Deliverable D7.1

Cross Plane Slice and Service Orchestrator

Editor(s):	José Cabaça and Pedro Neves, Altice Labs
Deliverable nature:	Report (R)
Dissemination level: (Confidentiality)	Public (PU)
Contractual delivery date:	31 October 2019
Actual delivery date:	11 November 2019
Suggested readers:	Infrastructure Providers, Communication Service Providers, Digital Service Providers Network Operators, Vertical Industries
Version:	1.0
Total number of pages:	128
Keywords:	Orchestration, Network Slice, 5G, SDN, NFV

Abstract

Following the overall architecture definition from task 2.1, this task will further specify and implement multidomain, cross-plane slice and service orchestrator. On one side, the SliceNet Orchestrator will empower cross-plane/vertical coordination of control, service and data planes to achieve system-level self-organised slicing control and slice management operation. On the other side, it will also coordinate multiple horizontal administrative domains to deliver the contracted end-to-end slice with the required characteristics. This task will also address the need to extend the legacy service & resources catalogues and inventories in order to integrate the slice concept as a new managed entity. For this the task 7.1 also aims the design and implementation of the slice and service catalogue and inventory that will provide all the required information about the slices, services and resources for the whole SliceNet architecture components.

Disclaimer

The research leading to these results has received funding from the European Union Horizon 2020 Programme under grant agreement number H2020-ICT-2014-2/761913.

Impressum

[Full project title] End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks

[Short project title] SLICENET

[Number and title of work-package] WP7 – Cross Plane Orchestration & Use Cases Prototyping

[Number and title of task] T7.1 - Cross Plane Slice and Service Orchestrator

[Document title] Cross Plane Slice and Service Orchestrator

[Editor: Name, company] José Cabaça and Pedro Neves Altice Labs

[Work-package leader: Name, company] Pedro Neves, Altice Labs (ALB)

Copyright notice

Executive summary

This deliverable first describes the role of the orchestrator in the SliceNet [1] architecture. Then, an overview of the main solutions/products working in orchestration activities is provided either under the auspices of Standard Definition Organizations (SDOs), Open Source Communities (OSCs) and other H2020 EU projects. Herein is important to highlight that the SliceNet Orchestrator is based on a solution developed in H2020 5G-Transformer project, which is extended to support slicing in multi-domain environments, as well as the required adaptations to the SliceNet logical, functional and informational architecture.

The next chapters provide the SliceNet orchestration approach in more detail focusing on the DSP and NSP orchestration aspects, from service orchestrator and slice orchestrator architecture to service level and network slice level requirements. Also a functional perspective of the orchestration is provided in the form of workflows, highlighting the preparation, subscription, runtime and decommission phases of a service/network slice. Finally, a detailed description of the orchestration interfaces are also given, as well as an explanation of the software prototype being delivered.

List of authors

Company	Author	Contribution
ALB	Pedro Neves, José Cabaça	Introduction, SliceNet orchestration approach, Orchestration workflows, Conclusion
NXW	Giacomo Bernini, Pietro G. Giardina, Ali Nejabati	SliceNet orchestration approach, SliceNet information model, Orchestration logical architecture, SliceNet orchestration interfaces, Software prototype, Orchestration related activities overview
UPC	Salvatore Spadaro, Fernando Agraz, Albert Pagès, Rafael Montero	SliceNet orchestration approach, SliceNet information model, Orchestration logical architecture, SliceNet orchestration interfaces, Software prototype, Orchestration related activities overview
IBM	Kenneth Nagin	Orchestration related activities overview
ECOM	Navid Nikaein, Osama Arouk	Orchestration related activities overview
TEI	Ciriaco Angelo	Orchestration related activities overview

Table of Contents

Contents

List of figures

List of tables

Abbreviations

3GPP	3rd Generation Partnership Project
5G	Fifth Generation (mobile/cellular networks)
5G PPP	5G Infrastructure Public Private Partnership
AI	Artificial Intelligence
API	Application Program Interface
BSS	Business Support System
DSP	Digital Service Provider
CN	Core Network
EIM	Extended Infrastructure Manager
eMBB	enhanced Mobile Broad Band
ENI	Experiential Network Intelligence
ETSI	European Telecommunications Standards Institute
FCAPS	Fault Configuration Accounting Performance Security
FMC	Fixed Mobile Convergence
KPI	Key Performance Indicator
mMTC	massive Machine Type Communications
ML	Machine Learning
MEC	Multi-access Edge Computing
NBI	North Bound Interface
NFV	Network Function Virtualization
NFVI	Network Function Virtualization Infrastructure
NFVO	Network Function Virtual Orchestrator
NMRO	Network Management Resource Orchestrator
NRM	Network Resource Models
NSD	Network Service Descriptor
NS	Network Slice
NSI	Network Slice Instance
NSSI	Network Slice Subnet Instance
NSO	Network Service Orchestrator
NSP	Network Service Provider
NST	Network Slice Template
NSS	Network Slice Subnet
NSST	Network Slice Subnet Template
ONAP	Open Network Automation Platform
ONOS	Open Network Operating System
OOM	ONAP Operations Manager
OSC	Open Source Communities
OSS	Operations Support System
OSM	Open Source MANO
P&P	Plug&Play
PLMN	Public Land Mobile Network
PNF	Physical Network Function
PoC	Proof Of Concept
QoE	Quality of Experience
QoS	Quality of Service

R&D	Research and Development
RAN	Radio Access Network
SBI	South Bound Interface
SDN	Software Defined Networks
SDK	Software Development Kit
SDO	Standard Definition Organizations
SliceNet	End-to-End Cognitive Network Slicing and Slice Management Framework in Virtualised Multi-Domain, Multi-Tenant 5G Networks
SO	Service Orchestrator
SON	Self-Organizing Networks
TS	Technical Specification
UE	User Equipment
URLLC	Ultra Reliable Low Latency Communications
VIM	Virtual Infrastructure Manager
VNF	Virtual Network Function
VNFD	Virtual Network Function Descriptor
VNFM	Virtual Network Function Manager
VM	Virtual Machine
VS	Vertical Slicer
VSB	Vertical Service Blueprint
VSD	Vertical Service Descriptor
VSI	Vertical Service Instance
ZSM	Zero-touch network Service Management

1 Introduction

1.1 Motivation and scope clarification

SliceNet aims to design, prototype and demonstrate an innovative, verticals-oriented, QoE-driven 5G network slicing framework focusing on cognitive network management and control for end-to-end slicing operation across multiple operator domains in SDN/NFV-enabled 5G networks. To achieve such an ambitious aim, SliceNet must thoroughly address a set of key areas, named as SliceNet Foundations, illustrated in Figure 1, including Network Slicing, Multiple Administrative Domains, Plug & Play capabilities, One Stop API, Cognitive Network Management and Cross-Plane Orchestration.

Figure 1: SliceNet Foundations

Orchestration is one of the most important aspects in today's service provider operations. It is a key element in the service fulfillment chain, addressing several important aspects, such as exposing service offers, provisioning new service subscriptions and orchestrating network resources according to the service requests. In tomorrow's operations, such as the ones envisaged in SliceNet, orchestration procedures will be paramount to achieve the desired service automation and vertical oriented scenarios. However, evolution is required at the currently existing orchestration procedures to be able to cope with several challenging requirements. For example, **(1)** besides orchestrating end-to-end services and network resources, orchestration solutions will have to cope with and manage the lifecycle of a new artifact – the **network slice**. Additionally, **(2)** in order to deliver vertical industries the desired end-to-end service offers, which will require more than a single service provider domain, the orchestration procedures will have to be able to address scenarios in which the end-to-end service is composed by network slices delivered in **multiple service provider domains**. Another key aspect in SliceNet vision is the **(3)** closed-loops automation, enhanced by the integration of cognitive Artificial Intelligence (AI) models, enabling service providers to proactively fix the network slices and maintain the desired service levels to the verticals. Closing the management and/or control loop requires orchestration to be agile and flexible enough to deal with real-time scenarios brought by the **cognitive network management**. Finally, another important aspect **(4)** is the **5G network**, which has significant modifications compared with its antecessor, 4G, and therefore requires also orchestration procedures to handle a multitude of new variables and approaches such as Network Function Virtualization (NFV), Software-Defined Networking (SDN), Multi-access/Mobile Edge Computing (MEC) when interacting with the network environment.

The purpose of the Cross-Plane Orchestration is to provide a set of coordination functions across several logical layers (i.e. service, slice, resources) with the aim of orchestrating the provisioning of end-to-end network slices. The cross-plane orchestration in SliceNet makes use of recursive orchestration abstractions that hide complex operations in multiple administrative domains (i.e. more than one network provider), multiple technologic domains (i.e. RAN, WAN and datacenters), and multiple layers (i.e. services, slices and resources).

Diving into the SliceNet logical architecture, orchestration procedures are involved in the Orchestration Sub-Plane itself, but also on the Information Sub-Plane, which holds the required information (catalogues and inventories) to enable orchestration procedures. Figure 2 illustrates these subplanes in the whole SliceNet framework. Further details about these sub-planes and their internal components will be provided within this document.

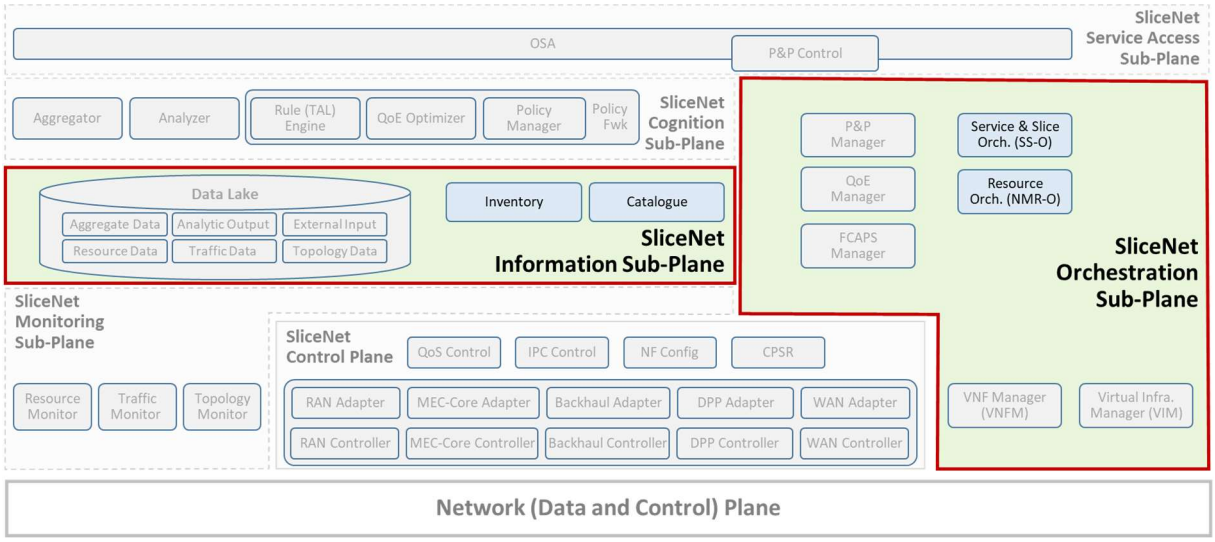

Figure 2: Cross-Plane Orchestration – Orchestration and Information Sub-Planes

1.2 Document outline

The document has the following structure:

- Section 2 reviews the ongoing activities (industry, open-source and R&D) related to orchestration;
- Section 3 provides a high-level description of the SliceNet orchestration vision;
- Section 4 describes the project's slice information model;
- Section 5 details the orchestration logical architecture;
- Section 6 defines the most important orchestration workflows;
- Section 7 describes the orchestration interfaces;
- Section 8 reports the software prototype being implemented;
- Section 9 concludes the document.

2 Orchestration related activities overview

This section provides an overview of the related orchestration activities in the SDOs, OSCs, Industry and R&D projects.

2.1 3GPP

Regarding 3GPP, the 5G management and orchestration aspects are represented using Network Resource Models (NRMs) and Interface models [2]. NRM can represent all functions and resources of the mobile network whose management is standardized, e.g. NRM for Core Network (CN) functions, NRM for RAN functions. The NRM specifications can be categorized into three parts (Figure 3 illustrates the 3GPP NRM in 5G):

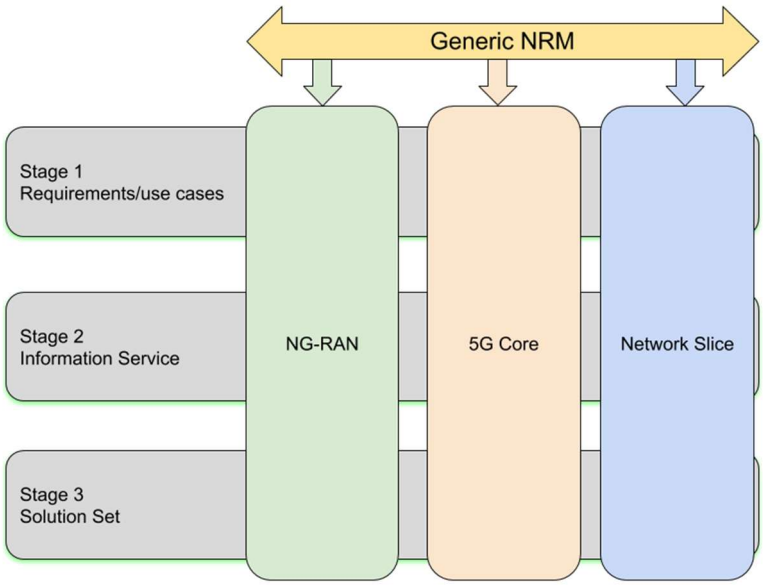

Figure 3: 3GPP NRM in 5G

2.1.1 Stage 1 - Requirements

The 5G NRM specifications define many requirements [3]:

a. NRM generic, without any domain specific definitions like CN entities.
b. Support network management that includes virtualized network functions
c. Requirements for managing NG-RAN, including multi Radio Access Technology (RAT) dual connectivity.
d. Requirements for managing 5G core network functions
e. Requirements for managing Network Slice (NS) and Network Slice Subnet (NSS)

2.1.2 Stage 2 - Information Model definitions

This stage [4] defines the semantic of Information Object Class (IOC) attributes and relations visible on the management interfaces in protocol and technology neutral way. The 5G NRM supports the modelling of NR and NG-RAN, 5G Core, and Network Slice [TS 28.540, TS 28.541, TS 28.542, TS 28.543], in addition to Generic NRM [TS 28.622]. This later mode can be reused or inherited by other domain specific models, e.g. NG-RAN. Figure 3 shows NRM for 5G networks.

a. Generic NRM: it specifies generic network resource information such as Link, Managed Service, Managed Function, subnetwork. This NRM supports also Federated Network Information Model (FNIM) in order to support Fixed Mobile Convergence (FMC) [TS 32.107]
b. NG-NRM: This model covers various 5G radio networks connectivity options, e.g. standalone and non-standalone radio node deployment, as well as architectural options, e.g. NR nodes with or without functional split.
c. 5G Core NRM: It supports the management of 5G core network functions, their respective, in addition to supporting of Access and Mobility Management Function (AMF) Set and AMF Region [TS 23.501].
d. Network Slice NRM: In order to support the Network Slice (NS) and Network Slice Subnet (NSS), an information model for NS and NSS is also supported [TS 28.541].

2.1.3 Stage 3 - Solution Set definitions

This stage [5] maps the information model definitions of stage 2 to a specific solution set definition used for implementation. These definitions for 5G include XML, JSON, and YANG solution set [TS 28.541].

2.2 ETSI

2.2.1 NFV

Started in 2012, the ETSI NFV Industry Specification Group (ISG) has been the driver of the network transformation activities in ETSI, being at the core of the NFV technology definition and standardization, starting from the NFV term itself. Up to now, the ISG has defined the initial framework (including the architecture diagram depicted in Figure 4 and continued in the following phases working on interfaces, information models and testing procedures. Detailed specifications for NFV related workflows, data structures and APIs are now under consolidation in the close-to-be-finished Release 3. At the current stage, the NFV community has produced a wide set of mature and stable specifications, has consolidated their applicability in the industry, and continues to work towards multi-vendor interoperability and addressing new architectural and application challenges. The available NFV standards of Release 3 are already used in the industry to implement NFV products. NFV was originally conceived to help network service providers in the challenge of cost reduction and agility improvement, and it has resulted to be a key framework to enhance how services are requested and consumed by users. It is a necessary component for next-generation networks, and in particular for 5G.

With NFV, network functions are implemented in software and can run on homogeneous, industry-standard commodity infrastructures. This software can then be moved to, or introduced in different

locations in the network as required. NFV simplifies the roll-out of network services, reduces deployment and operational costs and facilitates network management automation.

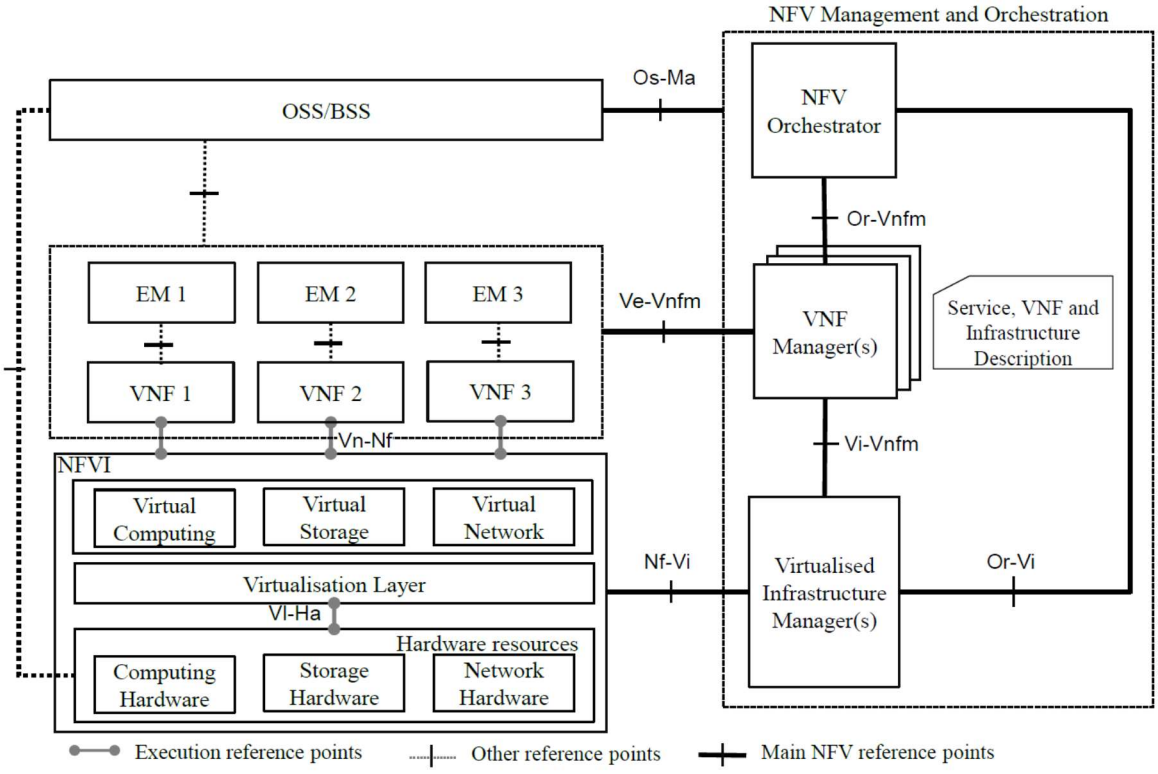

Figure 4: ETSI NFV architectural framework

As illustrated in the architecture diagram in Figure 4, the NFV scope is focused on the lifecycle of the Virtualized Network Functions (VNFs) and the services built by composing them, and with other components generally referred as PNFs (Physical Network Functions), irrespectively of their nature, scope, technology. Therefore, the NFV software-enabled function and service lifecycle management constitutes a new dimension to be considered in the network and service management arena, and it has to be:

- Integrated with other end-to-end management tools and system, such those for network slicing in 5G scenarios
- Considered as an essential enabling technology for network automation and simplification, enabling same services to be easily deployed in multiple instances in distributed locations
- Considered as a key target for new operational architectures and deployments where automation of virtualized environment and the services running on top is required

2.2.2 MEC

The ETSI Multi-access Edge Computing (MEC) ISG aims at creating an open environment where applications from vendors, service providers, and third-parties can be easily integrated across multi-vendor and multi-access Edge Computing platforms. Although all ETSI MEC specifications are defined

to enable a self-contained MEC cloud able to exist in different cloud environments, in most telco environment the real requirement is to extend NFV into the MEC realm, due to the maturity of the NFV specifications and initial compliant products and multi-vendor interoperability trials. For this reason, ETSI MEC has defined a MEC-in-NFV reference architecture in GS MEC 003 [6], which is show in Figure 5.

The MEC-in-NFV architecture takes full advantage of the NFV MANO principles and components and demonstrates how the architecture of ETSI MEC can fit and integrate with it. In particular, most of the ETSI MEC Applications can be treated by ETSI NFV entities as VNFs – even when they are not designed as such. In practice, with the proposed MEC-in-NFV architecture, the NFV scope is extended (from the MANO building blocks up to the virtualized infrastructure management) to cover also the MEC segment and functionalities with minimum impact on the available specifications, APIs, workflows and deployment scenarios. In this direction, Figure 5 also shows a number of interfaces that appear to require coordination and cooperation between ETSI MEC and ETSI NFV (i.e. those in red), resulting in integration complexities.

For what concerns the MEC information model and its mapping to NFV data structures, the MEC descriptor (AppD) for MEC applications, to be defined in the upcoming release of the ETSI GS MEC 010-2 specification, has to be linked to the NFV VNF descriptor (VNFD) for a proper integration of MEC with NFV. With the current NFV specification capabilities in terms of VNF onboarding, ETSI MEC defined AppD can be registered as a Non-MANO NFV artifacts in the NFV catalogues, still requiring dedicated lifecycle management routines within the NFV MANO to treat these artifacts as MEC ones.

In terms of future work, ETSI MEC expects to align with the emerging zero-touch management principles, such as those defined in ETSI Zero touch network and Service Management (ZSM) and update its reference architecture accordingly.

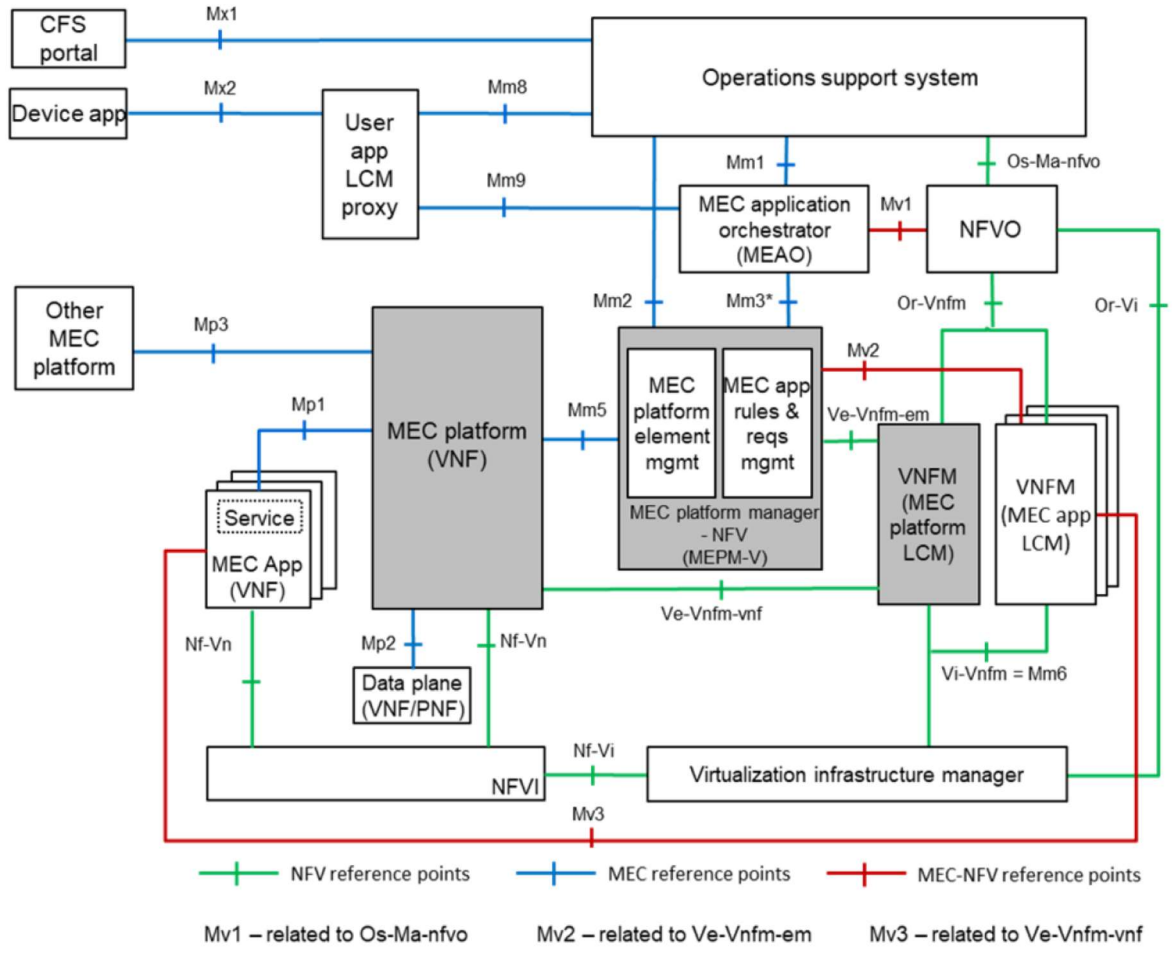

Figure 5: ETSI MEC-in-NFV reference architecture (source: ETSI MEC [6])

2.2.3 ZSM

The ETSI Zero touch network and Service Management (ZSM) ISG is aiming at defining a new, horizontal and vertical end-to-end operable framework enabling agile, efficient and qualitative management and automation of emerging and future networks and services. ZSM targets a network and service management architecture where all operational processes and tasks (e.g., delivery, deployment, configuration, assurance, and optimization) are executed automatically, ideally with full automation in multi-vendor environments. The work of ETSI NFV and ETSI MEC solves dedicated aspects of network and service management, and have defined management capabilities for their respective focus areas. On top of this, ETSI ZSM aims to provide a holistic end-to-end network and service management concept which, among others, enables the integration of NFV and MEC management demands.

