5G Solutions for European Citizens

D2.2B Specifications and design of CDSO plugins (V2.0)

Document Summary Information

Grant Agreement No	856691	**Acronym**	5G-SOLUTIONS
Full Title	5G Solutions for European Citizens		
Start Date	01/06/2019	**Duration**	36 months
Project URL	https://www.5gsolutionsproject.eu/		
Deliverable	D2.2B Specifications and design of CDSO plugins (V2.0)		
Work Package	WP2		
Contractual due date	M12	**Actual submission date**	25/05/2020
Nature	Report	**Dissemination Level**	Public
Lead Beneficiary	Nokia		
Responsible Author	Udi Margolin		
Contributions from	Nokia, LiveU, UoP, FNET		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 856691.

Revision history (including peer reviewing & quality control)

Version	Issue Date	% Complete[1]	Changes	Contributor(s)
V0.1	05/4/2020	0%	Baseline – Final D2.2A	Udi Margolin (Nokia)
V0.2	04/5/2020	50%	Add missing UC's, add sequence diagram for UC4.1, add preliminary flow table for UC 4.4 Add initial Flow table for UC4.4 + appendices	Udi Margolin (Nokia), Baruch Altman (LiveU), Christos Tranoris (UoP)
V0.3	06/5/2020	50%	Remove LiveU confidential code	Baruch (LiveU)
V0.3	07/5/2020	50%	Quality Check	Christos Skoufis (EBOS)
V0.4	12/5/2020	95%	Integrate UC flow tables for UC4.1, 4.4 & 4.6	Udi Margolin (Nokia), Moshe Elisha (Nokia), Yannis (FNET)
V0.5	13/5/2020	Ready for review	Last minute touch ups	Udi Margolin
V0.5	18/5/2020	Review Comments	Quality Check and Peer Review	Christos Skoufis (EBOS), Andrea Di Giglio (TIM), Anne-Marie Bosneag (LMI), Saman Feghhi (LMI), Takis Apostolopoulos (UoP)
V0.6	20/5/2020	After Review	Apply Review Comments	Udi Margolin
V0.7	24/5/2020	Final	Last review comments	Udi Margolin

Disclaimer

[1] According to 5G-SOLUTIONS Quality Assurance Process:
1 month after the Task started: Deliverable outline and structure
3 months before Deliverable's Due Date: 50% should be complete
2 months before Deliverable's Due Date: 80% should be complete
1 months before Deliverable's Due Date: close to 100%. At this stage it sent for review by 2 peer reviewers
Submission month: All required changes by Peer Reviewers have been applied, and goes for final review by the Quality Manager, before submitted

Copyright message

Table of Contents

List of Figures

List of Tables

Glossary of terms and abbreviations used

Abbreviation / Term	Description
AGV	Automated Guided Vehicle
AMQP	Advanced Message Queuing Protocol
CDN	Content Delivery Networks
CBND	Cloudband Network Director
CDSO	Cross Domain Service Orchestrator
C-RAN	Cloud Radio Access Network
CCTV	Closed Circuit Television
DSM	Demand Side Management
HA	High Availability
NFVO	Network Function Virtualization Orchestrator
NS	Network Service
NSD	Network Service Descriptor
PEV	Plugin Electric Vehicles
PNF	Physical Network Function
SDN	Software Defined Network
SD-WAN	Software Defined Networking in a Wide Area Network
TOSCA	Topology and Orchestration Specification for Cloud Applications
UoP	University of Patras
vEPC	Virtual Evolved Packet Core
VIM	Virtual Infrastructure Manager
VNF	Virtual Network Function
VNFD	Virtual Network Function Descriptor
VNFFG	Virtual Network Function Forwarding Gateway
VNFM	VNF Manager
VoLTE	Voice Over Long-Term Evolution

1 Executive Summary

Deliverable 2.2 is the bridge between the use case definition and requirements, the ICT-17 facility interfaces and the 5G-SOLUTIONS orchestrator. It has 2 main goals:

1) Describe the capabilities of the CDSO and the orchestration needs of the project. Give a detailed description of the extensibility mechanism of the CDSO and how we can use this mechanism to glue together the VNFs and networks into end-to-end use cases we can execute and report on
2) For each use case, identify the orchestration requirements and the gaps between the existing capabilities of the ICT-17 facility orchestrators and the needs of the use case. Then define how this gap will be closed by developing custom plugins to the CDSO.

The inputs to this deliverable come from WP1 (D1.1, D1.2) and from the ICT-17 projects (5G-VINNI, 5g-EVE) and their facility owners. The 5G-SOLUTIONS CDSO is based on Nokia CBND thus information in this deliverable is also based on CBND documentation.

These sources of information have different level of maturity (as of Mar 2020) thus the information we managed to receive is still not full for all Use Cases and Facilities. Since there are 4 drops for this deliverable, we expect that as more information becomes available when use case definition is more mature and the ICT-17 projects progress, we can complete the missing parts.

This deliverable is the main input for task T2.3 and completing the missing information is crucial to the success of T2.3.

Our focus in T2.3 will be 1st on the interface to the ICT-17 orchestrator first and after that we'll also handle those use cases that are not hosed in an ICT-17 facility and require specific direct orchestration actions. (called Category B use cases as defined in section 3.1.1 below).

Release D2.2B includes updates to the above for Cycle I of the experimentation. It covers in greater depth the Use Cases that are planned for Cycle I and specifically those that are planned to be integrated with CDSO in Cycle I.

Use cases that are planned to be integrated with CDSO in cycle I include a full flow of APIs and actions to document in full what the integration includes. As some of these steps are still being defined, if not all details are fully known, we indicate this in the flow. The missing details will be added as we progress with the integration and will be reported in the next version of D2.2.

2 Introduction

The 5G-SOLUTIONS project targets to explore various 5G verticals, set them up based on ICT-17 projects infrastructure, measure various KPIs from the network and application and show if the 5G infrastructure provides the required QoS for the different 5G use cases.

It was decided that the 5G-SOLUTIONS project will work with ICT-17 5G-EVE (Turin facility) and 5G-VINNI (Oslo & Patras facilities) to set up different use cases. Some use cases though, while having an association with 5G-EVE or 5G-VINNI, are deployed on other locations and not orchestrated end to end by the 5G-VINNI or 5G-EVE orchestrators.

We have created a classification of use cases, based on their orchestration needs. We thus differentiate between use cases that are orchestrated by an ICT-17 orchestrator (classified as Category A), and use cases not orchestrated by an existing ICT-17 orchestrator (classified as Category B).

5G-SOLUTIONS is using the Nokia CBND orchestrator, which was originally developed as an NFVO, and an end-to-end Orchestrator (CDSO == Cross Domain Service Orchestrator), leveraging the CBND Plugin mechanism to orchestrate multiple domains and provide end-to-end slice orchestration.

For Category A use cases, CBND will trigger the orchestration actions through an API to the ICT-17 orchestrator while for Category B use cases, end-to-end orchestration will be provided directly by CBND. This deliverable will present for all use cases that were defined up to this point the classification of Category A or Category B.

In D2.2B we identified a 3rd category of use cases – Category C, for those use cases that are operated manually and not integrated with the CDSO. These use cases are usually deployed inside factories / vertical location and connect to a 5G network via an on-site 5G component that is not connected to an ICT-17 facility.

In order to support these interfaces, there is a need to develop special plugins in CBND. The purpose of this deliverable is to define the required plugins.

This is the 2nd drop of 4 planned drops for deliverable D2.2. (D2.2B). This deliverable is based on information covered in more detail in Deliverables D1.1A and D1.2A which include the detailed requirements of each use case and the details of the ICT-17 projects and facilities on which 5G-SOLUTIONS is deploying its living labs.

This deliverable focuses on information relevant for Cycle I of the experiments and includes orchestration requirements submitted up to end of March 2020. Any use cases that have provided their requirements after this date will be included in the next releases of this document.

The Use cases confirmed for Cycle I experiments and that require CDSO integration include:

- UC4.1 - Ultra High-Fidelity Media
- UC4.4 – User & Machine Generated Content
- UC4.6 - Cooperative Media Production

In addition, there are several use cases which are defined as "best effort" for CDSO integration in Cycle I as detailed below. For the above 3 use cases, we tried to capture as much detail as possible for the integration flows with the CDSO as needed for implementation of T2.3. In this version, we decided to align the paragraph headers with the official use case numbering to avoid confusion.

2.1 Mapping Projects' Outputs

The purpose of this section is to map 5G-SOLUTIONS Grant Agreement commitments, both within the formal Deliverable and Task description, against the project's respective outputs and work performed.

Table 1: Adherence to 5G-SOLUTIONS GA Deliverable & Tasks Descriptions

Project GA Component Title	Project GA Component Outline	Respective Document Chapter(s)	Justification
DELIVERABLE			
D2.2B: Specifications and design of CDSO plugins (V2.0)	Defining the technical specifications and the low-level design to enable the implementation of the necessary plugins at CDSO.	Chapter 3.2 & 3.3 provide technical information on Nokia CBND and its plugin mechanism. Chapter 4 lists all use cases with their orchestration requirements, available interfaces, the gaps and the plugins that need to be developed to bridge that gap.	Plugin Specification specified for the Use Cases planned for integration with CDSO in Cycle I
TASKS			
T2.2: Cross-Domain Service Orchestrator (CDSO) plugins design & development for cross-domain orchestration	CDSO requirement analysis	Chapter 4 lists the use cases and their orchestration requirements	For each use case, we have a table consolidating the use case classification and its orchestration requirements from the CDSO.
	CDSO APIs design	Chapter 4 lists the use cases, and for each the available APIs and which APIs are missing and how to bridge that gap	APIs are detailed for UC's included in CDSO integration for Cycle I
	Deployment analysis	Chapters 3.2.3, 3.2.5 define the required resources and environment needed	Define where the CDSO will be deployed, identify the cloud resources on which it will be deployed and identify the required network resources.

2.2 Deliverable Overview and Report Structure

This deliverable is comprised of the following chapters:

Chapter 3: 5G-SOLUTIONS: The Orchestrator

In this chapter we will layout the high-level orchestration requirements of 5G-SOLUTIONS and see how these requirements are mapped to capabilities supported by Nokia CBND. This chapter will also cover the

deployment and resource requirements of Nokia's CBND and what configuration actions are needed. Finally, we will cover the plugin model and how integration is done with CBND.

Chapter 4: Integrating with 5G-SOLUTIONS Living Labs

In this chapter we will step through each use case supported by 5G-SOLUTIONS and cover:

- **Use Case Overview:** Short description of the use case. Additional details can be found in chapter 5 or deliverable D1.1A.
- **Use Case Information:** Category A or Category B, ICT-17 Facility and other available details.
    - **For the UC's included in Cycle I CDSO integration we also include: Existing Integration Points:**
        - For Class A use cases, this will point to the relevant ICT-17 orchestrator.
        - For Class B use cases, this will list available APIs we can use to orchestrate the use case.
    - **Integration Point Gaps:** Integration points that are not covered out of the box for this use case and need to be enhanced
    - **Plugins to develop:** Based on the gaps identified, which plugins need to be developed

Chapter 5: Conclusions & Next Actions

A summary and impact on project work.

3 5G-SOLUTIONS: The Orchestrator (Based on Nokia CBND)

3.1 Orchestration Tasks in 5G-SOLUTIONS

End-to-end orchestration of services and networks includes many activities as defined by different standardization bodies such as ETSI, TMF, ONAP etc. Orchestration of services include some or all actions as defined in the list below.

These may include:

- Catalog Management: Adding an application or a service entity to the orchestrator's catalog, removing from the catalog, listing the catalog etc.
- Management of the service descriptor files (the SD file includes a description of the components making up the service and how they are connected. Sometimes this is referred to as NSD, described in subsection 3.1.2)
- Managing the service itself (start, stop, status)
- Lifecycle actions on the service (scale up, scale down, heal)
- VNF Upgrade
- VNF Data Backup
- And other activities as needed for management of the services

3.1.1 Use Case Categories

As mentioned earlier, we have classified the use cases into two main categories based on their orchestration needs:

1) **Category A**: Use cases that are hosted in an existing ICT-17 facility and are already orchestrated by the orchestrator in that project. We mapped 3 facilities that will be used in 5G-SOLUTIONS:
    a. 5G-VINNI Norway
    b. 5G-VINNI Patras
    c. 5G-EVE Turin

In these use cases, we assume that the ICT-17 orchestrator is already integrated (or will be integrated) with the use case and we will work with the project owners to integrate the 5G-SOLUTIONS orchestrator with each facility. To this end, we'll perform the following (some or all) lifecycle actions:

a. Activate Service – will trigger the service deployment for the given service
b. Get Service Status – will query the local orchestrator for the status of the deployed service
c. Subscribe to notifications – In case the orchestrator supports this, we will subscribe to notifications and display them in our event stream
d. Terminate Service – will trigger termination of the service.

It is expected that in all ICT-17 projects we integrate with, the API exposed is HTTPs/REST based, that there is connectivity from the 5G-SOLUTIONS orchestrator to that API (directly or via VPN) and that credentials are provided.

2) **Category B**: Use cases that are **not** hosted in an existing ICT-17 facility. The use case may be associated with an ICT-17 project and even include some 5G components provided by that project (e.g. 5G radio (ETSI NFV IFA014 V2.1.1, n.d.)) but the use case will be considered as a Category B if the ICT-17 orchestrator is not handling this use case end to end.

In this case, there are several sub-cases:

a. The use case has its own MANO orchestrator that is orchestrating it – in this case, we should treat this like Category A and trigger the lifecycle actions as done in Category A use cases.
b. The use case needs to be orchestrated, but does not have an orchestration solution. In this case, the 5G-SOLUTIONS orchestrator should provide end-to-end orchestration (as a MANO NFVO).
c. The use case is orchestrated manually – in this use case, we should collaborate with the use case owner to decide how this use case is reflected in the 5G-SOLUTIONS orchestrator.

In these cases, we may also have to create the NSD and add it to our catalog.

3) **Category C**: Use cases that are not hosted by an ICT-17 facility and do not require any orchestration as they reside inside a factory and have a standalone 5G network. These use cases will have to perform all the experiments manually and will not be connected to CDSO.

3.1.2 Service Modelling

A service is composed of applications (either in a virtual environment or residing on a physical equipment) and networks connecting these applications. The service is described in a file via a modelling language which can be in one of several formats. One of the common formats to describe a service is to use the TOSCA language[2]

In order to fully model the network service, there is a need to create a Network Service Descriptor (NSD). The NSD describes the topology of the network service, including Virtual Network Functions (VNFs), Physical Network Functions (PNFs), network elements and the VNF Forwarding Graph (VNFFG). The NSD and additional files, such as scripts used for launching the service and configuring its components, are packaged together in an NS package. The Network Service (NS) package will be added to the orchestrator catalog for later creation of an instance of that service.

The connection between applications in a service is often described by a VNFFG (as defined by ETSI in IFA 014[3]).

3.1.3 Onboarding Applications

In order to on-board a VNF (Virtual Network Function), there is a need to allocate resources on the cloud for that application, set up these resources with the appropriate images, configure the application with the parameters and launch the application components. All these actions are usually handled by a VNFM (Virtual Network Function Manager) which could either be specific to that application or generic.

