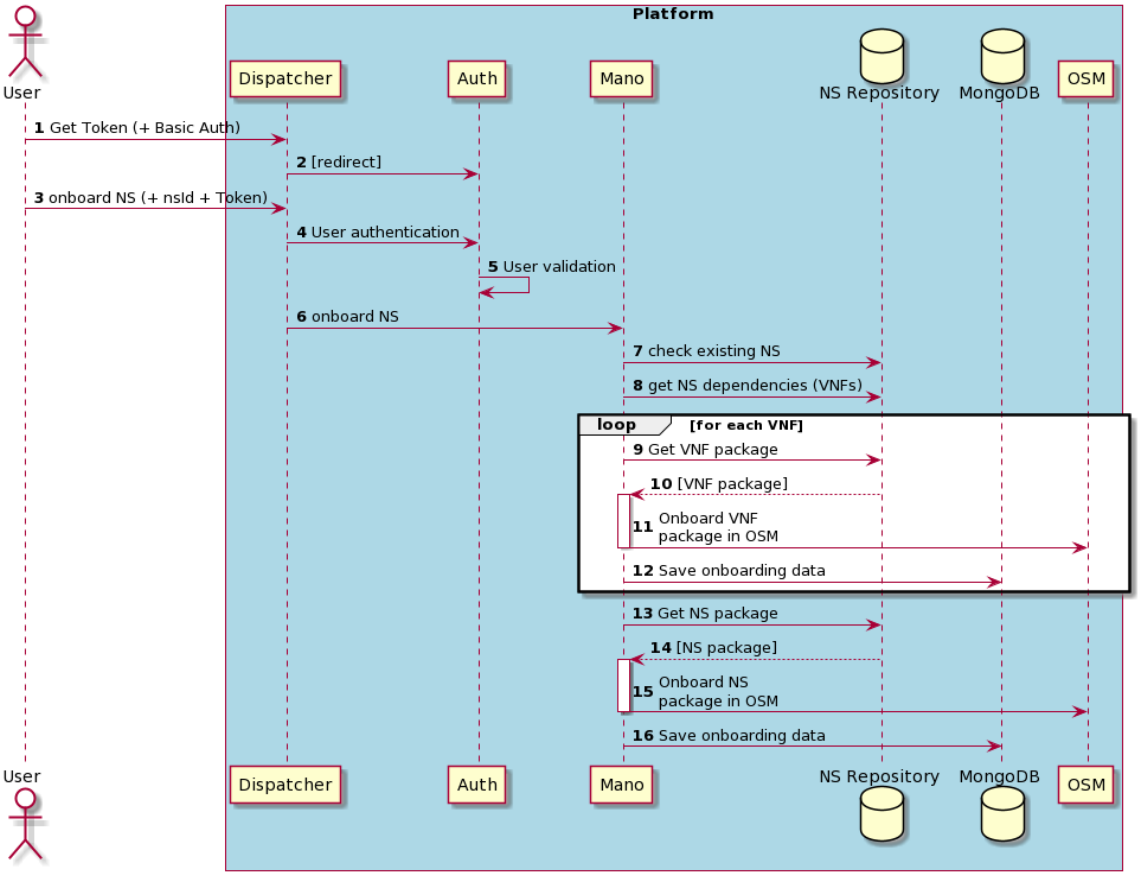

Figure 15 – Onboard NS flow

Through the following request of the Open APIs, the MANO Wrapper will onboard the NS in the OSM orchestrator:

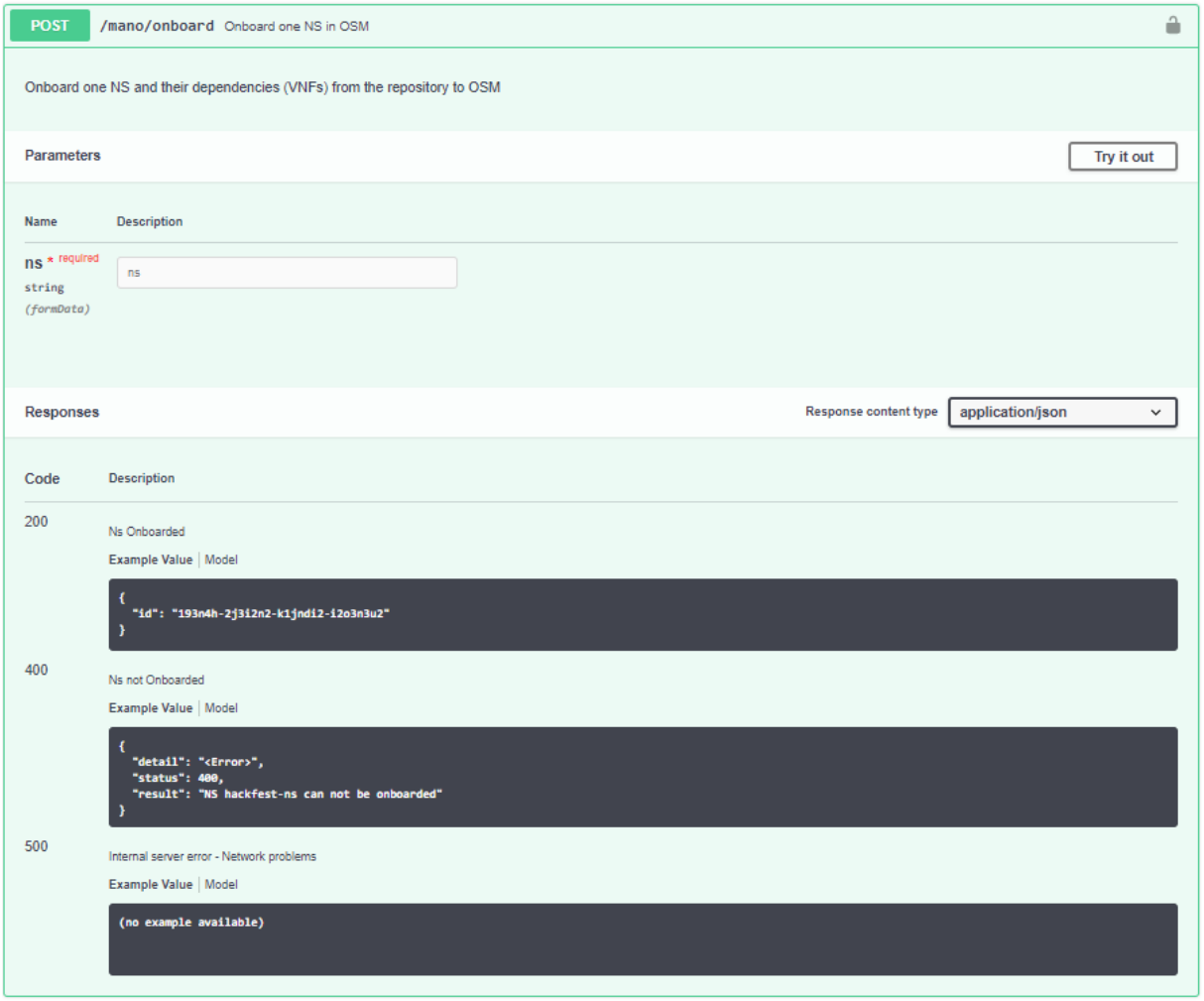

Figure 16 – NS Onboard

A successful request returns the onboarding ID generated by the NFVO.

Delete NS

If a NS is not required anymore in the platform, the owner can delete it both from the repository and in the NFVO, if it was already onboarded.

When deleting the NS, this function also deletes recursively all the orphan VNFDs stored in the repository, that is, all the VNFDs that were used by the deleted NS and are not used by any other NS.

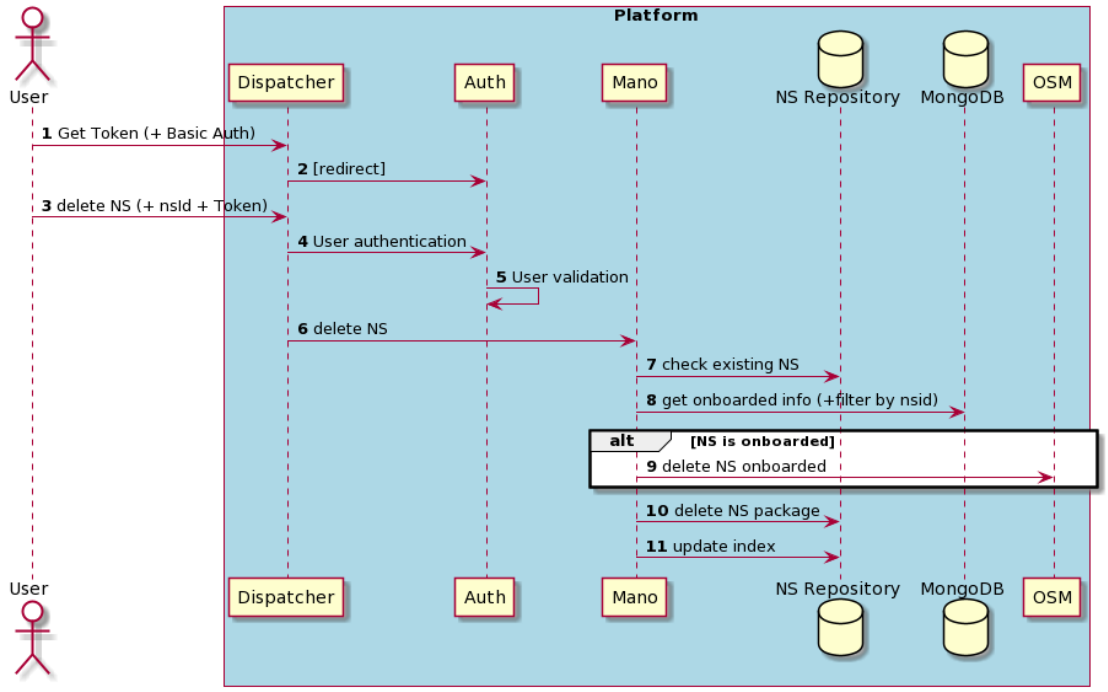

Figure 17 – Delete NS flow

Through the following request of the Open APIs, the MANO Wrapper Enabler delete NS from the repository:

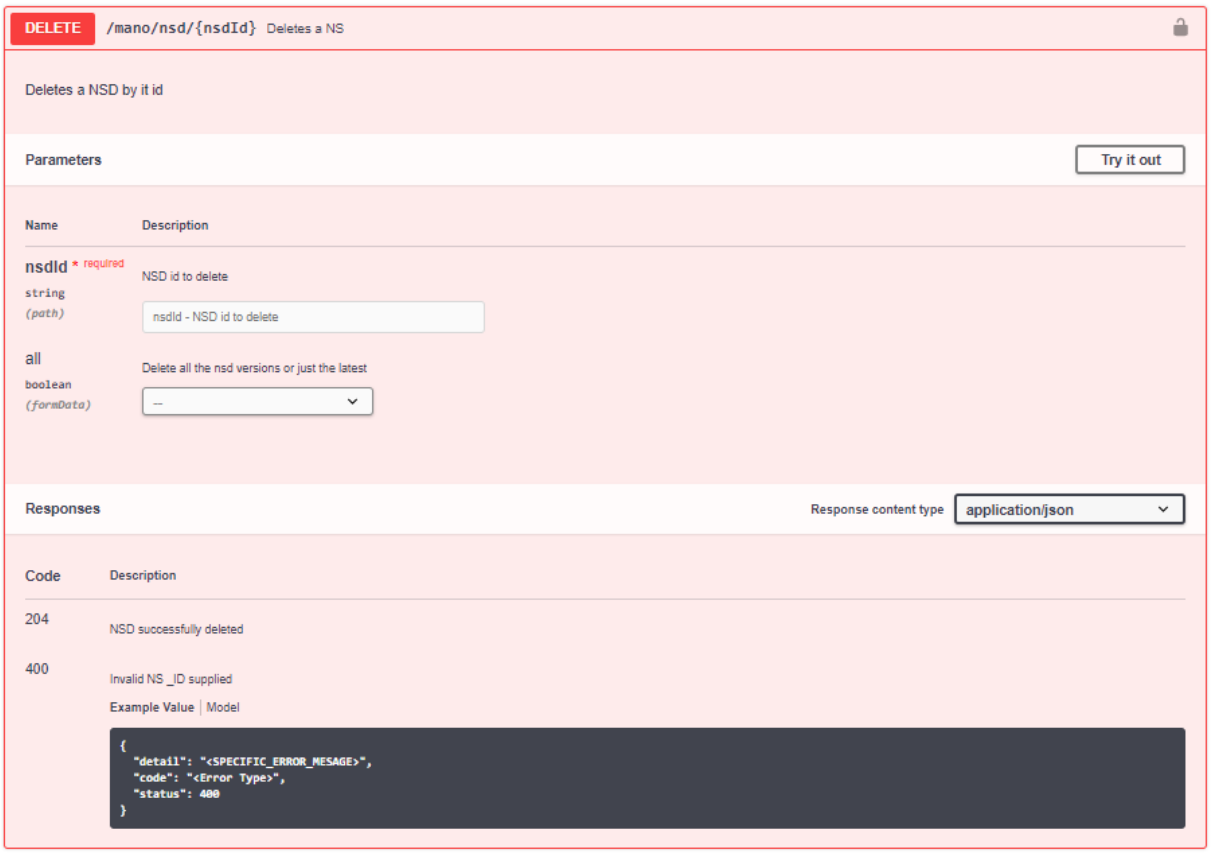

Figure 18 –NS deletion

List VIMs

List the VIMs available for deploying NSs in a specific location. This information is used to have the Point of presence (PoP) available in the infrastructure, that is used to define the location to upload the image files required by each VDU of the VNFs. To provide this list of available VIMs, the information must be previously registered, during the installation implying the configuration of the available VIMs.

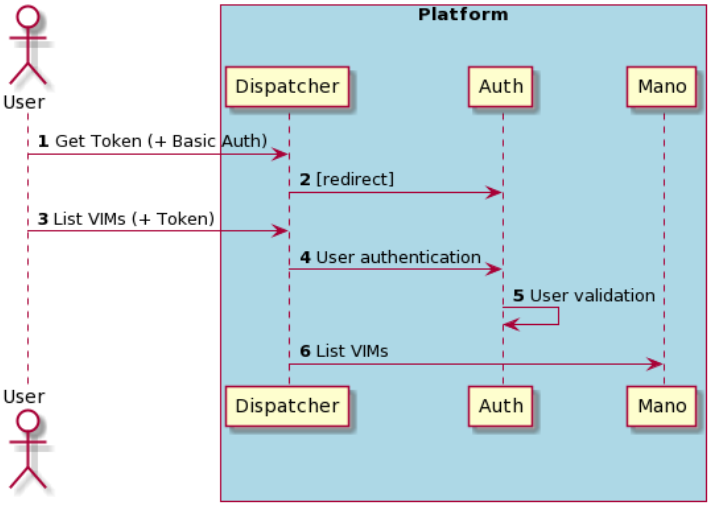

Figure 19 – List VIM flow

Through the following request to the MANO Wrapper, the list of the registered VIMs in the infrastructure is obtained:

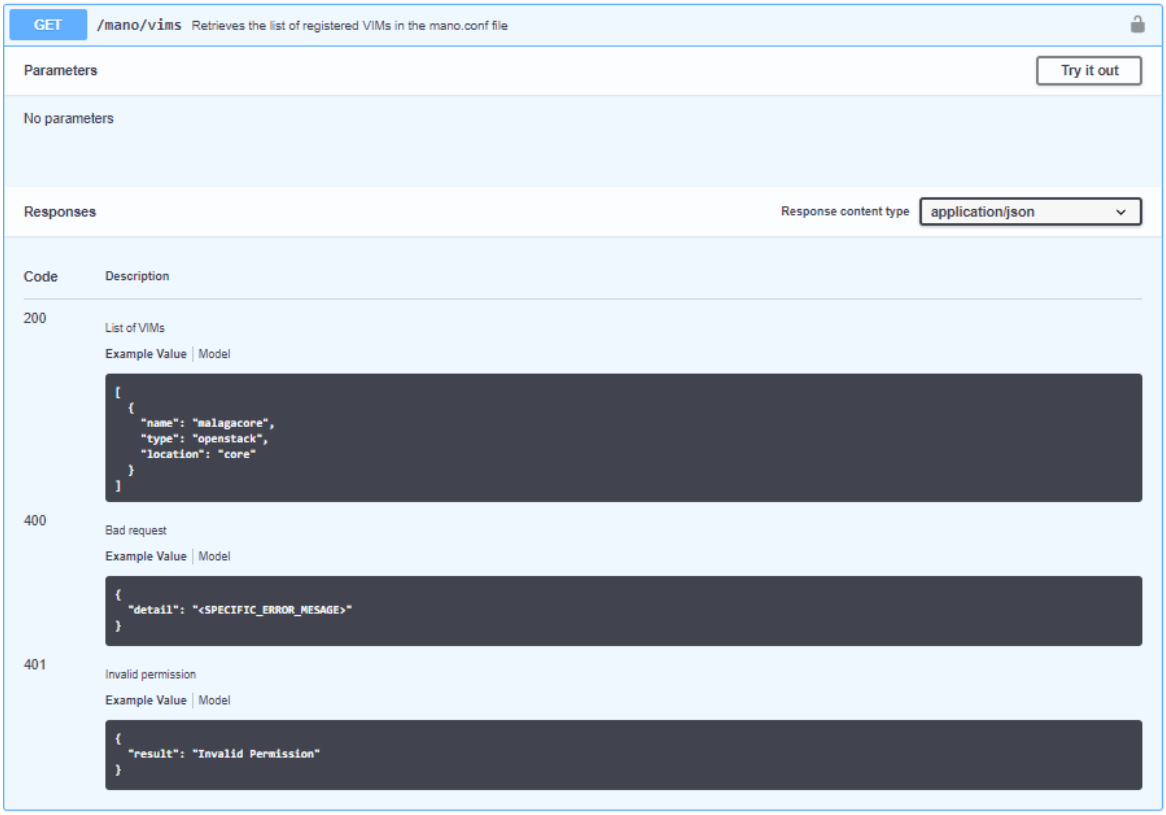

Figure 20 – VIM listing

A successful request returns the list of registered VIMs, including name, type and location.

Upload an image

The images required during the instantiation of the VNFs need to be uploaded prior to the uploading of the VNFD to avoid inconsistencies. Each image will be uploaded to a certain VIM, which has to be specified in the request. During this process, the images are validated with their file extension and checksum. The list of available VIMs can be obtained from the previous endpoint (List VIMs). More information of the uploading process is explained below in section 3.3.5

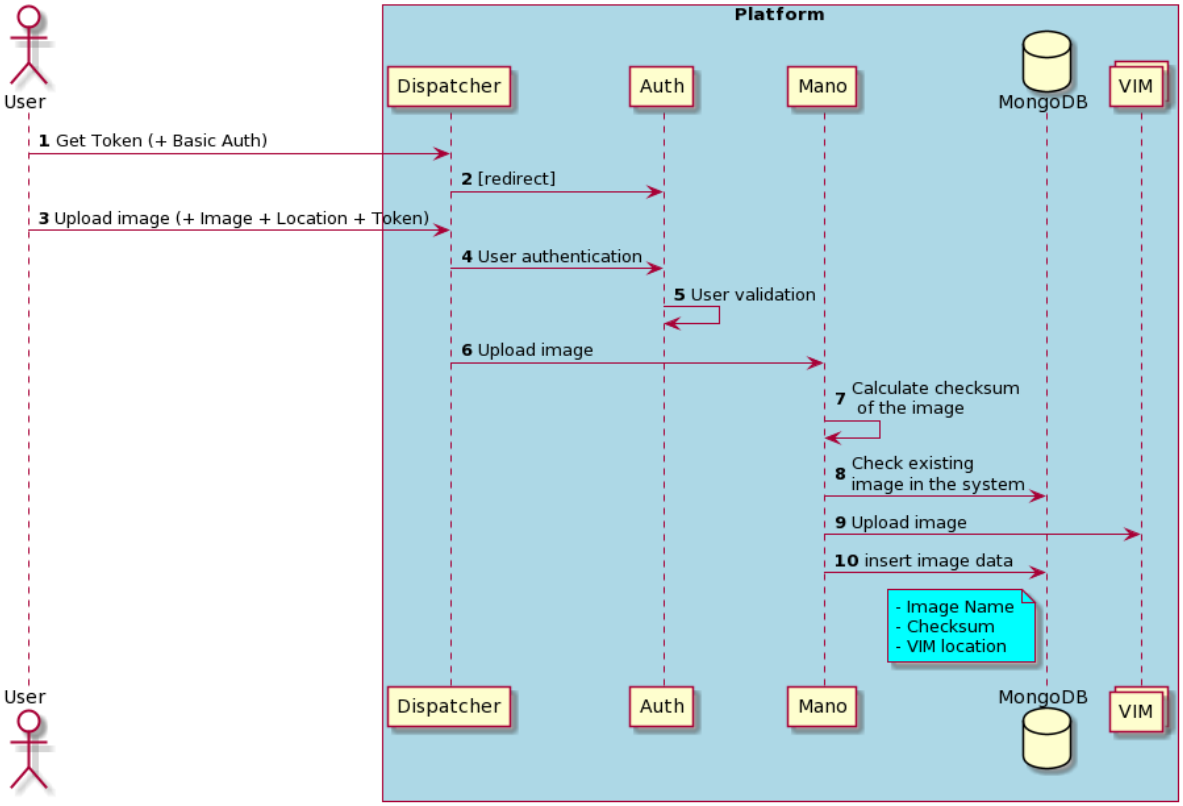

Figure 21 – Upload image flow

The Platform Admin can optionally use this endpoint to register manually an image that was already uploaded in a VIM. This alternative has been included because the process of uploading an image is very delicate: we are dealing with huge files and any cut during the uploading process (which is sensitively long), might break it, causing it to start over again.