The ZSM framework reference architecture is built around a set of building blocks that collectively enable construction of more complex management services and management functions. The clear identification and separation of management domains provide means to isolate management duties (possibly referring to very heterogeneous technologies), considering boundaries of different nature (technological, administrative, geographical, etc.). Each management domain provides a set of ZSM management services, realized by functions that expose and/or consume a set of end-points. An end-to-end service management domain is a special management domain responsible for the cross-domain

management and coordination, which glue all of the single domain management services, functions and end-points

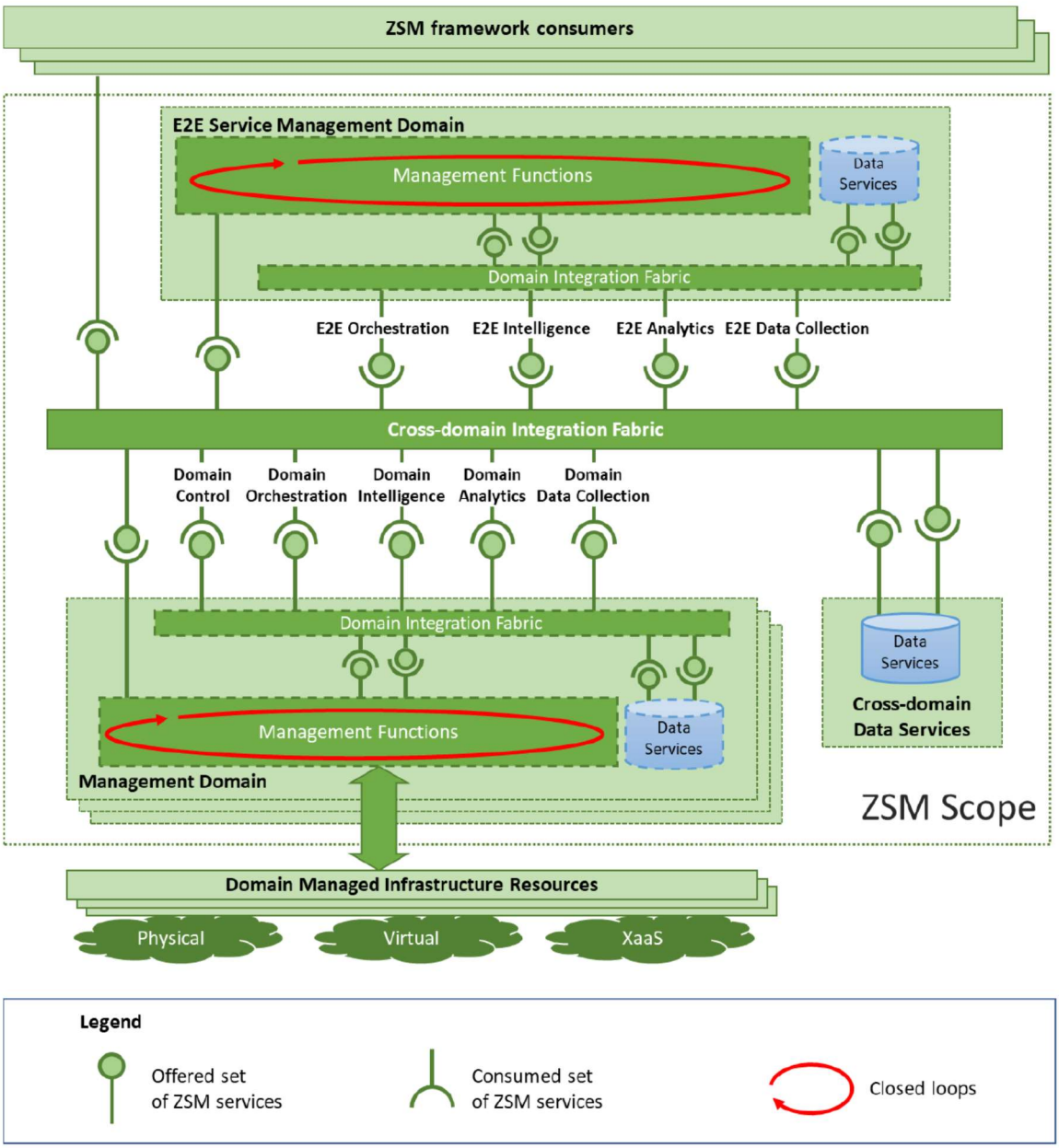

Figure 6: ETSI ZSM reference architecture (source ETSI ZSM 001 [8])

At the core of the ZSM architecture there is a cross-domain integration fabric, which facilitates the provision of services and the access to them through the related end-points across the various domains. It also includes specific services for the communication between management functions, which enable the exchange of management data to consumers. In addition, the cross-domain data services provide the mean to persist data and access it. According to ZSM principles, management services can be logically grouped according to the functionality offered (such as data collection, analytics, intelligence, orchestration, control). Figure 6 depicts the ZSM framework reference architecture. Due to its native openness and architecture flexibility, the ZSM framework is able to integrate management services as implemented by Open Source framework or other SDOs. Examples

include APIs from the TMForum ODA, management services as defined by 3GPP and from ETSI NFV. The possibility to flexibly compose management services, together with the exchange management data provide the foundation of closed loop automation.

2.3 Industry

With 5G, the telecommunication industry is more and more looking at comprehensive orchestration solutions to ease the deployment of heterogeneous vertical services and network slices across several technology domains. Indeed, the 5G network slicing principles and solutions are embracing heterogeneous technologies, spanning from 5G New Radio (5G NR) access technologies, to converged optical fronthaul and backhaul with SDN, and hybrid 4G LTE and 5G Core services. Each of these segments and technology has normally its own stack of management and control tools and protocols that need to be tightly integrated for being able to automatize the provisioning of end-to-end services and slices.

Moreover, the service and slice deployment processes have to cope with geographical distributed Point of Presence (PoP) locations where vertical applications as well as 5G related network functions can be dynamically deployed and migrated according to the specific vertical use cases. These normally include thousands of edge locations and tens of core locations for each telco operator.

As a consequence, the telecommunication industry is going in the direction of fulfilling these requirements with holistic orchestration solutions that integrate under a unique umbrella different kind of management tools for their products and services, as SliceNet is proposing with its cognitive slice management and orchestration platform. While most of the key industry players (both telco operators and vendors) are also participating in open source initiatives in the area of 5G network slices, NFV and SDN orchestration, all of them are anyway delivering their own products with more and more support of standard protocols and APIs.

This is evident in the context of the ETSI NFV Plugtests events for example [11], where in the last couple of years few NFV interoperability events have been organized by ETSI to validate the interworking of multi-vendor NFV-related products and interworking based on NFV APIs.

Among the European solution providers, Ericsson and Nokia are key players in the network slice orchestration arena, and both are offering to their customers end-to-end slice orchestration solutions to facilitate the delivery of 5G services customized for different vertical requirements.

Ericsson offers its Ericsson Orchestrator product as an integration of NFV Orchestration, Generic VNF management and Service Orchestration and Configuration Management features in support of end-to-end network slice coordination and provisioning [12]. In particular, for the emerging 5G use cases it provides network slice management by using TOSCA as the template language for cross-domain orchestration across access, transport, and core segments by interfacing with different network domain managers and transport SDN controllers.

Similarly, Nokia's network slice and NFV orchestration portfolio is built around the CloudBand product [13], which is an ETSI NFV MANO system with commercially proven reliability, automation, repeatability and security. It is flexibly deployed for any combination of CloudBand Infrastructure Software, CloudBand Application Manager, and CloudBand Network Director which provide respectively NFVI/VIM, VNF Management and NFV Orchestration functions. The CloudBand suite can be also integrated with the WaveFabric network slicing solution [14] for creating a flexible, scalable and dynamically reconfigurable virtual optical network spanning across access, metro and core

segments. Service driven and multi-layer flow steering enable Nokia WaveFabric to deliver efficient and cost-effective transport services that meet the diverse requirements of 5G services and verticals.

2.4 Open Source

2.4.1 Open Source MANO

Open Source MANO [15] is an open community-driven project led by operators, whose goal is to develop an NFV management and orchestration system aligned with the ETSI NFV standardization body. The main goal of OSM is to achieve end-to-end network service deployment for telco services through the orchestration and life-cycle management of VNFs, and local network services. To do this, OSM relies on four main aspects [16]:

- A well-defined information model, which is aligned with ETSI NFV, that allows modelling telco-oriented complex network services for automated deployment.
- A unified NorthBound Interface (NBI) that allows for the control, operation and supervision of the life-cycle of the network services and slices.
- An extended concept of 'Network Service' that spans across virtual, physical and transport domains, thus allowing for the control of the full end-to-end network service by transparently interacting with VNFs, PNFs and transport connections.
- Support for Network Slice management.

In this context, OSM is able to provide network as a service (NaaS) functionality in two ways, namely Network Service and Network Slice Instance (NSI). A Network Service is, in OSM, a set of interconnected network functions (VNFs and/or PNFs). OSM provides an interface, which is exposed by the NBI, to configure such Network Service. This API relies on a set of descriptors that follow the OSM information model, which is in turn compliant with ETSI standards. This modelling enables the creation of Network Service templates that can be further parameterized to create different instances of the service, each one having its own lifecycle. The NSI is the enabler for OSM to provide Network Slices as a service. In this context, OSM assumes the role of Slice Manager [18][19]. A NSI is seen here as a composition of Network Services that can be treated as a single entity.

To provide such Network Service and NSI functionalities, OSM consumes the services offered by the Virtual Infrastructure Managers (VIMs) and the WAN Infrastructure Managers (WIMs) it is connected to. VIMs provide the computation facilities (e.g. VMs) where the VNFs reside and the intra-DC connectivity, and WIMs offer the transport network configuration (e.g. inter-DC connectivity). OSM introduces here the concept of end-to-end network services that can span several domains.

Figure 7 depicts an overview of OSM and its relation to the other elements that take part in the NFV architectural framework [20]. At the bottom of the figure, the VIMs and WIMs are the elements that provide the infrastructure where the network functions will be hosted. OSM supports different VIM (e.g. OpenStack, OpenVIM, etc.) and WIM (e.g. ONOS, OpenDaylight, etc.) technologies. In the middle of the figure, the OSM provides the MANO functionalities to manage the life cycle of the network services and NSIs. As highlighted before, one of the key aspects of OSM is the definition of an information model, which is aligned with ETSI NFV. This information model is aimed to provide a unified view of the NSs, which are commonly composed of a set of heterogeneous functions (either virtual or physical) coming from different infrastructure management technologies, providers, etc. On top of the figure, the OSM NBI relies on this OSM information model to provide a set of API invocations and descriptors that allow creating the Network Services in a simple and effective way by the system clients (e.g. OSS and BSS).

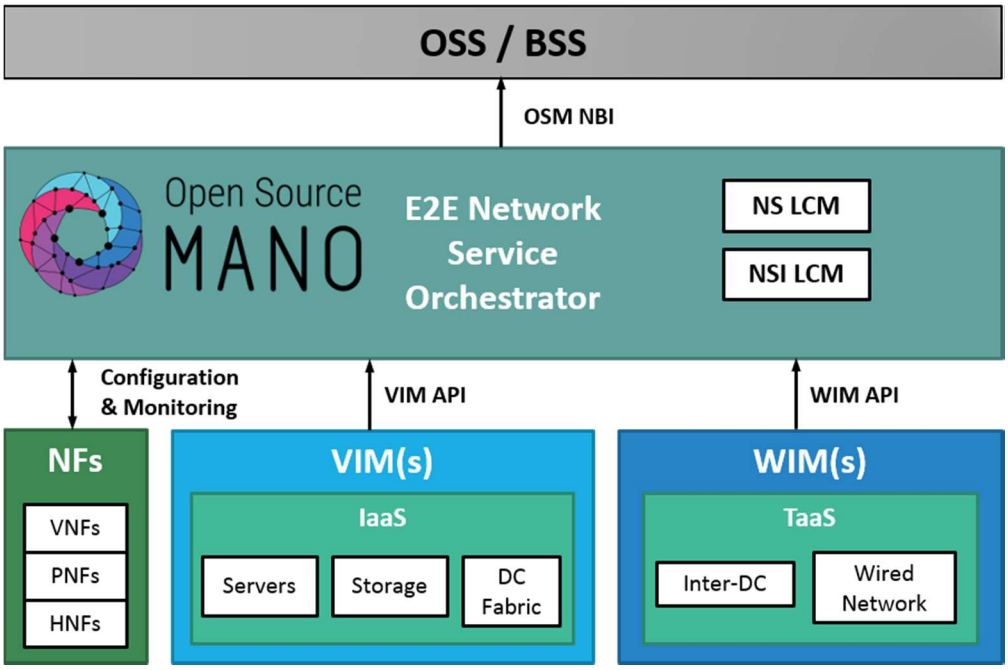

Figure 7: OSM general view

As mentioned above, OSM is currently targeting the support for NS as a service by means of the NSI element. Unfortunately, this is not sufficient to cope with the goal of SliceNet, which is to provision end-to-end cognition-based vertical-oriented services in support of 5G use cases. To achieve this, the SliceNet orchestration plane is divided into three levels of orchestration, namely vertical service, slice and resource, which are thoroughly explained in the forthcoming sections. In its current status, OSM is able to provide network services over the physical and virtual infrastructure of operators. These capabilities can cope with the resource level orchestration required by SliceNet. However, a more sophisticated slice orchestration than the one currently provided by OSM is needed for SliceNet purposes. Furthermore, the vertical service level is not considered in OSM. Hence, being OSM a mature and well-supported NFV-O, it has been chosen to be a core part of the resource level orchestration in SliceNet. In this context, OSM will be responsible for the configuration, lifecycle management and exposure of network services provided by the NSP. These network services will be part of the Network Slices that will be managed by the SliceNet Slice Orchestrator. In the upper level, the Network Slices will compose the vertical service that will be orchestrated by the Service Orchestrator.

2.4.2 ONAP

The ONAP Platform [21] enables product-independent capabilities for design, creation and lifecycle management of network services. ONAP uniquely provides a unified operating framework for vendor-agnostic, policy-driven service design, implementation, analytics and lifecycle management for large-scale workloads and services. With ONAP, network operators can synchronously orchestrate PNFs and VNFs.

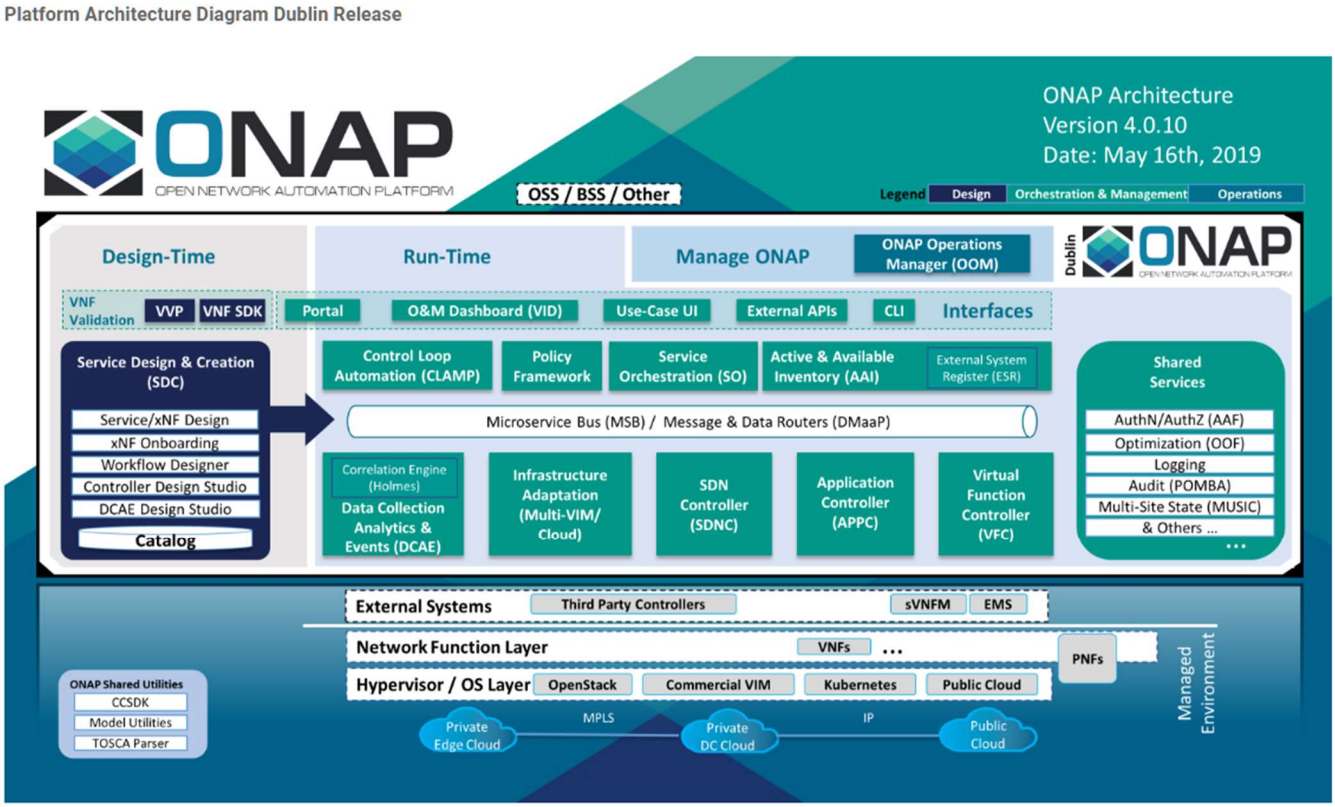

Figure 8: ONAP Architecture (Dublin release)

As a cloud-native application that consists of numerous services, ONAP requires sophisticated initial deployment as well as post-deployment management. The ONAP Operations Manager (OOM) is responsible for orchestrating the end-to-end lifecycle management and monitoring of ONAP components. It is integrated with the Microservices Bus, which provides service registration/discovery and support for internal and external APIs and key SDKs. OOM uses Kubernetes to provide CPU efficiency and platform deployment. In addition, OOM helps enhance ONAP platform maturity by providing scalability and resiliency enhancements to the components it manages.

The platform provides tooling for service designers as well as a model-driven run-time environment, with monitoring and analytics to support closed-loop automation and ongoing service optimization. Both design-time and run-time environments are accessed through the Portal Framework, with role-based access for service designers and operations personnel.

The design-time framework provides a comprehensive development environment with tools, techniques, and repositories for defining and describing resources, services, and products. This includes policy design and implementation, as well as an SDK with tools for VNF supplier packaging and validation.

The run-time environment executes the rules and policies distributed by the design and creation environment, as well as the Controllers that manage physical and virtual networks. The Active & Available Inventory (A&AI) component provides real-time views of a system's resources, services, products and their relationships with each other. In a fast-moving environment with rapid deployment and teardown of virtual resources, this real-time monitoring and mapping is critical to service assurance.

The run-time service execution components are in constant communication with the closed-loop automation modules, which provide real-time monitoring, analytics, alarm and event correlation, etc.

2.5 H2020 R&D Projects

2.5.1 5G Transformer

5G-TRANSFORMER is a 5G-PPP Phase 2 project that aims at developing an SDN/NFV-based platform for the delivery of vertical-tailored network slices as a mean to facilitate verticals industries in provisioning their services over mobile transport networks. In particular, 5G-TRANSFORMER enables verticals to easily meet their service requirements through customized 5G end-to-end slices. This is done through aggregation and federation of transport networking and computing fabric resources from the edge up to the core and cloud, for creation and management of slices on a federated and virtualized infrastructure.

This requires a transformation of current mobile transport networks into an SDN/NFV-based Mobile Transport and Computing Platform (MTP), bringing network slicing as a core paradigm for provisioning MTP slices tailored to the specific needs of vertical industries. To do this, the 5G-TRANSFORMER platform is based on extensions of the ETSI NFV MANO architecture: as depicted in Figure 9 below, it consists of three main functional components:

i. A Vertical Slicer (VS) for service and slice management [22].

ii. A Service Orchestrator (SO) for service and resource orchestration in multi-domain, federated scenarios [23].

iii. The MTP, representing the underlying unified transport stratum, responsible for providing the networking and computing resources required by the NFV NS orchestrated by the SO [24].

The main touching points of 5G-TRANSFORMER and SliceNet (targeting its orchestration framework) are the SO and the VS principles and functionalities. For what concerns the SO, 5G-TRANSFORMER addresses end-to-end service and resource orchestration across different administrative domains through a federation paradigm. In particular, end-to-end services are split into multiple segments deployed in different administrative domains, based on service requirements and resource availability. Federation is managed at the interface between SOs belonging to different domains and handling abstraction of services and resources. Therefore, it is clear that 5G-TRANSFORMER follows an approach with a fully distributed end-to-end service orchestration, without any clear separation of roles between Digital Service Providers (DSPs) and Network Service Providers (NSPs) as in the case of SliceNet, where more north-south interactions (i.e. between DSPs and NSPs) are foreseen rather than east-west (i.e. among NSPs).

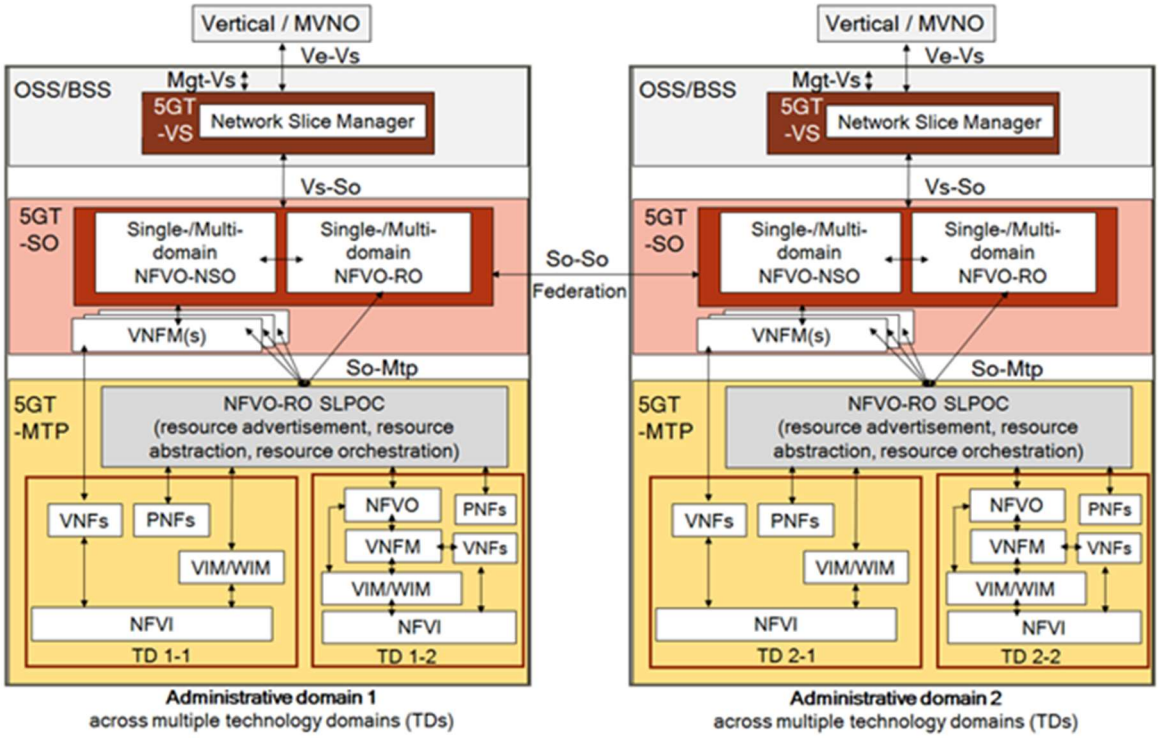

Figure 9: 5G-TRANSFORMER architecture

On the other hand, the VS aims at facilitating the specification, instantiation, monitoring and management of vertical services through network slices, and it creates and maps the services onto network slices according to the vertical requirements, managing their lifecycle. In this case, the VS can be considered as valuable reference baseline for the SliceNet orchestration at the DSP level, in terms of end-to-end network slicing features and lifecycle workflows. Following the VS approach, the SliceNet DSP orchestrator could translate the vertical experiments, service and slicing requests into per-domain slices and NFV Network Services to be deployed in the proper NSP domains.

2.5.2 SELFNET

SELFNET is a project focusing on 5G network management, with the main objective of developing an efficient self-organizing network management framework for 5G through the combination of a virtualized and software defined network infrastructure with artificial intelligence technologies, pursuing automated network monitoring, autonomic network maintenance, automated deployment of network management tools and automated network service provisioning.

SELFNET is driven by use cases designed to address major network management problems including Self-protection capabilities against distributed cyber-attacks, Self-healing capabilities against network failures, and Self-optimization to dynamically improve the performance of the network and the Quality of Experience (QoE) of the users.

SELFNET has the specific objectives of designing, implementing and validating a self-monitoring and detection subsystem, a distributed Self-Organising Network (SON) autonomic management engine subsystem and a SON orchestration and virtual infrastructure management subsystem. Through these automated and intelligence-based operations, SELFNET primarily contributes to significantly reducing service creation time in software-defined and virtualised 5G networks. Moreover, SELFNET expects to

help in realising the creation of a secure, reliable and dependable network with a “zero perceived” downtime for services.

In the Figure 10, the SELFNET reference architecture is depicted. The architecture is divided into 6 parts:

- Infrastructure Layer,
- Virtualized Network Layer,
- SON Control Layer,
- SON Autonomic Layer,
- NFV Orchestration and Management Layer,
- Access Layer.

The architecture is aligned with the most relevant standards which are the foundations of the project namely ETSI NFV, Open Networking Foundation (ONF), and TMForum.

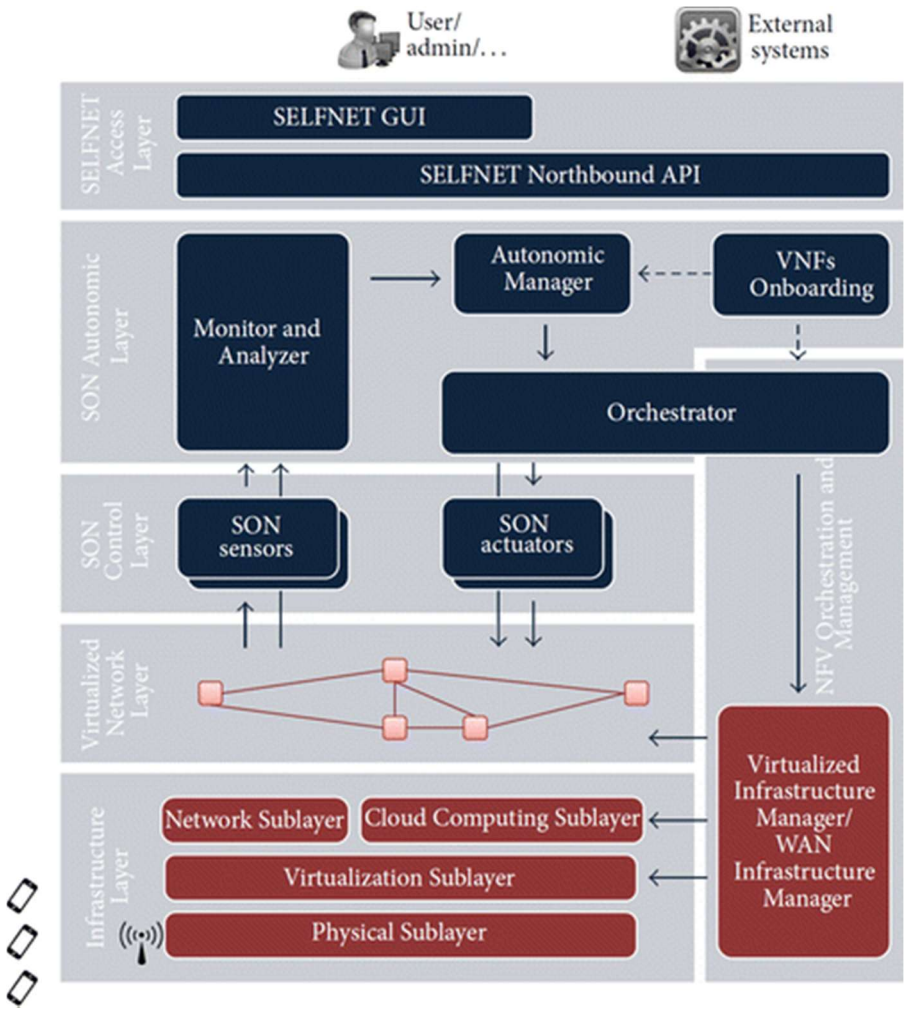

Figure 10: SELFNET reference architecture

Apart from SDN and NFV, SELFNET further explores Self-Organizing Networks (SON) for automatic 5G network management tasks. SON solutions in LTE (or LTE-Advanced) networks are typically classified into three categories including self-configuration, self-optimization, and self-healing. Self-configuration refers to the dynamic plug-and-play configuration capability of newly deployed LTE eNBs, whilst self-optimization enables already deployed eNBs to automatically adapt to radio conditions and network loads. The self-healing function attempts to recover from temporary bottlenecks or failures in the network, for example, eNB outage.

SliceNet differs from SELFNET in that SliceNet focuses on the orchestration of network slices with a clear separation of roles between Digital Service Providers (DSPs) and Network Service Providers (NSPs). In addition, SliceNet includes DSP and NSP data lakes that enables efficient sharing of data between SliceNet components, among other differences.

3 SliceNet Orchestration Approach

This chapter focuses on describing the SliceNet orchestration vision (section 3.1) and on listing the requirements to be taken into consideration for its design, implementation and validation (section 3.2).

3.1 Orchestration vision

One of SliceNet key pillars is to deliver end-to-end communication capabilities to verticals over multiple network providers / administrative domains. Towards this goal, as identified in deliverable D2.4 [25], SliceNet vision encompasses multiple business roles as follows:

- **Digital Service Customer (DSC) / Vertical**: subscribes and consumes end-to-end communication services from the Digital Service Provider (DSP); the Vertical should indicate only the required information about the service (e.g. end devices location, bandwidth requirements, latency requirements, etc.), without having to understand the network details and how the end-to-end service will be composed and offered;
- **Digital Service Provider (DSP)**: responsible for managing the end-to-end services lifecycle, including their creation and exposition to the Verticals, as well as their decomposition (during provision) into one or more Network Slices (NSs) across one or multiple network providers / administrative domains;
- **Network Service Provider (NSP)**: responsible for managing the NSs and involved network resources lifecycle, including their creation and exposition to the DSP (or DSPs), as well as their provision, monitoring and optimization.

Figure 11 illustrates the SliceNet business actors and their main interactions which can be aggregated in two core groups:

- **Actuation** (represented in orange) – provides all the functionalities and interfaces required for offering, provisioning and optimizing end-to-end services and NSs;
- **Supervision** (represented in grey) – provides all the functionalities required for monitoring the subscribed end-to-end services and NSs.