3.1.4 Configuring Networks

As described above, services are composed of applications (VNFs/PNFs) and networks. In many cases, we are talking about virtual networks created and configured via Software Defined Networking (SDN). In order to set up an SDN network, the orchestrator needs to "talk" to the SDN controller and configure the network according to the defined parameters and required QoS. The SDN controller can be an open source one (e.g. OpenDaylight[4]) or a commercial one (e.g. Nokia Nuage[5])

[2] https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca
[3] https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/014/02.01.01_60/gs_NFV-IFA014v020101p.pdf
[4] https://www.opendaylight.org/
[5] https://www.nuagenetworks.net/

3.1.5 Monitoring Application Status

An important part of the orchestrator's role is to monitor the service status and make sure it is functioning properly. In case a service or a component of the service is not functioning well or suffering from a performance bottleneck, there may be a need to trigger a healing or scaling action to overcome this situation. The first step is to have visibility to the up-to-date status. This could be done either through polling for the status via an API or by subscribing to notifications and listening to what is happening inside the system. Not all systems support event subscription, so each case should be evaluated and handled separately.

3.1.6 Monitoring Network Status

As the applications (or VNFs) composing the service are a critical part of the service, the networks connecting these applications are just as important. Monitoring the network status is just as important and again can be done via polling or by listening to events. The metrics used to monitor the networks can be from the infrastructure (In/Out/Error etc) or from the application level (API calls per second, etc.)

3.2 The Nokia CBND Orchestrator

3.2.1 Overview

New virtualized services require an efficient NFVO to realize agility and cost advantages. CloudBand Network Director (CBND) provides two main functions:

- As a Network Service Orchestrator, CBND onboards Network Services (NS), automates their lifecycle, and provides monitoring and troubleshooting tools.
- As a Resource Orchestrator, CBND administers, provisions, monitors and optimizes NFV Infrastructure (NFVI) resources across geographically distributed NFVI nodes.

CBND focuses on the automation of recurring service deployments and lifecycle processes, with services ranging from single-VNF network services to SD-WAN[6] with dynamic service chaining, VoLTE[7] (Voice over LTE), vEPC[8] (Virtual Evolved Packet Core), C-RAN[9] (Cloud Radio Access Network) and Gi-LAN[10] services. CBND automates network services delivery and operation in a distributed, multi-tenant, multi-vendor NFV environment, while optimizing and governing the usage of the platform resources. Through the support of TOSCA, an open, domain specific model for NS, and open source workflow and policy engines, operators benefit from the rapid pace of open source innovation in carrier grade software that is ready for deployment.

In addition to being a state of the art NFVO[11] (Network Functions Virtualization Orchestrator), CBND is also a configurable and versatile orchestrator which can be customized to a multi domain orchestrator handling complex multi slice scenarios using the Plugin mechanism embedded into CBND.

Figure 1 below shows the high-level architecture of CBND. As can be seen, CBND includes different modules that comprise of different elements of the service orchestration. The service itself is described in an NSD file (in TOSCA format) and CBND handles each service component based on the description in the NSD. The service component (VNFs or Networks) may reside on one or more NFVI instances or on an EDGE cloud.

Additional information on CBND features can be found in D1.2.

[6] https://www.silver-peak.com/sd-wan/sd-wan-explained#item1
[7] https://www.gsma.com/futurenetworks/technology/volte/
[8] https://searchnetworking.techtarget.com/definition/vEPC-virtual-Evolved-Packet-Core
[9] https://searchnetworking.techtarget.com/definition/cloud-radio-access-network-C-RAN
[10] https://www.a10networks.com/blog/gi-lan-functions-5g/
[11] https://www.sdxcentral.com/networking/nfv/definitions/nfv-orchestrator-nfvo-definition/

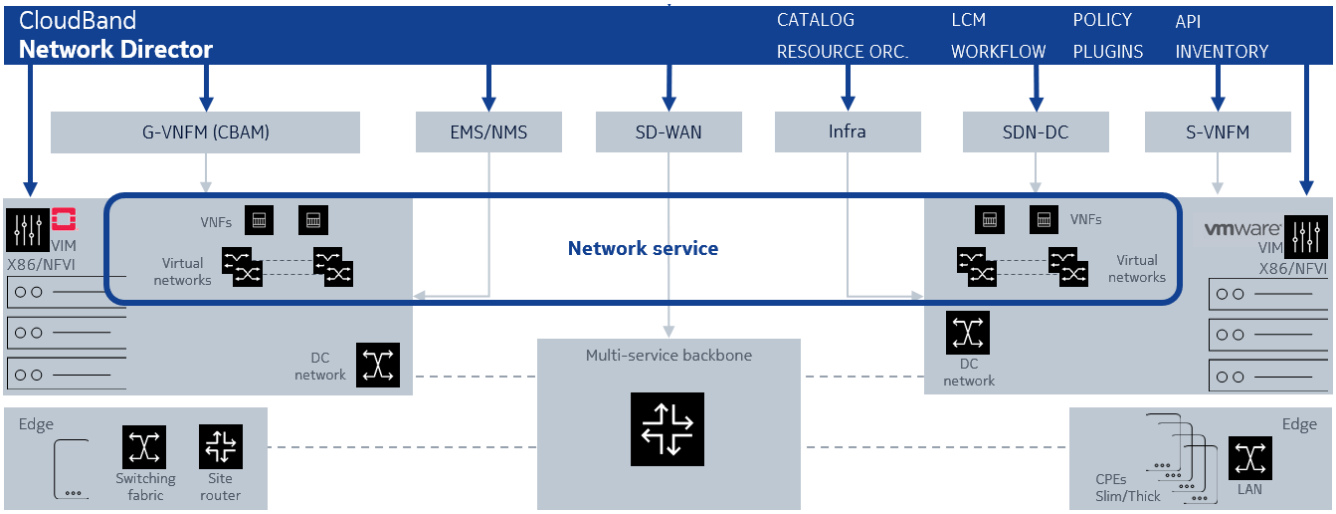


Figure 1: Cloudband Network Director High Level Architecture

3.2.2 Nokia CBND Main Features

CBND main features include Service Lifecycle Management using CLA (Closed Loop Automation) and VNF package management, Performance and Self-Monitoring, Notifications and placement capabilities based on a policy engine.

Figure 2: CBND Main Features

The screenshot below (Figure 3) shows an example of the CBND dashboard giving in a single screen the status of the network services, virtual and physical resources.

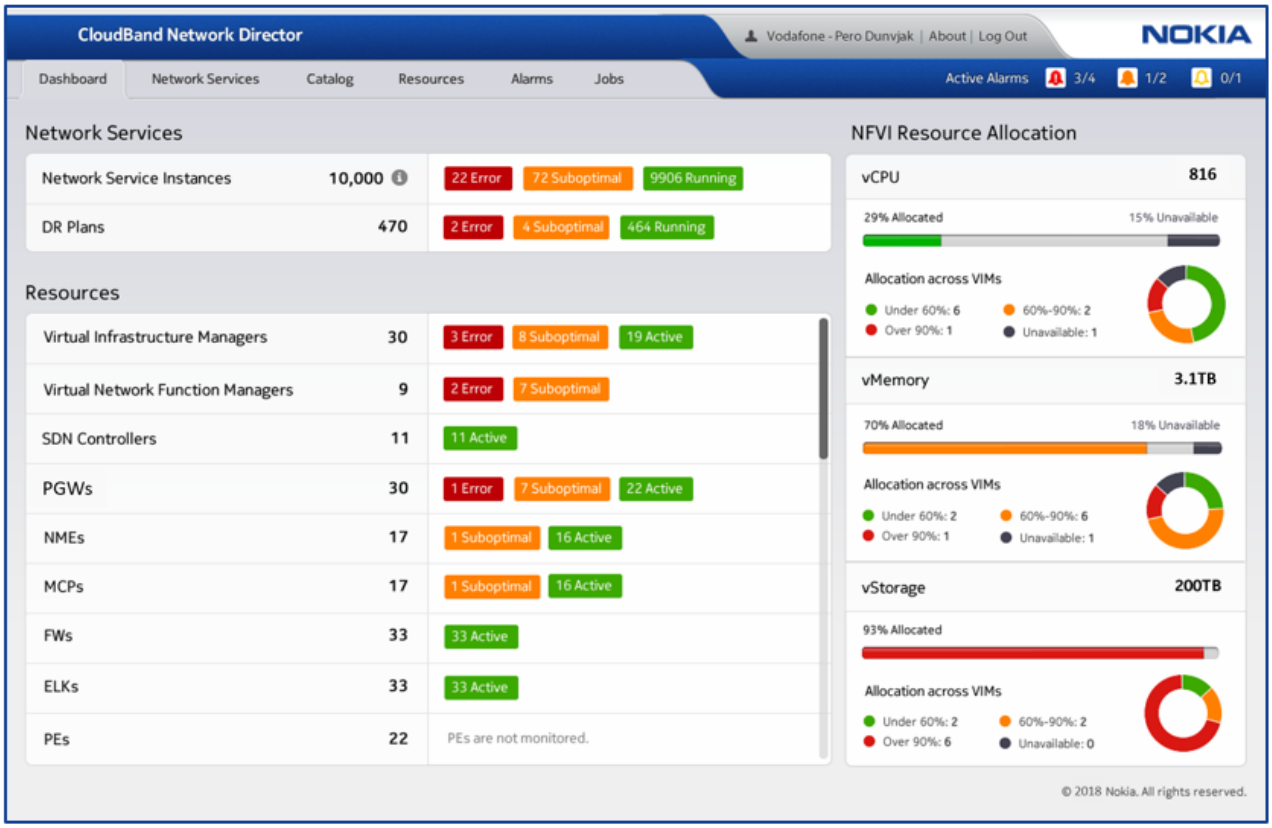


Figure 3: CBND Dashboard

3.2.3 Definition of a Network Service in CBND

Part of a service definition is the workflow defined for the services and implemented in CBND via an Openstack Mistral workflow. The screen below demonstrates a sample workflow together with the status of execution to allow debugging the workflow. In 5G-SOLUTIONS we may need to create custom workflows to accommodate for specific plugin needs. This will be determined in later stages of the project.

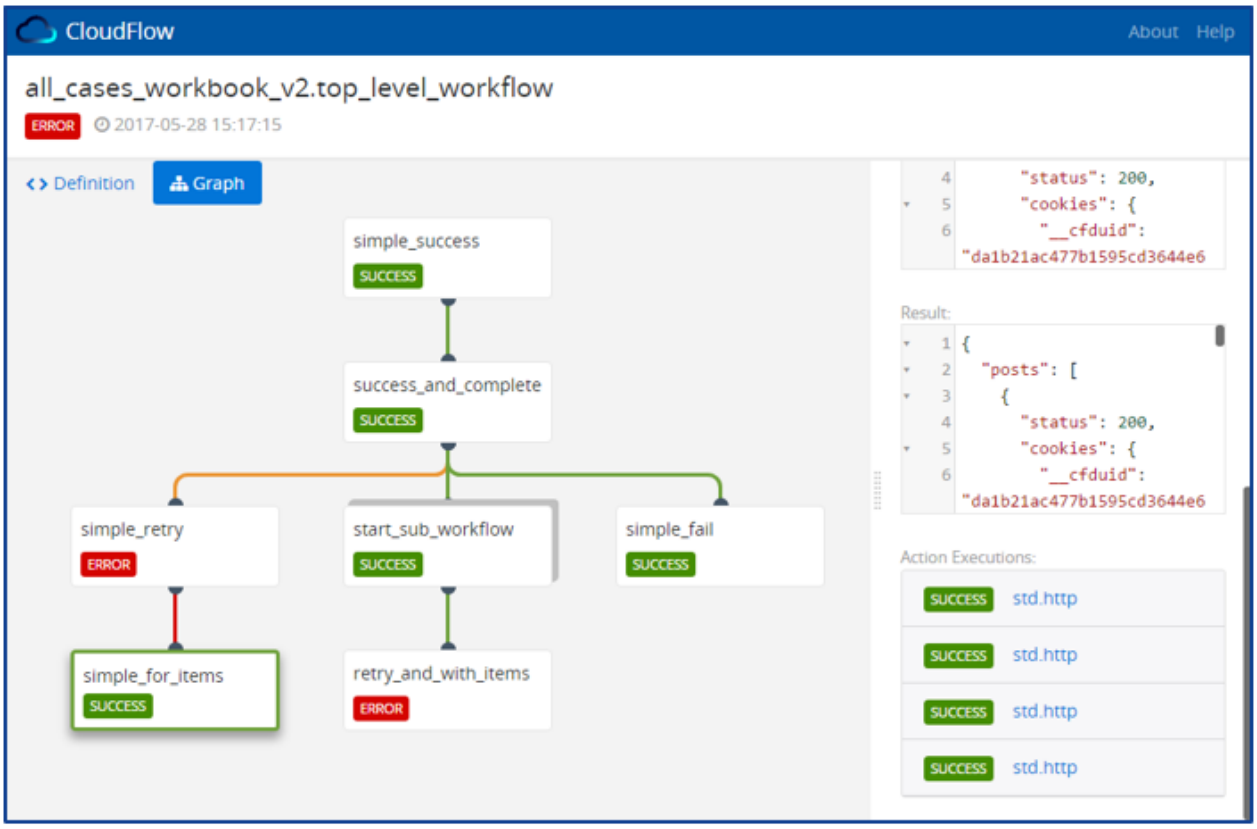


Figure 4: CBND Workflow

To define a Network Service, there is a need to create a "Network Service Descriptor (NSD)". The VNFFG (at the bottom of Figure 5) demonstrates the flow of the network service by illustrating the VNFs and how they are interconnected. The NSD contains information as listed below:

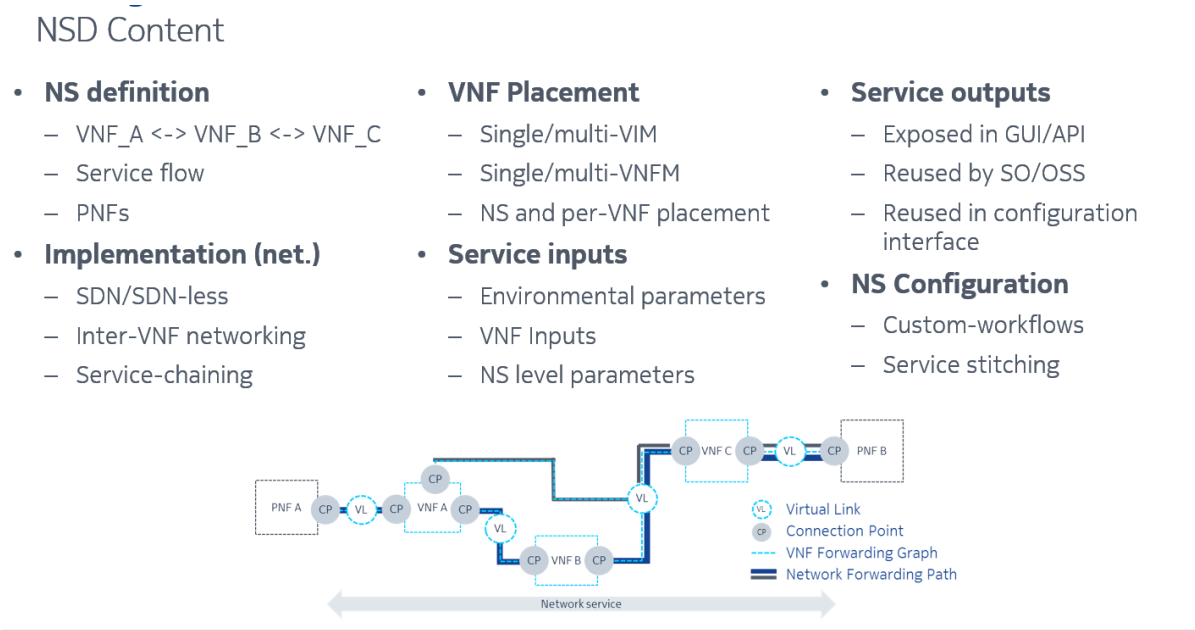


Figure 5: CBND NSD Content & VNFFG

3.2.4 Installation of CBND

This section provides information on the requirements for installing CBND, including sizing information, HW and virtual resource requirements, network requirements and other prerequisites. This is needed to make sure we start the installation after having all needed resources, so the installation is successful.