Through the following request to the MANO Wrapper, the image to the selected VIM is uploaded:

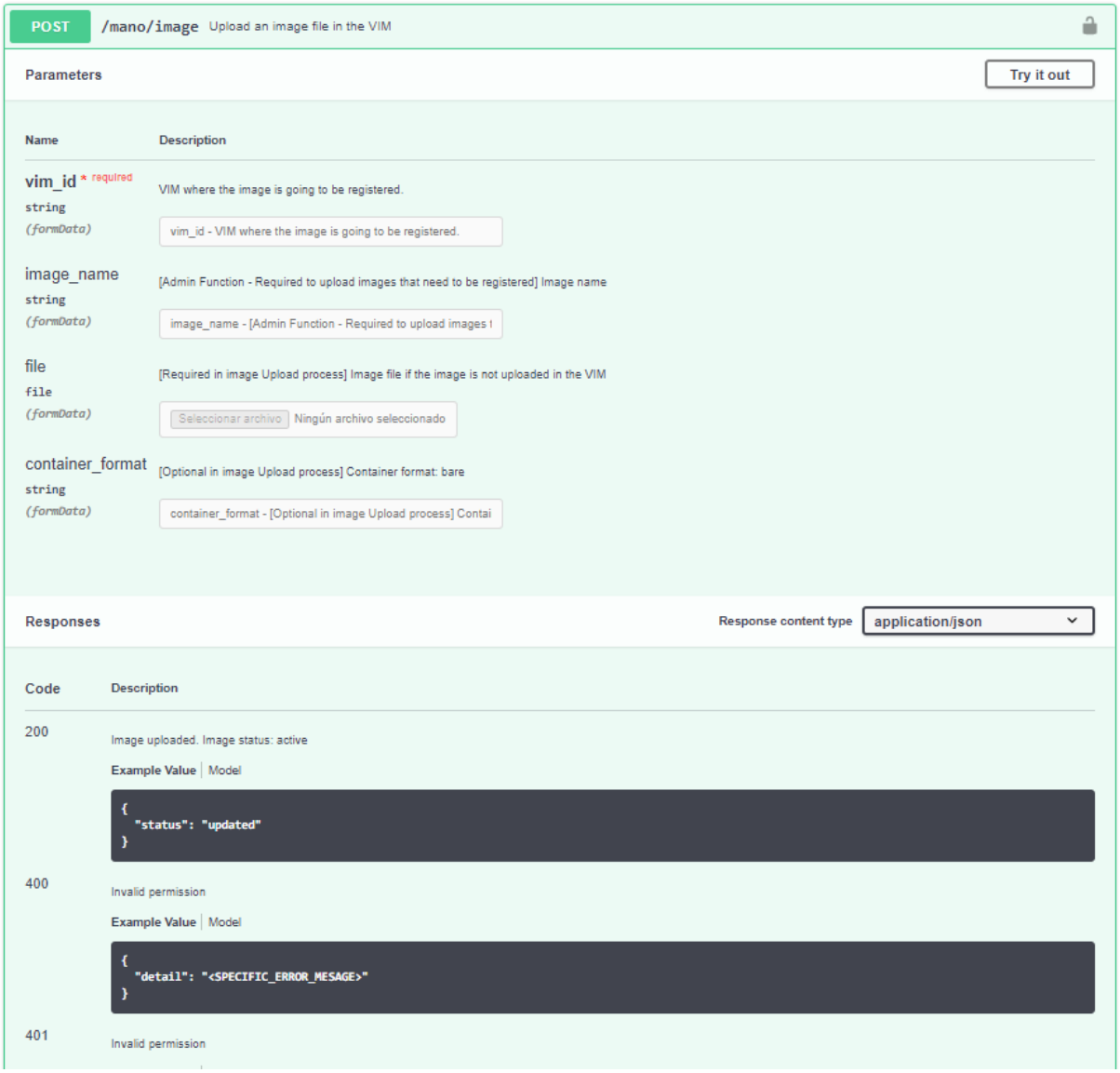

Figure 22 – VIM upload

The request responses with the confirmation that the image was correctly uploaded in the selected VIM.

List images

Lists the images uploaded in the different platform VIMs through the MANO Wrapper. The list comes from the local database registry, since the VIMs might not be dedicated only to 5GENESIS, but also shared with other applications.

Through the following request to the MANO Wrapper, we get the list of the registered images uploaded in the platform VIMs:

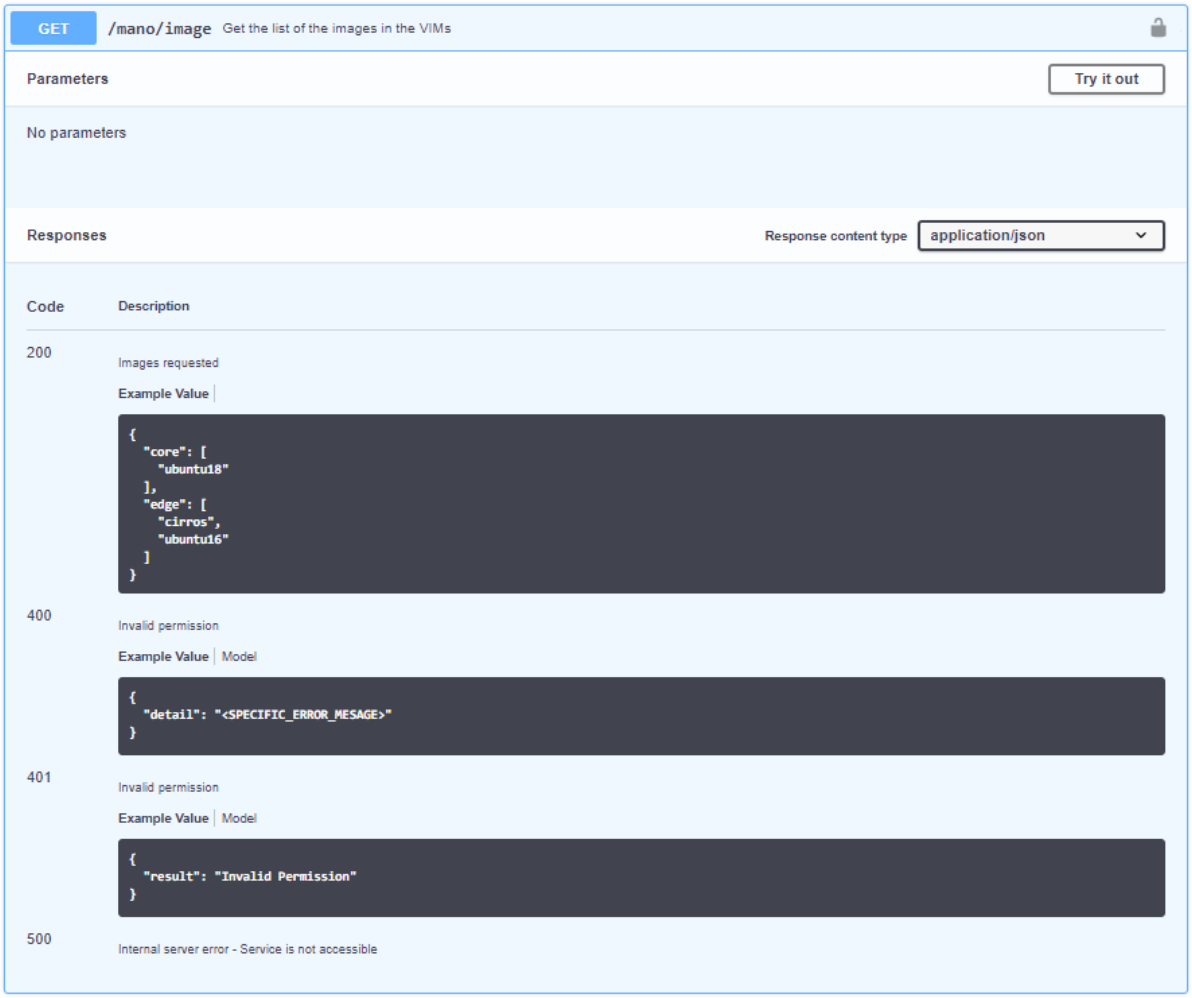

Figure 23 – List of the registered images

A successful request returns the list of registered images in the various platform VIMs, classified by their location.

Logs

Application logs are available in the application directory as 'mano.log'. The logs are also visible under the docker logs commands.

The structure of the logs is the following: UTC Datetime – Component Name – Severity Message. Some examples:

```
2020-11-13 17:26:42,678 -MANO API- INFO Validating VNFD
hackfest_cloudinit_vnf.tar.gz
2020-11-13 17:27:12,145 -MANO API- INFO Validating NSD file
2020-11-13 17:27:12,155 -MANO API- INFO Unpacking file for validation
2020-11-13 17:27:12,224 -MANO API- INFO Deleting temporary files
2020-11-13 17:29:21,939 -MANO API- INFO Retrieving available VNFDs
```

3.3.4. NS Validator

The validator inside the Mano Wrapper is in charge of validating the VNF and NS packages. Those packages are usually specified as VFND or NSD, with D standing for "Descriptor", the file that composes the packages and external dependencies. This descriptor, which is inside the packages, is the main file, as it describes the behaviour of the packages with their technical specification (networks, dependencies…). This descriptor is a YALM file that is analysed during the packages indexing. In order to validate its content and syntax, the Mano wrapper converts its content from yaml data structure to a JSON data structure.

With the JSON model structure, it is possible to validate the descriptor using a JSON schema, that specifies the structure that the descriptor must follow, the syntax, the required fields in the structure and the possible values that some fields must follow.

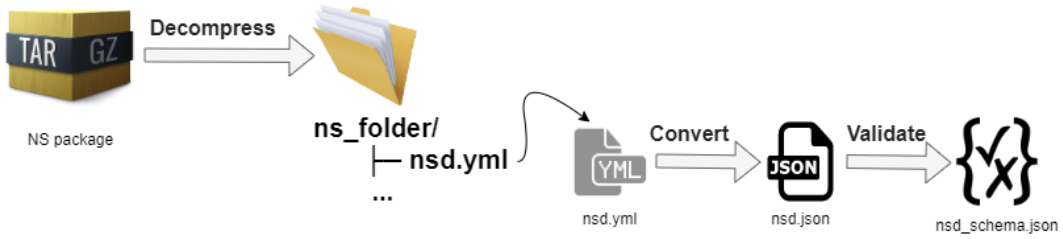

Figure 24: NS validation flow

There are two different JSON schemas, one for NSD [6] and other for VNFD [7]. Those JSON schemas have been inspired in the OSM specification[1]. Each schema has a different structure and critical fields that must exist in the descriptor, and some fields have to follow a simple logic or match with a regular expression.

For instance, both schemas require the following fields to index the packages in the repository:

- Package id.
- Name.
- Version.
- Vendor.

If a descriptor misses any of those fields, the validation will not be successful, and the file will be not indexed in the repository because it has not the required information to be properly managed by the NS repository.

After passing the JSON schema, the dependencies are checked. The VNF packages dependencies are the VIM images, referenced in the descriptor, which must be onboarded in the VIM through the dispatcher before trying to index the VNF. The NS packages dependencies are the VNF packages, referenced in the NS descriptor, which must be indexed before trying to index the NS package. Those dependencies are visible at the *metadata.yaml* in each NS artefact and their checking, if the dependencies are available, is a critical operation in the indexing process.

[1] OSM NS Packages specification: http://osm-download.etsi.org/ftp/osm-doc

3.3.5. NS Repository

The previous version of the NS catalogue relied on the internal OSM NS repository to manage the NS catalogue through the MANO Wrapper. This approach revealed several issues we had to deal with:

- If the NFVO instance is changed, the catalogue is lost or requires extra dedication to back it up.
- Multiple versions of the descriptors were uploaded during the development phase without any version control.
- The NFVO catalogue cannot be shared nor used by other platforms.
- The NFVO catalogue offers NS independently if the Experimenters wants to keep their services in private mode.