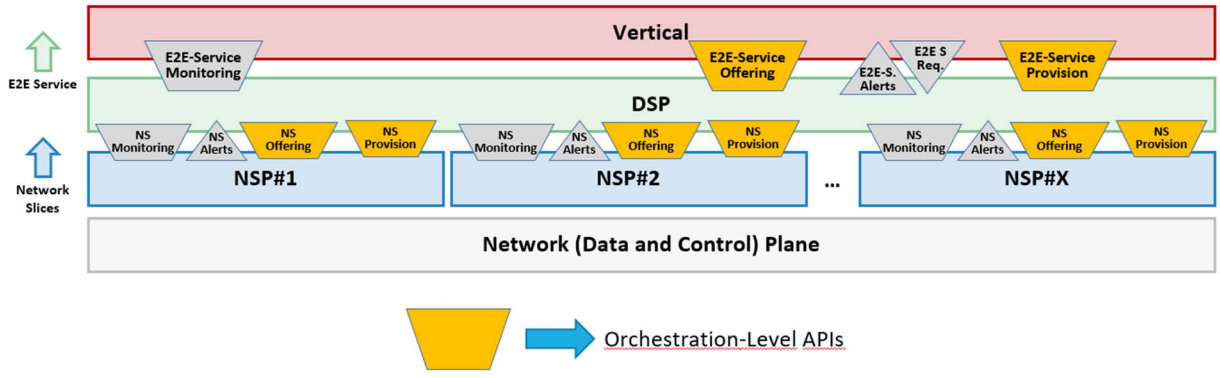

Figure 11: SliceNet Business Roles and Main Interactions

Concerning orchestration procedures, which is the focus of this deliverable, they are the core function of all the actuation procedures represented in Figure 11 (and not on supervision ones), either they are

related with NS and/or end-to-end-NS offering, provisioning and/or optimization. Therefore, these functionalities are mandatory at both SliceNet technical roles - DSP and NSP. From the SliceNet architecture perspective, two sub-planes are involved in the actuation procedures:

1. **Orchestration**: handles all the orchestration-related procedures;
2. **Information**: provides catalogue and inventory services.

The following subsections highlight the role and internals of each one of the aforementioned subplanes when instantiated at the NSP and at the DSP.

3.1.1 NSP level Orchestration

The NSP is responsible for managing (onboard, offer, provision, monitor, optimize) the Network Slices, NFV Network Services and all the underlying physical and virtual resources. Figure 12 illustrates the Orchestration and Information sub-planes at the NSP business role. To deliver the required SliceNet functionalities at the NSP-side, the following functional components (under the responsibility of task 7.1) are part of the Orchestration sub-plane:

3. **Service and Slice Orchestrator (SS-O)**: manages the association between the requested Network Slices by the DSP and the underlying NFV Network Services and resources;
4. **NFV, MEC and RAN Orchestrator (NMR-O)**: acts as a NFV NS and resource orchestrator, taking care of coordinating the lifecycle management of NFV NS instances composed by the combination of VNFs, PNFs, MEC applications interconnected by means of forwarding graphs.

From the Information sub-plane, the following logical architecture components are available:

1. **NS & NSS Catalogue**: manages the NSs and NSSs descriptors;
2. **Resource Catalogue**: manages the resources (VNF, PNF) descriptors;
3. **NS & NSS Inventory**: manages the NSs and NSSs instances;
4. **Resource Inventory**: manages the resources (VNF, PNF) instances.

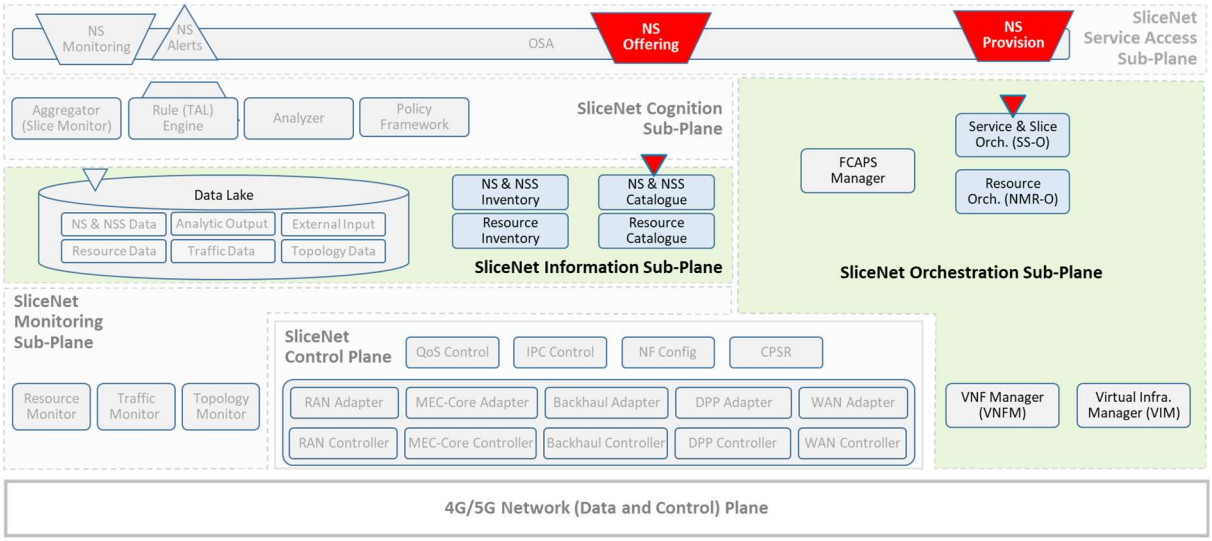

Figure 12: Orchestration and Information sub-planes at the NSP (Standalone Deployment)

3.1.2 DSP level Orchestration

The DSP main responsibility is to manage (onboard, offer, provision, monitor, optimize) end-to-end Network Slices delivered across multiple administrative domains. Figure 13 illustrates the

Orchestration and Information sub-planes at the DSP business role. To deliver the required SliceNet functionalities at the DSP-side, the following functional components are part of the Orchestration sub-plane (under the responsibility of task 7.1):

1. **Service and Slice Orchestrator (SS-O)**: handles the association of services offered to verticals and other service providers with network slices and their correspondent management functions. The SS-O at the DSP is also responsible for the end-to-end orchestration of services across multiple domains.

From the Information sub-plane, the following logical architecture components are available:

1. **end-to-end Network Slice Catalogue**: manages the end-to-end Networks Slices descriptors;
2. **end-to-end Network Slice Inventory**: manages the end-to-end Network Slices instances.

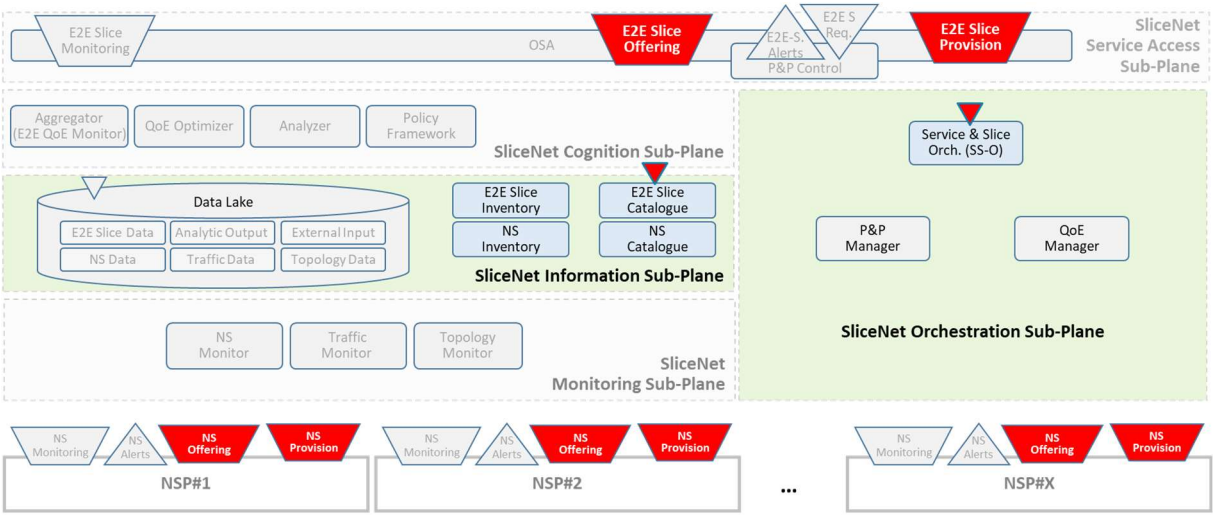

Figure 13: Orchestration and Information sub-planes at the DSP (Standalone Deployment)

3.1.3 DSP & NSP Deployment Combinations – Impact on Orchestration

The orchestration architectures described in sections **Error! Reference source not found.** and **Error! Reference source not found.** for the NSP and the DSP, respectively, are applied when the DSP and NSP business roles are independent (standalone deployment). In scenarios in which the DSP and the NSP business roles are fulfilled by the same business entity, the DSP and NSP orchestration architecture components are combined and therefore simplified. Figure 14 depicts the DSP and NSP combined deployment. In this case, DSP and NSP#1 are the same business entity, whereas NSP#2 and NSP#3 business roles are fulfilled by other service providers.

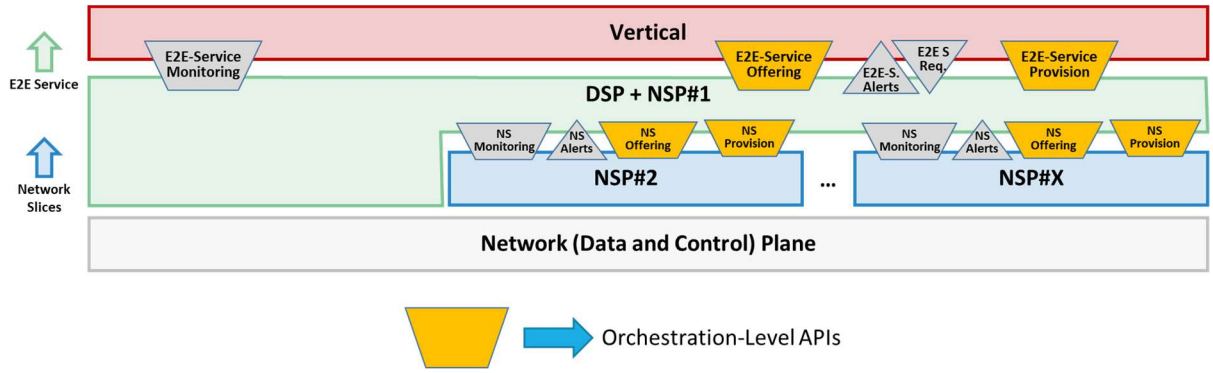

Figure 14: DSP and NSP deployment combinations

From an architectural perspective, in the DSP and NSP combined deployment, there is a single SS-O to manage and orchestrate the end-to-end-NSs, the NSs and the NSSs. Also the end-to-end-NSs, NSs and NSSs catalogues and inventories are owned by the same business entity. Figure 15 illustrates the DSP and NSP architecture in the combined deployment.

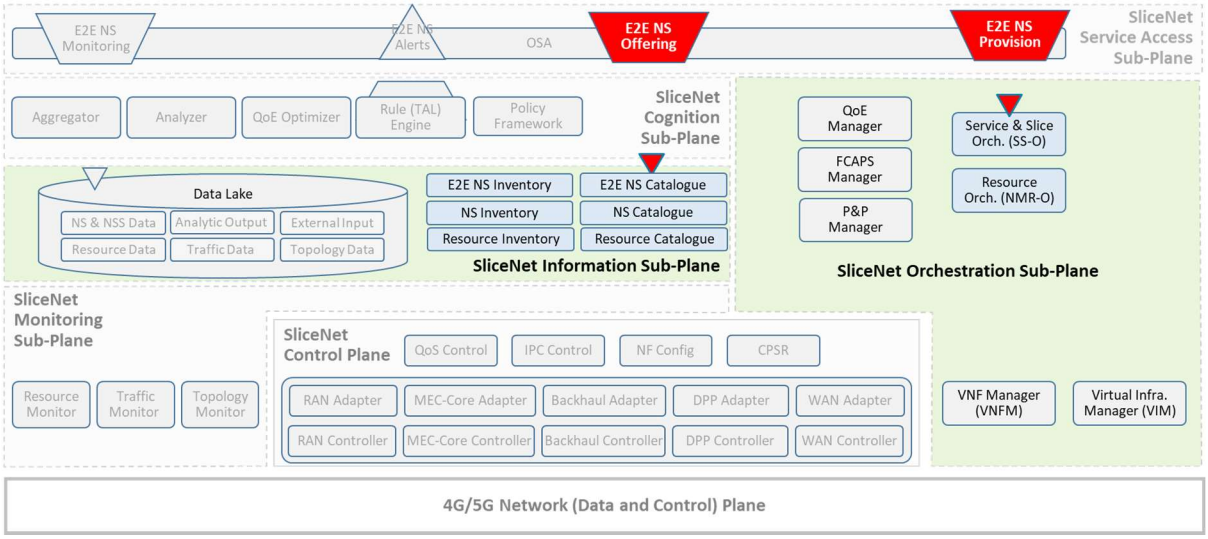

Figure 15: Orchestration and Information sub-planes at the DSP & NSP (Combined Deployment)

3.2 Orchestration requirements

As part of the SliceNet orchestration architecture definition, the identification of specific requirements is key to drive the functional decomposition and the definition of the building blocks of the architecture. Three levels of orchestration are envisioned in SliceNet, namely Service, Slice and Resource, which are implemented in three different functional modules: the Service Orchestrator (i.e. SS-O at DSP), the Slice Orchestrator (i.e. SS-O at NSP level) and the Resource Orchestrator (i.e. NMR-O at NSP level). For each of these levels a set of requirements has been defined.

The SS-O, at both DSP and NSP levels, is involved in the vertical service and network slice lifecycle at different phases, namely:

- Discovery phase - the SS-O capabilities supported by vertical services (at DSP level) and network slices (at NSP level) are computed and exposed to the verticals (at DSP level) and to

the DSPs (at NSP level) as offers. The NSP level offers are built over the resources exposed by the NMR-O.

- Fulfilment phase - the SS-O provides the mapping from the customer requests (i.e. vertical services at DSP level and network slices at NSP level) to end-to-end network slices (at DSP level) or network slice resources (at NSP level), taking care to provision them and manage their runtime lifecycle. At resource level, the NMR-O provides the network services to support the network slice.
- Assurance phase - the SS-O at both DSP and NSP levels takes care to support the vertical services and network slices monitoring, enabling the SliceNet FCAPS framework to access the vertical service and network slice instances information. Similarly, the NMR-O enables network service components monitoring at NSP level.
- Decommissioning phase - the SS-O participates and coordinates the release of network slices and related resources (through the NMR-O) whenever a vertical service or an end-to-end network slice is terminated.

In turn, the SS-O at NSP relies on the NMR-O, which is responsible for managing the infrastructure resources, to provide such management at a lower level.

For each phase and orchestration level, specific and dedicated requirements are identified. The following subsections list the orchestration requirements at vertical service, network slice and resource levels.

3.2.1 Vertical service level requirements

The Vertical service level requirements are listed in the following subsections, grouped in discovery, fulfilment, assurance and decommissioning requirements. The main functional component involved in these requirements identification is the Service Orchestrator, i.e. the SS-O at the DSP level.

3.2.1.1 Discovery requirements

Table 1 describes the service level orchestration discovery requirements.

Table 1: Service level orchestration discovery requirements

ID	Requirement Description
ReqSSO.DSP.Di.1	**Vertical Service Offers exposure** - The Service Orchestrator shall offer APIs towards the vertical to offer Vertical Service Blueprints to be customized by the vertical, and Vertical Service Descriptors
ReqSSO.DSP.Di.2	**Network Slice Offers collection** - The Service Orchestrator shall collect from NSPs Slice orchestrators the available Network Slice Descriptors (each NSP could/should expose also who are the potential peering NSPs)
ReqSSO.DSP.Di.3	**Vertical Service and Network slice descriptors association** - The Service Orchestrator shall keep the association among the network slices descriptors/templates gathered from the NSPs and the VSBs and VSDs.

ReqSSO.DSP.Di.4	**Private and public vertical service offers** - The Service Orchestrator should support both private (i.e. towards restricted set of verticals) and public exposure of vertical service offers.
ReqSSO.DSP.Di.5	**Vertical Service Descriptor Instantiation** - The Service Orchestrator shall allow a vertical to create several instances of the same VSD.
ReqSSO.DSP.Di.6	**Vertical Service Descriptors Onboarding** - The Service Orchestrator shall enable a vertical to store its service descriptions persistently, and to create, retrieve, update, and delete VSDs.
ReqSSO.DSP.Di.7	**Vertical Service Blueprints Onboarding** - The Service Orchestrator shall allow the DSP administrators to onboard and manage VSBs.

3.2.1.2 Fulfilment requirements

Table 1 describes the service level orchestration fulfillment requirements.

Table 2: Service level orchestration fulfillment requirements

ID	Requirement Description
ReqSSO.DSP.Fu.1	**Expose CRUD operations on Vertical Services** - The Service Orchestrator shall offer APIs to manage Vertical Services and end-to-end network slices. Verticals shall be able to create, modify and terminate Vertical Services.
ReqSSO.DSP.Fu.2	**Expose CRUD operations on end-to-end network slices** - end-to-end network slice runtime modification APIs (as part of FCAPS and Cognitive optimization) shall also be exposed to other DSP-level components
ReqSSO.DSP.Fu.3	**Translation of Vertical Service into Network Slices** - At Vertical Service activation/instantiation time, the Service Orchestrator shall select the proper Network Slice offers to activate/instantiate from the NSPs in order to fulfil the vertical requirements.
ReqSSO.DSP.Fu.4	**Network Slices sharing** - Existing Network Slice instances could be re-used (re-use could be in the scope of the same vertical, requesting more services to the same DSP)
ReqSSO.DSP.Fu.5	**Activation/Instantiation of single-domain Network Slices** - At Vertical Service activation/instantiation time, the Service Orchestrator shall interact with NSPs Slice Orchestrators to activate the correspondent slices (and providing any required runtime information)
ReqSSO.DSP.Fu.6	**Control/Management of Stitching of single-domain Network Slices into end-to-end multi-domain Network Slice** - The Service Orchestrator shall be able to request/issue the proper cross-domain slice configurations towards the single-domain NSPs slice orchestrators for assuring the required end-to-end QoS

ReqSSO.DSP.Fu.7	**CRUD operations on single-domain Network Slices (client-side)** - The Service Orchestrator shall be able to invoke the CRUD operations offered by the Slice Orchestrators in the NSPs. Therefore it shall implement the client-side of those single-domain slice lifecycle management APIs for creation, modification, termination, fault management, performance monitoring.
ReqSSO.DSP.Fu.8	**Activation of Vertical Plug & Play control instance** - The Service Orchestrator, upon successful instantiation of the end-to-end Vertical Service, shall activate the vertical P&P control instance through the P&P Manager, by indicating the required level of control exposure agreed with the vertical
ReqSSO.DSP.Fu.9	**Expose interface to QoE optimizer to apply optimization actions** - The Service Orchestrator shall offer dedicated APIs for QoE optimization purposes and end-to-end network slices runtime modifications to support the cognition loops

3.2.1.3 Assurance requirements

Table 3 describes the service level orchestration assurance requirements.

Table 3: Service level orchestration assurance requirements

ID	Requirement Description
ReqSSO.DSP.As.1	**Vertical service management isolation** - The Service Orchestrator should provide isolation among vertical services workflows and requests status coming from different vertical actors.
ReqSSO.DSP.As.2	**Vertical service monitoring** - The Service Orchestrator, upon creation of new vertical services, shall expose information about the new instances to the FCAPS framework to enable DSP monitoring jobs for the vertical service and to assure the verification of the vertical driven KPIs.
ReqSSO.DSP.As.3	**End-to-end network slice monitoring** - The Service Orchestrator shall ensure to request to NSP Slice Orchestrators the monitoring of proper KPIs in the per-domain network slices as a way to fulfil the vertical driven KPIs.
ReqSSO.DSP.As.4	**Vertical service arbitration** - The Service Orchestrator shall arbitrate network slices among vertical service instances of different verticals based on priorities, policies, SLA, and service requirements

3.2.1.4 Decommissioning requirements

Table 4 describes the service level orchestration decommissioning requirements.

Table 4: Service level orchestration decommissioning requirements

ID	Requirement Description

ReqSSO.DSP.De.1	**Network slices termination** - The Service Orchestrator shall be able to identify the network slices to be decommissioned as a result of a VSI termination.
ReqSSO.DSP.De.2	**Monitoring deactivation** - The Service Orchestrator shall be able to identify the monitoring mechanisms to be deactivated as a result of a VSI termination.
ReqSSO.DSP.De.3	**Network slices termination awareness** - The Service Orchestrator shall implement means for receiving acknowledgement of releasing network slices from the Slice Orchestrators.
ReqSSO.DSP.De.4	**Vertical service termination notification** - The Service Orchestrator should be able to notify verticals about their VSIs termination.

3.2.2 Network Slice level requirements

The slice level requirements are listed in the following subsections, grouped in discovery, fulfilment, assurance and decommissioning requirements. The main functional component involved in these requirements identification is the Slice Orchestrator, i.e. the SS-O at the NSP level.

3.2.2.1 Discovery requirements

Table 5 describes the slice level orchestration discovery requirements.

Table 5: Slice level orchestration discovery requirements

ID	Requirement Description
ReqSSO.NSP.Di.1	**Network Slice Offers exposure** - The Slice Orchestrator shall offer APIs towards the DSPs Service Orchestrators to offer Network Slice Descriptors to be activated/instantiated. This should include exposure of QoS capabilities, type of slice, coverage area, etc.
ReqSSO.NSP.Di.2	**NSP Resources Awareness** - The Slice Orchestrator shall be aware of NFV, MEC and RAN resources and services available for provisioning and configuration in its NSP domain. This is done by collecting information from NMR-O level catalogue(s)
ReqSSO.NSP.Di.3	**Network slice and per-domain resource descriptors association** - The Slice Orchestrator shall keep the association among the network slices descriptors/templates it exposes to the DSPs and the resource descriptors (NFV, RAN, etc.) for per-domain resource orchestration.
ReqSSO.NSP.Di.4	**Network Slice Descriptor Instantiation** - The Slice Orchestrator shall allow a DSP to create several instances of the same NSD.
ReqSSO.NSP.Di.5	**Network Slice Descriptor Onboarding** - The Slice Orchestrator shall allow the NSP administrators to onboard and manage NSDs.

3.2.2.2 Fulfilment requirements

Table 6 describes the slice level orchestration fulfillment requirements.

Table 6: Slice level orchestration fulfillment requirements

ID	Requirement Description
ReqSSO.NSP.Fu.1	**Expose CRUD operations for Network Slices** - The Slice Orchestrator shall expose APIs for single-domain slice lifecycle management APIs. These should cover creation, modification, termination, fault management, performance monitoring.
ReqSSO.NSP.Fu.2	**Translation of Network Slices into NFV, MEC, RAN services/configurations** - At Network Slice activation/instantiation time, the Slice Orchestrator shall select the proper Resource descriptors available in the NSPs to fulfil the DSP requirements. Existing NFV, MEC and RAN services and instances could be re-used.
ReqSSO.NSP.Fu.3	**Provisioning/configuration of single-domain Network Slices** - At Network Slice activation/instantiation time, the Slice Orchestrator shall coordinate the provisioning of per-resource-domain services and configurations. To do that, it shall interact with NFV orchestrator, RAN controller, any MEC orchestrator. It shall also take care to enforce proper QoS configurations where needed (including backhaul, inter-segment configuration and multi-domain).
ReqSSO.NSP.Fu.4	**QoS configuration in single-domain Network Slices** -The Slice Orchestrator shall take care to enforce proper QoS configurations in the domains involved in the given network slices (including backhaul, inter-segment configuration and multi-domain).
ReqSSO.NSP.Fu.5	**Activation of Plug & Play control instance** - The Slice Orchestrator, upon successful instantiation, shall activate the dedicated P&P control instance through the P&P Manager, by indicating the required level of control exposure agreed with the DSP.
ReqSSO.NSP.Fu.7	**Expose interface to FCAPS to close the loop at NSP level** - The Slice Orchestrator shall offer dedicated APIs for FCAPS optimization purposes and single-domain slice runtime modifications to support the FCAPS loops.

3.2.2.3 Assurance requirements

Table 7 describes the slice level orchestration assurance requirements.

Table 7: Slice level orchestration assurance requirements

ID	Requirement Description
ReqSSO.NSP.As.1	**Network slice management isolation** - The Slice Orchestrator should provide isolation among network slice workflows and requests status coming from different DSPs.

ReqSSO.NSP.As.2	**Network slice monitoring** - The Slice Orchestrator, upon creation of new network slice instances, shall dynamically enable the network slice monitoring jobs in the NSP FCAPS to assure the verification of the KPIs requested by the DSP.
ReqSSO.NSP.As.3	**Network slice arbitration** - The Slice Orchestrator shall arbitrate network slices resources among several network slice instances of different DSPs based on priorities, policies, and slice requirements.

3.2.2.4 Decommissioning requirements

Table 8 describes the slice level orchestration decommissioning requirements.

Table 8: Slice level orchestration decommissioning requirements

ID	Requirement Description
ReqSSO.NSP.De.1	**Network services and slice resource decommissioning** - The Slice Orchestrator shall be able to identify the NFV network services and the slice resources to be decommissioned as a result of a NST.
ReqSSO.NSP.De.2	**Monitoring de-activation** - The Slice Orchestrator shall be able to identify the monitoring mechanisms to be de-activated as a result of a NSI termination.
ReqSSO.NSP.De.3	**Network services and slice resource termination awareness** - The Slice Orchestrator shall implement means for receiving acknowledgement of releasing NFV network services and slice resources from the per-domain resource orchestrators and controllers.
ReqSSO.NSP.De.4	**Network slice termination notification** - The Slice Orchestrator should be able to notify DSPs about their NSIs termination.

3.2.3 Resource level requirements

The resource level requirements are listed in this section following the same split as the previous sections. The functional component responsible for fulfilling them is the Resource Orchestrator.

3.2.3.1 Discovery requirements

Table 9 describes the resource level orchestration discovery requirements.

Table 9: Resource level orchestration discovery requirements

ID	Requirement Description
ReqNMRO.NSP.Di.1	**Network Service Offers exposure** - The Resource Orchestrator has to expose the available resources (in terms of network services) to the Slice Orchestrator at NSP. It has to provide a set of Network Service offers that can be consumed by the Slice Orchestrator which, in turn, will build the Network Slice offer. This

	will be afterwards consumed at DSP level (by the Service Orchestrator) as part of the end-to-end Slice that will support the Vertical Service.
ReqNMRO.NSP.Di.2	**Infrastructure Resources Awareness** - The Resource Orchestrator has to be aware of the resources available at infrastructure level that will support the network services to be offered. This is done by collecting information from the Infrastructure Manager(s) of the NSP.
ReqNMRO.NSP.Di.3	**Network services descriptors and resources association** - The Resource Orchestrator shall keep the association among the NSDs/NSTs it exposes to the Slice Orchestrator and the infrastructure resources to be used.

3.2.3.2 Fulfilment requirements

Table 10 describes the resource level orchestration fulfillment requirements.

Table 10: Resource level orchestration fulfillment requirements

ID	Requirement Description
ReqNMRO.NSP.Fu.1	**Expose CRUD operations for Network Services** - The Resource Orchestrator shall expose APIs for network services lifecycle management. These should cover creation, modification, termination, fault management and performance monitoring.
ReqNMRO.NSP.Fu.2	**Translation of Network Services into NFV, MEC and other network configurations** - At Network Service activation/instantiation time, the Resource Orchestrator shall select the proper resources available in the NSP's infrastructure to fulfill the Network Slice requirements. Existing NFV, MEC and RAN services and instances could be re-used.
ReqNMRO.NSP.Fu.3	**Provisioning/configuration of the NFV and MEC services composing the Network Slice** - At Network Service activation/instantiation time, the Resource Orchestrator shall coordinate the provisioning of NFV and MEC services and configurations, as well as the network connectivity among them. To do that, it shall interact with the infrastructure of the NSP. This may be done through the VIM or other infrastructure management entity used by the NSP.
ReqNMRO.NSP.Fu.4	**Provisioning/configuration of the virtual network infrastructure associated to the Network Slice** - The Resource Orchestration is responsible for the life-cycle management of the Network Functions associated to the virtual infrastructure of the Network Slice.

3.2.3.3 Assurance requirements

Table 11 describes the resource level orchestration assurance requirements.

Table 11: Resource level orchestration assurance requirements

ID	Requirement Description
ReqNMRO.NSP.As.1	**Network service management isolation** - The Resource Orchestrator should provide isolation among network service workflows, requests and resources associated to different network slices.
ReqNMRO.NSP.As.2	**Network service and resources monitoring** - The Resource Orchestrator has to offer network service monitoring information to the NSP FCAPS to assure the verification of the KPIs requested by the upper layers of the system.