The step by step installation instructions are available to Nokia customers (as CBND is a commercial product) and will be provided by Nokia for the actual installation process in 5G-SOLUTIONS.

3.2.4.1 Virtual infrastructure requirements

CBND officially supports installation on OpenStack >= Queens only.

Table 2 Virtual infrastructure requirements lists the virtual infrastructure runtime environment requirements for CBND installation.

Non - HA

Table 2: *Virtual Infrastructure Requirements for non-HA CBND*

	Virtual resource	Required configuration
CBND VM	vCPUs	12
	RAM	33 GB
	Storage	100 GB
CBND Volumes	MariaDB	200 GB
	Varlog	50 GB
	Backup	40 GB
	ELK	50GB
	Metrics	16 GB

HA

Table 3: Virtual Infrastructure Requirements for HA CBND

	Virtual resource	Required configuration (per VM)	Cumulative configuration (3 VMs)
CBND VM	vCPUs	12	36
	RAM	33 GB	99 GB
	Storage	100 GB	300 GB
CBND Volumes	MariaDB	200 GB	600 GB
	varlog	50 GB	150 GB

	Backup	40 GB	120 GB
	ELK	50GB	150 GB
	Metrics	16 GB	48 GB

CPU pinning for CBND VM is enabled by default.

3.2.4.2 Networking requirements

CBND Installation is supported on both IPv4 & IPv6 networks. The required networks have to be pre-created. The configured subnets on the hosted networks of CBND VM and Installer VM should have IP and network requirements as per the following tables.

Table 4: IPV4 Networking Requirements

Installation on IPv4	No. of networks	IPs required (Non-HA)	IPs required (HA)	Network Details	Mandatory
Installer VM	2/1	2/1	2/1	External Provider network- IPv4	Yes
				Internal/External network- IPv4	No
CBND VM	2	4	8	External Provider network- IPv4	Yes
				Internal/External network- IPv4	Yes

Table 5: IPv6 networking requirements

Installation on IPv6	No. of networks	IPs required (Non-HA)	IPs required (HA)	Network Details	Mandatory
Installer VM	2	2	2	External Provider network- IPv6	Yes
				Internal/External network- IPv4	Yes
CBND VM	2	4	8	External Provider network- IPv6	Yes
				Internal/External network- IPv4	Yes

Prerequisites for CBND networking setup

- The Network where CBND is planned to be installed should be a non-NAT-ed network.
- Installer VM should be able to access the CBND VM's hosted network.
- In the CBND HA environment, both the external and internal networks should have the multicast support enabled.

3.2.5 CBND Deployment

It was agreed with 5G-VINNI that the Nokia CBND (5G-SOLUTIONS CSDO) will be installed on the 5G-VINNI instance in Oslo which is based on Nokia CBIS [12]and supports smooth deployment of CBND 19.5

Currently there is already an instance of Nokia CBND in this facility acting as a MANO NFVO for 5G-VINNI[13].

This new instance will be installed on the same environment but will have a totally different role – it will act as a cross-domain service orchestrator with slice management capabilities.

3.3 Integrating the Orchestrator

The following section describes mechanisms in Nokia CBND that allow to extend and configure the functionality to meet 5G-SOLUTIONS orchestration requirements.

3.3.1 The Plugin Model

CBND follows an Open Architecture where external systems like VIM, VNFM, SDN Controller and so on, can be integrated. Integration of these external systems into CBND happens through Plugins. CBND supports Standard Plugin Types for which Plugins can be developed.

3.3.2 Plugin Types

There are several built-in resource plugin types and custom resource types can be added using a plugin mechanism.

Commonly used plugin types are:

- VIM (Virtual Infrastructure Manager) – Commonly used as a deployment target for network services. The virtual resources of the VNFs defined inside the NS will be deployed on the VIM by the VNFM. Another common example for VIM is to act as the target for resource orchestration examples such as “upload VM image” or “create Heat stack”. Examples of plugins of type VIM can be OpenStack Rocky, OpenStack Stein, VMWare, etc.
- VNFM (Virtual Network Function Manager) – Commonly used as a deployment target for network services. The VNFs defined inside the NS will be managed by the VNFM. Examples of plugins of type VNFM can be CloudBand Application Manager, F5 VNFM, etc.
- SDN (Software Defined Networks) Controller – Commonly used as a deployment target for network services. The network resources (networks, subnets, etc.) that are defined inside the NS will be deployed on the SDN Controller. Examples of plugins of type SDN can be OpenStack Neutron, Nokia Nuage, OpenDaylight, etc.

CBND can also support the integration of Non-Standard Plugin types, which can be used to carry out certain operations. Note that non-standard plugin type integration into CBND will need data communication path to

[12] https://www.nokia.com/networks/products/cloudband-infrastructure-software/
[13] https://www.5g-vinni.eu/norway-main-facility-site/

be established between CBND Components and Plugin. As CBND is workflow -driven, there is a definite possibility to establish this communication path by using custom workflows.

3.3.3 Steps in implementing a CBND Plugin

3.3.3.1 Develop plugin

Plugin communication with CBND is done in two ways:

1. **Asynchronous Communication:** Used to send Asynchronous request between CBND Components and Plugins. Usually POST/PUT/DELETE operations are Asynchronous. Plugins need to listen to an AMQP exchange in order to get the Request. Once the request is processed the response is sent back to the caller (CBND Component) by putting a message into the AMQP Exchange.
2. **Synchronous Communication:** Used for Synchronous requests. Usually GET requests go via Synchronous method. In this type of communication, CBND Components invoke REST API of the Plugins. The response in the same thread. Standard Plugin Types of CBND have an NBI API which is used by CBND Components.

Plugins can be written in any language that supports RabbitMQ and HTTP. A plugin may include specific code or only configuration information. We have included 2 examples of plugin implementations in Annex A.

3.3.3.2 Packaging a Plugin

A Plugin package will contain the following artifacts. Some of these artifacts are optional, as mentioned below:

a. Plugin Packages: This is mandatory and will contain the Plugin logic.
b. Administration Template for Plugins: This is an XML file mandatory for each Plugin.
c. Custom TOSCA Entities: Custom TOSCA Entities, if developed as part of a Plugin need to be uploaded to CBND These TOSCA Entities are optional.
d. Custom Workflows: These are Custom Workflows developed as part of the Plugin.

3.3.3.3 Deploying a Plugin

Plugins can be deployed anywhere if they have communication via the external and internal networks of CBND (as defined in 3.2.4.2 above). As mentioned earlier, plugins can be written in several languages. However, Java and Python are the most common.

3.4 Registering Resources

CBND is also a Resource Orchestrator which allows you to manage and monitor resources. The definition of a resource for CBND is determined by the plugin type (VIM, VNFM, SDN Controller) and the VIM plugin (OpenStack, VMWare).

Resources are used for several use cases such as:

a. Target for a network service deployment (e.g., deploy NS in disaster recovery mode on VIM1+VNFM1+SDN1 and VIM2+VNFM2+SDN2).
b. Use resource status to influence network service status (for example, if VNFM is in ERROR, mark the VNFs and NSs as SUBOPTIMAL).

c. Targets for artifact execution (for example, create a flavor on multiple VIMs at once) – a "Flavor" is a template for a VM including the #of CPUs, the amount of memory etc.

The following figures show a listing of VIM resources as registered in CBND and the UI to define a new VIM instance. This is the very 1st step in defining and registering resources in CBND, after which one can define a network services using these resources.

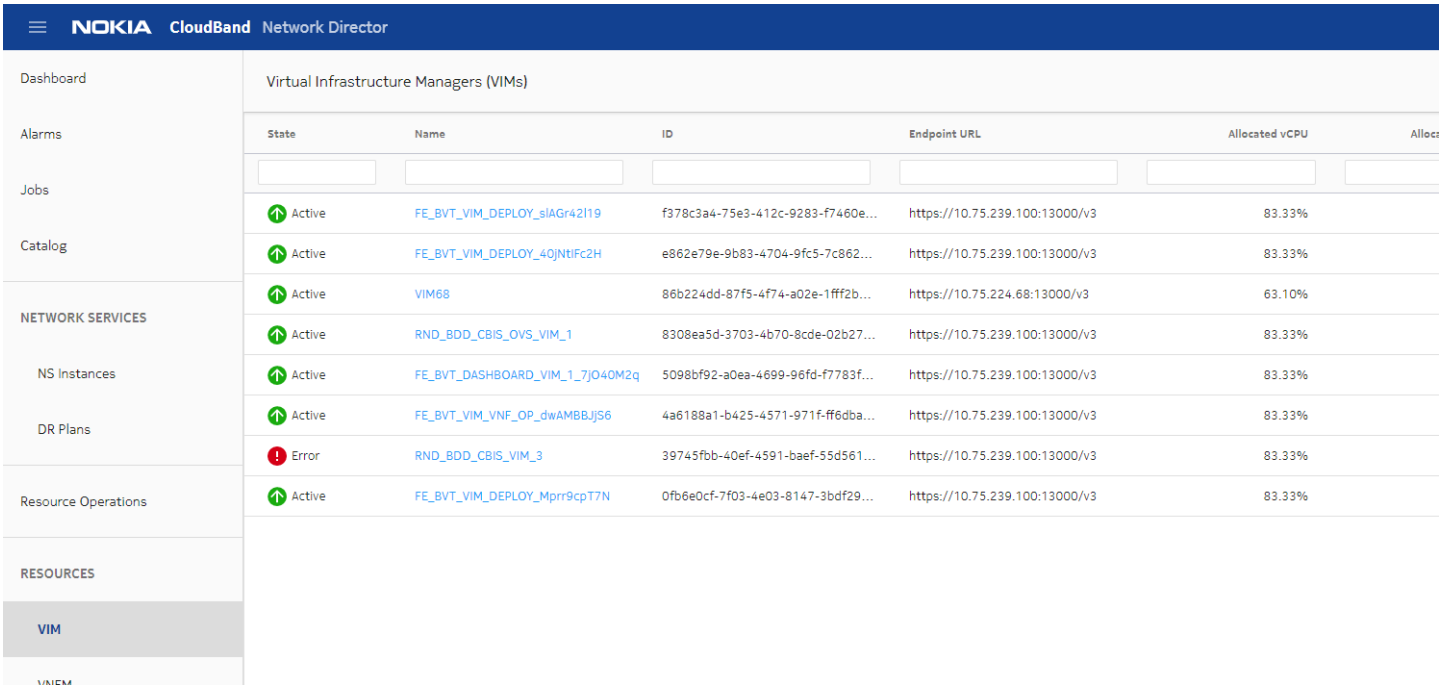


Figure 6: CBND List of VIMs

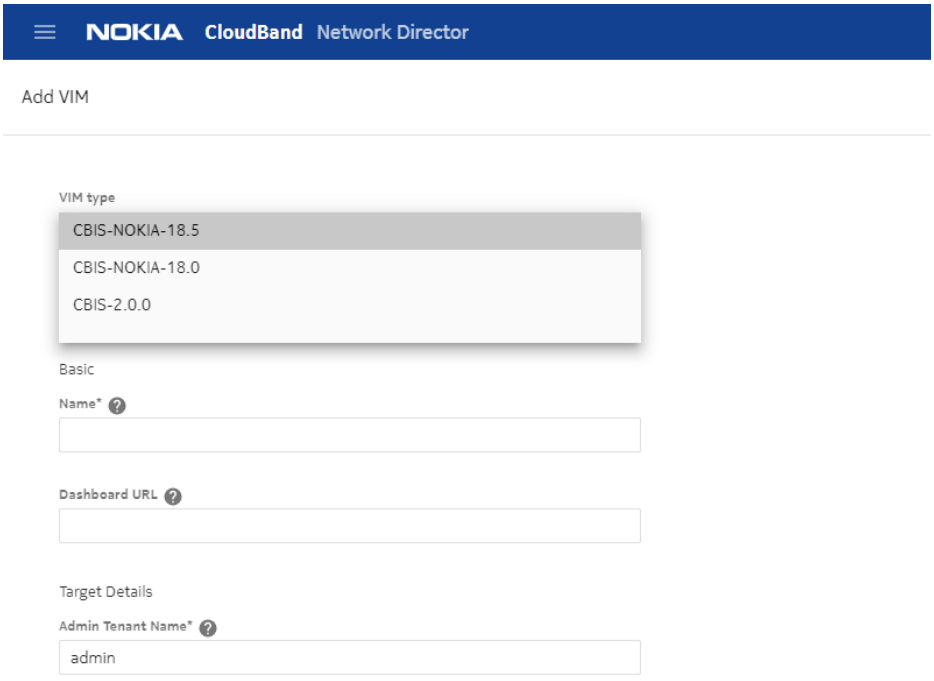


Figure 7: CBND: Add VIM

4 Integrating with 5G-SOLUTIONS Living Labs

4.1 Common Platform Interface: Interfacing with ICT-17 Facilities as Gateway to Living Labs

Many use cases are hosted in ICT-17 facilities and orchestrated by the ICT-17 orchestrator. These use cases (identified as Category A) will be integrated into the ICT-17 platform and onboarded to the platform orchestration system and 5G-SOLUTIONS will interact with the ICT-17 Orchestrator to trigger the service lifecycle actions which will then be executed by the ICT-17 orchestrator. The figure below demonstrates the interaction from the 5G-SOLUTIONS CDSO to ICT-17 5G-VINNI facility in the University of Patras in Greece.