To solve these issues, we have decoupled the repository functionality from the NFVO and created our own NS/VNF repository within the T3.1 activity, which contains the full catalogue of available services with version control, regardless the underlying NFVO.

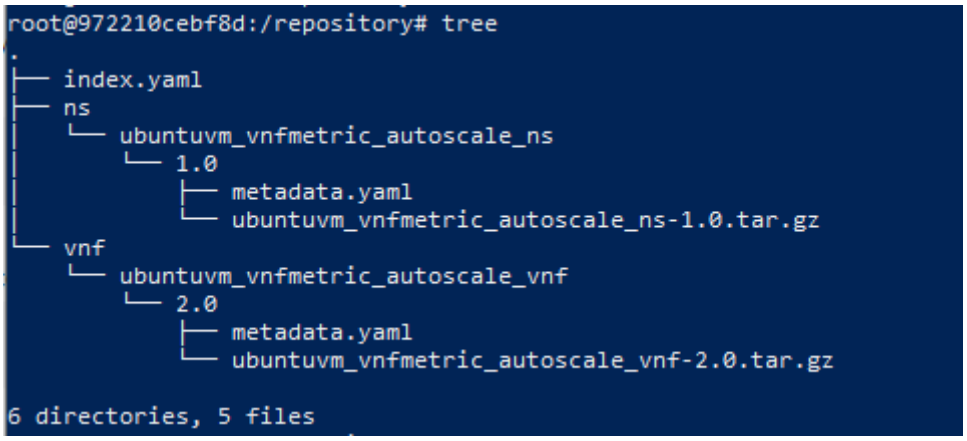

Figure 25 - Web view of the repository

Each package that is stored in the repository is validated and registered in the *index.yaml* file (Figure 26), which contains the minimum necessary set of information of each artefact, e.g., package checksum, name, description, version, and vendor.

The repository is structured as follows [Figure 25]:

- 'vnf' directory that hosts the VNFs along with extended metadata for each package.
- 'ns' directory for hosting the NSs along with extended metadata for each package.
- Index file (*index.yaml*) that keeps an index of the packages stored in the repository.

```
root@972210cebf8d:/repository# cat index.yaml
apiVersion: v1
generated: '2020-05-21T11:56:56.058907Z'
ns_packages:
  ubuntuvm_vnfmetric_autoscale_ns:
    '1.0':
      description: Ubuntu VM with VNF metrics and load-based autoscaling NS
      name: ubuntuvm_vnfmetric_autoscale_ns
      path: /ns/ubuntuvm_vnfmetric_autoscale_ns/1.0/ubuntuvm_vnfmetric_autoscale_ns-1.0.tar.gz
      vendor: Whitestack
    latest: '1.0'
vnf_packages:
  ubuntuvm_vnfmetric_autoscale_vnf:
    '2.0':
      description: Ubuntu VM with VNF metrics and load-based autoscaling
      name: ubuntuvm_vnfmetric_autoscale_vnf
      path: /vnf/ubuntuvm_vnfmetric_autoscale_vnf/2.0/ubuntuvm_vnfmetric_autoscale_vnf-2.0.tar.gz
      vendor: Whitestack
    latest: '2.0'
```

Figure 26 – *Index.yaml* file view

For each package, extra metadata, like the user that has indexed the service, is also stored to help with the package validation, crossed dependencies and consistency of the repository. This information is kept in the *metadata.yaml* file (Figure 27), located at the same path as the package.

The specific fields in the *metadata.yaml* are the user, the visibility of the package and the vnfd-id-ref with all the required VNFs.

```
root@972210cebf8d:/repository# cat ns/ubuntuvm_vnfmetric_autoscale_ns/1.0/metadata.yaml
checksum: 87017a59dee7f60fcd7542056fae4123
description: Ubuntu VM with VNF metrics and load-based autoscaling NS
id: ubuntuvm_vnfmetric_autoscale_ns
name: ubuntuvm_vnfmetric_autoscale_ns
path: /ns/ubuntuvm_vnfmetric_autoscale_ns/1.0/ubuntuvm_vnfmetric_autoscale_ns-1.0.tar.gz
user: No valid Token given
vendor: Whitestack
version: '1.0'
visibility: true
vnfd-id-ref:
- ubuntuvm_vnfmetric_autoscale_vnf
```

Figure 27 - *metadata.yaml* example

Image uploading

In order to keep consistency with the NS repository, there is the need to keep records of the available images for the NS deployment.

The image uploading is one of the functionalities that are available to the user as part of the Repository/MANO specification. To allow this functionality, the manager of the platform must configure the VIMs that will be available for the deployment in the platform by configuring the cloud computing infrastructure software at the moment of the installation. The required information is described as follows:

- Name of the VIM.
- Type: OpenNebula/Openstack.

- Location: edge/core.
- VIM Auth URL.
- Access credentials.

For the image uploading, the URL and the credentials are needed but it is important to register the rest of the details to improve later the deployment of the NSs and to keep the consistency in the repository, avoiding broken dependencies.

Currently, Openstack and OpenNebula are supported - these VIMs are the only ones used in 5GENESIS.

The image uploading must be transparent to the user regardless the underlying type of VIM used in the platform or the number of them. To achieve that, we had to develop libraries generic enough to consider the basic image attributes required for each type of VIM.

The images are to be validated before uploading them to the VIM. This validation is based in the extension file, that must be *'.qcow2', '.img', '.iso', '.ova' or '.vhd'*, and the operation "md5 checksum" of the file to assure that the same image is not uploaded twice in the same VIM.

Plugin for OpenStack [11]

Within this task, we have had to develop tools for performing the generic uploading of images and the listing of images present in an Openstack VIM. In this case, we have used the API provided by the application, that already includes those functionalities, but simplifying them to adapt them to the 5GENESIS needs, removing the implicit security (but adding our own) and translating the parameters, creating a layer on top of the regular application API that allows us to interact with Openstack as if it was a generic VIM.

Plugin for OpenNebula [12]

The interaction with OpenNebula is more complex, as its API does not include among its functionalities the image uploading. Therefore, in order to be consistent with the features offered by the Repository/MANO Wrapper, those functionalities need to be implemented. To summarise the procedure, the algorithm steps are the following ones:

- Load and save the image file in a buffer.
- Transfer the file to the OpenNebula VM.
- Register the image in OpenNebula calculating previously the deployment size and assigning the specific parameters translated from the generic ones included in the original request.
- Delete temporary files.

3.4. Wrapper Installation

The following subsections detail the process of installing the MANO Wrapper from the 5GENESIS public repository [13], including requirements, explanations, configuration, commands, etc.

3.4.1. Requirements

The machine (virtualised of not) where the MANO Wrapper is to be installed needs some previous software installed locally or remote but accessible for installing and running the

5GENESIS MANO wrapper, e.g., the NVFO or the MongoDB services do not need to be installed on the same machine but have to be reachable by http request from it:

- Locally installed
  - docker version >= 18.09.6
  - docker-compose version >= 1.17.1
- Optionally could be external but reachable
  - Authorization Module
  - MongoDB
  - NFVO + VIM

Software packages

Mano wrapper is developed in Python 3.x and requires the previous installation of the following libraries and releases to satisfy dependencies in the code:

```
requests==2.20.1
Flask==1.0.2
gevent==1.4.0
pymongo==3.8.0
Flask-RESTful==0.3.7
PyYAML==5.1
Flask-Cors==3.0.8
openstacksdk==0.38.0
configobj==5.0.6
jsonschema==2.6.0
packaging==20.3
```

3.4.2. How to install and configure

The code of the MANO Wrapper is open and available in the public repository of the project [13] and is structured in the following format:

```
mano/                                   Main Folder
│
├─ libs/                                Folder for host libraries
│   ├── osm_nbi_util.py                 OSM library
│   └── openstack_util.py               OpenStack library
│
├─ schemas/                             Folder with json schemas for validating
│   ├── nsd_schema.json                 Json schema for validate NSD
│   └── vnfd_schema.json                Json schema for validate VNFD
│
├─ mano.conf                            Configuration file to be fill up
├─ mano.py                              Mano Server
├─ utils.py                             Util functions
├─ validator.py                         Validation functions
├─ requirements.py                      Python requirements
└─ swagger.json                         Open APIs specification
```

Going deeper in this structure we must identify a critical file that is required to configure the

module before its installation: the *mano.conf*. It has to be modified to adapt it to the testbed needs, specifying one NFVO and from none to several VIMs.

The *mano.conf* must follow the following structure:

Table 5 – MANO configuration file template

```
[NFVO]
TYPE=<type of NFVO. Currently only 'OSM' is supported>
IP=<IP address of your NFVO>
USER=<username with admin rights within the NFVO>
PASSWORD=<Password to the previous user>

[VIM]
[[vim-1-name]]
    TYPE=<type of VIM. Currently 'openstack' and ‘opennebula’ are supported>
    LOCATION=<Location of the VIM: [core|edge]>
    AUTH_URL=<VIM 1 auth URL>
    USER=<VIM username>
    PASSWORD=<VIM username password>
    PROJECT=<VIM project>
[[vim-2-name]]
    TYPE=<type of VIM. Currently 'openstack' and ‘opennebula’ are supported>
    LOCATION=<Location of the VIM: [core|edge]>
    AUTH_URL=<VIM 2 auth URL>
    USER=<VIM 2 username>
    PASSWORD=<VIM 2 username password>
    PROJECT=<VIM 2 project>
...
```

An example of *mano.conf* file following the previous structure is presented below:

Table 6 - *mano.conf* example

```
[NFVO]
TYPE=OSM
IP=192.168.33.100
USER=adminosm
PASSWORD=*****

[VIM]
[[malaga-core]]
    NAME=malaga-core
    TYPE=openstack
    LOCATION=core
    AUTH_URL=http://192.168.33.11:5000/v3
    USER=admin
    PASSWORD=***********
    PROJECT=admin
```

MANO Wrapper is very easy to install and deploy in a docker container. By default, the Docker will expose port 5001, but this can be modified by changing the port within the Dockerfile if necessary. When ready, simply use the Dockerfile to build the image:

```
cd mano
```

```
docker build -t mano .
```

This will create the *mano* image and pull in the necessary dependencies. Once done, run the Docker. For running the image and map the container port (5001) to whichever port you wish on your local host (5001 also in this case):

```
docker run -p 5001:5001 mano
```

The service will be exposed in port 5001.

3.4.3. Manual

Once the MANO wrapper container is running, the service will be available on the port mapped when running the container (5001 by default). The available features are listed in Figure 28 with a short description, the endpoint and the HTTP method. All of them are accessible via HTTP with the following URL format:

```
http://<IP address>:<port|5001>/<MANO endpoint>
```

The API interfaces are described in the swagger service exposed in 5002 port.

Figure 28 - MANO Wrapper Open APIs

Each request has been explained individually in detail in section 3.3.3.

3.5. OSM contribution

As described in D7.6 Standardisation and regulation [15] about the contribution to standards, ETSI Open Source MANO has been the main contributed OSS community from the 5GENESIS MANO perspective. The technical development has been produced in the framework of WP3 in Task 3.1 “Management and Orchestration”. In this deliverable, we provide the overview of

the contribution provided, the specific meetings that were attended, and the 5GENESIS components involved.