3.2.3.4 Decommissioning requirements

Table 12 describes the resource level orchestration decommissioning requirements.

Table 12: Resource level orchestration decommissioning requirements

ID	Requirement Description
ReqNMRO.NSP.De.1	**Network functions and applications decommissioning** - The Resource Orchestrator shall be able to identify network and application functions to be decommissioned as a result of a NST.
ReqNMRO.NSP.De.2	**Monitoring deactivation** - The Resource Orchestrator shall be able to identify the monitoring mechanisms to be deactivated as a result of a NSI termination.
ReqNMRO.NSP.De.3	**Resources de-allocation awareness** - The Resource Orchestrator shall implement means for receiving acknowledgement from the infrastructure manager at NSP level for releasing the resources allocated to a network service.
ReqNMRO.NSP.De.4	**Network service termination notification** - The Resource Orchestrator should be able to notify the Slice Orchestrator about the NSIs termination.

4 SliceNet Information Model

4.1 Vertical service information model

This section describes the vertical service information model that is at the basis of the vertical service lifecycle management implemented by the SS-O at the DSP level. The focus is on the details of the information that shall be conveyed in VSDs and the notations used.

4.1.1 Vertical Service Blueprint

The Vertical Service Blueprint (VSB) is a high level description of a template for an end-to-end network slice service. It includes a high level description of the atomic functional components of the service and their interconnection. Moreover, it contains a set of parameters that can be used to customize the service. The VSB is used to describe in a formal and structured way the service that a vertical needs to deploy. It is provided by the DSP (as part of service design process), possibly interacting with the vertical to gather information about the required service and main capabilities.

In practice, the VSB represents a template of the vertical's service, as it defines a set of parameters that allow the customization of the service in some of its parts. Typically, parameters are used to correctly size the service in terms of resources and requirements. Specific translation rules are defined to take one or more parameter values as input and accordingly select the appropriate Network Slice Templates that will actually implement the vertical service.

The VSB can also be used to produce a graphical rendering of the topology of the service at the One Stop API level. In fact, while the textual format of the VSB aims at being human friendly, it can be difficult to understand the connections between the functional components of the blueprint.

To represent the service, the VSB makes use of three main pillars:

- Atomic functional components: they represent the building blocks of the service. Usually, each corresponds to a Network Service, a VNF or a PNF.
- End points: they are the anchor points to connect the atomic components together. They are included in the description of the related atomic component but they are also described separately as they can have specific properties.
- Connectivity services: they describe how endpoints and thus atomic components are connected to each other to create virtual links.

Figure 16 shows a graphical representation of a VSB inspired by the SliceNet eHealth Use Case. The green boxes are the atomic functional components, the black lines represent the endpoints, and the blue circles are the connectivity services. The ambulances are also shown in the figure and they can be considered as UEs. The vertical service is composed by a RAN component, by the two Telestroke assessment and BlueEye VNFs running in the edge which require an EPC service to connect to the hospital.

Table 13 reports the fields composing a VSB with their short description. It is left of the implementation of the One Stop API framework (as front-end of the DSP SS-O), how blueprints are presented to verticals and what support is provided to verticals to provide the parameters when preparing a VSD. A wide range of parameters are possible, e.g. values for latency constraints, paths to virtual application images, types of connection services, etc., could be left open in the blueprint. It is up to the One Stop API whether it just offers free text fields, range limited fields, e.g. for latency values, drop-down menus, e.g. for available traffic probes, etc., or whether it provides even wizard-like functionality to prepare a VSD from a blueprint.

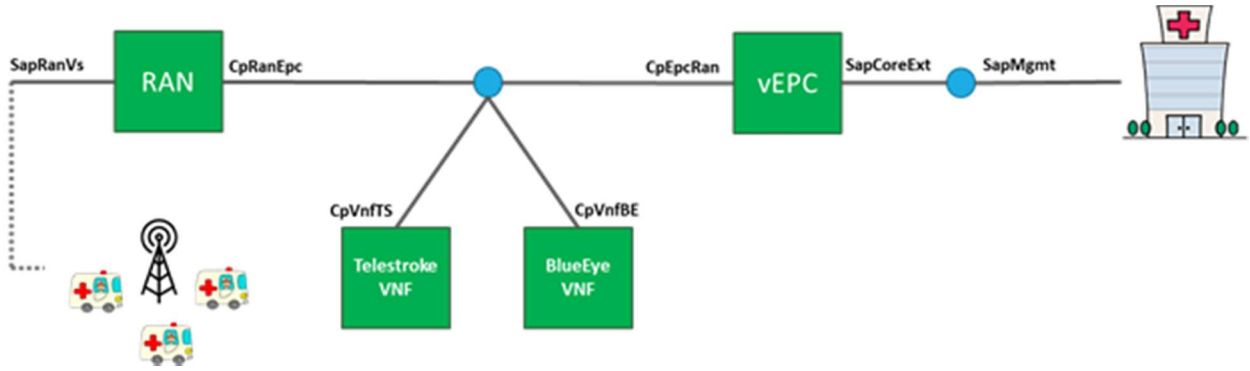

Figure 16: SliceNet eHealth VSB graphical representation

Table 13: Vertical Service Blueprint

Attribute	Description
blueprintId	Unique Identifier for the VSB.
version	A version number
name	Name for the VSB.
description	Short description of the VSB.
parameters	List of parameters that describe the service constraints the vertical has to fill (i.e. valorize) when filling the VSB to produce a new VSD. The list provides for each parameter its name, type, description, and the field of applicability
atomicComponents	List of atomic functional components (i.e., network functions and virtual applications in general) needed to implement the VSB.
endPoints	Specification of connection endpoints. They can be internal or external.
connectivityServices	List of virtual links and their relevant end points. Virtual links describe how the atomic functional components are connected.
serviceSequence	Description of how traffic flows among atomic components, supporting also multicast scenarios.
configurableParameters	Parameters that can be configured at instantiation time by the user for a specific instance of service derived from the given blueprint.
applicationMetrics	List of application metrics that this service can provide. Metrics reported here are strictly application related. No network metrics are included. These metrics are mapped to available metrics defined in the DSP FCAPS catalogue, as defined in D6.6 [26] and D6.7 [27].

qosMetrics	List of QoS metrics that this service can provide. Metrics reported here can refer to any QoS aspect, including network and service resource usage in general. These metrics are mapped to available metrics defined in the DSP FCAPS catalogue, as defined in D6.7 [27].
ppFunctions	List of P&P control functions that are supported for the given vertical service and can be requested by the vertical and activated to enable and expose specific runtime control functions, in accordance with P&P principles defined in D6.3 [28]. Each function is expressed with its name, exposure level (e.g. slice) and type (e.g. control, monitoring, management)
cognitiveFunctions	List of additional cognitive management and DSP optimization functions that are supported for the given vertical service. These can map to specific SliceNet optimization functions developed and reported in D5.5 [29] and D5.6 [30]. These can also group and include cognitive functions that one or more NSP have exposed through their Slice Orchestrators as part of their slice offers.

4.1.2 Vertical Service Descriptor

The VSDs are the vertical service descriptors that drive the provisioning of new vertical service instances, and are obtained after parameterizing the VSBs in the parts to be filled by the vertical, as described in the previous section. The structure of the VSD is therefore derived from the VSB one, in practice using most of the information expressed by the vertical when filling the VSB.

Table 14 shows the structure of a generic VSD.

Table 14: Vertical Service Descriptor

Attribute	Description
vsdId	Unique identifier for a VSD. This is provided by the SS-O when onboarding the VSD.
name	Name provided by the vertical for this VSD.
description	Short description of the VSD, e.g. “Sensor monitoring for plant B”.
version	A version number
blueprintId	The identifier of the blueprint from which this VSD was derived
Sst	Slice Service Type, as defined by 3GPP. Allowed values are therefore: eMBB, URLLC, mMTC
serviceConstraints	List of service related constraints that have to be fulfilled by vertical instances created starting from the given descriptor (e.g. geographical constraints, sharing rules, etc.).

qosConstraints	List of QoS related constraints that have to be fulfilled by vertical instances created starting from the given descriptor. This attribute contains the parameter types and values as filled by the vertical according to the parameterization of the related VSB.
monitoring	List of network and application KPIs that have to be monitored for this vertical service when provisioned. This attribute contains the parameter types and values as filled by the vertical according to the parameterization of the related VSB.
ppFunctions	List of P&P control functions that have been selected from those available in the VSB by the vertical and have to be supported for the instances based on the given descriptor.
cognitiveFunctions	List of cognitive management and DSP optimization functions that have been selected from those available in the VSB by the vertical and have to be supported for the instances based on the given descriptor.

4.1.3 End-to-end Network Slice Instance

When a vertical service is instantiated as the combination of single-domain network slices, the DSP creates for its internal end-to-end network slice lifecycle management logic an additional managed object called end-to-end network slice instance.
An end-to-end network slice is mapped 1:1 to a vertical service instance, and it is stored by the DSP as the combination of single-domain network slice instance information exposed by the NSPs. While the vertical service instance is only exposed to the verticals, the end-to-end network slice instances are kept internal in the DSP Service Orchestrator. Table 15 describes the end-to-end NSI Information Model.

Table 15: End-to-end Network Slice Instance information model

Attribute	Description
e2eNsiId	It is the identifier of the end-to-end NSI. It is allocated by the DSP Service Orchestrator when the VSI procedure is successfully completed.
vsiId	It is the vertical service instance identifier, as created by the DSP Service Orchestrator.
nsiList	List of NSI objects, according to the format defined in section 4.2.3. It includes the information of all the single-domain NSIs created in the NSP domains.

4.2 Slice information model

The SliceNet slice information model is specified in accordance to the 3GPP Network Resource Model [31], which provides the definition of New Radio (NR), Next Generation RAN (NG-RAN), 5G Core Network (5GC) and network slice instance data structures and managed objects. In particular, the

SliceNet slice information model that is at the base of the NSP slice orchestrator lifecycle management leverages on the network slice part of the 3GPP NRM, especially for what concerns the Network Slice Instance (NSI) and Network Slice Subnet Instance (NSSI).
The SliceNet slice information model is built around four main components:

- Network Slice Template (NST)
- Network Slice Subnet Template (NSST)
- Network Slice Instance (NSI)
- Network Slice Subnet Instance (NSSI)

These are the main elements that are managed by the NSP Slice Orchestrator in its catalogue and inventory functions, and exposed to the DSP to fulfil the end-to-end network slice lifecycle management.

4.2.1 Network Slice Template

The NST is a kind of network slice descriptor which describes the overall single domain network slice offer of an NSP, and therefore includes a set of attributes that characterize the network slice.
The NST is created and maintained by the NSP, and used by the Slice Orchestrator as the baseline to manage the lifecycle of the NSIs created starting from it. Moreover, the NST is exposed by the NSP to the DSPs to enable the composition of end-to-end slices spanning across several NSP domains. Indeed, specific attributes for the slice access points are included in the NST to describe how the network slices created from the NST can be accessed and interconnected to other slices offered by other NSPs. These slice access points can be considered as logical endpoints that model and abstract how the NSP allows interconnecting and accessing to its slices.
Even if the NST information model used by the Slice Orchestrator is defined by SliceNet, it follows the approach specified by 3GPP for the network slice NRM, where a network slice is composed by one or more network slice subnets.
Moreover, in SliceNet, most of the actual precise network slice performance requirements are expressed and requested at provisioning time, following the 3GPP approach defined in [32]. For this reason, the NST includes attributes to express the maximum capabilities in terms of network slice performances (reusing data structures defined by 3GPP for NSIs and NSSIs), which therefore allow DSPs to further apply their slice reuse and sharing policies. Table 16 describes the NST information model.

Table 16: NST information model

Attribute	Description
nstId	It is the identifier of the NST.
nstName	It is the name of the NST.
nstDescription	It provides a short description of the network slices that can created from the NST
nstProvider	It is the provider for the slice offer included in the NST. It identifies the NSP owning the NST.

sst	It is the slice Service Type, as defined by 3GPP and requested by the DSP when requesting the creation of the NSI. Allowed values are therefore: eMBB, URLLC, mMTC.
nsstList	It is the list of NSSTs associated to the NST. See section 4.2.2 for more details.
sapList	It is the list of slice access points for the NST, i.e. the logical endpoints that allow to access or interconnect the network slices created from the NST. Each slice access point is identified by: • name of the access point • type of the access point (access, external) • a list of allowed peering NST providers (i.e. NSPs) for end-to-end composition constraints
nssLinkList	It is a list of logical links that describe how the network slice subnets composing the NST can be interconnected. In includes references to the NSST endpoints defined in section 4.2.2.
ppFunctionList	List of Plug & Play control functions that are supported for the network slice in the NSP management domain, in accordance with P&P principles defined in D6.3 [28]. Each function is expressed with its name, exposure level (e.g. slice) and type (e.g. control, monitoring, management). NB. This attribute is not exposed to DSPs as part of the slice offer.
cognitiveFunctionList	List of additional cognitive management and QoE optimization functions that are supported for the network slice in the NSP management domain. These map to specific SliceNet optimization functions developed and reported in D5.5 [29] and D5.6 [30].
actuationList	List of actuations offerings that the NSP exposes (i.e. allows to invoke through the dedicated APIs) to the DSP for runtime optimization of network slices created from the NST. Each actuation is identified by: • a name, matching the network slice optimization or actuation function exposed to the DSP • a description, to specify the main purpose of the actuation list of key/value pairs describing the actuation parameters required to invoke the actuation on the NSP Slice Orchestrator This maps to the multi-domain FCAPS actuation capabilities and NSP actuation offering model defined in D6.7 [27].

4.2.2 Network Slice Subnet Template

The NSST represents the network slice subnet descriptor and details the capabilities and requirements of the individual components of the network slices offered by a given NSP.
The NSST is created and maintained by the NSP, and used by the Slice Orchestrator as the baseline to manage the lifecycle of the NSSIs created starting from it. Therefore, the whole set of NSSTs associated to an NST, including their attributes and characteristics, regulates the logic for the management of network slices.

The NSSTs are also exposed to the DSPs, but only as part of the whole network slice offers (i.e. within the master NSTs). Individual NSSTs are not offered to DSPs and cannot indeed be activated and provisioned as standalone NSSIs.
Following the 3GPP approach, network slice subnets can be implemented as NFV Network Services when the related network functions are provisioned in the virtualized infrastructure. For this, the NSST can include information about Network Service Descriptors to be used to implement the given network slice subnet. Table **15**Table 17 describes the NSST information model.

Table 17: NSST information model

Attribute	**Description**
nsstId	It is the identifier of the NSST.
nsstName	It is the name of the NSST.
nsstDescription	It provides a short description of the network slices that can created from the NSST
nsstType	It identifies the type of NSS that can be created from the NSST, e.g. RAN slice subnet, core slice subnet, etc. or specific application related slices, e.g. eHealth Telestroke slice subnet.
constituentNsstList	Following the 3GPP NRM approach, each network slice subnet can recursively be associated to other constituent subnets. This attribute provides the list of constituent NSSTs, if any.
endpointList	It is the list of logical endpoints associated to the network slices created from the NST. These endpoints are those referenced in the NST to describe how network slice subnets forming a network slice are interconnected. Each endpoint is identified by a name and a type (internal, external). External endpoints are those that maps to a slice access point.
maxPerfReq	This attribute describes the maximum performance capabilities that network slice subnets created from the NSST can support. It is a list of objects that depends on the “sst” attribute of the related NST. In particular: • If sST is eMBB, it is a list of eMBBPerfReq objects (ref. 3GPP 28.541), which include among the other attributes maximum uplink and downlink data rate, maximum user density • If sST is uRLLC, it is a list of uRLLCPerfReq objects (ref. 3GPP 28.541), which include among the other attributes maximum end-to-end latency, maximum jitter, maximum expected data rate. • If sST is mMTC, it is a list of mMTCPerfReq objects This attribute is inherited from the “perfReq” attribute defined in the 3GPP NRM [31].
maxNumberOfUEs	It identifies the maximum number of UEs that simultaneously can access network slice subnets created from the NSST

sharingLevel	It identifies if the resources of network slice subnets created from the NSST can be shared with other subnets. It can be: shareable, not-shareable
nsdInfo	If the network slice subnet is related to a NFV Network Service, this attribute provides a summary of the NFV NSD as maintained by the NMR-O. It is composed by: • A Network Service Descriptor identifier • A Network Service Descriptor name • A Network Service Descriptor type The NSD type identifies the type of service implemented through the Network Service (e.g. vEPC, eHealth Telestroke, eHealth BlueEye, etc.) This attribute allows to link the network slice subnets implemented as NFV Network Services with the NMR-O catalogue where full information about the NSD is kept.
kpiList	List of QoS and application metrics that the network slice subnets created from the NSST can provide through monitoring. These metrics are mapped to available metrics defined in the NSP FCAPS catalogue, as defined in D6.6 [26].
actuationList	List of actuations offerings that the NSP exposes (i.e. allows to invoke through the dedicated APIs) to the DSP for runtime optimization of network slice subnets created from the NSST. Each actuation is identified by: • a name, matching the network slice optimization or actuation function exposed to the DSP • a description, to specify the main purpose of the actuation • a list of key/value pairs describing the actuation parameters required to invoke the actuation on the NSP Slice Orchestrator This maps to the multi-domain FCAPS actuation capabilities and NSP actuation offering model defined in D6.7 [27].

4.2.3 Network Slice Instance

The NSI represents a provisioned network slice in a given NSP domain that has been instantiated based on an existing NST. Therefore, starting from an NSP slice offer (i.e. the NST), the DSP requests for the provisioning of an NSI. The NSI information model is based on the 3GPP NRM network slice definition [31], and includes attributes and characteristics that describe a network slice that is instantiated in a given NSP domain. Figure 17 shows how the main components of the NSI object relate to each other. In particular, an NSI is composed by one NSSI, and is described in terms of characteristics and capabilities by a Service Profile. The NSSI is a recursive object that can therefore include in it more NSSIs. This is useful to model network slice instances that are composed by multiple subnet slice instances, e.g. by one or more RAN slices and one or more NFV Network Services (for vEPC or 5GC services, as well as for vertical specific applications).
This relationship is inherited from the 3GPP NRM and is used as-is in SliceNet. The NSIs are managed at the NSP Slice Orchestrator level and related information is stored in its inventory, and exposed to the DSP Service Orchestrator.

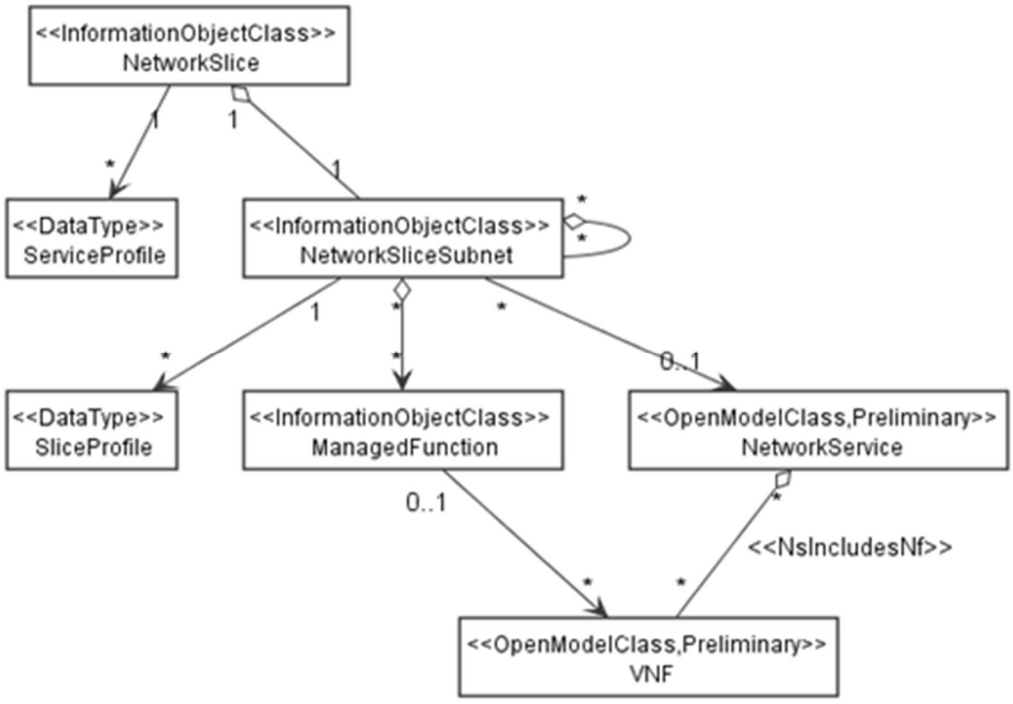

Figure 17: Network Slice objects relationships (ref. 3GPP)

The following table describes the NSI information model. As said the attributes are mostly inherited from the network slice object specified in the 3GPP NRM [31]. The SliceNet extensions are clearly identified in the table. Table 18 describes the NSI information model.

Table 18: NSI information model

Attribute	Description
operationalState	It is the operational state of the NSI, and indicates whether the slice resources are actually provisioned and working (ref. 3GPP 28.541)
administrativeState	It is the administrative state of the NSI, and indicates the permission to use the NSI (ref. 3GPP 28.541)
serviceProfileList	A list of ServiceProfile objects (see Table 19) which identify the NSI requirements and runtime attributes (ref. 3GPP 28.541)
nSSIId	It is the identifier of the NSSI associated to this NSI (see Figure 17). Therefore, it links the NSI with the NSSI implementing it (ref. 3GPP 28.541)
nstId	The identifier of the NST used to provision the NSI. This attribute is added in SliceNet to keep a proper reference with the NST from which the NSI has been created. This enables the proper implementation of arbitration and translation functions described in section 5.2.

Table 19: ServiceProfile information model (ref. 3GPP 28.541)

Attribute	Description
serviceProfileId	The unique identifier of the ServiceProfile object describing the NSI requirements (ref. 3GPP 28.541)
pLMNIdList	The list of Public Land Mobile Network (PLMN) identifiers associated and used by the NSI (ref. 3GPP 28.541)
perfReq	It describes the performance requirements associated to the NSI, and that have been requested by the DSP when requesting its creation. It is a list of objects that depends on the sST attribute of this ServiceProfile. In particular: • If sST is eMBB, it is a list of eMBBPerfReq objects (ref. 3GPP 28.541), which include among the other attributes expected uplink and downlink data rate, user density, coverage. • If sST is uRLLC, it is a list of uRLLCPerfReq objects (ref. 3GPP 28.541), which include among the other attributes end-to-end latency, jitter, expected data rate. • If sST is mMTC, it is a list of mMTCPerfReq objects Full details are provided in [31].
maxNumberofUEs	It identifies the maximum number of UEs that simultaneously access the NSI, as requested by the DSP when requesting its creation (ref. 3GPP 28.541)
latency	It identifies the packet transmission latency (in millisecond) through the RAN, core, and backhaul segments of the NSI, as requested by the DSP when requesting its creation (ref. 3GPP 28.541)
uEMobilityLevel	It identifies the mobility level of UEs accessing the NSI. It can be: stationary, nomadic, restricted mobility, fully mobility (ref. 3GPP 28.541)
resourceSharingLevel	It identifies if the NSI resources can be shared with other NSIs. It can be: shared, not-shared (ref. 3GPP 28.541)
sST	It is the slice Service Type, as defined by 3GPP and requested by the DSP when requesting the creation of the NSI. Allowed values are therefore: eMBB, URLLC, mMTC.
availability	It identifies the availability requirement for the NSI, expressed as a percentage (ref. 3GPP 28.541).
sNSSAIList	This attribute present in the 3GPP NRM network slice information model is not supported by the NSP Slice Orchestrator.
coverageAreaTAList	This attribute present in the 3GPP NRM network slice information model is not supported by the NSP Slice Orchestrator.

4.2.4 Network Slice Subnet Instance

The NSSI represents a constituent logical component of an NSI. Each NSI has at least one NSSI that actually implements the network slice itself. There are cases where an NSI is composed by multiple NSSIs, e.g. when an NSI in an NSP domain is the combination of a RAN slice and a NFV Network Service implementing a 4G LTE vEPC service.
The NSSI information model is also based on the 3GPP NRM network slice subnet definition [31], and includes attributes and characteristics that describe a network slice subnet that is instantiated in a given NSP domain. As depicted in Figure 17, an NSSI is composed by a set of Managed Functions (which can be VNFs and PNFs) and optionally an NFV Network Service, and it is characterized by a Slice Profile (which mostly provides the reference to the performance requirements).
In the SliceNet context, a slice subnet can be implemented as NFV Network Services (and thus provisioned through the NMR-O), as well as RAN, core and WAN slices which are enforced through the SliceNet Control Plane services. In general, a NSSI can be reused and shared across different NSIs.
The NSSIs are managed at the NSP Slice Orchestrator level and related information is stored in its inventory, and exposed to the DSP Service Orchestrator.
Table 20 describes the NSSI information model. Its attributes are mostly inherited from the network slice subnet object specified in the 3GPP NRM [31], and the SliceNet extensions are clearly identified in the table.

Table 20: NSSI information model

Attribute	Description
nSSIId	It is the identifier of the NSSI, as assigned by the Slice orchestrator. This attribute is added in SliceNet.
nSSTId	The identifier of the NSST used to provision the NSSI. This attribute is added in SliceNet to keep a proper reference with the NSST from which the NSSI has been created. This enables the proper implementation of arbitration and translation functions described in section 5.2.
mFIdList	It is the list of Managed Functions (i.e. VNFs and PNFs) instances identifiers that are associated with the NSSI. Some of them may have been reused from other NSSIs. This list allows to map the NSSI with the NMR-O inventory where full information about provisioned VNFs and PNFs is kept (ref. 3GPP 28.541)
constituentNSSIIdList	It is the list of identifiers of NSSIs which are associated to this NSSI. It is used when an NSI is composed by multiple NSSIs. If an NSI is built by a single NSSI this list is empty (ref. 3GPP 28.541)
operationalState	It indicates the operational state of the NSSI, and indicates whether the slice subnet resources are actually provisioned and working (ref. 3GPP 28.541)
administrativeState	It is the administrative state of the NSSI, and indicates the permission to use the NSSI (ref. 3GPP 28.541)

nsInfo	If the NSSI is implemented in the virtualized NSP infrastructure as an NFV Network Service, this attribute provides a summary of the NFV Network Service instance as provisioned by the NMR-O. It is composed by: • A Network Service instance identifier • A Network Service instance name • A Network Service instance description This attribute allows linking the NSSI implemented as an NFV NS with the NMR-O inventory where full information about the provisioned NS is kept.
sliceProfileList	A list of SliceProfile objects (see Table 21) which identify the NSSI requirements and runtime attributes (ref. 3GPP 28.541)
monitoring	It is the list of network and application KPIs that are monitored for the NSSI. This attribute contains the KPI types as requested by the DSP when requesting for the provisioning of the NSI (and related NSSIs). This attribute is added in SliceNet to keep a proper reference of the set of NSSI KPIs which are monitored through the FCAPS framework.

Table 21: Slice Profile information model (ref. 3GPP 28.541)

Attribute	**Description**
sliceProfileId	The unique identifier of the SliceProfile object describing the NSSI requirements (ref. 3GPP 28.541)
pLMNIdList	The list of PLMN identifiers associated and used by the NSSI (ref. 3GPP 28.541)
perfReq	It describes the performance requirements associated to the NSSI. It is a list of objects that depends on the sST attribute of this ServiceProfile. In particular: • If sST is eMBB, it is a list of eMBBPerfReq objects (ref. 3GPP 28.541), which include among the other attributes expected uplink and downlink data rate, user density, coverage. • If sST is uRLLC, it is a list of uRLLCPerfReq objects (ref. 3GPP 28.541), which include among the other attributes end-to-end latency, jitter, expected data rate. • If sST is mMTC, it is a list of mMTCPerfReq objects Full details are provided in [31].
maxNumberofUEs	It identifies the maximum number of UEs that simultaneously access the NSSI (ref. 3GPP 28.541)
latency	It identifies the packet transmission latency (in millisecond) requirement in the NSSI segment (ref. 3GPP 28.541)
uEMobilityLevel	It identifies the mobility level of UEs accessing the NSSI. It can be: stationary, nomadic, restricted mobility, fully mobility (ref. 3GPP 28.541)

resourceSharingLevel	It identifies if the NSSI resources can be shared with other NSSIs. It can be: shared, not-shared (ref. 3GPP 28.541)
sNSSAIList	This attribute present in the 3GPP NRM network slice information model is not supported by the NSP Slice Orchestrator.
coverageAreaTAList	This attribute present in the 3GPP NRM network slice information model is not supported by the NSP Slice Orchestrator.