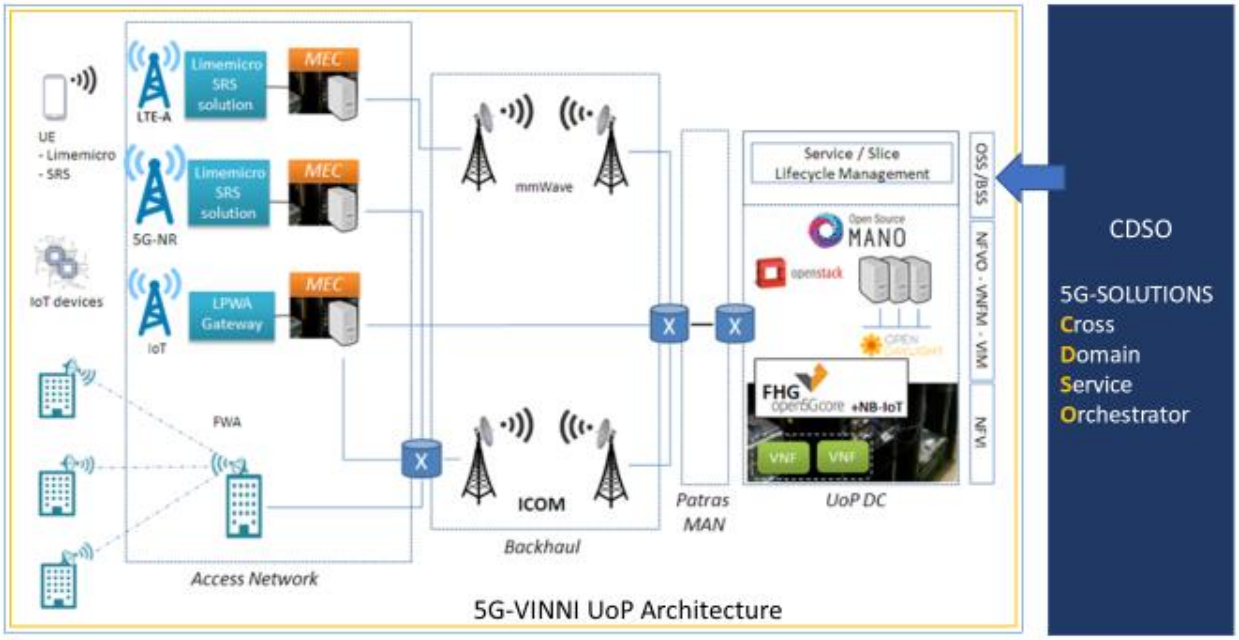


Figure 8: 5G-VINNI UoP Facility Architecture & Main Interfaces

There are 3 ICT-17 facilities which we plan to integrate with, as listed below and analysed in the following sections:

- 5G-VINNI University of Patras (UoP)
- 5G-VINNI Oslo
- 5G-EVE Turin

4.1.1 5G-VINNI University of Patras (UoP)

The UoP 5G-VINNI facility orchestrator is based on Open Source MANO [14]and exposes 2 types of interfaces:

1) OSM interface (as defined in https://osm.etsi.org/wikipub/index.php/NBI_API_Description); waiting for confirmation of OSM version supported from 5G-VINNI
2) TMF Defined Service Management APIs (TMF633 Service Catalogue Management API[15], TMF641 Service Ordering[16])

[14] https://osm.etsi.org/wikipub/index.php/OSM_Release_FIVE
[15] https://www.tmforum.org/resources/specification/tmf633-service-catalog-management-api-rest-specification-r16-5-1/
[16] https://www.tmforum.org/resources/specification/tmf641-service-ordering-api-rest-specification-r18-5-0/

It is our preference to use the TMF based APIs, as these are better supported by available tools and documentation. It is assumed that the provided interfaces are available and stable.

Integration with UoP is implemented by accessing Openslice (https://openslice.io) which is implementing TMF APIs as described above.

The first step in the integration is the creation of a service entry on the Openslice catalog. This will be done offline through the Openslice UI and will be a precondition to running the experiment on UoP.

The service defined includes also the slice definition and parameters and may require manual approval by the administrator. Some of the slice parameters are defined as updatable by the user and these will be offered to the user to override when actually running the experiment.

5G-VINNI defined a Test Manager component that is responsible to start/stop KPI collection during the experiment from the 5G components. At this point details of this component as well as the specific API has not yet been published by 5G-VINNI thus will not be included in Cycle I of the testing.

4.1.2 5G-VINNI Oslo

The Oslo 5G-VINNI facility orchestrator is based on Nokia FlowOne service orchestrator. The API available for us to use is planned to be the TMF Defined Service Management APIs TMF641 [17] and TMF633[18] (which is exposed to support inter-site connectivity for 5G-VINNI).

The CDSO, based on Nokia's CBND is installed on a VM in the 5G-VINNI Oslo facility. This is an additional instance of CBND and is used as an end-to-end orchestrator for 5G-SOLUTIONS and not as an NVFO. This instance of CBND will be the only instance of the CDSO thus it will have to communicate with all other facilities and all other use cases requiring orchestration services. As such, we expect that some of these connections may require a VPN connection which will have to be set up from the VM in 5G-VINNI Oslo to the other facilities. The first testbed to check this will be the connection to 5G-VINNI in UoP.

4.1.3 5G-EVE Turin

The Turin 5G-EVE facility orchestrator is based on Open Source MANO. It is still not clear exactly which interfaces will be made available. 5G-EVE is currently considering how to allow interfacing from other projects to the 5G-EVE environment. At this point, it is clear that:

- 5G-EVE exposes a UI to request onboarding of an experiment (service)
- Once such request is submitted, 5G-EVE personnel validate and actually do the onboarding
- 5G-EVE exposes a UI to execute the experiment (start, stop)

Currently what is considered is to allow the third step via a proprietary API. This section will be expanded as 5G-VINNI and 5G-EVE become ready for interacting with ICT-19 projects.

4.2 Integration with the KPI Visualization System

The KPI Visualization system subscribes to notifications with CDSO. The subscription is done once for all experiments. The Visualization system may also unsubscribe from CDSO notifications when it is shut down. The following table describes the flow:

[17] https://www.tmforum.org/resources/specification/tmf641-service-ordering-api-rest-specification-r18-5-0/
[18] https://www.tmforum.org/resources/specification/tmf633-service-catalog-management-api-rest-specification-r16-5-1/

Table 6: Integration flow between CDSO and KPI Visualisation System

	Step	Interface with	API	Parameters	Comments
1	Visualization System (VS) subscribes to experiment events	VS to CDSO			
1.1	VS creates auth token to CDSO	CDSO	OAUTH2	Credentials	Syntax Below
1.2	VS subscribes to CDSO Notifications	CDSO	LifeCycle Change Notification	Callback URL and optional basic auth credentials	See below syntax details See the Notification syntax
2	Visualization system (VS) unsubscribes from experiment events				
2.1	Visualization System (VS) Authenticates with CDSO	VS to CDSO	OAUTH2	Credentials	Syntax Below
2.2	Visualization System (VS) unsubscribes from experiment events	VS to CDSO	OAUTH2	Credentials	Syntax Below

4.3 Use Case 1.1: Time-critical process optimization inside digital factories

For more details on all use cases, please follow sections 5.1.x.y (where 'x' indicates the Living Lab and 'y' the Use Case within that Living Lab) of D1.1A "Definition and analysis of use cases/scenarios and corresponding KPIs based on LLs (v1.0)".

4.3.1 Use Case Overview

This use case's main focus is leveraging 5G technology to enable a real time product and process control. See Section 5.1.1 in D1.1A for a full description of the use case.

Table 7: Use Case 1.1: Time-critical process optimization inside digital factories

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Time-critical process optimization inside digital factories	P&G	
Use Case Number	1.1		

LL	1	IRIS	
ICT-17 Project	5G-VINNI		
Facility	Patras	UoP	Actual location in P&G in Belgium
Category	C		5G-CORE may be on UoP Facility. At this point the 5G network is implemented locally in a box provided by UoP but manually configured and standalone.
Orchestration	Minimal orchestration automation needed. Most setup is manual in P&G	UoP	Not yet clear if the 5G-CORE component will reside on site in P&G or in the 5G-VINNI/UoP facility. This will also have implications on how the orchestration will be performed.

4.3.2 Existing Integration Points

5G-VINNI/UoP interfaces (see section 4.1.1 above). If needed to orchestrate directly in P&G site, may need to "onboard" the service to the CDSO directly.

4.4 Use Case 1.2: Non-time-critical communication inside factories

4.4.1 Use Case Overview

This Use Case will focus on the 5G technology as enabler for multi IoT sensors data gathering to perform smart monitoring inside the factory facility. See Section 5.1.2 in D1.1A for a full description of the use case.

Table 8: Use Case 1.2: Non-time-critical communication inside factories

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Non-Time-critical process optimization inside digital factories	Glanbia	
Use Case Number	1.2		
LL	1	IRIS	
ICT 17 Project	5G-VINNI		
Facility	Patras	UoP	Actual location in Glanbia factory in Ireland
Category	C		5G-CORE may be on UoP Facility. At this point the 5G network is implemented locally in a box provided by UoP but manually configured and standalone.
Orchestration	Minimal orchestration automation needed. Most setup is manual in Glanbia	UoP	Not yet clear if the 5G-CORE component will reside on site in Glanbia or in 5G-VINNI/UoP facility. This will also have implications on how the orchestration will be

			performed.

4.4.2 Existing Integration Points

5G-VINNI/UoP interfaces (see section 4.1.1 above). If needed to orchestrate directly in Glanbia site, may need to "onboard" the service to the CDSO directly.

4.5 Use Case 1.3: Remotely Controlling Digital Factories

4.5.1 Use Case Overview

While in both UC1.1 and UC1.2, local on-site communication is assumed within the plant, in the case of UC1.3, the public wireless access network also plays a role in the E2E communication between remote workers and the factory. See Section 5.1.3 in D1.1A for a full description of the use case.

Table 9: Use Case 1.3: Remotely Controlling Digital Factories

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Remotely Controlling Digital Factories		
Use Case Number	1.3	NTNU	
LL	1	IRIS	
ICT 17 Project	5G-VINNI	TLNR	
Facility	Oslo	TLNR	Actual location in Trodenheim
Category	TBD		Still not clear if CDSO integration is required
Orchestration	None for Cycle 1. Still in discussion for later stages	Vinni, Oslo	

4.5.2 Existing Integration Points

TBD

4.6 Use Case 1.4: Connected Goods

This Use Case will assess the potential for 5G to enable full-duplex communication of appliances such as washing machine in tomorrow's home. See Section 5.1.4 in D1.1A for a full description of the use case. The setup and flow of the use case is still not defined well enough at this point in time to discuss its orchestration requirements.

4.7 Use Case 1.5: Rapid deployment, auto/re-configuration, testing of new robots

4.7.1 Use Case Overview

UC1.5 focuses on the possibilities of utilizing the core functionalities of 5G in order to achieve one major feature of the Industry 4.0 in the Factories of the Future (FoF), such as significantly lower expenses and reduced time in order to either commission or reconfigure robotized manufacturing. See Section 5.1.5 in D1.1A for a full description of the use case.

Table 10: Use Case 1.5: Rapid deployment, auto/re-configuration, testing of new robots

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Rapid deployment, auto/re-configuration, testing of new robots	NTNU	Early testing in Q2/2020
Use Case Number	1.5		
LL	1	IRIS	
ICT 17 Project	5G-VINNI		
Facility	Norway	TLNR	
Category	A		
Orchestration	5G-VINNI Norway will set up the network connection and interworking with CDSO.		

4.7.2 Existing Integration Points

No integration with CDSO is planned for cycle I.

4.8 Use Case 2.1: Industrial Demand Side Management

4.8.1 Use Case Overview

This use case deals with DSM at the level of business consumers (large, medium or small enterprises). The focus is on the optimal scheduling of industrial loads during normal plant operation, the computation and actuation of flexibilities offered on the Dispatching Market and the control actions needed to keep the peak power consumption limited. See Section 5.2.1 in D1.1A for a full description of the use case.

4.8.2 Existing Integration Points

Table 11: Use Case 2.1: Industrial Demand Side Management

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Industrial Demand Side Management	IREN	Planned for 2021
Use Case Number	2.1		
LL	2 (Smart Energy)	A2T	
ICT 17 Project	5G-EVE	TIM	
Facility	Turin		

Category	A		
Orchestration	5G-EVE	TIM	

4.8.3 Existing Integration Points

No integration with CDSO is planned for cycle I

4.9 Use Case 2.2: Electrical Vehicle Smart Charging

4.9.1 Use Case Overview

Smart charging is a form of DSM where the selection of the charging station used for recharging a PEV (Plugin Electric Vehicles) and the power absorbed during the recharging process are under the control of a charging point operator, according to a set of grid boundary conditions and drivers' charging preferences. See Section 5.2.2 in D1.1A for a full description of the use case.

Table 12: Use Case 2.2: Electrical Vehicle Smart Charging

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Electrical Vehicle Smart Charging	ENEL/IREN	Planned for 2021
Use Case Number	2.2		
LL	2 (Smart Energy)	A2T	
ICT 17 Project	5G-EVE	TIM	
Facility	Turin		
Category	A		
Orchestration	5G-EVE	TIM	

4.9.2 Existing Integration Points

No integration with CDSO is planned for cycle I

4.10 Use Case 2.3: Electricity network frequency stability

4.10.1 Use Case Overview

This use case deals with the involvement of active demand in the provisioning of the electricity services needed to maintain the electricity network frequency at its rated value (e.g., 50 Hz in Europe). See Section 5.2.3 in D1.1A for a full description of the use case.

Table 13: Use Case 2.3: Electricity network frequency stability

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Electricity network frequency stability	ENEL	Planned for 2021
Use Case Number	2.3		
LL	2 (Smart Energy)	A2T	
ICT 17 Project	5G-EVE	TIM	
Facility	Turin		
Orchestration	5G-EVE	TIM	

4.10.2 Existing Integration Points

No integration with CDSO is planned for cycle I

4.11 Use Case 3.1: Intelligent Street Lighting

4.11.1 Use Case Overview

UC 3.1 focuses on virtualizing intelligent street lighting where the sensors can detect the activities and only use the energy when there is a need. See Section 5.3.1 in D1.1A for a full description of the use case.

Table 14: Use Case 3.1: Intelligent Street Lighting

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Intelligent Street Lighting	NURO	
Use Case Number	3.1		
LL	3 (Smart Cities and Ports Living lab)	IBM	
ICT 17 Project	5G-VINNI	TLNR	
Facility	Oslo	TLNR	
Category	A		Detailed Orchestration Requirements still TBD. Defined as "best effort" for late Cycle I
Orchestration	TBD		

4.11.2 Existing Integration Points

integration with CDSO is for cycle I is defined as "best effort". Until this point, the UC orchestration flow was still ongoing work, thus it could not be documented. If progress is made in integrating this use case in cycle I, we will document it in the next version of this report.

4.12 Use Case 3.2: Smart Parking

In the Smart parking use case, we plan to use a mobile app to book a parking spot for a car arriving in the city and a sensor on the parking location to indicate the availability of the parking spot to other users. See Section 5.3.2 in D1.1A for a full description of the use case.

4.12.1 Use Case Overview

Table 15: Use Case 3.2: Smart Parking

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Smart Parking	IBM	Defined as "best effort" for late Cycle I
Use Case Number	3.2		
LL	3 (Smart Cities and Ports	IBM	

	Living lab)		
ICT 17 Project	5G-VINNI	TLNR	
Facility	Oslo	TLNR	
Category	A		Orchestration Requirements still TBD
Orchestration	TBD		

4.12.2 Existing Integration Points

Integration with CDSO is for cycle I is defined as "best effort". Until this point, the UC orchestration flow was still ongoing work, thus it could not be documented. If progress is made in integrating this use case in cycle I, we will document it in the next version of this report.

4.13 Use Case 3.3: Smart City Co-Creation

4.13.1 Use Case Overview

The target and ambition of this use case is to enable an explorative and wider approach to Smart City/IoT by facilitating for co-creation and exploring by external collaborators into 5G and advanced networking and experimental platform capabilities. See Section 5.3.3 in D1.1A for a full description of the use case.