Moreover, 5GENESIS is part of the research and contributors' projects that closely collaborate with the OSM community and provide feedback from the experiments and results executed in the different platforms. The project has been recently included in the OSM website [14] as part of the research activities that are using or contributing to OSM.

The technical committee and the community of OSM is already working in the mid-release of the Release 8 with all the new functionalities. The final release is planned to be launched in mid-June 2020 to be tested over cloud-native applications for NFV environments. It foresees to bring major enhancements designed to deploy container Network Function workloads on Kubernetes.

In this new OSM release, two main improvements have been developed to overcome 5GENESIS challenges, developments that have contributed to the release in the OSM community. The main modules focused on the management and operation of the NSs located in the Coordinator layer of the 5GENESIS architecture. The features included the development of a NS external repository to improve the VNF loading and onboarding to the ecosystem and plugging involved in the validation process while onboarding the NS in the repository catalogue.

3.5.1. Validator

Atos developed a module that validates the syntax of the descriptors uploaded in the MANO wrapper. The package defines the creation and validation of the NS onboarded in the system. It evaluates the mandatory fields needed to execute the functions and the different parameters and required format of the files. The validation tool implements a plugin for pyang JSON schema output, that for a given YANG file, produces a JSON schema and validates the correct content payload as defined in the RFC7951 [24]. The tool generates the output validation of the file, taking into account the schema and the file provided.

The scope of this contribution has been also extended to the validation of the Experiment Descriptor (ED). The experimenter fills the ED template with the required parameters; once it is instantiated in the coordinator layer for the execution of the experiment, it is validated to assure the details information for the experiment accomplishment. This functionality will be enhanced with the recommendations and outputs results of the platform trials.

3.5.2. OSM repository

The OSM Repository is the distributed NS version control. This feature is the first approach for managing a NS repository available across domains. NS is the key element to enable the network provided as a service. ETSI NFV have a set of functions straddle across the infrastructure. The network functions (VNFs, PNFs, KNFs and Hybrids) with the NFV infrastructures enable the broad ecosystem of VNF vendors, the interaction in the E2E scenario managed by Network Service Orchestrator.

NSs are described as network functions that can be composed, to provide more complex Network as a Service (NaaS); several components describe the function as VNFD, NSD and

images. The descriptors are packages distributed in the tar.gz format that encapsulate all necessary information in addition to the descriptor file:

- A VNF is the package that defines the resources needed to allocate the VNF described in the VNFD. Each VNFD has at least one VDU (Virtual Deployment Unit). Each VDU is referred to a VIM image that the VIM should host in order to deploy there when a VNF is instanced.
- The NS package define, by its descriptor (NSD), how the set of VNFs that compose the service need to be interconnected using the Virtual Links (VL).
- The image is the artefact that enable the deployment of the descriptors in the VM that will provide the functionality of the service.

Based on this information, it is crucial to have a clear knowledge on how this information is stored and the mechanism to retrieve by OSM to onboard and instantiate the Service in the underlying infrastructure. Accordingly, and aiming at alleviating the consumption of the NS in a standardized way, a distribution mechanism and efficient packaging standard is introduced as a new feature to be published [16][17].

OSM Release 8 brings several improvements over the management and consumption of NSs and the interaction with the repository. Release 8 defines a standardized model format to implement a consumption mechanism for a remote NS repository, which could be queried and managed by OSM abstracting from the actual storage mechanism and the interface will be exposed by HTTP requests. The repository follows the same index reference model based on the VNF repo specification, the interaction of OSM and storage services synchronizes the full repository folder, where developers and service providers can onboard and test with several NSs [16][17].

This approach will allow other NS developers publishing and onboarding their services, pushing local artefacts and dependencies of the VNFs to the remote repository. The following figure depicts the interconnection between the OSM and the external standardized repository [16][17].

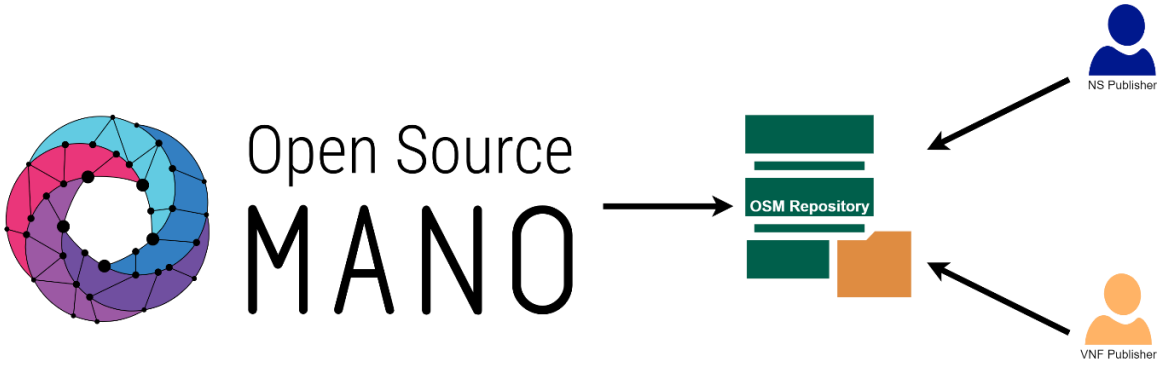

Figure 29: OSM Repository

The developed implementation requires the integration of the OSM client with an external OSM Repository. The repository model contains the artefacts of the VNF and NS that will be needed for the onboard process and the enquired of the catalogue. Those artefacts are indexing and managed by an index.yaml file in the root of the repository. The index has the basic information of each artefact, and its location.

The following format shows the baseline description to support the Repo Model defined with the required content information to manage the NS:

```
repository/
├── index.yaml
├── ns/
│   └── <ns_id>/
│       └── <ns_version>/
│           ├── metadata.yaml
│           └── <ns_id>-<ns_version>.tar.gz
└── vnf/
    └── <vnf_id>/
        └── <vnf_version>/
            ├── metadata.yaml
            └── <vnf_id>-<vnf_version>.tar.gz
```

As note, the text into **< >** are the identifiers for “ns” and “vnf” directories, that contains from 0 to N directories with the artefacts and versions of them.

The index file contains all the required information to be provided to the OSM about what is available in repo, regarding the VNFs and NSs artefacts; this means that it includes the array metadata and the dependencies of the artefact that is needed for the onboarding and instantiation of the service contained in the index repo.

```
apiVersion: v1
generated: '2020-03-26T13:34:44.828853Z'
ns_packages:
 hackfest1-ns:
  '1.0':
   description: Simple NS with a single VNF and a single VL
   name: hackfest1-ns
   path: ns/hackfest1-ns/1.0/ hackfest1-ns-1.0.tar.gz
   vendor: Atos
  latest: 1.0
vnf_packages:
 hackfest1-vnf:
  '1.0':
   description: A simple VNF descriptor w/ one VDU
   name: hackfest1-vnf
   path: vnf/hackfest1-vnf/1.0/ hackfest1-vnf-1.0.tar.gz
   vendor: Atos
  '2.0':
   description: A simple VNF descriptor w/ one VDU
   name: hackfest2-vnf
   path: vnf/hackfest2-ns/2.0/ hackfest2-ns-2.0.tar.gz
   vendor: Atos
  latest: 2.0
```

A more detailed description of the parameters is described as follows:

- packages: level of each package.
- name-vnf: level of each VNF/NS.
- version: “1.0” current version of VNF/NS.

- description: more detailed information of the package.
- name: naming.
- path: location of where the package is stored.
- vendor: network service provider developer.
- latest: the last uploaded version of the package.

The additional artefacts are included to represent the content of the components in the repo. Those are the software elements containing the require files and information for the specific component.

The repo considers two types of elements to be stored: VNF and NS. Every descriptor inside the artefact can have several versions of the instance, considering the latest version the default to be used while in other cases, the required version needs to be specified. Both elements have two subclasses: the metadata that contains more global significant information of the element and the package with compressed files in tar.gz format. An example of the yaml file is provided bellow for a NS metadata

```
checksum: c23729a2d91ce21279b6a55e19968844
description: Simple NS with a single VNF and a single VL
id: hackfest1-ns
name: hackfest1-ns
user: Admin
vendor: Atos
version: '1.0'
visibility: true
vnfd-id-ref:
- hackfest1-vnf
```

The information provided in this metadata yalm file is used to compose the index information of the repository.

In order to clarify the parameters considered in the file, some description is provided:

- checksum: hash of the package file required for OSM validation.
- description: more detailed information of the package.
- id: unique key that identifies the package.
- name: name of the package.
- user: person that store the package.
- vendor: network service provider developer.
- version: define the current versioning tag.
- visibility: defines if element can be shown public.
- vnfd-id-ref (only for VNF): list of VNFs that composed the service.
- images (only for NS): Type of image to instantiate the descriptor.

3.5.3. OSM Client

The development of the OSM Client is a key contribution of 5GENESIS [25]. This client is in charge of generating the Repository file structure from many unstructured packages with an

ETL (Extract, Transform & Load) data process, import repositories in OSM and the repository management operations from the OSM side (add, list and delete repositories, list and onboard artefacts). It is possible to view the OSM Repository documentation in the OSM user guide [18].

In what follows we have organized and explained the main functionalities of the OSM Client.

1. Repo Index: ETL to generate the directory structure for a repository

The OSM Client needs to build the directory structure from a directory full of VNFs and NS artefacts. The command is "repo-index" and it has two optional parameters:

- origin: To indicate the directory where the artefacts are. There are two kind of recognizable artefacts, compressed files with the extension ".taz.gz" and directories with the required configuration files. The descriptor file must be inside both structures. By default, if this parameter is not specified, the current route will be the origin.
- destination: To indicate the directory where the index is going to be created. By default, if this parameter is not specified, a folder called repository in the current route will be created.

In both parameters, it is possible to use absolute or relative routes.

Command example:

```
osm repo-index --origin <packages_folder> --destination <repository_folder>
```

2. Repo add: Aggregate an OSM Repository

The OSM Client needs to link a remote OSM Repository for listing and getting the VNF packages. The repository must follow the previous structure defined before. The command to aggregate the repository is "repo-add", previously used just for Kubernetes repositories.

Command example:

```
osm repo-add --description <repo description>  <repo name>  <repository_url>
```

3. Repo list: A list of the current repositories in OSM

The OSM Client needs to list the remote repositories. The command for this is "repo-list", previously used just for Kubernetes repositories and currently also listing the OSM Repositories.

Command example:

```
osm repo-list
```

4. VNF packages list: List the current VNFs packages in the repositories

Once the OSM repository is aggregated in OSM, through the OSM client, it is possible to consult the list of available VNFs in the added repository.

Command examples:

```
osm vnfpkg-repo-list
```

```
osm nfpkg-repo-list
```

5. NS packages list: List the current NSs packages in the repositories

It is possible for NS to consult the list of available NSs in the linked repository.

Command examples:

```
osm nsd-repo-list
```

```
osm nspkg-repo-list
```

6. Onboard VNF Packages: Onboard VNF Packages from OSM Repository

Onboarding from an OSM Repository is another key feature, that just requires passing the repository name and the VNF ID.

Command examples:

```
osm vnfpkg-create --repo <repo> <vnf>
```

```
osm nfpkg-create --repo <repo> <vnf>
```

7. Onboard NS Packages: Onboard NS Packages from OSM Repository

Onboarding from an OSM Repository is another key feature. Just passing the repository name and the NS ID.