4.3 Resource information model

The Resource Orchestrator information model is based on the ETSI NFV specifications to provide the network services that will compose the upper level network slices. In this regard, the Resource Orchestrator uses the NSD and VNFD specifications defined in ETSI NFV-IFA015 [33] to model the network services that will be offered to the Slice Orchestrator. Hence, these descriptors become the information unit used between the two levels of orchestration, namely slice and resource, to instantiate the network services at resource level that will be part of the upper layer slice. The interfaces to implement such communication are also specified by ETSI-NFV-IFA013 [34]. In particular, a set of CRUD operations over network services and the associated VNFs is defined. The NSD and VNFD are the basic transfer units in such operations and, thus, the main components of the information model of the SliceNet Resource Orchestrator. It is worth noting here that Open Source MANO, which is the core element of the Resource Orchestrator, is fully aligned with this information model as well OSM information model [35].

The NSD defined by OSM contains the information needed to deploy the network service. More specifically, the NSD contains references to the descriptors of the VNFs (VNFDs) conforming the service. These VNFs are interconnected by virtual links, and the descriptors of such virtual links (VLDs) are contained in the NSD as well. Additional connection points can be defined to provide external access to the network service. Table 22 summarizes the most relevant attributes of the NSD.

Table 22: NSD Information Model

Attribute	Description
id	Unique identifier for the NSD.
name	NSD name.
description	Description of the NSD.
connection-point	A list of references to network service connection points.
constituent-vnfd	List of VNFDs that are part of this network service.
ip-profiles-list	List of IP profiles. Allows establishing specific IP configurations to the network service.
vnf-dependency	List of VNF dependencies.

vld	List of Virtual Link Descriptors (VLDs).
monitoringparam	List of monitoring parameters at the network service level.

The VNFD defined by OSM contains the attributes that allow for the instantiation and lifecycle management of a particular network function. To do this, the Virtual Deployment Unit (VDU) is defined. The VDU contains the images to be used by the VNF, its network configuration, which includes the connection points to be assigned to the VNF for both internal (between the VNFs associated to the network service) and external communications. The physical resources required by the VNF, in terms of computing, storage and memory are defined here as well. Table 23 highlights the most relevant attributes of the VNFD.

Table 23: VNFD Information Model

Attribute	**Description**
id	Identifier for the VNFD.
name	VNFD name.
description	Description of the VNFD.
connectionpoint	The list for external connection points.
mgmtinterface	Interface over which the VNF is managed.
internal-vld	List of Internal Virtual Link Descriptors (VLD).
ip-profiles	List of IP profiles. An IP profile describes the IP characteristics for the virtual-link.
vdu	List of virtual deployment units (VDUs).
vdudependency	List of VDU dependencies, from which the orchestrator determines the order of startup for VDUs.
vnfconfiguration	Information about the VNF configuration for the management interface.
monitoringparam	List of monitoring parameters for the VNF.

Some efforts have been done in the framework of ETSI and 3GPP to provide network slicing (as defined by 3GPP TS 28.541 [31]) over the NFV-based infrastructure. The compatibility between these two paradigms is presented in ETSI NFV-IFA024 [36] and further analyzed in [37]. Figure 18 depicts the touch points between the network slicing model provided by 3GPP and the ETSI NFV model that supports the Resource Orchestrator. As shown in the figure, a NS or NSS at NSP level can be associated to a network service, which is in turn a composition of VNFs, configured in that NSP. Similarly, the figure also illustrates the association between the VNFs composing a network service and the managed functions that compose a NSS in the 3GPP NS model. As a matter of fact, the managed function object

is defined by 3GPP TS 28.622 [4], which was sent for transposition to ETSI TS 128 622 [38], as a function that can be either realized by software running on dedicated hardware or software running on NFV infrastructure.

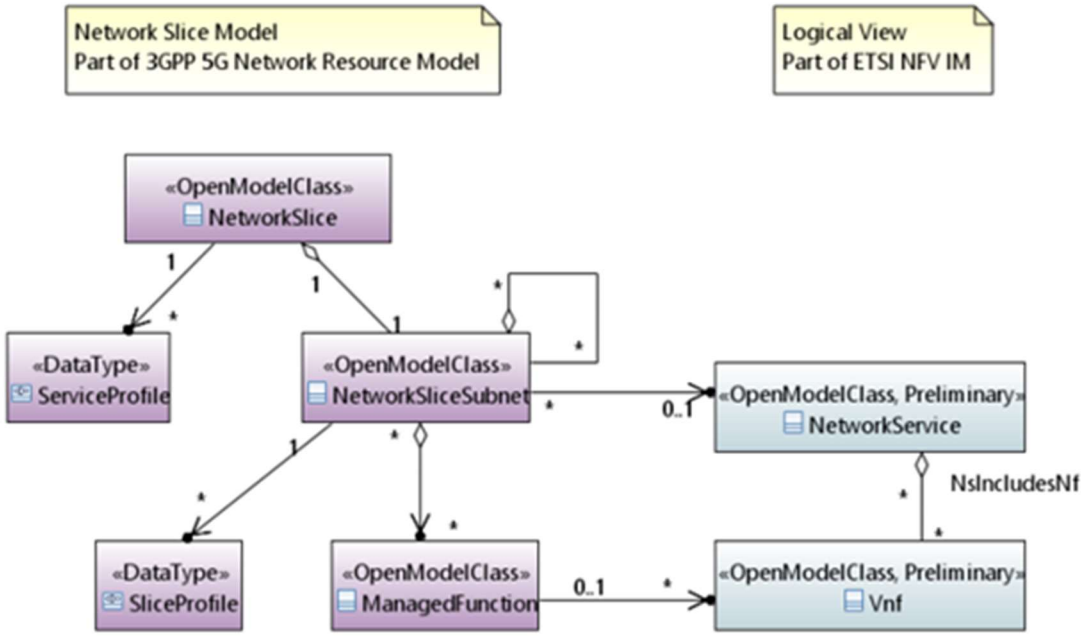

Figure 18: ETSI-3GPP: Touch points between the NFV information model and the Network Slicing information model

Hence, the above-mentioned association between the network slicing provided by the network resource model from 3GPP and the NFV information model provided by ETSI paves the way for the multi-level (slice and resource) NSP orchestration proposed for the SliceNet orchestration plane.

5 Orchestration Logical Architecture

This chapter provides a detailed description of the SliceNet orchestration logical architecture, as the cross-layer combination of Service Orchestrator at DSP level with Slice Orchestration and Resource Orchestration at NSP level. The overall design and development of this SliceNet orchestration is based on the existing work from the 5G-Transformer project [17], in particular on the Vertical Slicer. In the SliceNet context, the Service and Slice Orchestrators architecture design (and prototype implementation) is an evolution of the 5G-Transformer Vertical Slicer towards a multi-domain where the SliceNet principles for DSP and NSP split are supported.

The following sections describe each of the three main components in the SliceNet Orchestration architecture, introduced by a brief summary of the 5G-Transformer Vertical Slicer.

5.1 Reference Baseline: The 5G-Transformer Vertical Slicer

The 5G-Transformer Vertical Slicer (5GT-VS) is the common entry point for all verticals into the 5G-Transformer system, being part of the OSS/BSS of the administrative domain of a 5G- Transformer service provider (TSP). The 5GT-VS coordinates and arbitrates the requests for vertical services. Vertical services are offered through a high-level interface focusing on the service logic and the needs of vertical services. It allows defining vertical services from a set of vertical-oriented service blueprints, which, along with instantiation parameters, will result in Vertical Service Descriptors (VSD). Then, the 5GT-VS maps the vertical service descriptions and requirements defined in the VSD onto a network slice. We describe network slices with extended ETSI NFV Network Service Descriptors (NSD), which define forwarding graphs composed of a set of Virtual Network Functions (VNF) or Virtual Applications (VAs) connected with Virtual Links (VL), where some of them have the specific characteristics and constraints of MEC applications. Importantly, the 5GT-VS allows mapping several vertical service instances to one network slice, handling vertical-dependent sharing criteria and taking care of the necessary capacity changes in pre-established slices. In conclusion, the most fundamental tasks of the 5GT-VS are to provide the functionality for creating the vertical service descriptions, and to manage the lifecycle and the monitoring of vertical service instances and of the corresponding network slice instances.

In addition to such tasks, the 5GT-VS provides arbitration among several vertical service instances in case of resource shortage in the underlying infrastructure and based on global budgets for resource utilization, as defined in the SLAs. The arbitration component in the 5GT-VS maps priorities of services and SLA requirements to ranges of cardinalities of resources. These cardinalities are used by the 5GT-SO while deploying the NFV network services (NFV-NS) and, in case of actual resource shortage, to assign resources to the most important vertical service instances.

The architecture of the 5GT-VS is shown in Figure 19, as part of the service provider's OSS/BSS. It interacts with the vertical (or the M(V)NO) through its northbound interface (NBI) and with the service orchestrator through its southbound interface (SBI). The full description of the 5GT-VS architecture can be found in the 5G-Transformer deliverable D3.1 [22].

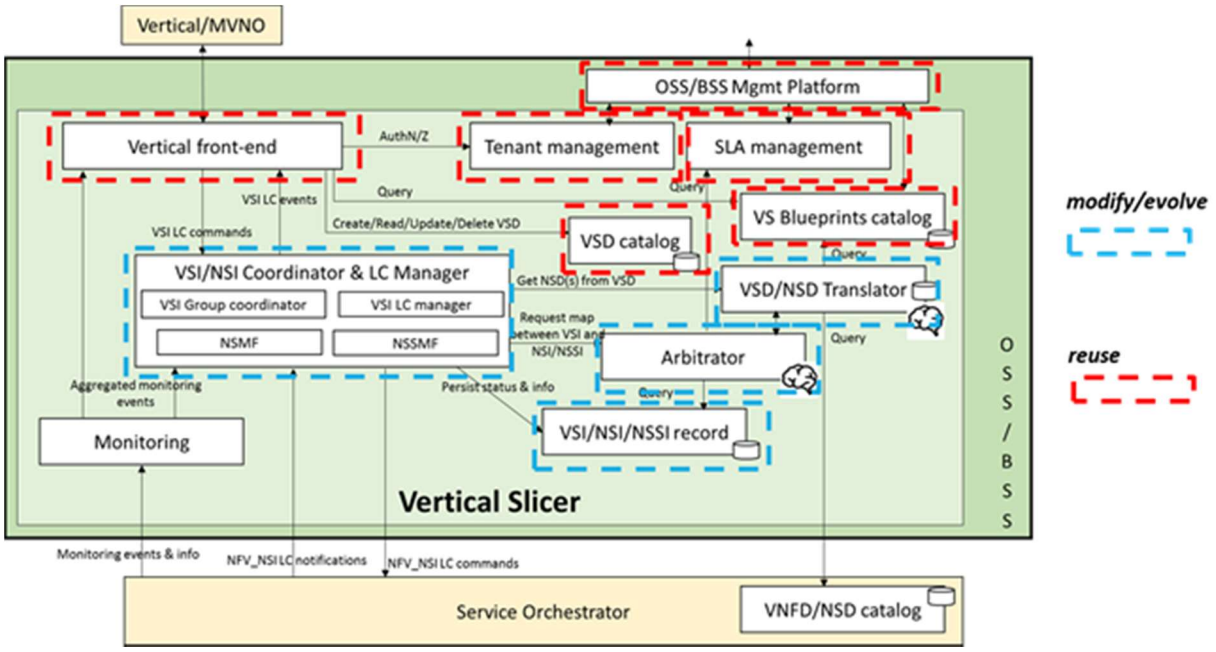

Figure 19: 5G-Transformer Vertical Slicer: high-level architecture and mapping to SliceNet SS-O re-use

The 5GT-VS NBI, is based on a REST API and implements the Ve-Vs reference point between the 5GT-VS and the vertical/MVNO, as well as the Mgt-Vs reference point between the 5GT-VS and the OSS/BSS Management Platform (as shown in Figure 20). In particular, the Ve-Vs reference point provides mechanisms for VSB queries and operation (e.g. instantiation, termination, queries and update) or monitoring (e.g. queries and subscriptions-notifications) of vertical services. The Mgt-Vs reference point provides management primitives, related to tenants, SLAs and VSBs.

For what concerns the 5GT-VS Monitoring Service and the 5GT-VS SBI, they can be considered out of scope for the work described in this document. Indeed, in the SliceNet management and orchestration approach the monitoring infrastructure is based on the FCAPS principles described in deliverables D6.6 [26] and D6.7 [27], while the SBI is completely re-designed to support the DSP and NSP split.

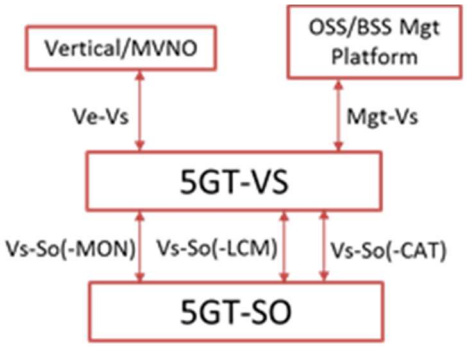

Figure 20: 5GT-VS reference points

5.2 Service Orchestrator Internal Architecture

The SS-O deployed at the DSP level provides orchestration of Vertical Services requested by verticals and takes care to map these into end-to-end network slices which are composed by per-domain

network slices offered by one or more NSPs. Therefore it implements the Service Orchestrator features of the SliceNet SS-O, managing the lifecycle of end-to-end slices and being the coordinator of the whole SliceNet cognitive management platform at the DSP level.

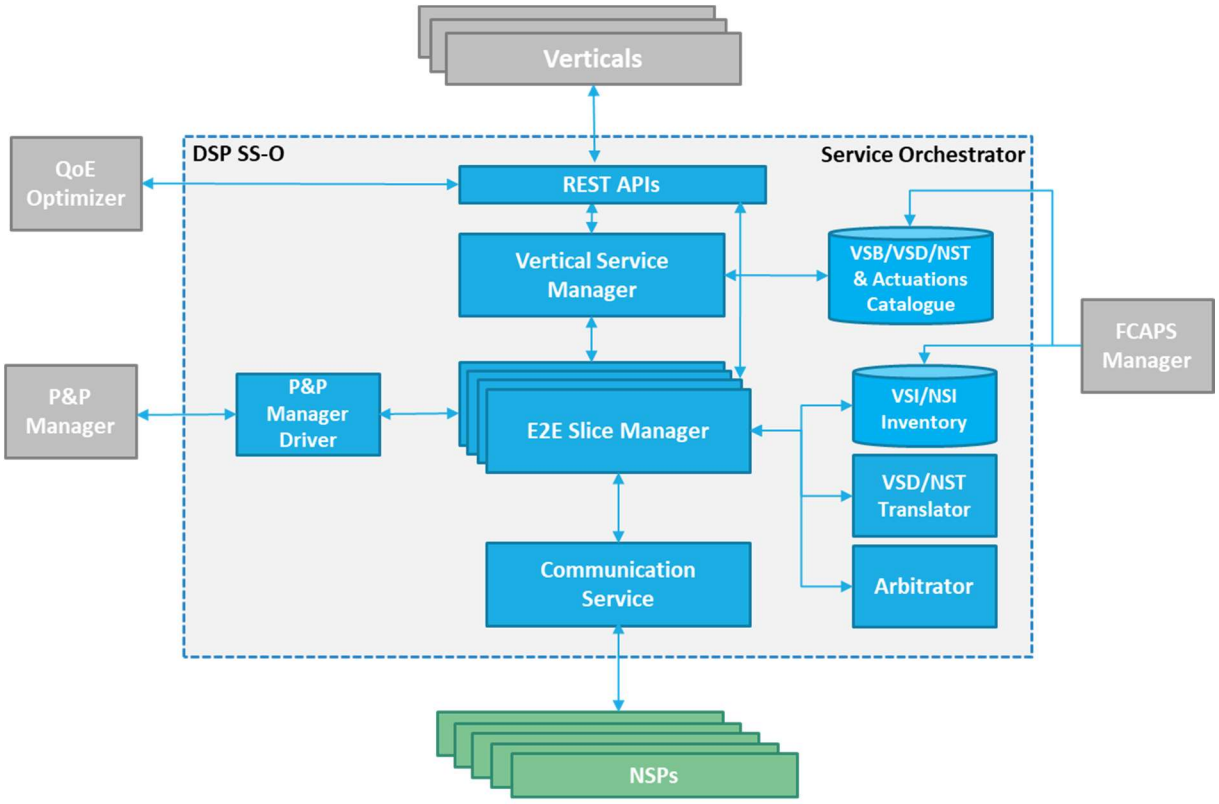

Figure 21: SliceNet Service Orchestrator functional architecture

Figure 21 shows the high level functional split of the Service Orchestrator, where the main service coordination functions are identified along with their interactions with other SliceNet architecture components. With reference to the 5GT-VS described in the previous section, this Service Orchestrator reuses part of the Vertical Service coordination logics, and enhance them to implement the mapping and translation of Vertical Services to end-to-end network slices spanning across several NSP domains. In particular, the front-end components of the 5GT-VS are mostly reused by the Service Orchestrator as they coordinate and manage the vertical service blueprints and descriptors which are very similar in terms of models between 5G-Transformer and SliceNet. On the other hand, the 5GT-VS components more related to network slice and NFV Network Service lifecycle management are widely enhanced and improved to support the SliceNet multi-domain DSP/NSP reference scenario.

In summary, the Service Orchestrator provides the following functionalities concerning the vertical service and end-to-end network slice management:

- Interaction with verticals for service offer exposure and customization, in terms of service components and additional capabilities and functions exposed
- Onboarding and maintenance of VSBs and VSDs
- Collection of network slice offers from NSPs, in the form of NSTs

- Lifecycle management of vertical services, which are mapped to end-to-end network slices that are built as the composition of single domain network slices as offered by the given NSPs in their exposed NSTs
- Maintenance of up to date vertical service and end-to-end network slice instances status and characteristics
- Coordination of Plug & Play control instances deployment (through the Plug & Play manager following the principles of D6.3 [28]), to expose to verticals the customized runtime control functions as requested in their vertical service request
- Coordination of multi-domain actuation operations for runtime modification and QoE optimization of end-to-end network slices across different NSP domains, following the multi-domain FCAPS principles defined in D6.7 [27]

For what concerns the multi-domain actuation capabilities, the Service Orchestrator acts as the main coordinator of the end-to-end network slice runtime actuations which are mostly driven by the QoE optimization and cognitive management logic described in D5.5 [29] and D5.6 [30]. In particular, this is implemented as part of the DSP to NSP interactions that occur between the Service Orchestrator and Slice Orchestrator through the APIs defined in section **Error! Reference source not found.**. In practice, the single domain NSP actuations exposed by the NSPs (as part of their slice offers in the NSTs) are processed by the DSP administrator and combined in the form of multi-domain actuation capabilities that are stored in the DSP Catalogue and offered to the DSP cognitive management and QoE optimization logic. Indeed, as part of the FCAPS design phase at the DSP level and described in D6.7 [27], part of these advanced QoE optimization functions may be designed and developed ad-hoc by the DSP according to the available multi-domain actuation capabilities.

The following subsections provide a brief description of each functional component depicted in Figure 21, highlighting the main features provided and the mapping to equivalent 5GT-VS building blocks (when applicable).

5.2.1 Vertical Service Manager

The Vertical Service Manager is the front-end component of the Service Orchestrator and is responsible to collect the requests from the verticals and manage them according to their purpose. In particular, the Vertical Service Manager presents the VSBs and the VSDs through the SS-O REST APIs to allow verticals to select the vertical services to be instantiated. In this context, it manages the parametrization process to obtain VSDs from VSBs upon specific vertical service characterizations. Therefore, it mostly interacts with the VSB and VSD Catalogues to expose the service offers to the verticals and to upload new descriptors and blueprints in them.

With respect to the 5GT-VS architecture, it maps to the Vertical Front-End component, augmented with the Tenant Management and SLA Management functionalities. It is mostly reused as-is from the 5GT-VS apart from the management of the extended models of the VSBs and VSDs as described in section 4.1.

5.2.2 Vertical Service and Network Slice Catalogue

The Service Orchestrator is equipped with a Catalogue service that stores the various blueprints and descriptors used by the end-to-end network slice manager to coordinate the lifecycle of vertical services and end-to-end network slices. This Catalogue, as shown in Figure 21, keeps up to date information related to VSBs, VSDs and NSTs, each with different procedures. In particular, VSBs are onboarded in the Catalogue as part of DSP admin operations following the model defined in section

4.1.1. On the other hand, VSDs are onboarded in the Catalogue as a result of the vertical-driven parametrization of VSBs, thus the trigger for the creation and storage of VSDs come directly from verticals through the Service Orchestrator REST APIs. For what concerns the NSTs, the Service Orchestrators collects them from the various NSPs with whom the given DSP has relationships, following the onboarding procedures defined in section 6. These NSTs stored in the Catalogue represent the per-domain network slice offers that NSPs offer to the given DSP for composing end-to-end slices, and follow the model defined in section 4.2.

In practice the VSB/VSD/NST Catalogue is implemented as a database service which stores the VSBs, VSDs and NSTs and is managed by the Vertical Service Manager to support the different logics described above. Through the Vertical Service Manager and the REST APIs, other components in the DSP cognitive management platform (e.g. the FCAPS Manager) can access the Catalogue to retrieve information about vertical slice and end-to-end network slice offers and related capabilities as described in the related descriptors and templates.

Moreover, as part of the P&P and monitoring related information included in VSBs, VSDs and NSTs, the Catalogue provides to the end-to-end Network Slice Manager the required knowledge to coordinate, beyond the lifecycle of network slice, the creation of additional artifacts and per-slice control and management functions related to P&P control (according to the principles and logics described in deliverable D6.3 [28]). In addition, the catalogue keeps track of the available multi-domain actuation capabilities that are being collected from the various NSPs actuation offerings and combined by the DSP administrator. Such actuation capabilities follow the model defined in D6.7 [27] for the actuation offerings.

5.2.3 End-to-end Network Slice Manager

The end-to-end Network Slice Manager is the core lifecycle engine of the Service Orchestrator, and is responsible for mapping and associating Vertical Service instances (VSIs) with end-to-end Network Slice instances (NSIs), leveraging on the Translator and Arbitrator functionalities for composing the end-to-end network slice from the different NSP network slice offers and regulating the sharing of network slices among different vertical services.

In practice, the end-to-end Network Slice Manager handles the lifecycle of VSIs and end-to-end NSIs, coordinating commands and events associated with them. The end-to-end NS management takes care of instantiation, modification and termination operations of the corresponding per-domain NSs by interacting with the different NSP Slice Orchestrators. The status and the current attributes and capabilities of VSIs and end-to-end NSIs are stored and maintained in the Vertical Service and Network Slice Inventory to enable other components within the SliceNet DSP cognitive management framework to access it.

The Vertical Service Manager takes care to allocate a dedicated manager for each new end-to-end NSI that implements a Vertical Service. Therefore each end-to-end NS associated to a VSI has its own end-to-end Network Slice Manager, which takes care of the management of its lifecycle, including on the one hand the decomposition into per-domain network slices, but also coordinating the P&P and FCAPs functions to be activated within the DSP cognitive management platform in support of the advanced SliceNet features. To fulfil this two-fold management approach, the end-to-end Network Slice Manager retrieves the different vertical service and network slice capabilities from the Catalogue (in terms of application, QoS and geographical constraints on the one hand, and KPI monitoring and vertical-driven P&P features on the other) and applies the vertical-specific coordination logics to create, maintain and terminate vertical services.

5.2.4 VSD/NST Translator

The Service Orchestrator during the coordination and management of each vertical service instance has to implement some decision logics to associate different of kind of descriptors (e.g. VSDs to end-to-end NSTs) and to decide whether some per-domain network slice instances can be reused for multiple vertical services. Indeed, while the end-to-end Network Slice Manager is the actual coordinator of the different steps and operations in the lifecycle of vertical services and end-to-end network slices (i.e. it implements the various workflow logics to ensure vertical services are instantiated, maintained and terminated in accordance with vertical requirements), the Translator and Arbitrator provide the required decision logic of the Service Orchestrator.

In particular, the VSD/NST Translator maps the vertical requirements into more technical characteristics and attributes that are required by the NSPs Slice Orchestrators to deploy per-domain network slices. In practice, it selects the per-domain NSTs (as collected from the various NSPs) that have to be composed for matching the requested vertical services constraints. The Translator identifies the per-domain NST capabilities in terms of QoS capabilities, monitored KPIs, actuation capabilities, geographical constraints, application requirements (i.e. specific type of NFV services offered by the NSPs) most suitable to guarantee the performance and the capabilities defined in the VSD.

When deploying a vertical service, the vertical first selects a VSB, which contains (as defined in section 4) attributes that have to be filled by the vertical itself to express the fine-grained requirements it demands. This makes the VSB becoming a VSD that is passed to the end-to-end Network Slice Manager for being deployed. Here, the VSD/NST Translator is invoked to map the VSD into an end-to-end NST following these steps:

1. the Translator queries the Vertical Service and Network Slice Catalogue to retrieve the VSD

2. the Translator queries the Vertical Service and Network Slice Catalogue to retrieve the per-domain available NSTs, applying a preliminary filter if possible

3. the Translator filter-out those per-domain NSTs not matching the vertical QoS, geographical, monitoring, actuation and application requirements

4. the Translator composes the set of per-domain NSTs into and end-to-end network slice chain that is passed back to the Network Slice Manager for coordinating the vertical service deployment

With respect to the 5GT-VS Translator, this SliceNet Service Orchestrator version is basically refactored to support the mapping and translation from VSDs towards the end-to-end network slices following a different information model with respect to the one used in 5G-Transformer.

5.2.5 Arbitrator

The Service Orchestrator Arbitrator is the other decision point of vertical services and network slices lifecycle coordination logics, as it takes care of regulating the sharing of network slices among different vertical services. In general, the vertical service and the network slice lifecycle operations are effective when trade-offs are found among the specific vertical requirements (technical and non-technical) and the overall status of the system where other services and slices are running. Accommodating new services and slices, possibly reusing part of the existing ones, is a typical service deployment problem requiring the management of such trade-offs.

The Arbitrator is the Service Orchestrator component that is in charge of resolving these trade-offs and thus decide for each vertical service instance to be created if some of the already existing per-domain network slices instances can be reused without affecting any of the involved service. As part of the resolution of the trade-offs, some of the vertical services requests may be also refused. In particular, the Arbitrator decides how to map new vertical service requests into network slice instances, enabling vertical services to share one or more per-domain network slice instances. To do that, the Arbitrator applies its decision logic mostly based on the QoS and geographical constraints expressed in the VSD matching them against the information stored in the Vertical Service and Network Slice Inventory for what concerns the actual status (and resource usage) of the existing vertical services and network slices. A relevant example of network slice sharing and reuse could be the sharing of the same 4G or 5G core services and virtual functions (e.g. virtual EPC or 5G Core) among different vertical service instances with similar requirements in terms of mobile access.

Upon invocation from the end-to-end Network Slice Manager, and starting from the available NSTs in the Catalogue, the Arbitrator verifies if one or more existing network slice instances can be reused to accommodate the new vertical service, or at least part of it. In the case of no suitable existing network slice instances available, the Arbitrator decides to create new network slice instances for the whole set of per-domain NSTs selected by the Translator. At this point, the end-to-end Network Slice Manager can start with the vertical service deployment workflows and trigger the creation of the new per-domain network slices in the given NSP domains. On the other hand, when the Arbitrator decides that at least part of the end-to-end network slice can be covered by existing network slice instances, this is passed-back to the Network Slice Manager to take care of the required actions for reusing them in the new vertical slice deployment process.

Similar to the Translator, the SliceNet Service Orchestrator Arbitrator is a refactored version of the 5GT-VS one due to the substantially different logic to be implemented, mostly in terms of target network slice instance model used in SliceNet.

5.2.6 Vertical Service and Network Slice Inventory

The Vertical Service and Network Slice inventory is the counterpart of the Service Orchestrator Catalogue service component, as it stores and maintains up to date information for the provisioned VSIs and end-to-end NSIs. Detailed information about vertical service characteristics, as well as per-domain network slice attributes and service access points are persisted in the inventory. The Vertical Service and Network Slice inventory is also implemented as a database following the vertical service and network slice models described in section 4.

The information stored in this inventory component is mostly used by the Arbitrator for deciding upon network slice reuse and share, as well as by the Vertical Service Manager to expose such information to the vertical through the Service Orchestrator REST APIs and the One-Stop-API. In addition, other SliceNet DSP cognitive management framework components (e.g. the FCAPS Manager) can access the inventory through the REST APIs to retrieve VSIs and NSIs information to fulfill their management logics.