Table 16: Use Case 3.3: Smart City Co-Creation

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Smart City Co-Creation	NTNU	
Use Case Number	3.3		
LL	3 (Smart Cities and Ports Living lab)	IBM	
ICT 17 Project	5G-VINNI	TLNR	
Facility	Oslo	TLNR	
Category	A		Orchestration Requirements still TBD
Orchestration	TBD		

4.13.2 Existing Integration Points

No integration with CDSO is planned for cycle I.

4.14 Use Case 3.4: Smart buildings / Smart campus

UC 3.4 focuses on leveraging 5G to accurately locate IoT devices, find the density of nodes required to cover a specific building, explore the architecture and delay of mapping IoT devices in a building. See Section 5.3.4 in D1.1A for a full description of the use case.

4.14.1 Use Case Overview

Table 17: Use Case 3.4: Smart buildings / Smart campus

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Smart buildings / Smart campus	IBM	
Use Case Number	3.4		
LL	3 (Smart Cities and Ports Living lab)	IBM	
ICT 17 Project	5G-VINNI	UoP	
Facility	Patras	UoP	
Category	A		
Orchestration	OSM/TMF		See 4.1.1 above

4.14.2 Existing Integration Points

No integration with CDSO is planned for cycle I.

4.15 Use Case 3.5: Autonomous Assets and Logistics for Smart Port

4.15.1 Use Case Overview

The smart port UC is focused on automation of container transport and migration to zero-emission transport. See Section 5.3.5 in D1.1A for a full description of the use case.

Table 18: Use Case 3.5: Autonomous assets and logistics for smart harbor/port

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Autonomous Assets and Logistics for Smart Port	YARA, NTNU	Starting Q2/2020
Use Case Number	3.5		
LL	3 (Smart Cities and Ports Living lab)	IBM	
ICT 17 Project	5G-VINNI	TLNR	
Facility	Oslo	TLNR	
Category	A		
Orchestration			See 4.1.1 above

4.15.2 Existing Integration Points

Integration with CDSO is for cycle I is defined as "best effort". Until this point, the UC orchestration flow was still ongoing work, thus it could not be documented. If progress is made in integrating this use case in cycle I, we will document it in the next version of this report.

4.16 Use Case 3.6: Port Safety: monitor & detect irregular sounds

4.16.1 Use Case Overview

The UC focuses on detection, analysis and transmission of notifications triggered by gut-shot sounds and other irregular sounds that could be considered a hazard. See Section 5.3.6 in D1.1A for a full description of the use case.

Table 19: Use Case 3.6: Port Safety: monitor & detect irregular sounds

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Port Safety: monitor & detect irregular sounds	NURO	Early testing planned for Q2/2020
Use Case Number	3.6		
LL	3 (Smart Cities and Ports Living lab)	IBM	
ICT 17 Project	5G-VINNI	TLNR	
Facility	Oslo	TLNR	
Category	A		
Orchestration			See 4.1.1 above

4.16.2 Existing Integration Points

No integration with CDSO is planned for cycle I.

4.17 Use Case 4.1: Ultra High-Fidelity Media

4.17.1 Use Case Overview

UC 4.1 focuses on measuring latency and performance in mass distribution of UHFM (Ultra High-Fidelity Media) digital media over 5G networks. See Section 5.4.1 in D1.1A for a full description of the use case.

Table 20: Use Case 4.1: Ultra High-Fidelity Media

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Ultra-High-Fidelity Media	FNET	Planned for Cycle I, Started deployment of FNET servers in UoP
Use Case Number	4.1		
LL	4 (Media & Entertainment Living lab)	LiveU	
ICT 17 Project	5G-VINNI	UoP	
Facility	Patras	UoP	
Category	A		
Orchestration	OSM/TMF/Openslice		See 4.1.1 above

4.17.2 Existing Integration Points

The Orchestration of UC 4.1 is done through integration of CDSO with Openslice on UoP. The FNET application server is deployed in UoP as a VNF and activated by the UoP MANO orchestrator. The FNET servers expose an SNMP interface used in UoP to control the application server.

The Network Slice used for the experiment is defined in the service catalog on Openslice. Some of the slice parameters can be also set by the experiment owner from the outside, thus they will be exposed by CDSO to the experiment owner in a later stage of the integration process.

The UC High level architecture is described in the following diagram:

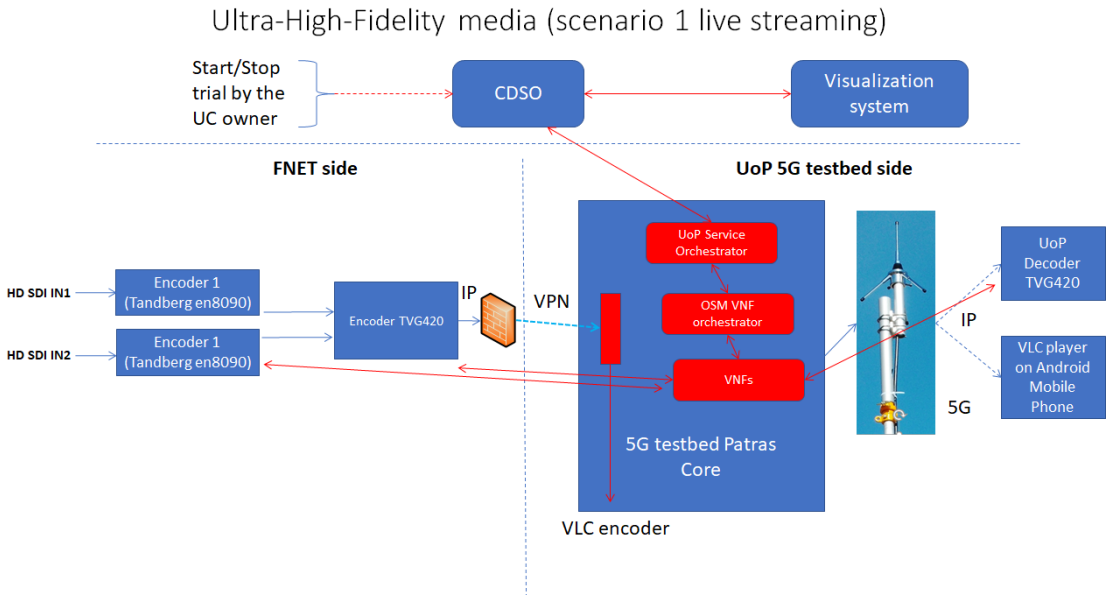


Figure 9: High level architecture of UC4.2

Note: The test manager is not described in the flow since it is not installed currently in UoP.

The flow of UC4.1 (and a similar sequence from CDSO perspective is valid for UC4.4 & UC4.6) is split to 2 parts:

1) Service specification – this part is done offline on the UoP orchestrator via Openslice interface. The following sequence diagram illustrates this flow:

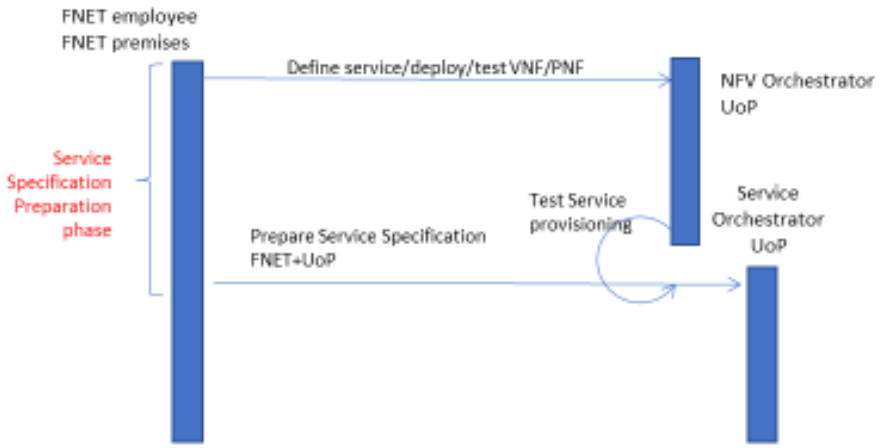
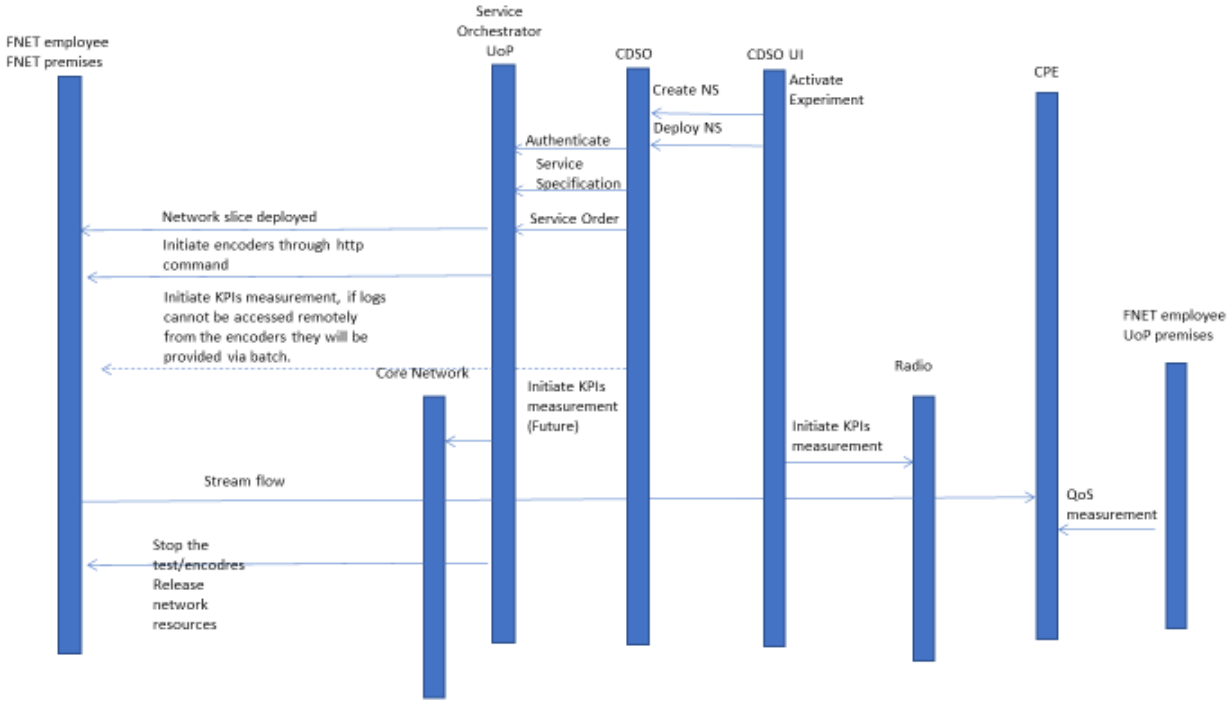


Figure 10: UC4.1 Service Specification Flow

2) Service Activation – this part is done for each running of the experiment.
3) The following sequence diagram shows the flow of control in UC4.1 activation.

Figure 11: Flow of control in UC4.1



The following table describes the step-by-step integration flow between the CDSO and the UoP where the UC is deployed:

The proposed flow assuming that the end-to-end infrastructure is established is:

Table 21: Integration flow between the CDSO and the UoP

	Step	Interface with	API	Parameters	Comments
1	User activates experiment	UI/API to CDSO			
1.1	Activate experiment on CDSO (API) – Create NS	UI/API to CDSO	NS LCM	Slice ID, UseCase ID, …	The CDSO provides the interface to the UC owner to provide information about the slice parameters and the video stream activation. If needed, separate order will be given by the UC owner to configure the slice and activate the streaming. See below syntax details Notification sent – see here
1.2	Activate experiment on CDSO (API) – Deploy NS	UI/API to CDSO	NS LCM	Slice ID, Use Case ID, …	See below syntax details Notification sent – see here
1.3	Create connection from CDSO to UoP & Authenticate	UoP Orchestrator	OAUTH2	Credentials	See below syntax details
1.4	Slice Configuration	UoP Orchestrator		Slice Parameters	This will be a Service Order configuration. We assume that a Service Specification already Exists and included in the order. (of course, a Service Specification could be done also programmatically but it needs a level of maturity of the use case) See below syntax details
1.5	Activate Slice	UoP Orchestrator		Slice ID	Here the Service Order is made. After a while, the status of the order can be requested (Completed or not) and the status of Services (ACTIVATED, etc.) See below syntax details
1.6	Activate encoders 2x Tandberg en8090, TVG420. In addition, at a later a TVG420 decoder in UoP premises will be	No interface to orchestrator. The process is managed by the UoP		Credentials	Equipment: Encoders in FNET premises, Decoder in UoP premises Note: A VLC encoder is also installed in UoP premises. This will be continuously operational and there no need to be

	activated	service orchestrator .			orchestrated. Nevertheless, the visualization system will collect KPIs from the VLC encoder as well. In needed a VNF will be implemented to control the VLC encoder as well. Process: The UoP services orchestrator will request from the OSM VNF orchestrator to activate the respective VNF. Either one VNF per equipment will be implemented or one VNF addressing all equipment. Within VNF the appropriate SMNP calls will be made towards the respective equipment e.g. start, stop, etc.)
1.7	Activate Video Stream	No interface to orchestrator . The process is managed by the UoP service orchestrator .		Unit ID	This process is activated by the VNFs
1.8	Visualisation system activation, test start	Visualisation system, UoP Orchestrator		Credentials	When VNFs are activated according to the previous process the UoP service orchestrator informs CDSO that the infrastructure is setup. Then the CDSO informs the visualization system that it should start measuring the KPIs.
2	Ongoing Status				
2.1	Get Slice Status	UoP Orchestrator			**Service Order ID** must be known – syntax below. The **Service Order** includes the Services IDs that are instantiated. To get a Service ID use the Service Inventory API as described below.
2.2	Get streaming equipment status	UoP Orchestrator		Unit ID	Show status. The VNFs will get the equipment status. This information will be given to CDSO through the UoP service orchestrator.
3	User Terminates Experiment	UI/API to CDSO			

3.1	Terminate Network Service	UI/API to CDSO	NS LCM	Network Service ID	Syntax Below Notification sent – see here
3.2	Terminate Video Stream	UoP orchestrator			Then CDSO informs the UoP service orchestrator to terminate the streaming. The UoP services orchestrator sends the command to the OSM VNF orchestrator and the respective VNFs. The VNFs send the stop command to the equipment (via SNMP)
3.3	Terminate Slice	UoP Orchestrator		Slice ID	Then CDSO informs the UoP service orchestrator to terminate the slice.

4.17.3 Development work required in CDSO to support the integration

As part of the 4.1, 4.4 and 4.6 use cases, CDSO will develop the following:

1. Plugin for OpenSlice, used to interact with OpenSlice instances such as https://patras5g.eu/ and http://portal.openslice.io
    a. A Java web app that will be deploy on top of CDSO.
    b. Basic monitoring capability to show the OpenSlice instance status and warn if needed
    c. TOSCA types to model a service inside a Network Service Descriptor (NSD)
    d. Workflows to interact with OpenSlice (create service order, create and delete service, etc).
    e. Allow selection of an OpenSlice instance when deploying a Network Service that requires an OpenSlice service
2. For UC 4.4 & 4.6, a Plugin for LiveU, used to interact with LiveU server such as https://luc-staging.liveu.tv:<port_number>
    a. A Java web app that will be deploy on top of CDSO.
    b. Basic monitoring capability to show the LiveU server status and warn if needed
    c. TOSCA types to model LiveU server inside a Network Service Descriptor (NSD)
    d. Workflows to interact with LiveU (start and stop streaming, etc.)
    e. Allow selection of a LiveU instance when deploying a Network Service that requires such server.
3. Network Service package for each use case
    a. A network service package is a ZIP archive containing files in a standard structure.
    b. The package contains the Network Service Descriptor and additional files (workflows, scripts) that might be needed for the specific use case.
    c. By deploying and terminating the NS package, the end user will be able to start and stop the experiment for the specific use case.