Command examples:

```
osm nsd-create –repo <repo>  <ns>
```

```
osm nspkg-create --repo <repo> <ns>
```

8. Show VNFs Packages: Show the VNFs details from OSM Repository

Showing the VNF packages. Just passing the repository name and the NS ID.

Command examples:

```
osm vnfpkg-repo-show --repo <repo> <vnf>
```

```
osm nfpkg-repo-show --repo <repo> <nsd>
```

9. Onboard NS Packages: Onboard NS Packages from OSM Repository

Onboarding from an OSM Repository is another key feature. Just passing the repository, the NS ID and the repository.

Command examples:

```
osm nsd-repo-show --repo <repo> <vnf>
```

```
osm nspkg-repo-show --repo <repo> <vnf>
```

3.5.4. OSM GUI

The new OSM LW-UI (Light Weight User Interface) includes in the main menu panel an option for accessing the *OSM repositories* feature to see and manage the registered repos (see figure below). The repository listing is organised by name, repo identifier, type ("OSM" for this type of repo), location, and creation and modification dates.

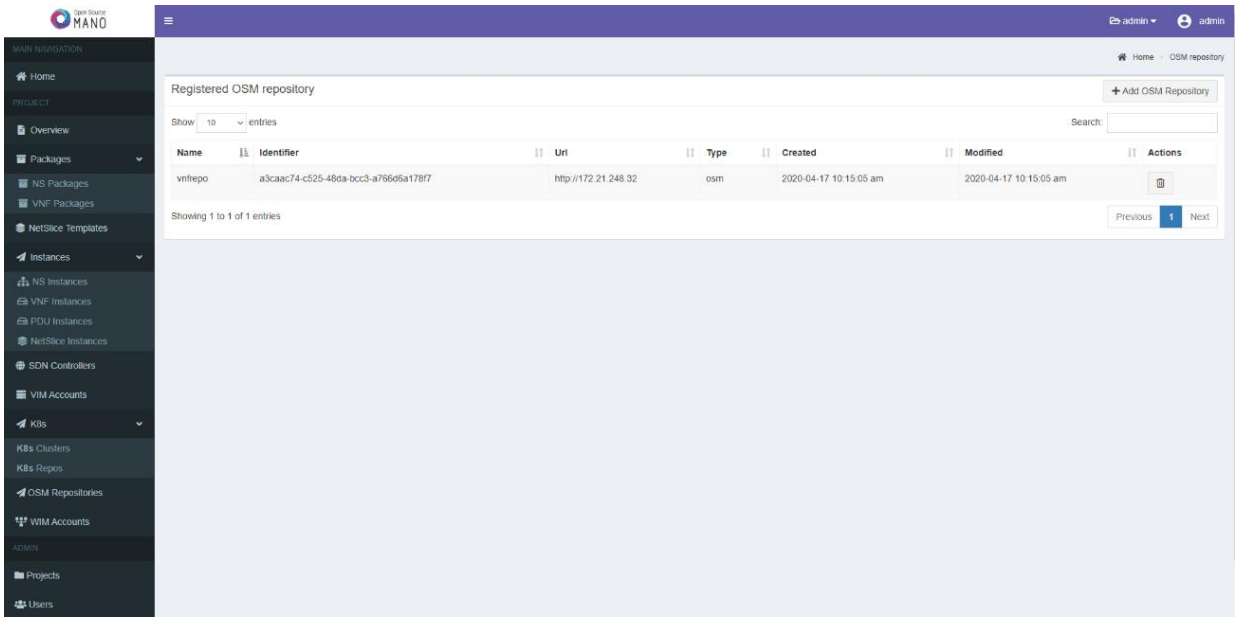

Figure 30 - OSM repositories menu screen

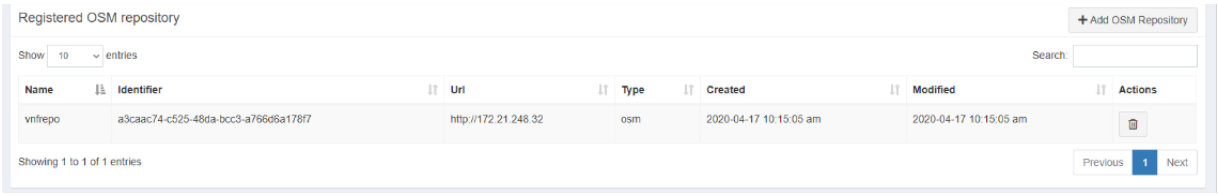

Figure 31 - OSM repositories list screen

From this screen (Figure 30, Figure 31), the user can also register a new repository (see figure below) by filling up a form with the following information:

1. Repo name.
2. Repo URL.
3. Repo description.
4. Type ("OSM" by default).

The registration view of the OSM panel is visualized for the as follows (Figure 32):

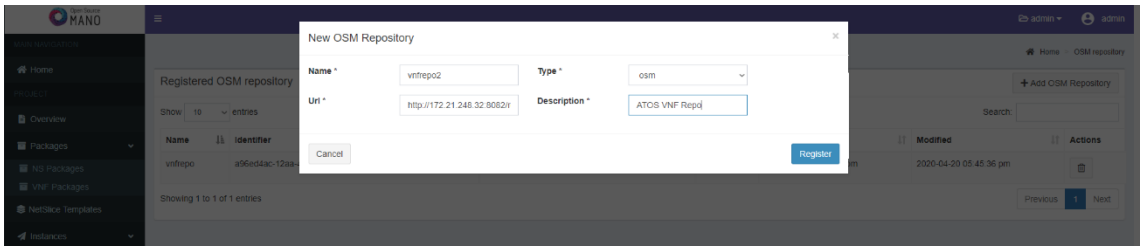

Figure 32 - OSM repository registration

In the current version, we consider that the external repositories are open and do not need access credentials.

For the current release of the NS repository the management of images has not been considered, as it is a much more complex instance to be stored in a remote repository due to the size of the file and the complexity of uploading to the VIM and there is still not a common agreement how they have to be handled. For example, for several VIMs that can be managed and used by a OSM instance, the current strategy will be to duplicate images in all of the VIMs for the instantiation process, this can simplify the preparation of the NS indexing due to the time depending on the image load.

3.5.5. Functional Testing

In order to validate the developments, some tests have been developed to verify the good behaviour of the OSM contribution.

The tests have been designed to validate and verify the new functionalities of the OSM Repository feature. A full set of automated testing will be also reported in D3.8 Open APIs, service level functions and interfaces for verticals (Release B) , also delivered at the same milestone as D3.2, describing the integrated component in the Dispatcher environment that will cover all the features involved with the MANO Wrapper.

The tests that have been executing to assure the correct functioning of the NS repository have been the following:

- FT1 - Repository Creation: It is possible to create a directory with the OSM repository structure.

FT1

```
osm_hackfest_48@osmvnfonb:/tmp/hola$ osm repo-index --origin . --destination repo
WARNING VNF /tmp/hola/gateway_pnf.tar.gz already exists

Final Results:
VNF Packages Indexed: 18
NS Packages Indexed: 18
```

- FT2 - Repository addition: It is possible to add a repository in OSM server.

FT2

```
osm_hackfest_48@osmvnfonb:~$ osm repo-add --description "OSM Repository" osmrepo
https://osm-download.etsi.org/ftp/vnf/vnf-catalog
c70c8750-2d9e-4c2d-9626-4a5f87432809
```

- FT3 - Repository List: It is possible to list all the repositories saved in OSM server.

FT3

```
osm_hackfest_48@osmvnfonb:~$ osm repo-list
+---------+--------------------------------------+------+-----------------------------------------------+----------------+
| Name    | Id                                   | Type | URI                                           | Description    |
+---------+--------------------------------------+------+-----------------------------------------------+----------------+
| osmrepo | c70c8750-2d9e-4c2d-9626-4a5f87432809 | osm  | https://osm-download.etsi.org/ftp/vnf-catalog/ | OSM Repository |
+---------+--------------------------------------+------+-----------------------------------------------+----------------+
```

- FT4 - Repository VNF List: It is possible to list all the VNF packages in a repository.

FT4

```
osm_hackfest_48@osmvnfonb:~$ osm vnfpkg-repo-list
+------------------------------+------------+
| nfpkg name                   | repository |
+------------------------------+------------+
| fb_magma_knf                 | osmrepo    |
| squid-vnf                    | osmrepo    |
| hackfest_gateway_vnfd        | osmrepo    |
| hackfest_magma-agw-enb_vnfd  | osmrepo    |
+------------------------------+------------+
```

- FT5 - Repository NS List: It is possible to list all the NS packages in a repository.

FT5

```
osm_hackfest_48@osmvnfonb:~$ osm nspkg-repo-list
+------------------------------+------------+
| nfpkg name                   | repository |
+------------------------------+------------+
| hackfest_magma-agw-enb_nsd   | osmrepo    |
| squid-cnf-ns                 | osmrepo    |
| fb_magma_ns                  | osmrepo    |
+------------------------------+------------+
```

- FT6 - NS Onboarding fail for existing NS package: Trying to upload a NS, from the repository, that already exists in the system.

FT6

```
osm_hackfest_48@osmvnfonb:~$ osm nsd-create --repo osmrepo hackfest_magma-agw-enb_nsd
ERROR: Error 409: {
    "code": "CONFLICT",
    "status": 409,
    "detail": "nsd with id 'hackfest_magma-agw-enb_nsd' already exists for this project"
}
```

- FT7 - NS Onboarding fail for no existing VNF reference: Trying to upload a NS, from the repository with a VNF dependency that does not exist in the system.

FT7

```
osm_hackfest_48@osmvnfonb:~$ osm nsd-create --repo osmrepo fb_magma_ns
ERROR: Error 409: {
    "code": "CONFLICT",
    "status": 409,
    "detail": "Descriptor error at 'constituent-vnfd':'vnfd-id-ref'='fb_magma_knf' references a non existing vnfd"
}
```

- FT8 - VNF Onboarding: It is possible to create a VNF from a repository, just indicating his ID.

FT8

```
osm_hackfest_48@osmvnfonb:~$ osm vnfpkg-create --repo osmrepo fb_magma_knf
7d9eaac9-07c5-44f3-8ca8-e5ff08863e36
```

- FT9 - NS Onboarding: It is possible to create a NS from a repository, just indicating his ID.

FT9

```
osm_hackfest_48@osmvnfonb:~$ osm nsd-create --repo osmrepo fb_magma_ns
fe0e5edc-b434-477f-b671-10901c296108
```

4. EXTENSIONS FOR OPENTAP

In addition to the solution described in the sections above, we describe an approach to deploy and instantiate VNFs and NS in OSM, developed as an alternative to the mainstream approach recommended in the 5GENESIS project [26]. This alternative gives platform operators the option to leverage their existing test cases (based on OpenTAP) during their migration to the 5GENESIS' recommended approach. The OSM Plugin for OpenTAP provides VNFD and NSD developers the opportunity to automate and continuously test and develop their network functions with either a local developer environment (OSM with *emu-vim* VIM emulator) or team-wide MANO environment (e.g. OSM with OpenStack / OpenVIM combination).

OpenTAP [19] is an open-source automated test execution solution that allows developers to define test steps and algorithms. It provides the functionality and APIs [17] to define, configure and control plugins in the form of test steps, instruments, result listeners and device-under-tests (DUTs). An overview of the OpenTAP architecture is illustrated below in Figure 33.