5.2.7 Communication Service

The Communication Service regulates the interactions of the DSP Service Orchestrator with the underlying NSP Slice Orchestrators, as it offers common internal APIs to other Service Orchestrator components (mostly the end-to-end Network Slice Manager) to fulfill the vertical service and end-to-

end network slice lifecycle coordination operations. The main rationale behind this Communication Service is to expose a unified set of APIs for per-domain slice lifecycle management (i.e. create, modify, terminate), including onboarding related operations for per-domain NSTs collection, and implement through specific technology-specific clients the APIs of different slice orchestrators.

In the context of the SliceNet DSP Service Orchestrator, the Communication Service is equipped with a REST client that implements the Slice management APIs exposed by the NSP Slice Orchestrators as documented in section 7.2.

This component is newly introduced as part of the SliceNet, with respect to the 5GT-VS, and is one of the key enhancements to implement the coexistence of both the single-domain and multi-domain approaches of 5G-Transformer and SliceNet. In particular, the Communication Service for the 5GT-VS implements a RabbitMQ client to support the internal 5GT-VS communications (i.e. between vertical service and slice level management components).

5.2.8 P&P Driver

As part of the vertical service and end-to-end network slice lifecycle management, the Service Orchestrator takes also care of coordinating the creation of the additional per end-to-end slice control and management functions required for the SliceNet P&P features. For each vertical service, the end-to-end Network Slice Manager interacts with the P&P Manager through the dedicated driver which implements the related APIs described in deliverable D6.3 [28]. In accordance with the P&P requirements expressed in the VSBs and VSDs, the end-to-end Network Slice Manager instructs the P&P Manager to allocate the required per end-to-end slice control and management functions.

These interactions occur mostly as final steps during the vertical service instantiation and termination, respectively to activate and deactivate the customized control and management functions.

5.3 Slice Orchestrator Internal Architecture

The Slice Orchestrator within the SliceNet cognitive management platform is responsible for the coordination of per-domain slices lifecycle, thus interacting with different domain-specific controllers and orchestrator for the actual creation and configuration of slice resources. The Slice Orchestrator is deployed at the NSP level, and it manages single administrative domain slices, which are offered to the DSPs for being composed in end-to-end multi-domain scenarios and deployments. In terms of management logics, the Slice Orchestrator operates at a finer resource granularity with respect to the Service Orchestrators within the DSPs, as it coordinates the actual resources in the NSP physical and virtual infrastructure.

The Slice Orchestrator high level functional architecture is depicted in Figure 22. Here, the main slice coordination functions are highlighted with their interactions and positioning with respect to other SliceNet architecture components. Similarly to the Service Orchestrator, the Slice Orchestrator reuses part of the 5GT-VS, mostly for what concerns the functionalities related to the network slice lifecycle, including the NFV Network Services coordination and modelling, as well as the unified lifecycle procedures and specific drivers towards technology-specific NFV orchestrators. These are anyway enhanced to integrate the RAN control and slicing features of SliceNet, together with the QoS enforcement through the SliceNet control plane.

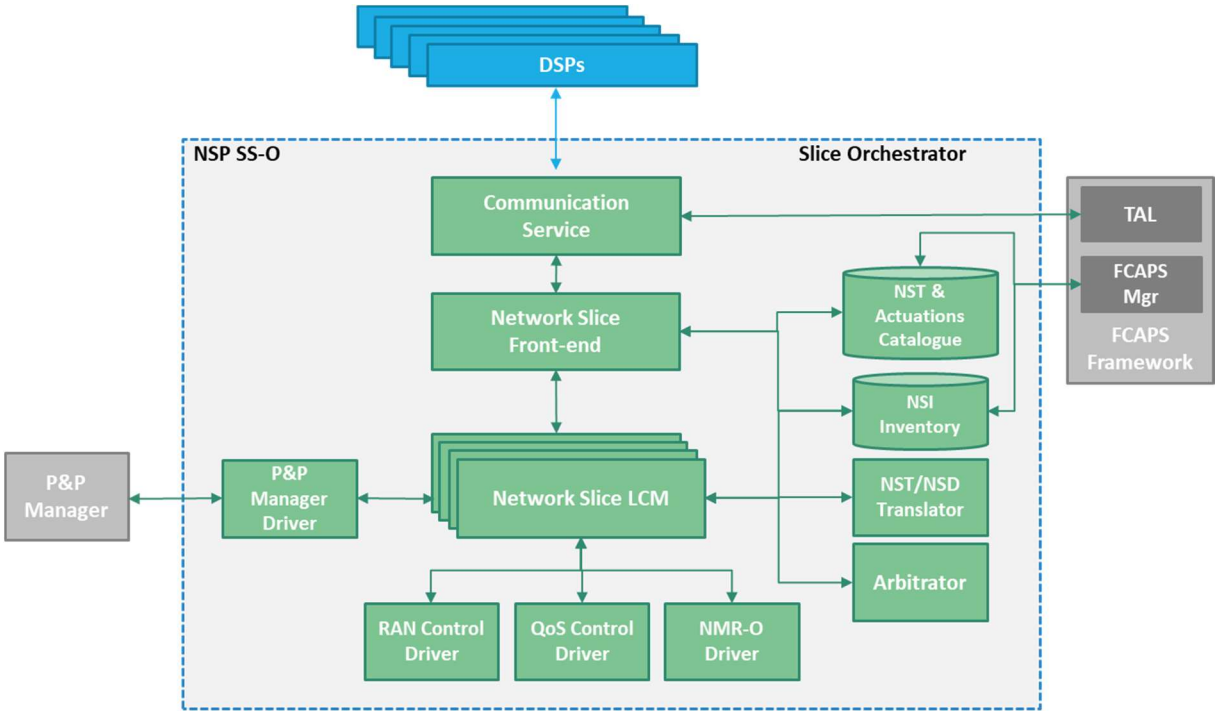

Figure 22: SliceNet Slice Orchestrator functional architecture

The Slice Orchestrator provides the following functionalities for network slice management:

- Interaction with DSPs for single domain network slice offer exposure, in terms of slice components, capabilities and actuation functions exposed
- Onboarding and maintenance of NSTs and NSSTs
- Lifecycle management of NSIs and NSSIs, including mapping and translation of network slice requirements into resource provisioning constraints and operations in the RAN, core, backhaul and NFV segments. Sharing of network slice resources is also a key fundamental feature implemented as part of the lifecycle management
- Maintenance of up to date NSIs and NSSIs status and characteristics in dedicated inventories
- Coordination of Plug & Play control instances deployment (through the Plug & Play manager following the principles of D6.3 [28]), to allocate per-slice control and management functions (including SliceNet Control Plane Services – CPSs – and cognitive/QoE functions)
- Coordination of single-domain actuation operations for runtime modification and cognitive optimization of network slices and network slice subnets, by interacting with SliceNet CPSs and NMR-O for applying the actuations at the resource level

The Slice Orchestrator exposes dedicated APIs towards the NSP FCAPS framework (ref. D6.6 [26]), for single-domain actuation purposes. Indeed, the Tactical Autonomic Language (TAL) Rule Engine in the FCAPS framework captures optimization events as generated by NSP cognitive algorithms (D5.5 [29] and D5.6 [30]), and invokes such network slice actuation APIs to apply the given actuation. All of the single domain NSP actuations available for the Slice Orchestrator are maintained in the Slice Orchestrator catalogue, following the model defined in D6.7 [27] for the NSP offerings and design phase. And they can be offered to DSPs in the context for multi-domain actuation purposes by including them in the NST slice offers, as defined in section 4.2

The following subsections provide a functional description of the components depicted in Figure 22, in terms of main features provided and mapping to equivalent 5GT-VS building blocks (when applicable).

5.3.1 Network Slice Front-end

The Network Slice Front-end represents the functional entry point of the SliceNet Slice Orchestrator and it coordinates all the requests related single-domain network slices coming from the DSPs. On the other hand, the Network Slice Front-end regulates the per-domain NSTs that have to be offered to the DSPs as part of the multi-domain onboarding process. For this, it interacts with the NST Catalogue to expose the slice offers to the peering DSPs through the Communication Service, as well as to upload new NSTs as part of the single-domain onboarding process (i.e. when new slice offers are designed by the NSP administrators).

The Network Slice Front-end also coordinates the per-slice Network Slice Lifecycle Managers (LCM), allocating them and forwarding the related actions and operations as requested by the DSPs' Service Orchestrators or by other SliceNet NSP cognitive management platform components.

The Network Slice Front-end is not mapped to any of the 5GT-VS functional components, as it is required upon the split in DSP and NSP orchestration functionalities in support of the multi-domain network slice deployments. So, it is newly added and implemented from scratch as part of the SliceNet SS-O.

5.3.2 Communication Service

The Communication Service within the Slice Orchestrator has a similar role with respect to the one in the Service Orchestrator, as it provides a unified communication interface for the Slice Orchestrator internal components towards the external world (i.e. other SliceNet architecture components). It is basically the counterpart of the same Service Orchestrator component, and allows to support the coexistence of both single-domain and multi-domain slice management.

In the Slice Orchestrator, the Communication Service is equipped with a REST controller that provide the SliceNet technology specific implementation of the single-domain slice management APIs described in section 7.2. It is linked to the Network Slice Front-end which regulates the access to other internal Slice Orchestrator services, like the NST Catalogue and the lifecycle managers.

As in the Service Orchestrator, the Communication Service is a new component with respect to the 5GT-VS and allows integrating different technology-specific APIs from different slice orchestrators.

5.3.3 Network Slices Catalogue

The per-domain NSTs modelling the NSP slice offers are stored and maintained the Network Slices Catalogue. The Catalogue content is exposed to the DSPs through the Network Slice Front-end and the Communication Service in support of the composition of per-domain slice offers at the DSP level for deployment of vertical services.

The NSTs stored in the Network Slices Catalogue follow the model defined in section 4.2, and provide to the Network Slice Lifecycle Managers (LCMs) the required information to coordinate the lifecycle of per-domain network slices. The NSTs are onboarded in the Catalogue as part of NSP administrator operations, as the result of the network slice design process, which is out of the scope of this document. The Catalogue itself is implemented as a database, and its access is regulated by the Network Slice Front-end. In particular, other components in the NSP cognitive management platform

(e.g. the FCAPS Manager within the FCAPS framework) can access the Catalogue (still through the Communication Service) to retrieve information about NSTs and related capabilities.

Moreover, similarly to the Service Orchestrator Catalogue, as part of the P&P and monitoring related information included in the NSTs, the Network Slice Catalogue provides to the Network Slice LCM the required knowledge to coordinate, beyond the lifecycle of network slices, the creation of additional per-slice control and management functions related to P&P control (according to the principles and logics described in deliverable D6.3 [28]) by interacting with the P&P manager. In addition, the Service Orchestrator Catalogue maintains the available single actuation capabilities that offered and implemented by the SliceNet CPSs and the NMR-O mostly. Such actuation capabilities follow the model defined in D6.7 [27] for the single domain NSP actuation offerings.

5.3.4 Network Slice Lifecycle Manager

The Network Slice LCM is the core component of the Slice Orchestrator, as it handles the lifecycle of per-domain NSIs, applying the required management logics for instantiation, modification and termination of network slices. As part of its lifecycle coordination duties, the Network Slice LCM decomposes each of the network slices to be created into specific resource domain actions according to the slice resource requirements expressed in the NSTs. To do that, it leverages on the Translator and Arbitrator functionalities for selecting proper resources to create slices, and for deciding if reuse and share part of the slice resources among different per-domain network slices.

The Network Slice LCM manages the lifecycle management requests coming from the Network Slice Manager, and interacts with the technology specific resource orchestrators and controllers to create and configure the required RAN resources, NFV Network Services and QoS in the different network domains, according to the slice requirements in the NSTs. Therefore, the Network Slice LCM processes the network slice requirements stored in the Catalogue, and upon per-domain network instantiation and runtime management takes care to maintain into the Network Slice Inventory the information related to NSIs.

A dedicated Network Slice LCM is allocated by the Network Slice Manager for each new per-domain network slice. This allows to have isolated slice lifecycle management workflows, and also to apply the coordination logics at scale as the management load is distributed among different logical entities. In addition to these lifecycle management duties, the Network Slice LCM also takes care to drive the activation of per-slice control and management functions for P&P purposes, still according to the related requirements expressed in the NSTs. This is done through the P&P driver, which translate these NST requirements into proper API calls to the P&P Manager following the specifications in D6.3 [28].

The Network Slice LCM in the SliceNet SS-O is a refactored version of the NSMF and NSSMF features of the 5GT-VS that required enhancements to support the integration with the NSP slice management approach and with the information model defined in section 4.

5.3.5 NST/NSD Translator

Similar to the Service Orchestrator, the Slice Orchestrator implements its decision logic with the combination of an NST/NSD Translator and an Arbitrator, with the aim of mapping NST attributes and characteristics into technology-specific resource requirements on the one hand, and of deciding if reusing slice resources among different network slices.

The NST/NSD Translator is invoked by the Network Slice LCM, mostly at network slice instantiation time, to map the network slice level requirements expressed into the NSTs into finer grain resource

level requirements. In practice, the Translator associates an NST with a set of specific RAN slice configurations, NFV Network Service Descriptors (NSDs) and QoS configurations (to be applied through the SliceNet control plane) that are then used by the Network Slice LCM to enforce the slice resource instantiation and configuration.

With respect to the 5GT-VS Translator, this Slice Orchestrator version is significantly enhanced in support of the SliceNet NST model defined in section 4.2, as well as to include in the translation process the specific RAN slice configurations and QoS enforcements.

5.3.6 Arbitrator

The Arbitrator in the Slice Orchestrator decision maker for regulating the slice resource sharing among different network slices. In particular, it assists the Network Slice LCM when new network slices have to be provisioned in the NSP domain, by evaluating if some of the existing (i.e. already provisioned and configured) slice resources can be reused without affecting the running services and slices.

In practice, the Arbitrator applies its decision logic by matching, for each network slice to be provisioned, the QoS and resource requirements expressed in the NST against the available NSIs in the inventory and the related slice resources used and configured. As an example, as part of the provisioning of a new per-domain network slice, the Arbitrator may decide to reuse an entire existing RAN slice configuration as it may be evaluated as enough to accommodate the requirements of both network slices (i.e. the new and the existing one).

When the Arbitrator decides that it is not possible to reuse existing slice resources, the Network Slice LCM is informed that for the given new network slice instance to be created, new slice resources have to be instantiated or configured in all the technology-specific domains (e.g. new NFV Network Services and new RAN slices configurations). On the other hand, when the Arbitrator that at least some of the slice resources can be shared, the information related to these slice resource instances is passed-back to the Network Slice LCM to take care of the required actions for reusing them in the new network slice instance deployment process.

As for the Translator, the SliceNet Slice Orchestrator Arbitrator is a widely enhanced version of the 5GT-VS one due to the substantially different decision logic to be supported, mostly in terms of network slice model and RAN resources used in SliceNet.

5.3.7 Network Slice Inventory

The Network Slice inventory in the Slice Orchestrator is intended to store and maintain up to date information related to the provisioned and running NSIs and NSSIs. The main relevant characteristics and configurations of slice resources are kept into the inventory, in terms of RAN slices, used NFV Network Service Instances, and reference to the instantiated Network Functions (NFs), either virtual or physical. The Network Slice inventory is implemented as a database service, and the data stored follows the network slice models described in section 4.

The Network Slice Inventory is mostly accessed by the Arbitrator for deciding upon slice resource reuse and share, and by the Network Slice Manger to expose such information to other SliceNet NSP cognitive management framework components (e.g. the FCAPS Manager in the FCAPS framework) that may require it to fulfill their management logics.

5.3.8 P&P driver

As it happens in the Service Orchestrator, the Slice Orchestrator is also responsible for coordinating the allocation of those per-slice control and management functions which provide additional customized runtime features for P&P aspects. Indeed, in SliceNet, the regular per-domain slice management is augmented with specific per-slice runtime control functions that are differentiated from slice instance to slice instance according to the specific requirements included in the NSTs.

As a result, for each network slice instance, the Network Slice LCM interacts with the P&P Manager through the dedicated driver that implement the APIs described in deliverable D6.3 [28]. This is done to fulfil the P&P requirements expressed in the NSTs (as defined in section 4.2).

Being these P&P features specific and novel functionalities integrated in the SliceNet cognitive management and orchestration framework, this driver is newly introduced with respect to the 5GT-VS.

5.3.9 Southbound resource control drivers

As part of its lifecycle coordination workflows, the Network Slice LCM has to interact with various technology-specific resource controllers and orchestrators to perform the slice resource instantiation and configuration following the requirements described in the NSTs.

Following the SliceNet overall principles, and the NSP cognitive management and control platform architecture, three main interactions are required for the Slice Orchestrator and external SliceNet components:

i) the RAN controller for creating slices in the radio domain,

ii) the QoS controller for enforcing QoS configurations in the different NSP network domains at runtime,

iii) the NFV Orchestrator for deploying and configuring Network Services and VNFs.

For each of these interactions, a southbound driver is included in the Slice Orchestrator to implement the specific APIs exposed by the slice resource controllers and orchestrators selected in SliceNet to implement the three features above:

- The RAN Adapter, on top of the Mosaic-5G FlexRAN controller, for the RAN slices creation
- The SliceNet QoS Control Plane Service, for runtime QoS configurations in RAN, edge, core and WAN network segments

- The SliceNet NMR-O, on top of the ETSI OSM [15], for the NFV Network Services Orchestration and VNF Management, in accordance with the functionalities described in section 5.3

5.4 NMR-O Internal Architecture

This section describes the functional split of the NMR-O to fulfill the functional requirements detailed in section 3.2.3. As detailed in [25], the NMR-O stands for NFV, MEC, RAN Orchestrator and was designed to act as the network domain orchestration, that is, the entity responsible for managing the network services lifecycle and the resources associated with them. It is worth noting here that,

although RAN slicing responsibility was assigned to the NMR-O during the design phase [25], it has now been moved to the Slice Orchestrator. The reason behind this is that the Resource Orchestrator does not have the view of the NS, so it should not have any slicing function. Hence, in its final design, the NMR-O provides the functionalities listed below:

- Exposure of network service offers to the Slice Orchestrator
- Onboarding and maintenance of network service descriptors and VNF descriptors
- Lifecycle management of the network service instances and their associated VNFs
- Up-to-date maintenance of the status and characteristics of the network service instances and their associated VNFs
- Enabling network service instances monitoring following the principles defined in D6.3 **Error! Reference source not found.**

Figure 23 depicts the functional architecture of the Resource Orchestrator of the SliceNet Orchestration plane. The main work element of the NMR-O is the Network Service, which is basically a set of virtual and physical functions and configurations that are configured at the infrastructure of the NSP and work in a coordinated way to provide a more complex functionality to the NS. Hence, the NMR-O receives Network Service requests from the SS-O at NSP level through the NBI. These requests are processed by the Network Service Orchestrator (NSO), which splits them into the lower level configurations required by the NFV-O (which is implemented by Open Source MANO, OSM) and the Extended Infrastructure Manager (EIM). The NMR-O relies on OSM as the enabler for the provisioning of the VNFs that will compose the Network Service. In addition, the Extended Infrastructure Manager component is responsible for the infrastructure configurations that are not supported by OSM, such as some kinds of low level network configurations. The NMR-O maintains a catalogue of Network Services that can be offered to the SS-O, and an inventory to keep track of the running ones.

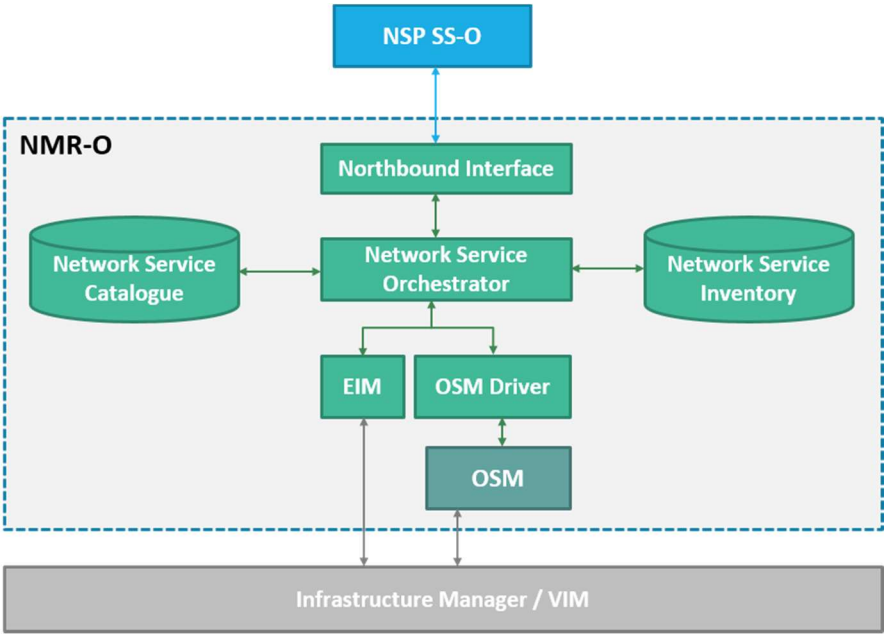

Figure 23: NMR-O functional architecture

5.4.1 Northbound Interface

The NBI of the NMR-O provides a set of RESP APIs that allow for the allocation of the Network Services associated to a slice. To this end, the NBI implements a set of operations to expose the available Network Services offerings to the SS-O, which are stored in the catalogue. These services can be

instantiated and configured through a set of operations offered by the NBI. The network service instances, which are stored in the inventory, can be consulted, as well.

5.4.2 Network Service Orchestrator

The NSO collects the requests coming from the SS-O at NSP through the NBI and processes them. In this regard, the NSO is responsible for orchestrating the network service lifecycle management, which is mainly supported by OSM, and the additional configurations (by means of the EIM) that may be needed to be done over the infrastructure to fulfill the request. In addition, the NSO also manages the exposure of the network service offers that are stored in the catalogue and the network service instances status that is tracked in the inventory.

5.4.3 Catalogue and Inventory

The NMR-O maintains a catalogue of the onboarded network services that can be instantiated by the NSP. It is worth noting here that the network service onboarding is done in a previous phase and, in the current version, manually. This catalogue is currently based on the one provided by OSM although some extension can be needed to provide complete automated network service allocation. This is expected to be analyzed and developed during the integration of the orchestration plane in WP8.
Similarly, the inventory maintained by the NMR-O, which contains the network services instantiated over the infrastructure, is currently based on the one provided by OSM but further extensions will be analyzed during the integration phase.

5.4.4 Open Source MANO

The core of the NMR-O is implemented by Open Source MANO [15]. OSM is based on the ETSI NFV Architecture proposed in [39] and focuses on covering the VNF Manager component, which is related to the VNF lifecycle management, and the NFV Orchestrator, which handles the general resource orchestration and network service lifecycle. The main modules of OSM are the Resource Orchestrator (RO), the Lifecycle Manager (LCM) and the VNF Configuration and Abstraction (VCA). The VCA is in charge of configuring the VNFs associated to the network service, the LCM manages the instantiation, maintenance and deletion of the VNFs and the network services and, finally, the RO acts as the orchestration module providing blueprints, implementing methods and coordinating operations between modules. It is worth noting here that OSM also provides a NBI module that enables REST-based network service configuration. Inter-module communication in OSM is realized by means of a message bus.
From release four on, OSM has been adding and extending monitoring capabilities as well as including some policy-based actuations. Two new modules, namely Monitoring (MON) and Policy (POL) have been introduced in the architecture for this purpose. In brief, monitoring parameters and associated thresholds can be added to the VNFD. These parameters are collected by the POL which requests the creation of a new alarm to the MON module through the message bus. The MON, in turn, creates the alarm with the specified parameters and starts the monitoring process. In case the established thresholds are surpassed, the MON raises an alarm, which is inserted in the message bus and detected by the POL. Then, the POL triggers the actuation defined in the descriptor, which is applied by means of the LCM. In the context of SliceNet, the monitoring mechanism (not the actuation) is connected to the FCAPS management system [28]. In particular, the FCAPS system is connected to the message bus of OSM to consume the monitoring information of the virtual infrastructure. For the actuation the loop defined in the framework of SliceNet is used.
In NMR-O functional architecture, OSM is expected to provide the orchestration of the VNFs composing the network services requested from the SS-O. In this regard, the NSO forwards the network service related operations received through its NBI to the OSM NBI. The OSM Driver sub-

module enables this communication. Such requests are based on the NSDs and VNFDs supported by OSM, which are compliant with the ETSI standards as described in section **Error! Reference source not found.**. Upon the reception of these requests, OSM contacts the Virtual Infrastructure Manager(s) of the NSP to allocate and configure the computational and network resources to provide the network service(s).

5.4.5 Extended Infrastructure Manager

As said in section 2.4.1, OSM still has some lacks to support end-to-end vertical-oriented cognition-based service provisioning. The EIM aims at implementing the operations needed to configure the full network service in support of the slices provided by the SS-O. For example, additional configurations may be needed to interconnect independent network service instances associated to the same slice, or to provide network service instances connectivity to services allocated in a different domain (shall it be different VIM, operator, NSP, etc.). To do this, the EIM has to contact the infrastructure manager of the NSP as depicted in Figure 23.

5.5 Cross-layer orchestration considerations

The SliceNet orchestration architecture lays its foundations on the business roles split described in section 3, where verticals, DSPs and NSPs have clearly separated responsibilities and have access to specific managed entities, which are respectively the vertical services, the end-to-end network slices and the single domain network slices composed by technology and domain specific resources. Indeed, dedicated orchestration functions are required to manage the lifecycle of these different managed entities which identify three main management domains and scopes:

- Vertical services and end-to-end network slice orchestration domain
- Network slice orchestration domain
- Resource orchestration domain

This gives to the SliceNet orchestration approach a first cross-layer dimension, where three main orchestration engines cooperate for the final aim of delivering to verticals tailored end-to-end services provisioned as a combination of single domain network slices with added value capabilities and functionalities like cognitive based QoE end-to-end optimization, Plug & Play, multi-domain actuation and fault management. This is done by keeping logically separated the management logics for services, slices and resources which natively require different logics with different granularities. In particular, having the interactions among DSPs and NSPs at the network slice level (assuming the use of the models in section 4), guarantees a proper resource abstraction in the exposure of management capabilities and actions from NSPs to DSPs. The gluing and integration of these orchestration engines (Service Orchestration at the DSP level, and Slice and Resource Orchestration at the NSP level) is driven by the workflows defined in section 6 and by the APIs defined in section 7.