4.18 Use Case 4.2: Multi CDN selection

4.18.1 Use Case Overview

The trials for this use case will explore the possibilities of providing the optimum QoS for users by means of a dynamic selection of the CDN cache where the user is receiving content or even the prediction of congestion and the use of edge computing resources in order to instantiate new caches on demand.

See Section 5.4.2 in D1.1A for a full description of the use case.

Table 22: Use Case 4.2: Multi CDN selection

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Multi CDN selection	IRT	Detailed deployment will be scheduled according to the trials plan. An approach with incremental complexity will be followed
Use Case Number	4.2		
LL	4 (Media & Entertainment Living lab)	LiveU	
ICT 17 Project	5G-VINNI	UoP	
Facility	Patras	UoP	
Category	A		
Orchestration	OSM/TMF	UoP	See 4.1.1 above UoP will set up the network connection and orchestrator.

4.18.2 Existing Integration Points

No integration with CDSO is planned for cycle I.

4.19 Use Case 4.3: On-site Live Event Experience

4.19.1 Use Case Overview

The focus of UC4.3 is on measuring QoE in cases many people in an event transmit & consume video content all at the same time. The channels include 4G, 5G & Wifi all together. See Section 5.4.3 in D1.1A for a full description of the use case.

Table 23: Use Case 4.3: On-site Live Event Experience

Parameter	Value	Responsible Partner	Remarks
Use Case Name	On-site Live Event Experience	LiveU	
Use Case Number	4.3		
LL	4 (Media & Entertainment Living lab)	LiveU	
ICT 17 Project	5G-VINNI	UoP	
Facility	Patras	UoP	
Category	A		
Orchestration	OSM/TMF	UoP	See 4.1.1 above UoP will set up the network connection and orchestrator.

4.19.2 Existing Integration Points

No integration with CDSO is planned for cycle I.

4.20 Use Case 4.4: User & Machine Generated Content

4.20.1 Use Case Overview

This use case is about media-related experiences in generating professional and semi-professional content for various purposes, including live news coverage, live sports and other entertainment coverage, telemedicine-related live transmission such as video from rural medical centers to medical experts or live from moving ambulances, real time security cameras, automotive-generated content such as multiple cameras enabling teleoperation or uploading huge amounts of sensory-generated data. See Section 5.4.4 in D1.1A for a full description of the use case.

Table 24: Use Case 4.4: User & Machine Generated Content

Parameter	Value	Responsible Partner	Remarks
Use Case Name	User & Machine Generated Content	LiveU, FNET	
Use Case Number	4.4		
LL	4 (Media & Entertainment Living lab)	LiveU	
ICT 17 Project	5G-VINNI	UoP	
Facility	Patras	UoP	
Category	A		
Orchestration	OSM/TMF	UoP	See 4.1.1 above

In UC4.4 a first test could be to manually set up the field unit to certain parameters, use an eMBB slice, and run several sessions (each of which is a start/stop of the video stream of one or more of the field units). A later scenario, under the same UC4.4, may be to set the field unit (manually) to different parameters and run several more sessions. Another phase/scenario in the UC is to set the network to a different configuration. For example, to change from the eMBB slice to a NEM media slice. Other configuration options will be explored.

In this specific UC we plan to have two devices transmitting (in cycle I). Some tests would be run with a single one, others with 2 units. We can simplify and always run with two field units, setting one of them manually to not use any modem or similar, if it makes things simpler, but in any case, the CDSO needs to run the start/stop/poll for every UNIT ID separately.

4.20.2 Existing Integration Points

The flow of commands from the CDSO perspective is similar to UC4.1 above. The integration of the CDSO for UC 4.4 includes the following:

Table 25: Integration points between CDSO and UC4.4

	Step	Interface	API	Paramaters	Comments

		with			
1	User activates experiment	UI/API to CDSO			
1.1	Activate experiment on CDSO (API) – Create NS	UI/API to CDSO	NS LCM	Slice ID, Use Case ID, …	See below syntax details Notification sent – see here
1.2	Activate experiment on CDSO (API) – Deploy NS NS	UI/API to CDSO	NS LCM	Slice ID, Use Case ID, …	See below syntax details Notification sent – see here
1.3	Create connection from CDSO to UoP & Authenticate	UoP Orchestrator	OAUTH2	Credentials	See below syntax details
1.4	Slice Configuration	UoP Orchestrator		Slice Parameters	This will be a Service Order configuration. We assume that a Service Specification already Exists and included in the order. (of course, a Service Specification could be done also programmatically but it needs a level of maturity of the use case) See below syntax details
1.5	Activate Slice	UoP Orchestrator		Slice ID	Here the Service Order is made. After a while, the status of the order can be requested (Completed or not) and the status of Services (ACTIVATED, etc.) See below syntax details
1.6	Create connection from CDSO to LiveU Central & Authenticate	LiveU Central		Credentials	LiveU Central is in the cloud. See below syntax details
1.7	Activate Video Stream	LiveU Central		Unit ID	See below syntax details
1.8	Create connection from CDSO to Test Manager & Authenticate	Test Manager?		Credentials	Assuming not the same entity as UoP Orch.
1.9	Activate Test	Test		Test ID	TBD (Test Manager API not yet

		Manager?			available)
2	Ongoing Status				
2.1	Get Slice Status	UoP Orchestrator			**Service Order ID** should be known – syntax below. The **Service Order** includes the Services IDs that are instantiated. To get a Service ID use the Service Inventory API as described below.
2.2	Get stream status	LiveU Central		Unit ID	Syntax Below
3	User Terminates Experiment	UI/API to CDSO			
3.1	Terminate Network Service	UI/API to CDSO	NS LCM	Network Service ID	Syntax Below Notification sent – see here
3.2	Terminate Test	Test Manager / TBD			TBD
3.3	Terminate Video Stream	LiveU Central			Syntax Below
3.4	Terminate Slice	UoP Orchestrator			Not implemented at this point in OpenSlice. Experiment includes Start & End dates
4	User Deletes Experiment	UI/API to CDSO			Cleanup stage
4.1	User deletes network service	UI/API to CDSO	NS LCM	Network Service ID	Syntax Below Notification sent – see here

4.21 Use Case 4.5: Immersive and Integrated Media and Gaming

4.21.1 Use Case Overview

UC 4.5 focuses on running a multiplayer game with VR. The game will include multiple players going into the same environment, where they stand and defend themselves from various enemies. The game and the multiplayer servers will completely be virtualised. Experiment will measure the QoE and performance over 5G.

See Section 5.4.5 in D1.1A for a full description of the use case.

Table 26: Use Case 4.5: Immersive and Integrated Media and Gaming

Parameter	Value	Responsible	Remarks

		Partner	
Use Case Name	Immersive and Integrated Media and Gaming	NORU, LiveU, CTTC	
Use Case Number	4.5		
LL	4 (Media & Entertainment Living lab)	LiveU	
ICT 17 Project	5G-VINNI	UoP	
Facility	Patras	UoP	
Category	A		Detailed Orchestration Requirements still TBD
Orchestration	OSM/TMF	UoP	See 4.1.1 above

4.21.2 Existing Integration Points

No integration with CDSO is planned for cycle I.

4.22 Use Case 4.6: Cooperative Media Production

4.22.1 Use Case Overview

In this use case, several cameras are each connected in the field to cellular-based transmission devices, including bonding devices to provide the utmost reliability and bandwidth. These field devices then transmit the video stream over the cellular and the standard public internet (ISP) to a single receiving/decoder server with several physical SDI outputs. See Section 5.4.6 in D1.1A for a full description of the use case.

Table 27: Use Case 4.6: Cooperative Media Production

Parameter	Value	Responsible Partner	Remarks
Use Case Name	Cooperative Media Production	LiveU	
Use Case Number	4.6		
LL	4 (Media & Entertainment Living lab)	LiveU	
ICT 17 Project	5G-VINNI		
Facility	Patras	UoP	
Category	A		
Orchestration	TBD		

4.22.2 Existing Integration Points

UC 4.6 is quite similar to UC 4.4 in terms of orchestration; thus, the same sequence of commands will be used in both experiments.

4.22.3 Development work required in CDSO to support the integration

See UC 4.1 above for a detailed description of the work done on the CDSO.

4.23 Future Use Cases

There are several use cases that were defined as part of D1.1 but are not yet defined well enough to determine if they are Category A, B or C. These will be further elaborated in future releases of this deliverable. In addition, all Multi Living Lab use cases have not yet been defined (part of WP7) and will be covered in future drops of this deliverable.

5 Conclusions and Next Actions

This deliverable is intended to bridge the gap between the use case definition and requirements (D1.1) and what is needed to orchestrate in order to connect each use case to the hosting facility and enable triggering the experiments.

It also gives more detailed information on the CSDO and information on the extensibility options and plugin formats available to support the previous action.

Since this second drop (D2.2B) is released before any experiments were performed and no integration with CDSO was done (M12and from other EU projects (5G-EVE & 5G-VINNI), there are still some important pieces of information missing at this point in time. This is also due to the following reasons:

- ICT-17 projects we are planning to integrate with are still in the process of defining some of the interfaces and mechanisms (e.g. Test Manager).
- Some 5G-SOLUTIONS use cases are still in early stages of definition and as such the orchestration requirements are not yet defined for the use cases. We did manage to gather detailed orchestration requirements for 3 use cases and these are included in this report.

As most of the use cases in 5G-SOLUTIONS are hosted by ICT-17 projects, we intend to rely on the orchestration services provided by these ICT-17 facilities and trigger the activation or deactivation of the experiments in 5G-SOLUTIONS via the interfaces provided by each ICT-17 facility. This will only work in UCs where orchestration is provided by the underlying platform and will not hold in cases where the use case is not part of the ICT-17 setup (there are several such use cases e.g. UC1.1) and in the multi living lab scenario which will go across use cases, thus requiring additional orchestration.

Each use case in Chapter 4 includes a short description (taken from D1.1A).More complete information is available in D1.1A. It also includes a table consolidating the current information regarding the use case including the formal name & number, the owners, the ICT-17 facility, the category of the use case and the orchestration requirements available.

For those Use Cases that are fully defined for CDSO orchestration, we added a table of detailed step by step flow of the API and actions required for integration with very detailed command line examples for executing these APIs.

In future versions of this deliverable, we shall complete the missing information per use case including:

- Complete the orchestration requirements for other use cases.
- Complete gaps in APIs we need to use that are not currently supported by the CSDO
- Define the exact plugins/flows we need to develop in order to bridge these gaps.

This will be used as the input for T2.3 to actually implement these plugins and perform the orchestration and integration of all use cases.

6 References

5G-EVE Deliverable D3.1, https://www.5g-eve.eu/deliverables/

5G-EVE Turin site high level architecture, https://www.5g-eve.eu/wp-content/uploads/2018/09/5g-eve-facility-site-italy.png

5G-VINNI Patras Site, https://www.5g-vinni.eu/greece-main-facility-site/

5G-VINNI Norway Site, https://www.5g-vinni.eu/norway-main-facility-site/

5G-SOLUTIONS Deliverable D1.1A available on the project Teamworks platform.

5G-SOLUTIONS Deliverable D1.2A available on the project Teamworks platform.

7 Annex A: Code Samples

7.1 Deploying Plugins

The following are examples of how to deploy plugins developed in Java or Python

7.1.1 Deploying Plugins Developed in Java

The WAR file which contains the Plugin needs to be deployed in the Backend VM (where the JBoss process runs). The JBoss CLI command to deploy a WAR file is shown below. In order to execute the WAR

a. Login into CBND Backend node where JBoss is running
b. SSH (using Putty or any SSH Client) to Backend node (VM) using root credentials
c. Check the status of the JBoss service using command " ***systemctl status jboss-as-domain*** " command
d. Copy the Plugin WAR file into /tmp folder
e. Execute Jboss CLI command to deploy the WAR file

$JBOSS_HOME_DIR/bin/jboss-cli.sh --connect --command="deploy /tmp/<<PLUGIN>>.war

7.1.2 Deploying Plugin Developed in Python

The Python package can then be deployed in " ***Workflow VM*** " where there is a Python runtime installed.

a. Login to the Workflow VM using root credentials
b. SSH (using Putty or any SSH Client) to Workflow node (VM) using root credentials
c. Copy the Plugin Artefacts into Workflow VM
d. Deploy Plugin Artefacts into Python Runtime

7.2 CDSO Plugin Code Examples

Below are 2 examples of plugin implementations:

7.2.1 A configuration only plugin:

This example is a plugin that checks if an object can be deleted.

- **1st part is an XML describing the properties and the PRE_DELETE action:**

```
<?xml version="1.0" encoding="UTF-8"?>

<administrationData>
    <pluginInfo>
        <type displayName="Network Functions">Network Functions</type>
           <name>NetworkFunction</name>
           <version>1.0</version>
           … // more parameters for the plugin info
    </pluginInfo>
    <configurationData>
        <parameterGroup name="Basic">
            <parameter name="type" type="string" displayName="Type"
source="user" mandatory="true"
```

```
                        meta="{'search':'true'}"  description="Type  of
the resource. Ex: vnf, nms, ems etc.">
                <constraint operator="pattern">^.{1,80}$</constraint>
            </parameter>
            <parameter name="name" type="string" displayName="Name"
source="user"
          mandatory="true"
                        meta="{'search':'true'}"
          description=" 'MyTelecom_NF_124', 'JohnDoe-1'">
               <constraint operator="pattern">^.{1,80}$</constraint>
            </parameter>

     …// More parameters and parameter groups

</configurationData>
    <actions>
        <action name="Action to validate NetworkFunction is removable"
workflowHandler="Mistral">
            <stages>
                <stage>PRE_DELETE</stage>
            </stages>

<workflow>nfvo.check_for_active_batch_ops_executions</workflow>
            <arguments>
                <argument name="instance_id">${id}</argument>
                <argument                    name="node_type">network
functions</argument>
            </arguments>
            <expectedMessage>No Active executions, NETWORK FUNCTIONS
can be deleted.
     </expectedMessage>
        </action>
    </actions>
</administrationData>
```

- **2nd part is a YAML describing the workflow to execute:**

```
nfvo.check_for_active_batch_ops_executions:
  description : Check if there are any active batch ops executions
before deleting the instance.
  type : direct
  input:
    - instance_id
    - node_type
  output :
    result: <% $.result %>
  tasks :
    get_target_to_aes_inventory_mapping:
      action: std.http
      input:
         url:                                                    <%
env().aes_service_url+'?query=state:neq:DELETED&query=targetResourc
eId:eq:'+  $.instance_id %>
```

```
          method: GET
          allow_redirects: true
          headers:
                    "Content-Type": "application/json"
                    "Accept": "application/json"
                    "Authorization": "bearer <% env().authToken %>"
                    "x-client-request-id": <% uuid() %>
          verify: false
      on-success:
        -               send_target_can_be_deleted:               <%
task().result.content.results.len() = 0 %>
        -                target_cannot_be_deleted:                <%
task().result.content.results.len() > 0 %>
      publish:
          inventory : <% task().result.content.results %>

    send_target_can_be_deleted:
      action: std.noop
      publish:
        result : 'No Active executions, <% $.node_type.toUpper() %>
can be deleted.'

    target_cannot_be_deleted:
      action: std.noop
      publish:
          result : '<% $.node_type.toUpper() %> cannot be deleted
as there are one or more resource operations still available in aes
inventory <% $.inventory.select($.id) %>'
```