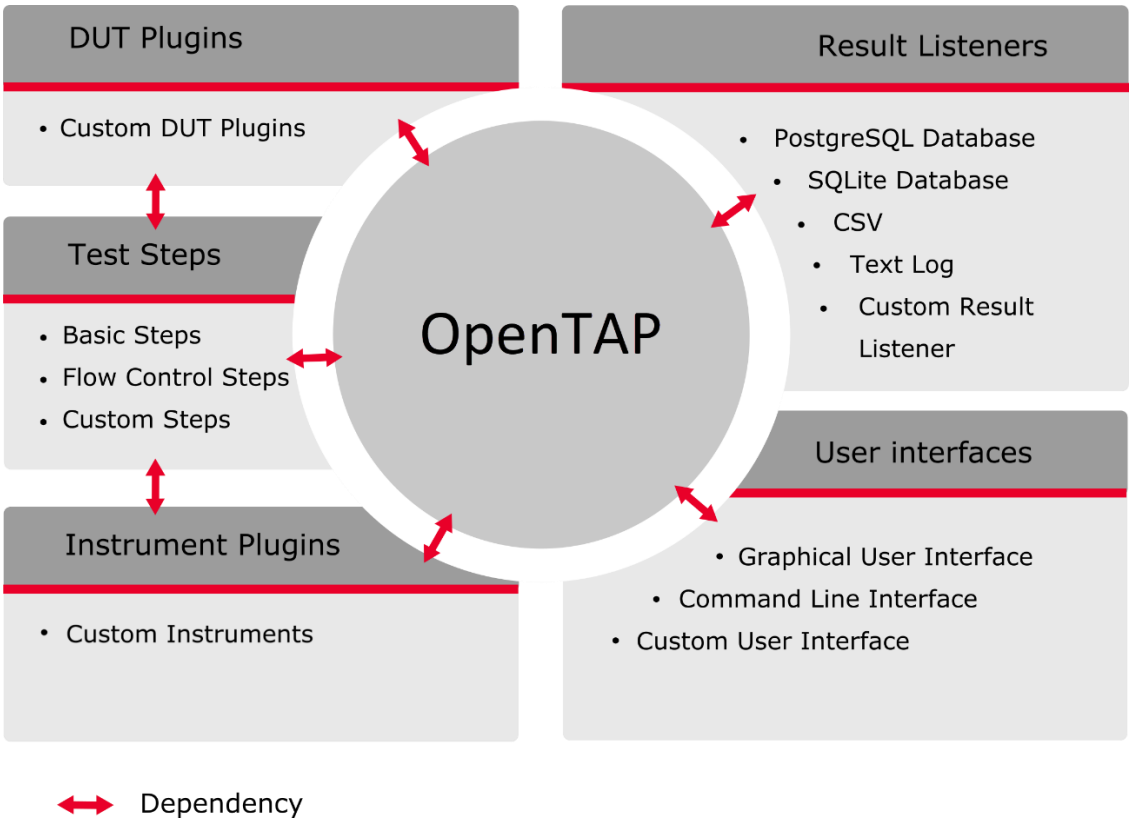

Figure 33 – Overview of the OpenTAP architecture

As part of the *5GENESIS Plugin Suite for OpenTAP* [21], we provide the OSM package as illustrated below in Figure 35. The *5GENESIS Plugin Suite* is developed as a C# application based on the OpenTAP SDK (Community Edition v9.4.2) [20]. The *Suite*, illustrated in Figure 34, consists of a number of packages that provide various OpenTAP *Test Instruments* (iPerf, Ping, MONROE, Prometheus, SSH, OSM, VIM), *Result Listeners* (InfluxDB, AutoGraph, CSV) and *Test*

Steps used in 5GENESIS. The *Suite* is distributed as an OpenTAP Package (*.TapPackage file) and can be installed on any OpenTAP Test Editor tool.

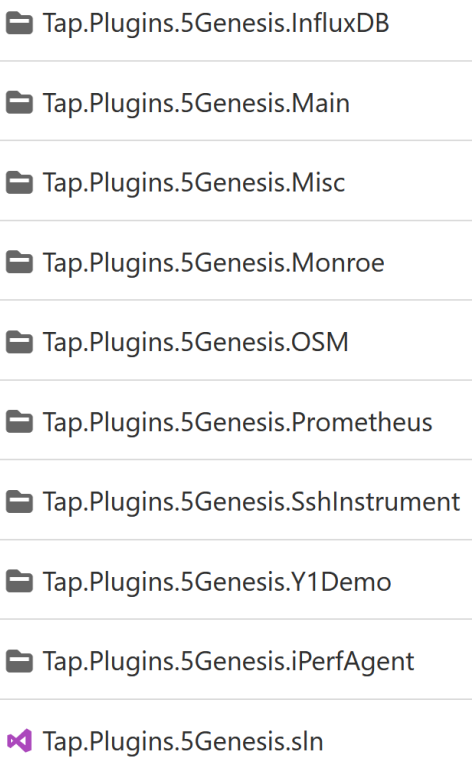

Figure 34 – 5GENESIS Plugin Suite for OpenTAP

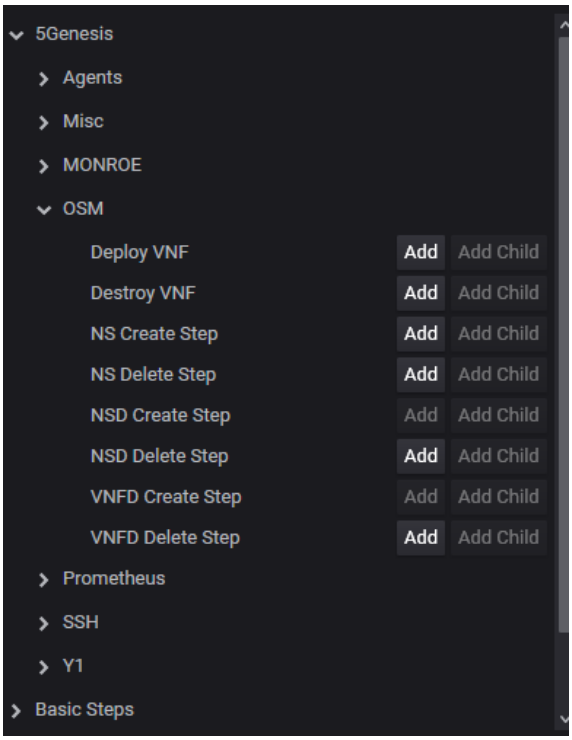

Figure 35 – OSM Package as part of the 5GENESIS plugin suite for OpenTAP

The OSM Package for OpenTAP consists of eight OSM related tests steps. Of these, two tests are purely parent steps, i.e., they cannot be the children of any other test step. These are:

- *Deploy VNF* – Parent step to instantiate a new Virtual Network Functions (VNFs) to OSM.
- *Destroy VNF* – Parent step to destroy currently deployed VNFs in OSM.

These two steps provide OpenTAP the functionality to nest other OSM related test steps and to choose the instruments and result listeners that would be used by all the underlying child steps.

Four pure child steps allow the test developer to create and destroy VNFDs and NSDs. These child steps are:

- *VNFD Create* – Atomic step to upload and create a new VNFD package in OSM.
- *VNFD Destroy* – Atomic step to delete a VNFD package uploaded to OSM.
- *NSD Create* – Atomic step to upload and create a new NSD in OSM.
- *NSD Destroy* – Atomic step to delete an existing NS Descriptor in OSM.

The *VNFD Create* atomic child step allows the test developer to specify the VNFD package (*.tar.gz file) and the availability zone to upload and place the VNFD. The *VNFD Destroy* step allows the tester to specify / reference the ID of the VNFD package that was either previously uploaded using OpenTAP or manually uploaded to OSM.

The *NSD Create* child step allows the tester to specify / browse and choose the NS Descriptor package (*.tar.gz file) that would be uploaded to the OSM for the instantiation of the NS. The *NSD Destroy* atomic step allows the test developer to specify / reference the ID of the NSD package that was previously uploaded during the OpenTAP test run or manually uploaded to OSM by the tester.

Finally, two parent/child steps provide the functionality to create and destroy the NS in OSM. These are:

- *NS Create* – Hybrid parent/child step to instantiate a NS in OSM.
- *NS Destroy* – Hybrid parent/child step to destroy and delete the NS instance in OSM.

The *NS Create* step provides attributes to input / reference the ID of the NSD that was uploaded to OSM and the VIM in which the NS would be instantiated. The *NS Destroy* steps allow the test developer to destroy an instantiated NS. The step provides attributes to choose the ID of an instantiated NS, the number of retries to perform and the duration of each retries when the *NS Destroy* step initially fails.

In addition to the eight OSM test steps, two OpenTAP test instruments are provided in the OSM Plugin package. These are:

- *OSM Instrument* – Allows the configuration of OSM REST endpoint used by the Test steps. This instrument takes care of the AccessToken generation.
- *VIM Instrument* – Allows the user the choose the OSM VIM account that would be used in the Test steps.

All test steps and instruments are developed in C# and use the Northbound REST API (NBI) [21] provided by OSM to provide the functionalities. An overview class-diagram of the OSM Test Steps and Instruments is provided below in Figure 36.