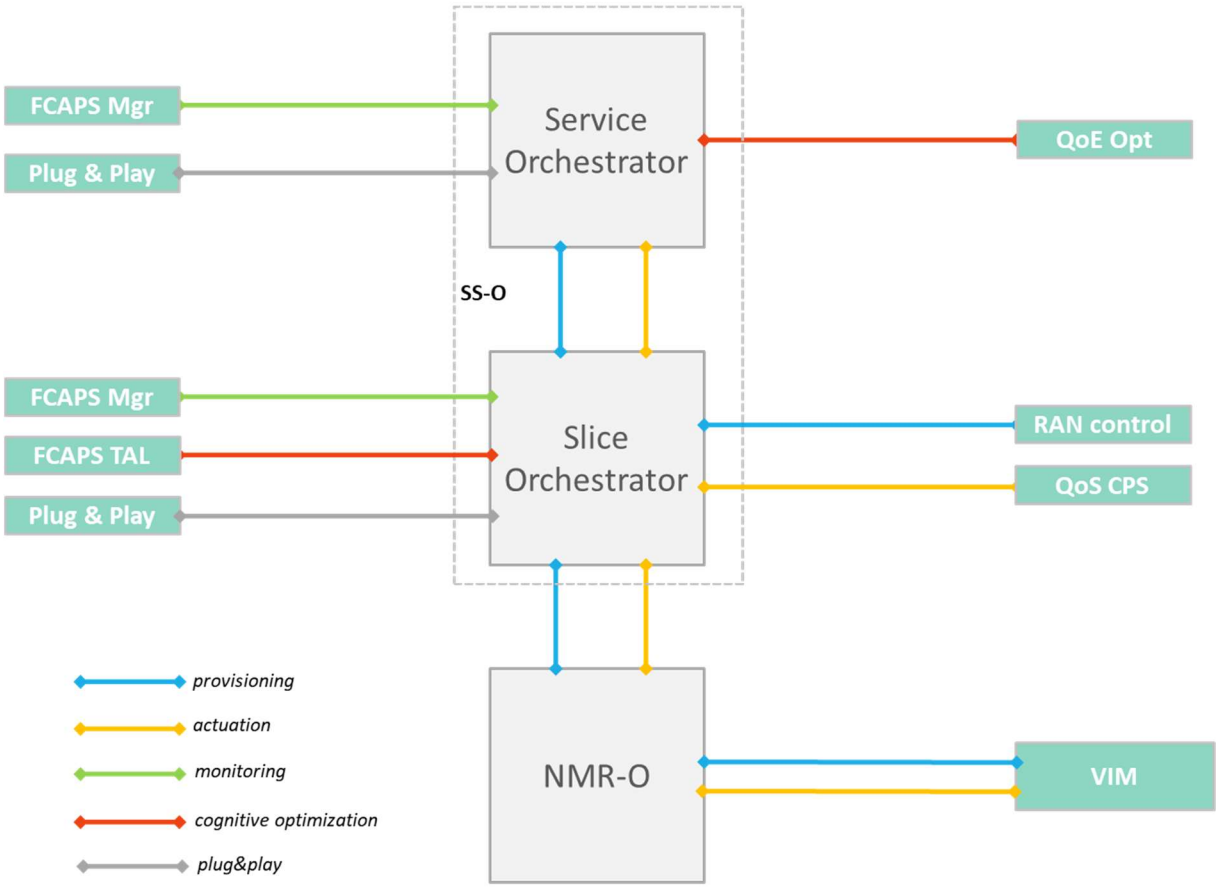

Figure 24 Cross-layer Slice Orchestration approach: interfaces and coordination actions

Besides this macroscopic cross-layer orchestration dimension, each of the Service, Slice and Resource orchestrators in turn provide to the SliceNet cognitive management platform key cross-layer coordination features within their domain scope that aim at gluing the lifecycle management operations towards a seamless integration of various aspects of services, slices and resources management including, among others, resource provisioning, FCAPS, Plug & Play, actuation, cognitive optimization. Figure 24 provides an overview of how each of the SliceNet orchestration engine is interfaced and integrates the different SliceNet cognitive management logics. The SS-O (as combination of DSP Service and NSP Slice Orchestrators) and the NMR-O are therefore clearly fundamental cross-layer coordination entities that make consistent heterogeneous logics enabling a flexible and customizable approach. The different interactions and coordination sources of each of the SliceNet orchestration engine are shown in Figure 24 with different colours, and can be summarized as follows:

- DSP Service Orchestrator: coordinates end-to-end network slice provisioning, multi-domain FCAPS, QoE optimization, Plug & Play
- NSP Service Orchestrator: coordinates network slice and slice subnets provisioning, single-domain FCAPS, cognitive optimization, Plug & Play
- NSP NMR-O: coordinates the provisioning and actuation of NFV related resources in the virtualized infrastructure

With respect to new emerging management and orchestration approaches, like those proposed and under definition in ETSI ZSM (see section 2) for end-to-end service management automation, the

SliceNet proposed orchestration architecture makes a first step towards flexible and scalable coordination of different managed domains and entities. While ETSI ZSM proposes a pioneer approach where all of the management domains are loosely coupled and coordinated by end-to-end service management logics through a cross-domain integration fabric, SliceNet takes a more conservative approach where at least part of the management interactions are occurring through traditional well-defined interfaces and procedures (e.g. based on REST). However, SliceNet follows an architectural approach compatible with the ETSI ZSM for what concerns the exposure of well-defined services between DSPs and NSPs through the related orchestrator interactions for domain control (as combination of provisioning and actuation), domain monitoring data collection, domain cognitive functions. Moreover, the interactions between the DSP Service Orchestrator and NSP Slice Orchestrator are enabled by the Communication Services defined in section 5.2 and 5.3 which offer a flexibility in the implementation of various options for the orchestrator to orchestrator interfaces, enabling solutions like an ETSI ZSM like approach based on publish/subscribe message bus mechanisms.

6 Orchestration Workflows

6.1 Design, Onboard and Offer

This section describes the design, onboarding and offer workflows at the DSP and NSP levels. Basically it includes services, network slices and resources. Figure 25 illustrates the design, onboarding and offer workflow:

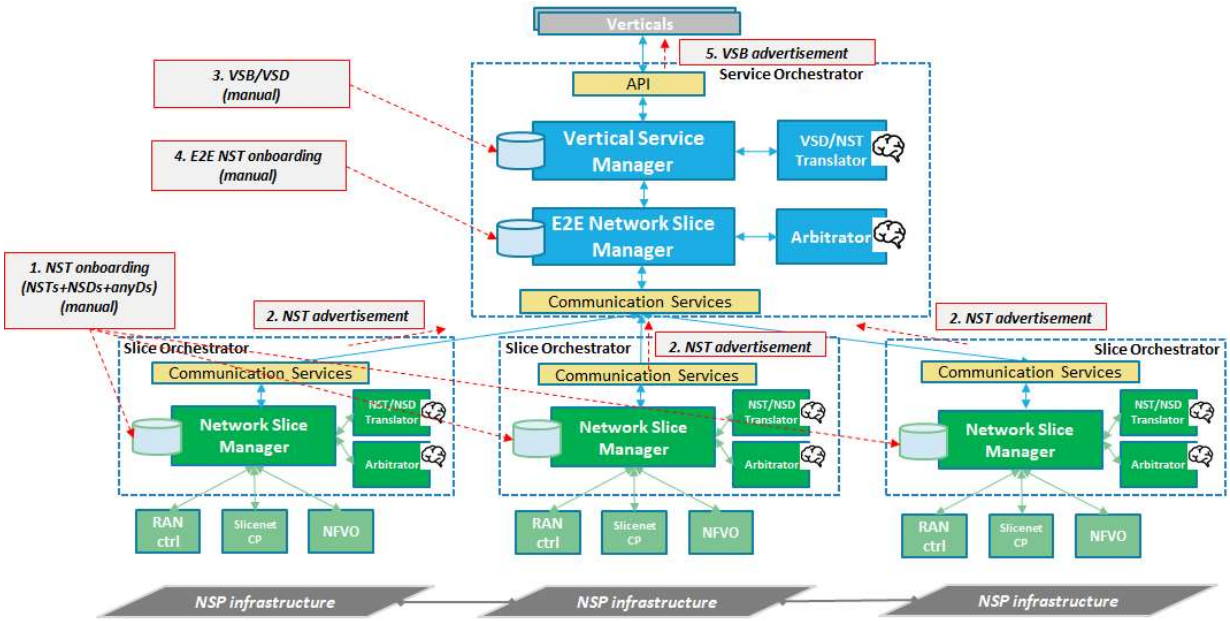

Figure 25: Service and Slice Design, onboarding and Offer workflow

Table 24 describes the workflow steps shown in Figure 25.

Table 24: Service and Slice Design, Onboard and Offer workflows

Step	Workflow step description
1	Each NSP will manually design and onboard their NSTs in a catalogue.
2	The next step will be to offer their NSTs through an API (Communication Services) which will be consumed by the DSPs.
3	Each DSP will manually design and onboard their VSBs in a catalogue.
4	Each DSP will manually design and onboard their end-to-end NSTs in a catalogue.
5	Each DSP will offer their services through an API which will be consumed by the Verticals.

6.2 Instantiation

This section describes the Vertical service request and the respective service/slices instantiation workflow at DSP and NSP levels. It includes Vertical services, end-to-end NSs and NSs instantiations. Figure 26 illustrates the instantiation workflow:

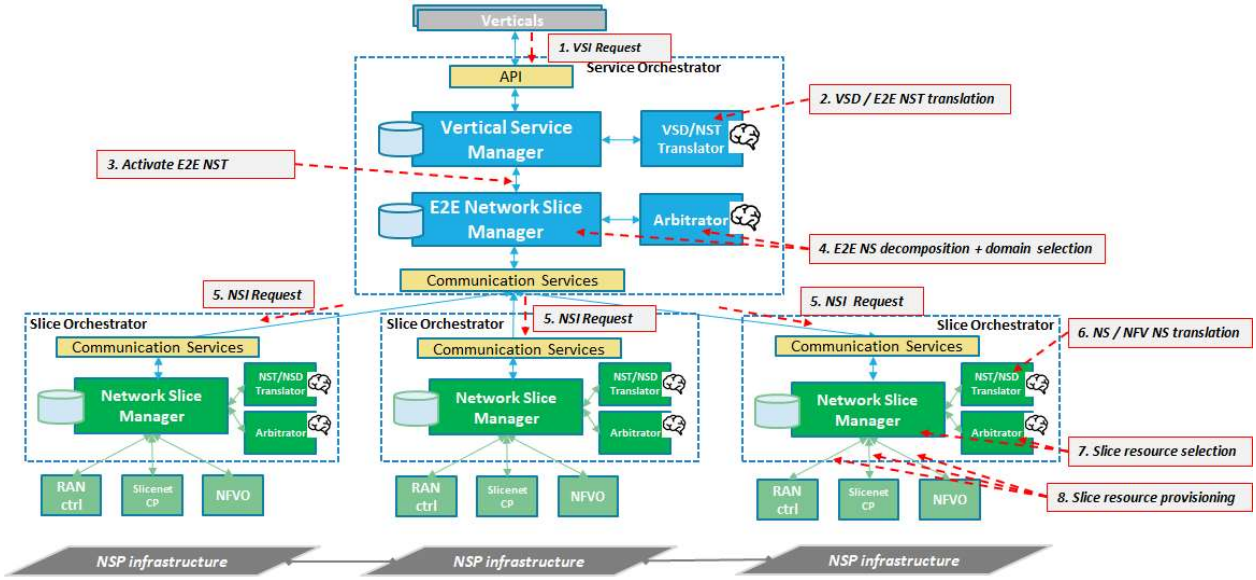

Figure 26: Service and Slice Instantiation workflow

Table 25 describes the workflow steps shown in Figure 26.

Table 25: Service and Slice Instantiation workflow

Step	Workflow step description
1	The Vertical selects the appropriate VSB, taken into account his needs in terms of requirements (latency, bandwidth, SLA, geography, etc.), providing all the information necessary for the DSP to instantiate a VSD.
2	The Service Orchestrator maps this VSD into an end-to-end NST, describing the end-to-end NS details for this specific Vertical service.
3	The Vertical Service Manager requests the instantiation of the end-to-end NST to the Network Slice Manager.
4	The arbitrator decomposes the end-to-end NST and selects the appropriate NSs that comply with the Vertical requirements. Finally it informs the end-to-end NS Manager to request the network slices instances to the selected NSPs.
5	NSI request to the selected NSPs.
6	The NSI request is translated into the required NFV network services.

7	The arbitrator decomposes the NS and selects the appropriate resources (PNFs/VNFs) that comply with the NSI request. Finally it informs the NS Manager of the selected resources.
8	The NS Manager will contact the NMR-O in order to provision the selected resources (see Figure 27).

At the NSP level the NMR-O will be responsible for the orchestration of the resources involved in the respective NS. Figure 27 describes the NMRO Slice instantiation workflow.

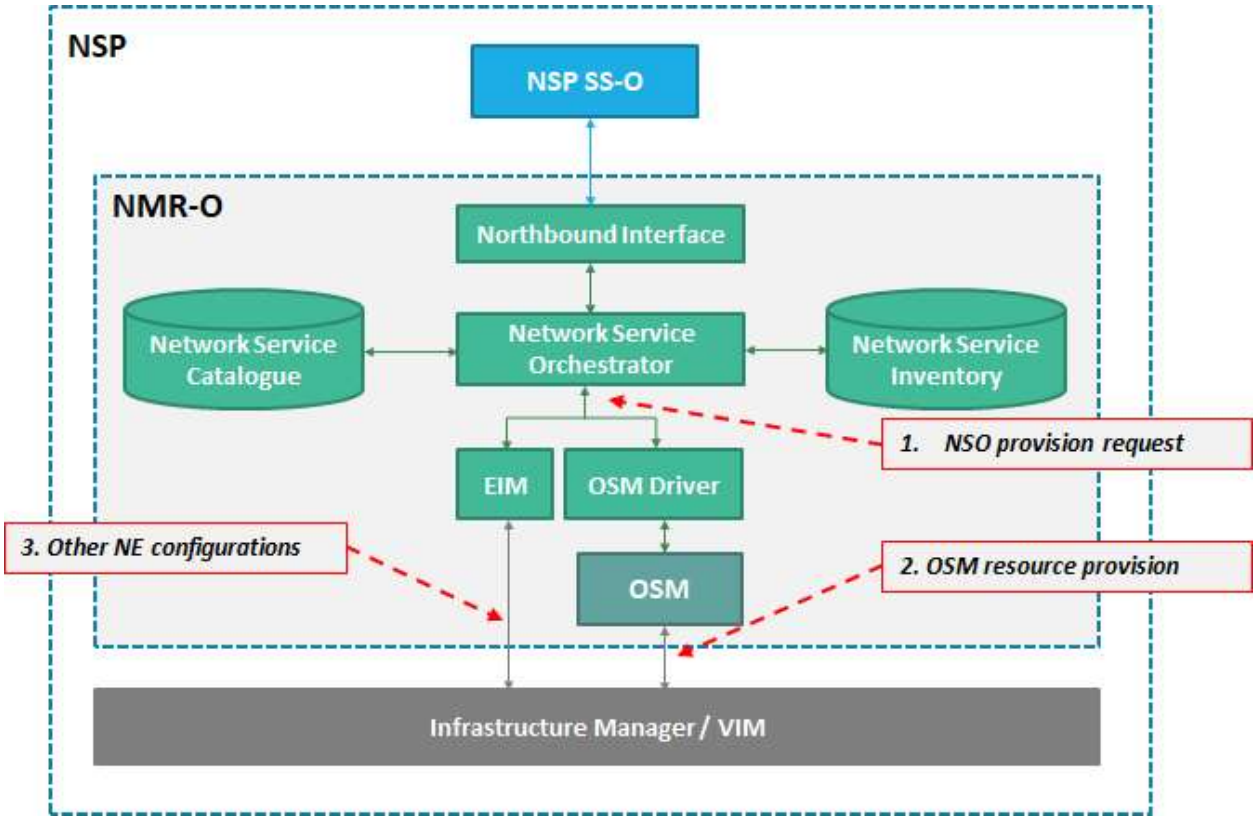

Figure 27: NMRO Slice instantiation workflow

Table 26 describes the workflow steps shown in Figure 27.

Table 26: NMRO Slice Instantiation workflow

Step	Workflow step description
1	NSO provision request to OSM
2	OSM resource provisioning
3	Other Network Element configurations

6.3 NSP NS Optimization

During runtime the NS can and will experience alarms and/or performance degradation. Some performance degradation events, if not addressed carefully and in the correct time, will cause an alarm that most likely will cause a NS downtime or severe degradation. The permanent monitoring of the resources belonging to a specific NS can prevent this to occur, by performing an optimization in the NS that "heals" the resource(s) causing the unwanted degradation. If this NS optimization is not done the NSP will face a violation of the SLA agreed with the DSP, either in the form of requirements degradation (bandwidth, latency, etc.) or even and more serious an out of service NS. Figure 28 shows the NSP optimization workflow:

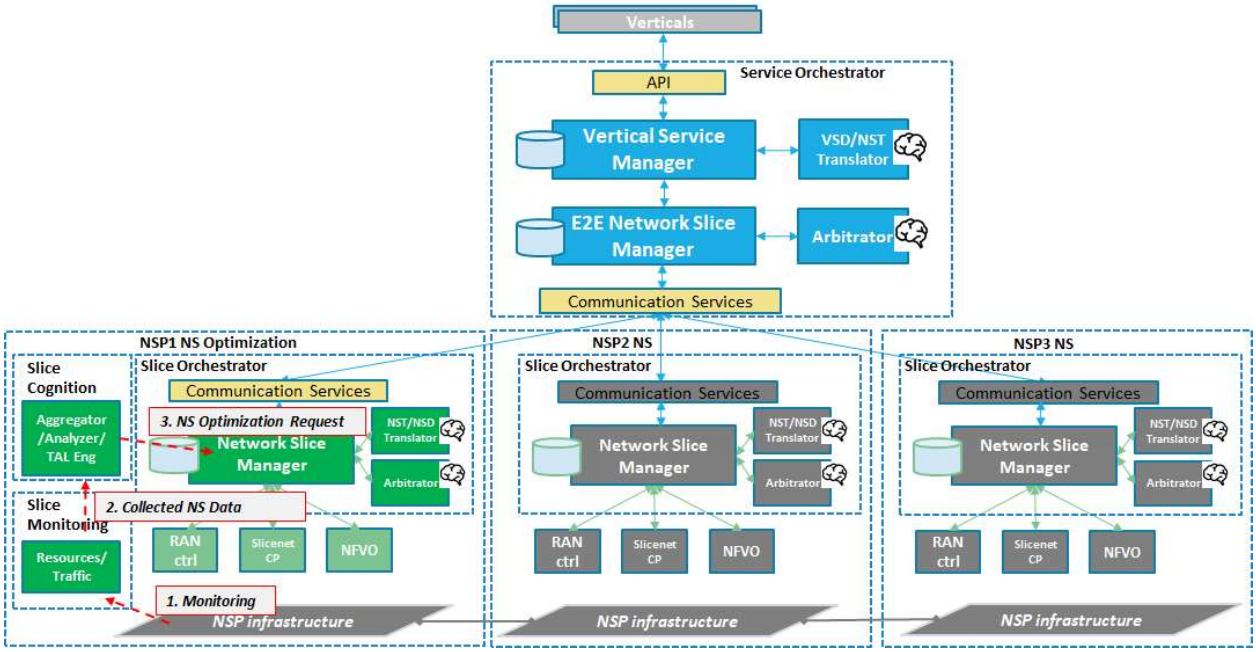

Figure 28: NSP NS Optimization workflow

Table 27 describes the workflow steps in more detail.

Table 27: NSP NS Optimization workflow

Step	Workflow step description
1	Resources and traffic monitoring.
2	Feed a Data Lake with all this monitoring information which will be consumed by the Slice Cognition module.
3	If a NS Optimization is necessary the Network Slice Manager is triggered in order to make all the necessary actions in the NS to keep the SLA with the DSP.

6.4 DSP End-to-end NS Optimization

Also during runtime the end-to-end NS can and will experience alarms and/or performance degradation in the same way as what happens in the NSP domain. Some performance degradation events, if not addressed carefully and in the correct time, will cause an alarm that most likely will cause an end-to-end NS downtime or severe degradation. The permanent monitoring of the DSP resources and corresponding NSs belonging to a specific end-to-end NS can prevent this to occur, by performing an optimization in the faulty/degraded NS. If this end-to-end NS optimization is not done the DSP will face a violation of the SLA agreed with the Vertical, either in the form of requirements degradation (bandwidth, latency, etc.) or even and more serious an out of service end-to-end NS. The following figure shows the DSP end-to-end NS optimization workflow, where in the presented scenario we have an end-to-end NS composed of 2 NS, one from NSP1 and the other from NSP2. At some point in time the NSP2 NS violates the SLA and the DSP will replace this faulty NS by a healthy NSP3 NS that fulfills the requirements and the SLA. Figure 29 shows the DSP end-to-end NS Optimization workflow.

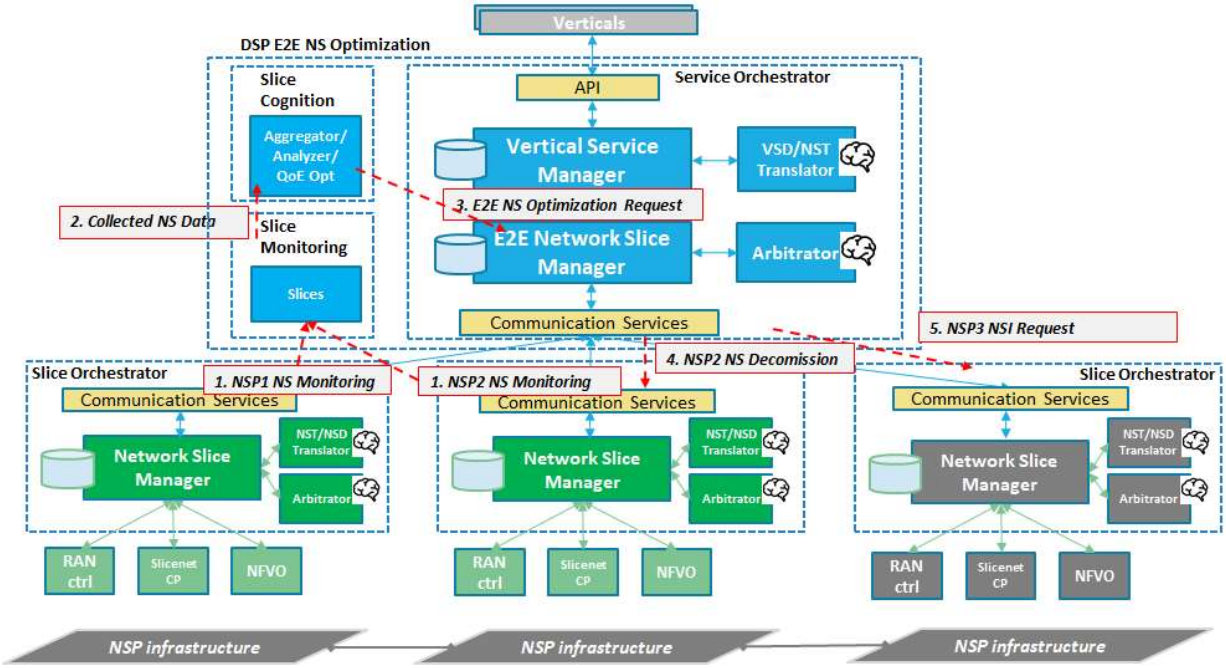

Figure 29: DSP end-to-end NS Optimization workflow

Table 28 describes the workflow steps in more detail.

Table 28: DSP end-to-end NS Optimization workflow

Step	Workflow step description
1	DSP is monitoring all the NSs that belongs to a particular end-to-end NS
2	This NSs monitoring information is fed into a Data Lake and will be consumed by the Slice Cognition module at the DSP level

3	If a given NS fails or is performing under the SLA an end-to-end NS Optimization is necessary. Thus the end-to-end Network Slice Manager is triggered in order to make all the necessary actions in the end-to-end NS to keep the SLA with the Vertical
4	The faulty NSP2 NS is informed that will be decommissioned
5	The NSP3 is requested to instantiate a NS with the required parameters

Figure 30 shows the new DSP end-to-end NS that fulfils the necessary service SLA, and is being permanently monitored to verify the agreed SLA.

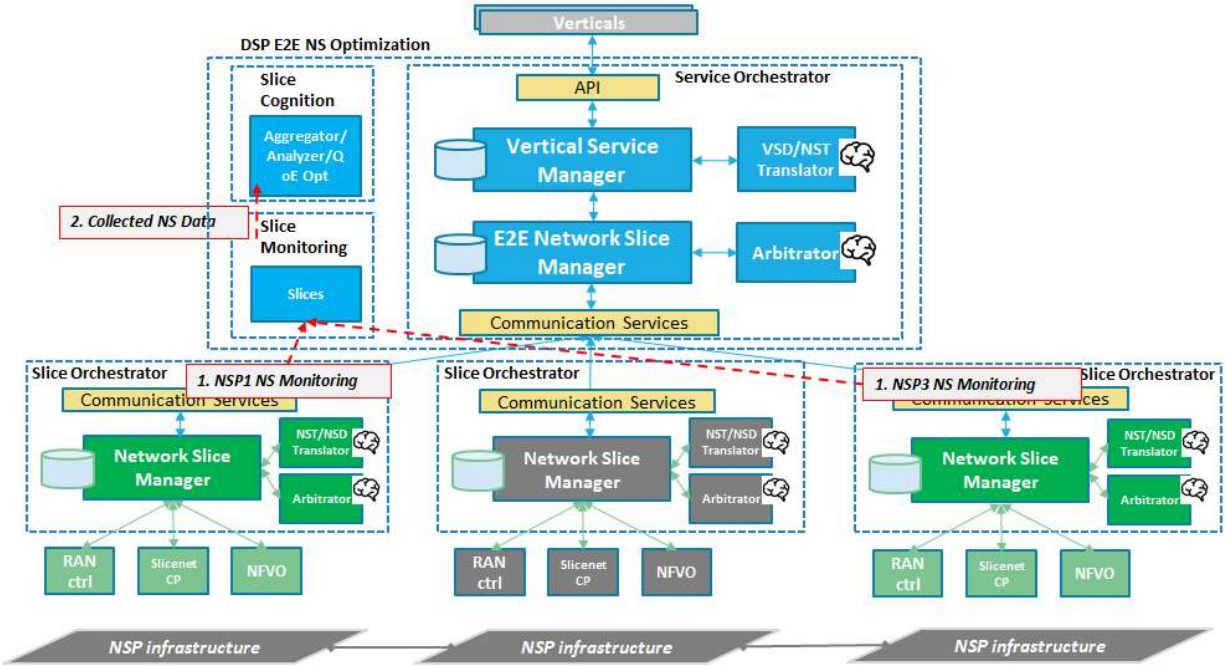

Figure 30: DSP end-to-end NS Optimization workflow (final state)

6.5 Vertical service reconfiguration through P&P

During the service lifetime the Vertical may want to change a specific requirement (e.g., latency, bandwidth, etc.) or may want to add new vertical devices in new locations. This can have an impact in the DSP and/or NSP and it must be checked whether the new reconfiguration is feasible or not. Two considerations should be taken into account, the first one is related with the requirements and it should be checked if the DSP/NSP can deal with the new requirements reconfiguration, the second one is related with the new vertical location devices as this cannot be done in the actual end-to-end NS, for instance if the new devices falls out of the domain of the existent NSPs, and thus new NSPs should contacted to provision the necessary NSs to fulfil the new end-to-end NS. This later reconfiguration scenario may, and most certainly will, have an impact in the Vertical service as this involves new hardware to be installed. Figure 31 shows the Vertical service reconfiguration through P&P workflow.

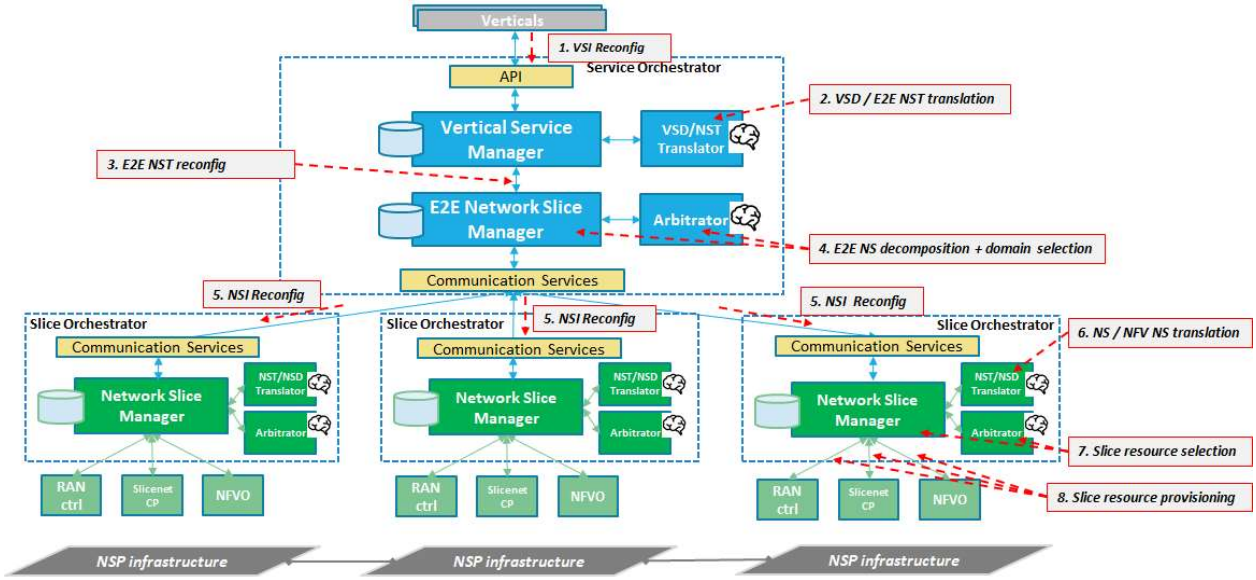

Figure 31: Vertical reconfiguration through P&P workflow

Table 29 describes the workflow steps in more detail.

Table 29: DSP end-to-end NS Optimization workflow

Step	Workflow step description
1	The Vertical selects the appropriate VSB, and makes the necessary modifications according to his needs in terms of requirements (latency, bandwidth, SLA, geography, etc.).
2	The Service Orchestrator maps the new reconfigured VSD into an end-to-end NST, describing the end-to-end NS details for this specific Vertical service.
3	The Vertical Service Manager requests the reconfiguration of the new end-to-end NST to the Network Slice Manager.
4	The arbitrator decomposes the new end-to-end NST and informs the end-to-end NS Manager to request a NSI reconfiguration to the selected NSPs. If additional NSPs are required the DSP shall contact them and send a NSI request for each one.
5	NSI reconfiguration request to the selected NSPs.
6	The NSI reconfiguration request is translated into the required NFV network services.
7	The arbitrator decomposes the NS and selects the appropriate resources (PNFs/VNFs) that comply with the NSI reconfiguration request. Finally it informs the NS Manager of the selected resources.
8	The NS Manager will contact the NMR-O in order to provision the selected resources (see Figure 32).