7.2.2 A plugin with specific code:

This **example** is a plugin that checks the state of the VNFM:

- **Define the API to be implemented**:

```
@GET
   @Path("/{vnfm_id}")
   @Produces(MediaType.APPLICATION_JSON)
   @TypeHint(VnfmDescriptor.class)
   @ApiOperation(value = "API to get the status of the VNFM.",
        response =VnfmDescriptor.class )

   @ApiResponses(value = {
        @ApiResponse(code = 401, message = "Unauthorized Request.",
             response = ErrorResponse.class),
        @ApiResponse(code = 403, message = "Forbidden.",
             response = ErrorResponse.class),
        @ApiResponse(code = 500, message="Internal Server error.",
             response = ErrorResponse.class)
   })
   Response getVNFMStatus(@ApiParam(value = "Identifier of VNFM
        which can be fetched by using the vnfms/list API or from
        administrated VNFMs in CBND") @PathParam("vnfm_id") String
```

```
            vnfmId);
```

- **Implement the API**:

```
@Override
 public Response getVNFMStatus(String vnfmId) {
     LOGGER.info("Getting the status of Vnfm from CBAM for id
           [{}]", vnfmId);
     validateVnfManagerExists(vnfmId);
     VnfmDescriptor vnfmStatus = getVnfmStatusHelper
           (getCBAMServiceUtil().getVnfmInfo(vnfmId)).
                getVnfmStatus();
     return Response.status(Response.Status.OK).entity(
           vnfmStatus).build();
 }
  public VnfmDescriptor getVnfmStatus() {
       LOGGER.trace("Entering into status of vnfm id " +
       vnfmInfo.getVnfmUUID());
       VnfmDescriptor vnfmDescriptor = new VnfmDescriptor();
       vnfmDescriptor.setId(vnfmInfo.getVnfmUUID());
       try {
           // Returns list of vnf instances that exist
                  on the VNFM.
           getConnectionHelper().getVnfInstances();
           vnfmDescriptor.setStatus(Status.Resource.ACTIVE.name())
;
       } catch (CbamHttpException e) {
           vnfmDescriptor.setStatus(Status.Resource.ERROR.name());
           LOGGER.error("Error while fetching status of vnfm : {},
              cause : {} ",
               vnfmInfo.getInterfaceEndpoint(), e.getMessage(), e);
       }
       return vnfmDescriptor;
    }
```

7.3 CDSO API usage samples

7.3.1 Authenticate with CDSO

curl --location --request POST
'https://10.0.92.15:7443/auth/realms/CloudBandNetworkDirector/protocol/openid-connect/token' \

--header 'Content-Type: application/x-www-form-urlencoded' \

--data-urlencode 'username=cbndadmin' \

--data-urlencode 'password=Admin@123' \

--data-urlencode 'grant_type=password' \

--data-urlencode 'client_id=cbnd' \

--data-urlencode 'scope=offline_access'

Example response:

```
{
```

```
    "access_token": "…I",
    "expires_in": 14400,
    "refresh_expires_in": 0,
    "refresh_token": "…o",
    "token_type": "bearer",
    "not-before-policy": 0,
    "session_state": "72054367-a746-4458-9726-2fe09992dee3",
    "scope": "profile email offline_access"
}
```


7.3.2 Subscribe to CDSO notifications:

```
curl --location --request POST 'https://10.0.92.15:9443/api/cbnd/nslcm/v1/subscriptions' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer e...I' \
--header 'Content-Type: text/plain' \
--data-raw '{
   "callbackUri": "https://callback.url",
   "authentication": {
         "authType": [ "BASIC" ],
         "paramsBasic": {
               "userName": "myself",
               "password": "p@ssw@rd"
         }
   }
}'
```

Example response:


```
{
    "id": "3be698f6-b13f-4da2-b0eb-10221ddccecb",
    "filter": null,
    "callbackUri": "http://melisha.free.beeceptor.com?authUsername=myself&authPassword=p%40ssw%40rd",
    "_links": {
        "self": {
            "href": "https://10.0.92.15:9443/api/cbnd/nslcm/v1/subscriptions/3be698f6-b13f-4da2-b0eb-10221ddccecb"
        }
    }
}
```


Note: For HTTPS callback to work, the user needs to manually import the target callback server SSL cert into CDSO. NOKIA team can do this import. See Appendix A for full documentation regarding LifeCycle Change Notifications.

7.3.3 Unsubscribe from CDSO Notifications

curl --location --request DELETE 'https://10.0.92.15:9443/api/cbnd/nslcm/v1/subscriptions/3be698f6-b13f-4da2-b0eb-10221ddccecb' \

--header 'Authorization: Bearer ...I'

Example response:

```
204 No Content
```

7.3.4 Create Network Service

curl --location --request POST 'https://10.0.92.15:9443/api/cbnd/v1/networkservices' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer e...I' \

--header 'Content-Type: text/plain' \

--data-raw '{

"name": "TestExpirement",

"nsdId": "8b3cc4d0-8211-4702-87bb-ffc5ced9e1e3",

"parameters": [

{

"name": "use_case_id",

"value": "4.4"

}

]

}'

7.3.5 Deploy Network Service

curl --location --request POST 'https://10.0.92.15:9443/api/cbnd/v1/networkservices/691f31d7-1b69-4cf2-a101-17c42ac3bc1a/operations' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer e...I' \

--header 'Refresh-Token: e...g' \

--data-raw '{

"operationType": "DEPLOY"

}

'

7.3.6 Terminate Network Service

curl --location --request POST 'https://10.0.92.15:9443/api/cbnd/v1/networkservices/691f31d7-1b69-4cf2-a101-17c42ac3bc1a/operations' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer e...I' \

--header 'Refresh-Token: e...g' \

--data-raw '{

"operationType": "TERMINATE"

}

'

Notes:

1. Two notifications (termination start and termination end) will be sent to the subscriber. Details here
2. Once a notification is received, the subscriber can use the CDSO "Get Network Service" API to get details about the service/experiment (use case ID, etc.)

7.3.7 Delete Network Service

curl --location --request DELETE 'https://10.0.92.15:9443/api/cbnd/v1/networkservices/691f31d7-1b69-4cf2-a101-17c42ac3bc1a' \

--header 'Content-Type: application/json' \

--header 'Authorization: Bearer e...I'

Notes:

1. A notification will be sent to the subscriber. Details here.

7.4 UoP (Openslice) API Usage Examples

7.4.1 Authenticate

Site address is on https://patras5g.eu/

curl -X POST https://patras5g.eu/auth/realms/openslice/protocol/openid-connect/token -H 'Content-Type: application/x-www-form-urlencoded' -d 'username=demouser' -d 'password=demouser' -d 'grant_type=password' -d 'client_id=admin-cli'

7.4.2 Access to a Service Specification

curl -H 'Authorization: Bearer ...-sY2WIu32Wfa3IPEDw' http://portal.openslice.io/tmf-api/serviceCatalogManagement/v4/serviceSpecification/0d5551e6-069f-43b7-aa71-10530f290239

The following will be used the define the slice:

https://bscw.5g-ppp.eu/pub/bscw.cgi/d322688/NEM%20Networld2020%205G%20media%20slice%20V1-2_24092019.pdf

7.4.3 Request a Service Order

Curl -X POST -H 'Authorization: Bearer ...-sY2WIu32Wfa3IPEDw'

http://portal.openslice.io/tmf-api/serviceOrdering/v4/serviceOrder

7.4.3.1 Service Order Payload:


```
{
 "orderItem": [
  {
   "service": {
    "serviceSpecification": {
     "id": "0d5551e6-069f-43b7-aa71-10530f290239",
     "name": "Upstream journalist case",
     "version": "0.1.0"
    },
    "serviceCharacteristic": [
     {
      "name": "Location of Mobile Cameras",
      "value": {
       "value": "2610",
       "alias": "Patras"
      }
     },
     {
      "name": "Location of NASS",
      "value": {
       "value": "2310",
       "alias": "Patras"
      }
     },
     {
      "name": "Maximum Number of cameras",
      "value": {
       "value": "3",
       "alias": null
      }
     },
     {
      "name": "Number of reserved hours",
      "value": {
       "value": "2",
       "alias": "2"
      }
     },
     {
      "name": "Video quality of the mobile cameras",
      "value": {
```

```
          "value": "0",
          "alias": "SD"
         }
        },
        {
         "name": "Video response time of the mobile cameras",
         "value": {
          "value": "1",
          "alias": "Live (1s)"
         }
        }
       ]
     },
     "action": "add"
   }
  ],
  "note": [
   {
    "author": "my application",
    "text": "please start this",
    "date": "2020-04-28T14:32:15.249Z"
   }
  ],
  "requestedStartDate": "2020-04-28T14:32:06.008Z",
  "requestedCompletionDate": "2020-04-29T14:32:06.008Z"
}
```


7.4.4 Get Service Order ID

Curl http://portal.openslice.io/tmf-api/serviceOrdering/v4/serviceOrder/**975ea4ae-6b2a-45c5-a0cd-d2934cbf70da**

7.4.5 Get Service Order

Based on above Service Order ID now get the Service Order record:

curl http://portal.openslice.io/tmf-api/serviceInventory/v4/service/**d6a03828-bda1-4e25-9784-926af76537ed**

7.5 LiveU Central API Usage Examples

LiveU full sample code will be available for integration purposes only. Samples below do not contain any confidential information.

7.5.1 Authenticate

POST **Error! Hyperlink reference not valid.**

Activate Video Stream

POST

Set bearer token, Application-Id header and content-type header

7.5.2 Get Stream Status

7.5.3 GET
set bearer token, Application-Id header and content-type header
Delete Stream

DELETE
set bearer token, Application-Id header and content-type header

8 Annex A: CDSO Notifications

8.1 CDSO (CBND) NS LCM notifications

An NFVO admin would like to notify NBI clients when there are changes in the state of network service.

On CBND 19.5 the ability to get notifications for NS operations (create, deploy, update, terminate, delete) was added - user needs to subscribe first (LCM SOL-005 subscriptions).

Notifications are SOL005 complaint (ETSI GS NFV-SOL 005 V2.5.1 with some exceptions, some fields are not available yet, see chapter 6 "NS Lifecycle Management interface").

In this annex there is a short explanation on how to use CBND Notifications and also some real examples for notifications.

8.1.1 Subscription

This clause describes the procedure for creating, reading and terminating subscriptions to notifications related to NSD management.

Create Subscription request:

POST https://{{cbnd-ex-vip}}:{{cbnd-port}}/api/cbnd/nslcm/v1/subscriptions

```
{
   "callbackUri": "https://cbnds.free.beeceptor.com",
   "authentication": {
     "authType": [
       "BASIC"
     ],
     "paramsBasic": {
       "userName": "myuser",
       "password": "mypass"
     }
   }
 }
```

Create response:

```
{
```

```
  "id": "51c7327c-7ba3-4400-ab9e-86cffd7a67ce",
  "filter": null,
  "callbackUri": "https://cbnds.free.beeceptor.com?authUsername=myuser&authPassword=mypass",
  "_links": {
    "self": {
      "href": "https://135.248.18.66:9443/api/cbnd/nslcm/v1/subscriptions/51c7327c-7ba3-4400-ab9e-
86cffd7a67ce"
    }
  }
}
# get by ID
GET https://{{cbnd-ex-vip}}:{{cbnd-port}}/api/cbnd/nslcm/v1/subscriptions/{{subs-id}}
  {
    "id": "3d886b5b-117f-48b3-bc76-9b18d65cd591",
    "filter": null,
    "callbackUri":
"https://testendpoint.free.beeceptor.com/my/api_1/path?authUsername=myuser&authPassword=mypass",
    "_links": {
      "self": {
        "href": "https://135.248.18.66:9443/api/cbnd/nslcm/v1/subscriptions/3d886b5b-117f-48b3-bc76-
9b18d65cd591"
      }
    }
  }

# get by ID
GET https://{{cbnd-ex-vip}}:{{cbnd-port}}/api/cbnd/nslcm/v1/subscriptions/{{subs-id}}
  {
    "id": "3d886b5b-117f-48b3-bc76-9b18d65cd591",
    "filter": null,
    "callbackUri":
"https://testendpoint.free.beeceptor.com/my/api_1/path?authUsername=myuser&authPassword=mypass",
    "_links": {
      "self": {
        "href": "https://135.248.18.66:9443/api/cbnd/nslcm/v1/subscriptions/3d886b5b-117f-48b3-bc76-
9b18d65cd591"
      }
    }
  }
```


The DELETE method terminates an individual subscription.


```
DELETE https://{{cbnd-ex-vip}}:{{cbnd-port}}/api/cbnd/nslcm/v1/subscriptions/{{subs-id}}
```

8.1.2 Manual SSL import - for https urls

In order to use 'https' with SSL certificate, please use the following commands/scripts to import them into CBND (for HA you need to execute on all nodes):

```
// use mistral user for B&R
[cbnd@cbnd-19-5-server]$ su mistral
[mistral@cbnd-19-5-server ~]$



# 1
########
# first download certificates, in case you have it already skip to next step "to store them on cbnd truststore.."
[mistral@cbnd-19-5-server ~]$ /opt/oodee/bin/ood_find_certificate_from_remotenode.sh
Usage: ./ood_find_certificate_from_remotenode.sh [Options]
Description: This script can be used to fetch all public key from a remote node.
Allowed Options are : -
      NODE IP/FQDN:  IP or FQDN of the node where certificates are present. Mandatory.
      PORT:  port where service is running. Mandatory.

#for example:
[mistral@gp-jul-19-0-server root]$ /opt/oodee/bin/ood_find_certificate_from_remotenode.sh  beeceptor.com
443
Cert files generated are :
 /var/lib/mistral/certs//1.remote_cert_beeceptor.com_443.pem
 /var/lib/mistral/certs//2.remote_cert_beeceptor.com_443.pem




# 2
#######
# in case you have the certificate and didnt use the script from step #1 please put the certificate under:
'/var/lib/mistral/certs/'
# to store them on cbnd truststore use following script (for HA on all cbnd nodes) :
[mistral@cbnd-19-5-server]$ /opt/oodee/bin/ood_import_public_key_to_client_truststore.sh
Usage: ./ood_import_public_key_to_client_truststore.sh [Options]
Description: This script can be used to import a certificate to truststore of Jboss/Tomcat.
Allowed Options are : -
      keyAlias:  Alias of the public key to be imported. Mandatory.
      certFileName :  File name with full path. Supports key in the DER encoded certificate (cer, crt, pem).
Mandatory.

#for example:
[mistral@gp-jul-19-0-server root]$ /opt/oodee/bin/ood_import_public_key_to_client_truststore.sh lccn_genos
/var/lib/mistral/certs//2.remote_cert_beeceptor.com_443.pem
INFO: Executing -import on backend node (Wildfly)
```