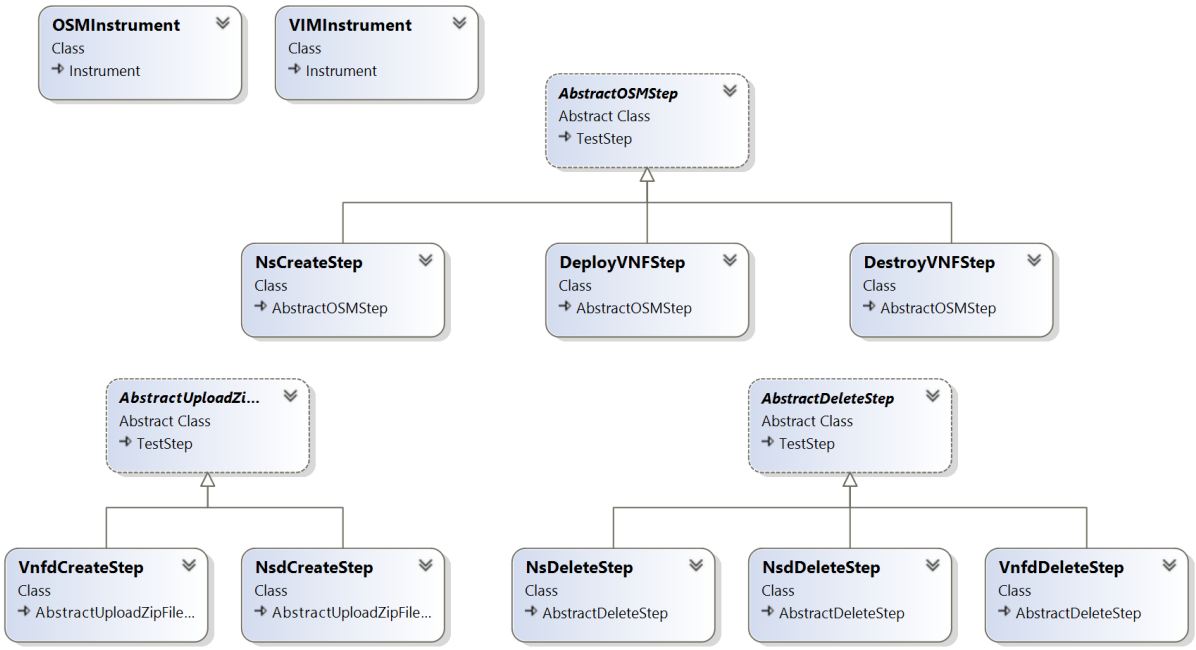

Figure 36 – Overview Class Diagram of the OSM Test Steps and Instruments

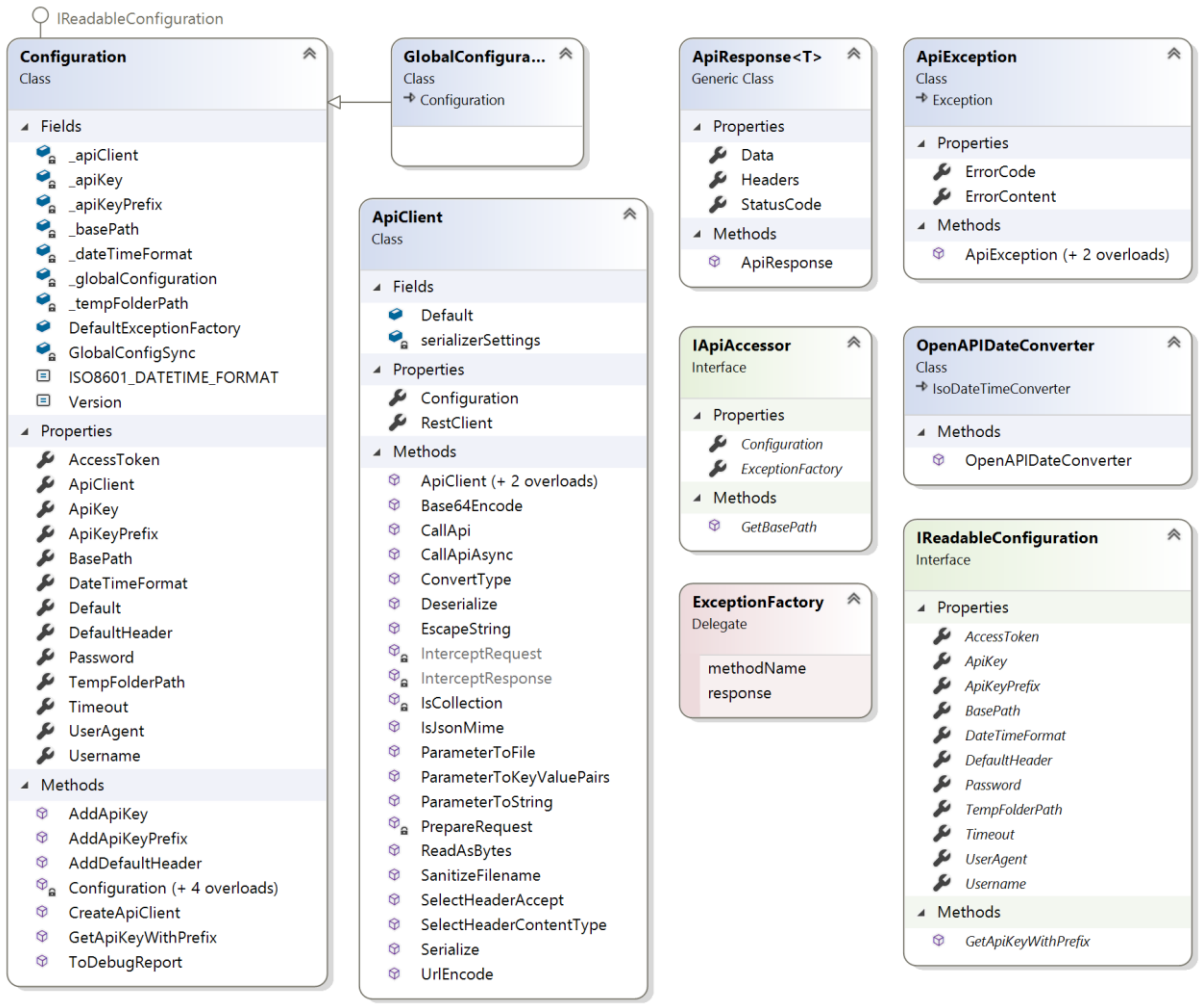

Figure 37 –Class Diagram of the API Client used for OSM NBI

Usage of the OSM Plugin for OpenTAP is pretty straightforward. As a prerequisite, ensure that your OSM instance is configured correctly. Either configure your OSM instance with the *emu-vim* VIM emulator for local OSM based testing and development, or with OpenStack / OpenVIM when using OSM deployed in a *staging* / team environment.

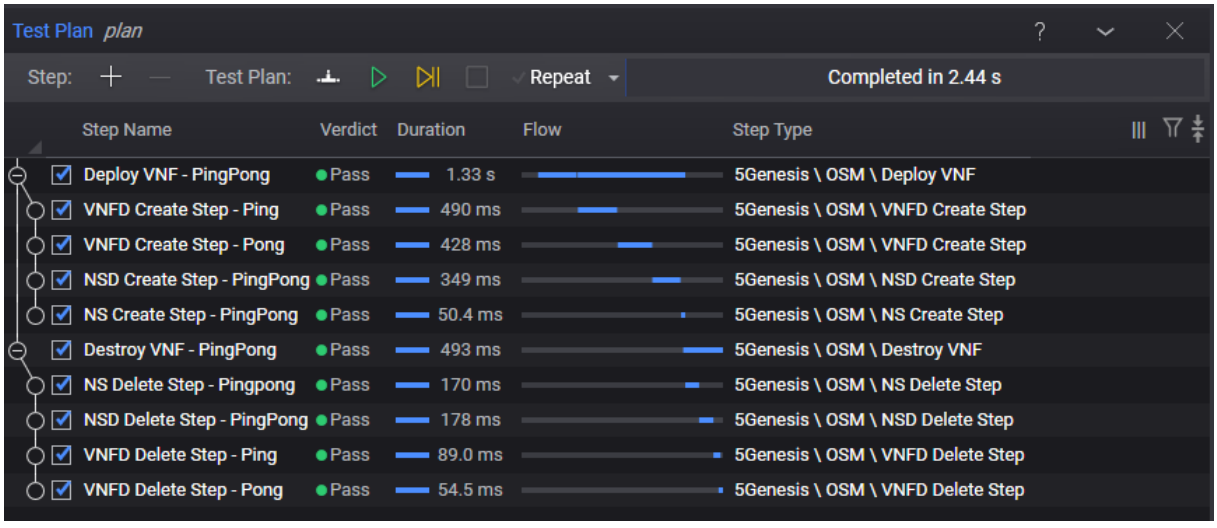

Figure 38 – Typical OSM Plugin usage for testing and development

As the first step, create instruments that reflect your OSM and VIM environment. Thereafter add test steps to create VNFDs and NSD, followed by the test step to instantiate the NS. Destroy the VNFDs and NSDs in the reverse order of the creation. A snapshot of the typical OSM Plugin usage is show in Figure 38.

5. CONCLUSIONS

This document provides the final version of the design and implementation of the Management and Orchestration of 5GENESIS, with the provided features and flow description of the developed functionalities. Based on the 5GENESIS requirements and the feedback provided by the cycle's evaluation, the component has been improved in order to comply with the 5GENESIS platform requirements.

The work has been organized as follow:

- Recap of Release A status and major improvement towards Release B.
- Architecture update and components interconnection.
- Description of features with the interaction flow and operations with the NS and related artifacts.
- Manual to install and operate with the component.
- Contribution to the Release 8 of OSM.
- Extensions for Open TAP plugins.
- Testing and validation of the implemented features.

The result of this work can be considered as the contribution to the MANO layer inside the 5GENESIS architecture, the software component can be found in the 5GENESIS repository.

REFERENCES

[1] D3.1 Management and orchestration https://5genesis.eu/wp-content/uploads/2019/10/5GENESIS_D3.1_v1.0.pdf.
[2] OSM Release 8 https://www.etsi.org/newsroom/press-releases/1799-2020-07-etsi-launches-osm-release-eight
[3] Open Nebula https://opennebula.io/
[4] MANO architecture https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf
[5] OSM release notes https://osm.etsi.org/wikipub/images/5/56/OSM_Release_EIGHT_-_Release_Notes.pdf
[6] VNFD Model http://osm-download.etsi.org/ftp/osm-doc/vnfd.html
[7] NSD Model http://osm-download.etsi.org/ftp/osm-doc/nsd.html
[8] OSM guide https://osm-download.etsi.org/ftp/osm-6.0-six/8th-hackfest/presentations/8th%20OSM%20Hackfest%20-%20Session%202.1%20-%20Creating%20a%20basic%20VNF%20and%20NS.pdf
[9] https://osm-download.etsi.org/ftp
[10] Authorization component https://github.com/5genesis/Dispatcher/tree/master/auth
[11] Plugin for OpenStack https://github.com/5genesis/Dispatcher/blob/master/mano/libs/openstack_util.py
[12] Plugin for OpenNebula https://github.com/5genesis/Dispatcher/blob/master/mano/libs/opennebula_util.py
[13] MANO Wrapper repository https://github.com/5genesis/Dispatcher/tree/master/mano
[14] https://osm.etsi.org/wikipub/index.php/Research#5Genesis
[15] D7.6 Standardisation and regulation https://5genesis.eu/wp-content/uploads/2020/07/5GENESIS-D7.6_v1.0.pdf
[16] https://osm.etsi.org/docs/user-guide/_sources/06-osm-platform-configuration.md.txt
[17] https://osm.etsi.org/gitlab/osm-doc/osm-user-guide/commit/5eb3c0f5150e4f85daee1f7ff347a27e4cc641d8#6581028c17633719d2f9ae8939012409029170a5
[18] OSM user guide https://osm.etsi.org/docs/user-guide/
[19] OpenTAP.io, OpenTAP Developer Guide, accessed 17 Feb. 2021, https://www.opentap.io/docs/OpenTAP%20Developer%20Guide.pdf
[20] OpenTAP.io, OpenTAP API Reference 9.12, accessed 17 Feb. 2021, https://doc.opentap.io/api/
[21] OSM, Annex 4 – NBI API Description, accessed 17 Feb. 2021, https://osm.etsi.org/docs/user-guide/12-osm-nbi.html
[22] ETSI NFV Information Models :https://www.etsi.org/technologies/nfv
[23] Prometheus monitoring system https://prometheus.io/
[24] JSON Encoding of Data Modeled with YANG https://tools.ietf.org/html/rfc7951
[25] H. Koumaras et al., "5GENESIS: The Genesis of a flexible 5G Facility," 2018 IEEE 23rd International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Barcelona, Spain, 2018, pp. 1-6, doi: 10.1109/CAMAD.2018.8514956.

[26] A. Díaz Zayas, G. Caso, Ö. Alay, P. Merino, A. Brunstrom, D. Tsolkas, and H. Koumaras, “A Modular Experimentation Methodology for 5G Deployments: The 5GENESIS Approach,” Sensors, vol. 20, no. 22, p. 6652, Nov. 2020 [Online]. Available: http://dx.doi.org/10.3390/s20226652