INFO: Certificate /var/lib/mistral/certs//2.remote_cert_beeceptor.com_443.pem was imported successfully at /opt/wildfly/keystores/truststore.jks
INFO: Certificate /var/lib/mistral/certs//2.remote_cert_beeceptor.com_443.pem was imported successfully into the truststore

8.1.3 Notifications examples

8.1.3.1 NS Create (Table 6.5.2.6-1: Definition of the NsIdentifierCreationNotification data type):


```
{
   "NsIdentifierCreationNotification" : {
      "notificationType" : "NsIdentifierCreationNotification",
      "subscriptionId"  :  "fdbc9c19-dd2b-4b5a-9e27-25b22db58e6a",
      "timestamp" : "2019-08-04T12:21:13Z",
      "nsInstanceId" : "5f300f61-f34e-4f7c-965c-e5dbe01395aa",
   "_links": {
      "nsInstance": {
         "href": "https://135.248.18.66:9443/api/cbnd/v1/networkservices/5f300f61-f34e-4f7c-965c-
e5dbe01395aa"
      },
      "subscription": {
         "href": "https://135.248.18.66:9443/api/cbnd/nslcm/v1/subscriptions/fdbc9c19-dd2b-4b5a-9e27-
25b22db58e6a"
      },
      "lcOpOcc": {
         "href": null
      }
   }
   }
}
```


8.1.3.2 NS Deploy start :


```
{
  "NsLcmOperationOccurrenceNotification": {
   "id": "f4c07a73-8219-4665-bdae-9c3483e96696",
   "nsInstanceId": "5f300f61-f34e-4f7c-965c-e5dbe01395aa",
   "nsLcmOpOccId": "a2135071-7a2e-47af-937d-c405e2e478ff",
   "notificationType": "NsLcmOperationOccurrenceNotification",
   "subscriptionId": "fdbc9c19-dd2b-4b5a-9e27-25b22db58e6a",
   "timestamp": "2019-08-04T12:21:25Z",
   "isAutomaticInvocation": false,
   "notificationStatus": "START",
   "operation": "INSTANTIATE",
   "operationState": "PROCESSING",
   "affectedVnf": [
   ],
   "affectedPnf": [
   ],
   "affectedVl": [
     ],
```

```
  "_links": {
    "nsInstance": {
      "href": "https://135.248.18.66:9443/api/cbnd/v1/networkservices/5f300f61-f34e-4f7c-965c-
e5dbe01395aa"
    },
    "subscription": {
      "href": "https://135.248.18.66:9443/api/cbnd/nslcm/v1/subscriptions/fdbc9c19-dd2b-4b5a-9e27-
25b22db58e6a"
    },
    "lcOpOcc": {
      "href": null
    }
  }
 }
}
```


8.1.3.3 NS Deploy completed:


```
{
 "NsLcmOperationOccurrenceNotification": {
  "id": "b2802070-1ddd-406e-9932-751143fb75a3",
  "nsInstanceId": "5f300f61-f34e-4f7c-965c-e5dbe01395aa",
  "nsLcmOpOccId": "a2135071-7a2e-47af-937d-c405e2e478ff",
  "notificationType": "NsLcmOperationOccurrenceNotification",
  "subscriptionId": "fdbc9c19-dd2b-4b5a-9e27-25b22db58e6a",
  "timestamp": "2019-08-04T12:22:49Z",
  "isAutomaticInvocation": false,
  "notificationStatus": "RESULT",
  "operation": "INSTANTIATE",
  "operationState": "COMPLETED",
  "affectedVnf": [
    {
      "vnfInstanceId": "56d753a3-330b-4318-bdcc-52c7d096af10",
      "vnfdId": "simple_powerful_v3",
      "vnfProfileId": "",
      "vnfName": "gateVnf",
      "changeType": "INSTANTIATE",
      "changeResult": "COMPLETED",
      "changedInfo": {
       "changedVnfInfo": {
         "vnfInstanceId": "56d753a3-330b-4318-bdcc-52c7d096af10"
         ,"vnfInstanceName": "gateVnf"
       }
      }
    }
  ],
  "affectedPnf": [
  ],
  "affectedVl": [
```

```
    {
      "nsVirtualLinkInstanceId": "fe836ab5-4a64-4620-9d9e-43344a4371ee",
      "nsVirtualLinkDescId": "vl1",
      "vlProfileId": "",
      "changeType": "ADD",
      "changeResult": "COMPLETED"
    }
    ],
  "_links": {
    "nsInstance": {
      "href": "https://135.248.18.66:9443/api/cbnd/v1/networkservices/5f300f61-f34e-4f7c-965c-
e5dbe01395aa"
    },
    "subscription": {
      "href": "https://135.248.18.66:9443/api/cbnd/nslcm/v1/subscriptions/fdbc9c19-dd2b-4b5a-9e27-
25b22db58e6a"
    },
    "lcOpOcc": {
      "href": null
    }
  }
 }
}
```


8.1.3.4 NS Deploy failed:

In case you deployment failed you will get this (instead completed). error 500 with details


```
{
 "NsLcmOperationOccurrenceNotification": {
  "id": "b1ad4a00-f0e6-4537-8acc-3b14f64c1ddb",
  "nsInstanceId": "3972514a-665a-48d2-abe7-03dece3202cf",
  "nsLcmOpOccId": "3b0f334a-5540-44f4-b3c4-b4f851385cd6",
  "notificationType": "NsLcmOperationOccurrenceNotification",
  "subscriptionId": "3d886b5b-117f-48b3-bc76-9b18d65cd591",
  "timestamp": "2019-08-07T13:27:08Z",
  "isAutomaticInvocation": false,
  "notificationStatus": "RESULT",
  "operation": "INSTANTIATE",
  "operationState": "FAILED",
  "affectedVnf": [
    {
      "vnfInstanceId": "e5ebb05d-5926-4709-9934-03074e761864",
      "vnfdId": "proxy",
      "vnfProfileId": "",
      "vnfName": "proxy_vnf_v2",
      "changeType": "INSTANTIATE",
      "changeResult": "FAILED",
      "changedInfo": {
       "changedVnfInfo": {
```

```
            "vnfInstanceId": "e5ebb05d-5926-4709-9934-03074e761864"
            ,"vnfInstanceName": "ProxyVnfNameAfterDeploy"
          }
         }
      }
   ],
   "affectedPnf": [
   ],
   "affectedVl": [
     {
        "nsVirtualLinkInstanceId": "9624a04b-7dab-4cb0-a112-1985e2ff0dc5",
        "nsVirtualLinkDescId": "vl1",
        "vlProfileId": "",
        "changeType": "ADD",
        "changeResult": "FAILED"
     }
     ],
   "error": {
      "status": 500,
      "detail": "Network service DEPLOY operation failed for network service name: tettet, id: 3972514a-665a-
48d2-abe7-03dece3202cf, error info: Task 'send_create_vnf_request' is in ERROR with state info: {u'externalId':
u'CBAM-99893e676e334315bc122ebd16c93e38', u'error': {u'_embedded': None, u'userMessage':
u'WorkflowHeatOperationError: Stack operation error! Stack id: 43880eb8-f3d8-4ca8-a8e8-de73cc2944c5,
expected status: COMPLETE, actual status: FAILED, reason: Resource CREATE failed: BadRequest:
resources.proxy: The requested availability zone is not available (HTTP 400) (Request-ID: req-0ae72636-eb0f-
401a-8c7b-446695ab42a3)', u'errorKey': None, u'entity': None, u'errorCode': 400, u'developerMessage':
u'WorkflowHeatOperationError: Stack operation error! Stack id: 43880eb8-f3d8-4ca8-a8e8-de73cc2944c5,
expected status: COMPLETE, actual status: FAILED, reason: Resource CREATE failed: BadRequest:
resources.proxy: The requested availability zone is not available (HTTP 400) (Request-ID: req-0ae72636-eb0f-
401a-8c7b-446695ab42a3)', u'httpStatusCode': 500, u'serverRequestId': None}}"
   },
   "_links": {
      "nsInstance": {
         "href": "https://135.248.18.66:9443/api/cbnd/v1/networkservices/3972514a-665a-48d2-abe7-
03dece3202cf"
      },
      "subscription": {
         "href": "https://135.248.18.66:9443/api/cbnd/nslcm/v1/subscriptions/3d886b5b-117f-48b3-bc76-
9b18d65cd591"
      },
      "lcOpOcc": {
         "href": null
      }
   }
  }
}
```

8.1.3.5 NS Update (start / completed):


```
{
  "NsLcmOperationOccurrenceNotification": {
    "id": "f977ade1-73bd-479d-a8b4-5281602147ae",
    "nsInstanceId": "5f300f61-f34e-4f7c-965c-e5dbe01395aa",
    "nsLcmOpOccId": "686da878-9c4c-4a63-aff1-3ea4b71a6a71",
    "notificationType": "NsLcmOperationOccurrenceNotification",
    "subscriptionId": "fdbc9c19-dd2b-4b5a-9e27-25b22db58e6a",
    "timestamp": "2019-08-04T12:22:59Z",
    "isAutomaticInvocation": false,
    "notificationStatus": "START",
    "operation": "UPDATE",
    "operationState": "PROCESSING",
    "affectedVnf": [
    ],
    "affectedPnf": [
    ],
    "affectedVl": [
      ],
    "_links": {
      "nsInstance": {
        "href": "https://135.248.18.66:9443/api/cbnd/v1/networkservices/5f300f61-f34e-4f7c-965c-
e5dbe01395aa"
      },
      "subscription": {
        "href": "https://135.248.18.66:9443/api/cbnd/nslcm/v1/subscriptions/fdbc9c19-dd2b-4b5a-9e27-
25b22db58e6a"
      },
      "lcOpOcc": {
        "href": null
      }
    }
  }
}


{
  "NsLcmOperationOccurrenceNotification": {
    "id": "d0bad4ce-697c-4b36-a1e6-7188219dd66a",
    "nsInstanceId": "5f300f61-f34e-4f7c-965c-e5dbe01395aa",
    "nsLcmOpOccId": "686da878-9c4c-4a63-aff1-3ea4b71a6a71",
    "notificationType": "NsLcmOperationOccurrenceNotification",
    "subscriptionId": "fdbc9c19-dd2b-4b5a-9e27-25b22db58e6a",
    "timestamp": "2019-08-04T12:22:59Z",
    "isAutomaticInvocation": false,
    "notificationStatus": "RESULT",
    "operation": "UPDATE",
    "operationState": "COMPLETED",
    "affectedVnf": [
```

```
  ],
  "affectedPnf": [
  ],
  "affectedVl": [
    ],
  "_links": {
    "nsInstance": {
      "href": "https://135.248.18.66:9443/api/cbnd/v1/networkservices/5f300f61-f34e-4f7c-965c-
e5dbe01395aa"
    },
    "subscription": {
      "href": "https://135.248.18.66:9443/api/cbnd/nslcm/v1/subscriptions/fdbc9c19-dd2b-4b5a-9e27-
25b22db58e6a"
    },
    "lcOpOcc": {
      "href": null
    }
  }
 }
}
```


8.1.3.6 NS Terminate (start / completed):


```
{
 "NsLcmOperationOccurrenceNotification": {
  "id": "edcf1530-7464-40dc-8d5d-332048305f0e",
  "nsInstanceId": "5f300f61-f34e-4f7c-965c-e5dbe01395aa",
  "nsLcmOpOccId": "031f1261-1940-47c8-9872-1b6c2e04219c",
  "notificationType": "NsLcmOperationOccurrenceNotification",
  "subscriptionId": "fdbc9c19-dd2b-4b5a-9e27-25b22db58e6a",
  "timestamp": "2019-08-04T12:23:10Z",
  "isAutomaticInvocation": false,
  "notificationStatus": "START",
  "operation": "TERMINATE",
  "operationState": "PROCESSING",
  "affectedVnf": [
  ],
  "affectedPnf": [
  ],
  "affectedVl": [
    ],
  "_links": {
    "nsInstance": {
      "href": "https://135.248.18.66:9443/api/cbnd/v1/networkservices/5f300f61-f34e-4f7c-965c-
e5dbe01395aa"
    },
    "subscription": {
      "href": "https://135.248.18.66:9443/api/cbnd/nslcm/v1/subscriptions/fdbc9c19-dd2b-4b5a-9e27-
25b22db58e6a"
```

```
      },
      "lcOpOcc": {
        "href": null
      }
    }
  }
}


{
  "NsLcmOperationOccurrenceNotification": {
    "id": "92173166-5840-436f-aace-dc2275c5fdf8",
    "nsInstanceId": "5f300f61-f34e-4f7c-965c-e5dbe01395aa",
    "nsLcmOpOccId": "031f1261-1940-47c8-9872-1b6c2e04219c",
    "notificationType": "NsLcmOperationOccurrenceNotification",
    "subscriptionId": "fdbc9c19-dd2b-4b5a-9e27-25b22db58e6a",
    "timestamp": "2019-08-04T12:24:11Z",
    "isAutomaticInvocation": false,
    "notificationStatus": "RESULT",
    "operation": "TERMINATE",
    "operationState": "COMPLETED",
    "affectedVnf": [
      {
        "vnfInstanceId": "56d753a3-330b-4318-bdcc-52c7d096af10",
        "vnfdId": "simple_powerful_v3",
        "vnfProfileId": "",
        "vnfName": "gateVnf",
        "changeType": "TERMINATE",
        "changeResult": "COMPLETED",
        "changedInfo": {
         "changedVnfInfo": {
           "vnfInstanceId": "56d753a3-330b-4318-bdcc-52c7d096af10"
           ,"vnfInstanceName": "gateVnf"
         }
        }
      }
    ],
    "affectedPnf": [
    ],
    "affectedVl": [
      {
        "nsVirtualLinkInstanceId": "fe836ab5-4a64-4620-9d9e-43344a4371ee",
        "nsVirtualLinkDescId": "vl1",
        "vlProfileId": "",
        "changeType": "DELETE",
        "changeResult": "COMPLETED"
      }
      ],
    "_links": {
      "nsInstance": {
```

```
        "href": "https://135.248.18.66:9443/api/cbnd/v1/networkservices/5f300f61-f34e-4f7c-965c-
e5dbe01395aa"
      },
      "subscription": {
        "href": "https://135.248.18.66:9443/api/cbnd/nslcm/v1/subscriptions/fdbc9c19-dd2b-4b5a-9e27-
25b22db58e6a"
      },
      "lcOpOcc": {
        "href": null
      }
   }
  }
}
```


8.1.3.7 NS Delete:


```
{
   "NsIdentifierDeletionNotification" : {
      "notificationType" : "NsIdentifierDeletionNotification",
      "subscriptionId"  :  "fdbc9c19-dd2b-4b5a-9e27-25b22db58e6a",
      "timestamp" : "2019-08-04T12:24:13Z",
      "nsInstanceId" : "5f300f61-f34e-4f7c-965c-e5dbe01395aa",
   "_links": {
      "nsInstance": {
        "href": "https://135.248.18.66:9443/api/cbnd/v1/networkservices/5f300f61-f34e-4f7c-965c-
e5dbe01395aa"
      },
      "subscription": {
        "href": "https://135.248.18.66:9443/api/cbnd/nslcm/v1/subscriptions/fdbc9c19-dd2b-4b5a-9e27-
25b22db58e6a"
      },
      "lcOpOcc": {
        "href": null
      }
   }
   }
}
```