At the NSP level the NMR-O will be responsible for the orchestration of the resources involved in the respective NS. Figure 32 shows the NMRO Slice reconfiguration workflow.

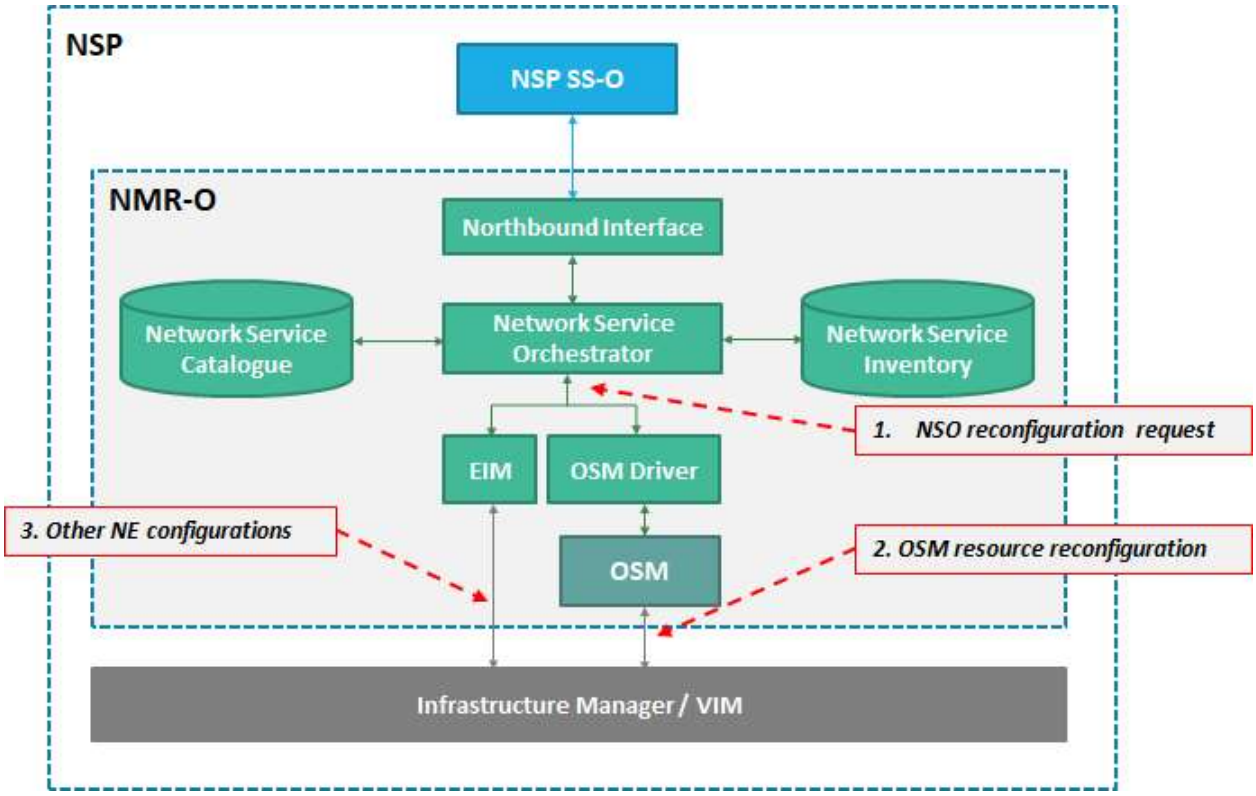

Figure 32: NMRO Slice reconfiguration workflow

Table 30 describes the workflow steps shown in Figure 32.

Table 30: NMRO Slice reconfiguration workflow

Step	Workflow step description
1	NSO reconfiguration request to OSM
2	OSM resource reconfiguration
3	Other Network Element configurations

6.6 Vertical service decommission

If the Vertical wants to end the service contracted with the DSP it will ask for a Vertical service decommission request. The DSP will act accordingly, e.g. removing the vertical service and the end-to-end NS from the respective inventories among other internal actions, and will also inform, via an NSI decommission request, each of the involved NSPs in the end-to-end NS. Figure 33 shows the Vertical service decommission workflow.

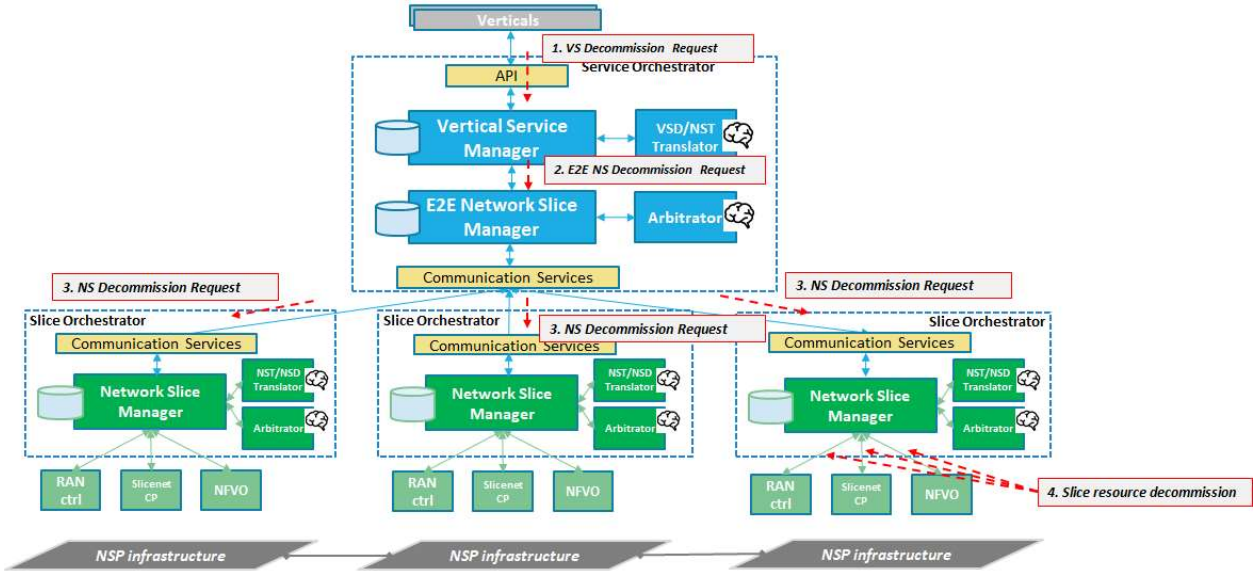

Figure 33: Vertical service decommission workflow

Table 31 describes the workflow steps in more detail.

Table 31: Vertical service decommission workflow

Step	Workflow step description
1	Vertical initiates a service decommission request
2	The DSP VS Manager delivers the notification to the end-to-end NS Manager. The VS Inventory is updated accordingly.
3	The request is then delivered to all the NSPs, involved in the end-to-end NS, informing that the respective NS should be decommissioned. The end-to-end NS Inventory is updated accordingly.
4	The NSP NS Manager will make decommission of all the resources involved in the respective NS. The NS Inventory is updated accordingly.

7 SliceNet Orchestration Interfaces

7.1 Service-level interfaces

The Service Orchestrator at the DSP level exposes its northbound interface through a set of REST APIs based on the HTTP protocol and JSON messages, offering different kind of features towards the verticals (through the One Stop API framework) and the DSP administrator. In particular, the following two set of APIs are defined at the northbound of the Service Orchestrator:

- Vertical service management APIs, which provide a set of endpoints exposed towards verticals (through the One Stop API) for vertical service lifecycle management aspects, including retrieval of VSBs, management of VSDs and operational actions VSIs (instantiation, termination, modification, etc.).
- Management and administrative APIs, which are offered to the DSP system administrators and allow managing tenants and VSBs. They are intended to be used internally to the DSP (i.e. not exposed to third parties)

The Service Orchestrator implements the REST server side of these APIs, while the verticals (through the One Stop API) and the DSP OSS platform provide REST client sides. These APIs also support asynchronous notifications generated by the Service Orchestrator as defined in the following subsections.

Authentication and authorization functionalities over the Service Orchestrator REST APIs will be provided by the One Stop API framework.

With respect to the 5GT-VS northbound interface, these Service Orchestrator APIs are enhanced to support the SliceNet vertical service model (described in section 4) and to fulfil the requirements of the whole DSP cognitive management platform.

7.1.1 Vertical service management APIs

The Vertical service management APIs offered by the Service Orchestrator implement the following list of vertical service lifecycle management operations:

- Query VSBs
- Create, query, update, and delete VSDs
- Instantiate, query, modify and terminate VSIs
- Actuation for end-to-end NSI optimization
- Notifications about vertical service lifecycle events

The following subsections detail how these operations translate into REST resources and methods in the form of tables.

7.1.1.1 Query Vertical Service Blueprints

Table 32: Service Orchestrator REST APIs: Get all VSBs

Endpoint	/vs/catalogue/vsblueprint/

HTTP Verb	GET
Description	Retrieve all the VSBs (associated to a given vertical) from the Service Orchestrator catalogue. The blueprints can then be used by the vertical to create the VSDs for the vertical services to be instantiated.
Caller	Vertical / One Stop API
Request	• Vertical ID (i.e. tenant identification)
Response	• List of VSBs (following the format specified in section 4)
Response Codes	• 200 - VSBs retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

Table 33: Service Orchestrator REST APIs: Get VSB

Endpoint	/vs/catalogue/vsblueprint/<vsb_id>
HTTP Verb	GET
Description	Retrieve an individual VSB (associated to a given vertical) from the Service Orchestrator catalogue. The blueprint can then be used by the vertical to create the VSD for the vertical service to be instantiated.
Caller	Vertical / One Stop API
Request	• Vertical ID (i.e. tenant identification) • VSB ID
Response	• VSB matching the issued VSB ID (following the format specified in section 4)
Response Codes	• 200 – VSB retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.1.2 Create Vertical Service Descriptor

Table 34: Service Orchestrator REST APIs: Create VSD

Endpoint	/vs/catalogue/vsdescriptor/

HTTP Verb	POST
Description	Create a new VSD to be added into the Service Orchestrator catalogue. The VSD is built by parameterizing an existing VSB. The VSD can then be used by the vertical to request the instantiation of a new vertical service.
Caller	Vertical / One Stop API
Request	• Vertical ID (i.e. tenant identification) • VSD (following the format specified in section 4.1) • Is Public (set to true if the VSD can be shared to other verticals)
Response	• VSD Identifier
Response Codes	• 201 – VSD created successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.1.3 Query Vertical Service Descriptor

Table 35: Service Orchestrator REST APIs: Get all VSDs

Endpoint	/vs/catalogue/vsdescriptor/
HTTP Verb	GET
Description	Retrieve all the VSDs (owned by a given vertical) from the Service Orchestrator catalogue. The descriptors can then be used by the vertical to request for the instantiation of vertical services
Caller	Vertical / One Stop API
Request	• Vertical ID (i.e. tenant identification)
Response	• List of VSDs (following the format specified in section 4)
Response Codes	• 200 - VSDs retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

Table 36: Service Orchestrator REST APIs: Get VSD

Endpoint	/vs/catalogue/vsdescriptor/<vsd_id>
HTTP Verb	GET
Description	Retrieve an individual VSD (owned by a given vertical) from the Service Orchestrator catalogue. The blueprint can then be used by the vertical to create the VSD for the vertical service to be instantiated.
Caller	Vertical / One Stop API
Request	• Vertical ID (i.e. tenant identification) • VSD ID
Response	• VSD matching the issued VSD ID (following the format specified in section 4)
Response Codes	• 200 – VSD retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.1.4 Update Vertical Service Descriptor

Table 37: Service Orchestrator REST APIs: Update VSD

Endpoint	/vs/catalogue/vsdescriptor/<vsd_id>
HTTP Verb	PUT
Description	Update an individual VSD (owned by a given vertical) into the Service Orchestrator catalogue. A VSD can be updated only if there are no VSIs created based on it. The updated VSD will have a new VSD ID, and the previous version will be available with the old VSD ID.
Caller	Vertical / One Stop API
Request	• Vertical ID (i.e. tenant identification) • Updated VSD (following the format specified in section 4.1) • Is Public (set to true if the VSD can be shared to other verticals)
Response	• New VSD ID
Response Codes	• 201 – VSD updated successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.1.5 Delete Vertical Service Descriptor

Table 38: Service Orchestrator REST APIs: Delete VSD

Endpoint	/vs/catalogue/vsdescriptor/<vsd_id>
HTTP Verb	DELETE
Description	Delete an existing VSD from the Service Orchestrator catalogue. A VSD can be deleted only if there are no VSIs created based on it.
Caller	Vertical / One Stop API
Request	• Vertical ID (i.e. tenant identification) • VSD ID
Response	• None
Response Codes	• 204 – VSD deleted successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.1.6 Create Vertical Service Instance

Table 39: Service Orchestrator REST APIs: Create VSI

Endpoint	/vs/basic/vslcm/vs
HTTP Verb	POST
Description	Create a new VSI, given its reference VSD. The Service Orchestrator starts the instantiation procedure and returns the VSI ID that can be used by the vertical to retrieve information about the status and the attributes of the VSI, using the Query VSI operation. In parallel, lifecycle notifications are generated asynchronously towards the vertical / One Stop API in case a suitable notification endpoint was provided at the request time.
Caller	Vertical / One Stop API
Request	• Vertical ID • VSI name • VSI description • Reference VSD ID • Notification endpoint, where the Vertical / One Stop API can receive notifications about the lifecycle or failure events of the VSI) • VSI configuration parameters set by the vertical (and referred in the VSB following the model specified in section 4.1)

Response	• VSI Identifier
Response Codes	• 201 – VSI resource created successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.1.7 Query Vertical Service Instance

Table 40: Service Orchestrator REST APIs: Get all VSIs

Endpoint	/vs/basic/vslcm/vs/
HTTP Verb	GET
Description	Retrieve all the VSIs (owned by a given vertical) from the Service Orchestrator inventory. This operation can be invoked also by any other SliceNet DSP cognitive management platform component to fulfil their specific logic.
Caller	Vertical / One Stop API / SliceNet DSP cognitive management platform components
Request	• Vertical ID (i.e. tenant identification)
Response	• List of VSIs and related information (following the format described in the next table for individual VSI retrieve operation)
Response Codes	• 200 - VSIs retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

Table 41: Service Orchestrator REST APIs: Get individual VSI

Endpoint	/vs/basic/vslcm/vs/<vsi_id>
HTTP Verb	GET
Description	Retrieve an individual VSI (owned by a given vertical) from the Service Orchestrator inventory. This operation can be invoked also by any other SliceNet DSP cognitive management platform component to fulfil their specific logic.
Caller	Vertical / One Stop API
Request	• Vertical ID (i.e. tenant identification) • VSI ID
Response	• VSI ID

	• VSI name • VSI description • VSD ID • VSI status and attributes (i.e. instantiating, instantiated, under modification, terminating, terminated, failed) • per-domain NSI status and attributes (according to the model specified in section 4.2)
Response Codes	• 200 – VSI retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.1.8 Terminate Vertical Service Instance

Table 42: Service Orchestrator REST APIs: Delete VSI

Endpoint	/vs/basic/vslcm/vs/<vsi_id>
HTTP Verb	DELETE
Description	Terminate an existing VSI, given its VSI ID. The Service Orchestrator starts the termination procedure and returns an HTTP code. In parallel, VSI termination notification is generated asynchronously towards the vertical / One Stop API when the VSI is deleted, in case a suitable notification endpoint was provided at the VSI creation time.
Caller	Vertical / One Stop API
Request	• Vertical ID (i.e. tenant identification) • VSI ID
Response	• None
Response Codes	• 200 – VSI termination procedure started successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.1.9 Modify Vertical Service Instance

Table 43: Service Orchestrator REST APIs: Optimize e2e NSI

Endpoint	/vs/basic/vslcm/vs/<vsi_id>
HTTP Verb	PUT

Description	Modify an already instantiated VSI, providing a new VSD. This VSI modification procedure is conceived to enable the vertical to manually modify a running VSI by providing a full new VSD configuration. This means that starting from the same VSB, the vertical needs to first produce multiple VSDs to characterize different options of the same vertical services (e.g. different number of UEs, different geographical locations, etc.). The Service Orchestrator starts the modification procedure and returns an HTTP code. In parallel, VSI modification notification is generated asynchronously towards the vertical / One Stop API when the VSI is modified, in case a suitable notification endpoint was provided at the VSI creation time.
Caller	Vertical / One Stop API
Request	• Vertical ID (i.e. tenant identification) • VSI ID • New VSD ID
Response	• None
Response Codes	• 200 – VSI modification procedure started successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.1.10 Actuation for end-to-end Network Slice Instance optimization

Table 44: Service Orchestrator REST APIs: end-to-end NSI actuation

Endpoint	/vs/basic/vslcm/e2ens/<e2e_nsi_id>/actuate
HTTP Verb	POST
Description	This operation is used by the QoE optimizer to request for an end-to-end network slice actuation, in response to the outcome of a DSP cognitive optimization process (as defined in D5.5 [29] and D5.6 [30]). The Service Orchestrator starts the end-to-end NSI actuation procedure and returns an HTTP code. In parallel, the actuation notification is generated asynchronously towards the caller when the actuation is applied, in case a suitable notification endpoint was provided in the request.
Caller	QoE Optimizer
Request	• end-to-end NSI ID • Actuation name (matching one of the actuations available in the DSP Catalogue and mapped to NSP actuation offerings in NSTs) • Actuation parameters ((according to the parameters defined for the actuation in the DSP Catalogue, and mapped to those from NSP actuation offerings in NSTs) • Notification endpoint, where the caller can receive notifications about the actuation result

Response	• None
Response Codes	• 200 – end-to-end NSI actuation procedure started successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.1.11 Notification of Vertical Service Instance lifecycle event

Table 45: Service Orchestrator REST APIs: Notify VSI lifecycle event

Endpoint	/<endpoint-provided-by-the-vertical-or-onestopapi>
HTTP Verb	POST
Description	This operation is used by the Service Orchestrator to notify the vertical (or the One Stop API) about the results of VSI lifecycle operations that have been executed. The notifications are asynchronous and provide the result of actions that have either triggered by vertical (e.g. instantiation, termination, modification) or by autonomous cognitive driven actuations.
Caller	Service Orchestrator
Request	• VSI ID • Notification type (e.g. instantiation result, termination result, modification result) • Result (success or failure)
Response	• None
Response Codes	• 200 – Notification received successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.2 Management and administrative APIs

The Management and administrative APIs offered by the Service Orchestrator implement the following list of operations:

- Create, query and delete Tenants
- Create, query and delete VSBs

The following subsections detail how these operations translate into REST resources and methods in the form of tables.

7.1.2.1 Create Tenant

Table 46: Service Orchestrator REST APIs: Create Tenant

Endpoint	/vs/admin/tenant/
HTTP Verb	POST
Description	Create a new tenant in the Service Orchestrator. A tenant corresponds to a vertical who has established a business relationship with the DSP. Each tenant is able to request VSIs.
Caller	One Stop API
Request	• Tenant ID (which will be used as Vertical ID by verticals) • Tenant name • Tenant credential
Response	• None
Response Codes	• 201 – Tenant created successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.2.2 Query Tenant

Table 47: Service Orchestrator REST APIs: Query Tenant

Endpoint	/vs/admin/tenant/<tenant_id>
HTTP Verb	GET
Description	Retrieve all the information associated with an existing tenant, including the IDs of the VSDs created by the vertical and the VSIs instantiated for it. Detailed information about each VSD, VSI and SLA can then be obtained through the related query operations.
Caller	One Stop API
Request	• Tenant ID
Response	• Tenant ID • Tenant name • Tenant credential • List of VSD IDs • List of VSI IDs

Response Codes	• 200 – Tenant retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.2.3 Delete Tenant

Table 48: Service Orchestrator REST APIs: Delete Tenant

Endpoint	/vs/admin/tenant/<tenant_id>
HTTP Verb	DELETE
Description	Delete an existing tenant in the Service Orchestrator. A tenant can be removed from only when all its VSIs have been terminated.
Caller	One Stop API
Request	• Tenant ID
Response	• None
Response Codes	• 204 – Tenant deleted successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.2.4 Create Vertical Service Blueprint

Table 49: Service Orchestrator REST APIs: Create VSB

Endpoint	/vs/catalogue/vsblueprint/
HTTP Verb	POST
Description	On-board a new VSB in the Service Orchestrator catalogue, so that it can be used as a baseline by the verticals to create their own VSDs. A VSB is typically designed (off-line), and can include the definition of suitable translation rules between VSDs that can be derived from the blueprint and the corresponding per-domain NSTs, i.e. translation rules between QoS related parameters and NST parameters.
Caller	One Stop API
Request	• VSB (following the model specified in section 4.1) • List of translation rules
Response	• VSB Identifier

Response Codes	• 201 – VSB created successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.1.2.5 Delete Vertical Service Blueprint

Table 50: Service Orchestrator REST APIs: Delete VSB

Endpoint	/vs/catalogue/vsblueprint/<vsb_id>
HTTP Verb	DELETE
Description	Delete and existing VSB from the Service Orchestrator catalogue
Caller	One Stop API
Request	• VSB ID
Response	• None
Response Codes	• 204 – VSB deleted successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2 Slice-level Interfaces

Similar to the Service Orchestrator, the Slice Orchestrator exposes its northbound interface through a set of REST APIs based on the HTTP protocol and JSON messages, offering different kind of features towards the DSPs (through the One Stop API framework) and the NSP administrator. In particular, two set of APIs are defined at the northbound of the Slice Orchestrator:

- Network slice management APIs, which provide a set of endpoints exposed towards DSPs (through the One Stop API) for network slice lifecycle management aspects, including retrieval of NSTs, operational actions NSIs (instantiation, termination, modification, etc). Additional endpoints exist also for network slice subnet lifecycle management (for operations on NSSTs and NSSIs), but these are not exposed to the DSPs. They can be indeed accessed by other NSP cognitive management platform components to apply their single domain optimization logic (e.g. for runtime adaptation and actuation over individual network slice instances)
- Management and administrative APIs, which are offered to the NSP system administrators and allow managing tenants, NSTs and NSSTs. They are intended to be used internally to the NSP (i.e. not exposed to third parties)

The Slice Orchestrator implements the server side of these REST APIs, while the DSP Service Orchestrators and the NSP OSS platform provide REST client sides. These APIs also includes

asynchronous notifications that are generated by the Slice Orchestrator as defined in the following subsections.

Authentication and authorization functionalities over the Slice Orchestrator REST APIs will be provided by the One Stop API framework.

With respect to the 5GT-VS APIs, these Slice Orchestrator APIs are defined from scratch as part of the explicit split of vertical service and network slice lifecycle management between the Service Orchestrator (at the DSP) and the Slice Orchestrator (at the NSP).

7.2.1 Network slice management APIs

The network slice management APIs exposed by the Slice Orchestrator implement the following list of lifecycle management operations:

- Query NSTs and NSSTs
- Instantiate, query, modify and terminate NSIs and NSSIs
- Notifications about NSIs and NSSIs lifecycle events

Subscriptions and notifications for NST related events These Slice Orchestrator REST APIs are based on the RESTful HTTP-based APIs defined by 3GPP in [40] for provisioning of network slice and network slice subnet instances. Apart from the resource URIs that are re-defined in SliceNet, the Slice Orchestrator REST APIs makes use of the 3GPP approach of requesting the provisioning of NSIs and NSSIs by using the Service Profile and Slice Profile data structures (see sections 4.2.3 and 4.2.4) to express the slice and slice subnet performance requirements.

The following subsections detail how these operations translate into REST resources and methods in the form of tables.

7.2.1.1 Query Network Slice Template

Table 51: Slice Orchestrator REST APIs: Get all NSTs

Endpoint	/ns/catalogue/nstemplate/
HTTP Verb	GET
Description	Retrieve all the NSTs from the Slice Orchestrator catalogue. The templates can then be used by the DSP to request for the instantiation of new network slices
Caller	DSP Service Orchestrator / One Stop API
Request	• DSP ID (i.e. tenant identification)
Response	• List of NSTs (following the format specified in section 4)
Response Codes	• 200 - NSTs retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.2 Query Network Slice Subnet Template

Table 52: Slice Orchestrator REST APIs: Get all NSTs

Endpoint	/ns/catalogue/nsstemplate/
HTTP Verb	GET
Description	Retrieve all the NSSTs from the Slice Orchestrator catalogue. The templates can then be used to request for the instantiation of new network slices subnets. This operation is not allowed to be invoked by the DSP Slice Orchestrator, while can be called by the NSP administrator or other SliceNet NSP cognitive management platform component to fulfil their specific logic.
Caller	NSP administrator / SliceNet NSP cognitive management platform components
Request	• DSP ID (i.e. tenant identification)
Response	• List of NSTs (following the format specified in section 4)
Response Codes	• 200 - NSTs retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

Table 53: Slice Orchestrator REST APIs: Get NSST

Endpoint	/ns/catalogue/nsstemplate/<nsst_id>
HTTP Verb	GET
Description	Retrieve an individual NSST from the Slice Orchestrator catalogue. The template can then be used to create new network slice subnet instances. This operation is not allowed to be invoked by the DSP Slice Orchestrator, while can be called by the NSP administrator or other SliceNet NSP cognitive management platform component to fulfil their specific logic.
Caller	NSP administrator / SliceNet NSP cognitive management platform components
Request	• NSST ID
Response	• NSST matching the issued NSST ID (following the format specified in section 4.2.2)
Response Codes	• 200 – NSST retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.3 Create Network Slice Instance

Table 54: Slice Orchestrator REST APIs: Create NSI

Endpoint	/ns/basic/nslcm/nsi
HTTP Verb	POST
Description	Create a new NSI, given its reference NST. The Slice Orchestrator starts the instantiation procedure and returns the NSI ID that can be used by the DSP and One Stop API to retrieve information about the status and the attributes of the NSI, by using the Query NSI operation. In parallel, lifecycle notifications are generated asynchronously towards the DSP / One Stop API in case a suitable notification endpoint was provided at the request time.
Caller	DSP Service Orchestrator / One Stop API
Request	• DSP ID • NSI name • NSI description • Reference NST ID • Notification REST endpoint, where the DSP Slice Orchestrator / One Stop API can to receive notifications about the lifecycle or failure events of the NSI • NSI ServiceProfile attributes and performance requirements (according to the data structure of section 4.2.3)
Response	• NSI Identifier
Response Codes	• 201 – NSI resource created successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.4 Create Network Slice Subnet Instance

Table 55: Slice Orchestrator REST APIs: Create NSI

Endpoint	/ns/basic/nslcm/nssi
HTTP Verb	POST

Description	Create a new NSSI, given its reference NSST. The Slice Orchestrator starts the instantiation procedure and returns the NSSI ID that can be used by the caller to retrieve information about the status and the attributes of the NSSI, by using the Query NSSI operation. In parallel, lifecycle notifications are generated asynchronously towards the caller in case a suitable notification endpoint was provided at the request time. This operation is not allowed to be invoked by the DSP Slice Orchestrator, while can be called by an NSP administrator or other SliceNet NSP cognitive management platform component to fulfil their specific logic.
Caller	NSP Administrator / SliceNet NSP cognitive management platform components
Request	• NSSI name • NSSI description • Reference NSST ID • Notification REST endpoint, where the caller can to receive notifications about the lifecycle or failure events of the NSSI • NSSI SliceProfile attributes and performance requirements (according to the data structure of section 4.2.4)
Response	• NSSI Identifier
Response Codes	• 201 – NSSI resource created successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.5 Query Network Slice Instance

Table 56: Slice Orchestrator REST APIs: Get all NSIs

Endpoint	/ns/basic/nslcm/nsi
HTTP Verb	GET
Description	Retrieve all the NSIs from the Slice Orchestrator inventory. This operation can be invoked also by any other SliceNet NSP cognitive management platform component to fulfil their specific logic.
Caller	DSP Slice Orchestrator / One Stop API / SliceNet NSP cognitive management platform components
Request	• DSP ID (i.e. tenant identification)
Response	• List of NSIs and related information (following the format described in section 4.2.3)

Response Codes	• 200 - NSIs retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

Table 57: Slice Orchestrator REST APIs: Get individual NSI

Endpoint	/ns/basic/nslcm/nsi/<nsi_id>
HTTP Verb	GET
Description	Retrieve an individual NSI from the Slice Orchestrator inventory. This operation can be invoked also by any other SliceNet NSP cognitive management platform component to fulfil their specific logic.
Caller	DSP Slice Orchestrator / One Stop API / SliceNet NSP cognitive management platform components
Request	• DSP ID (i.e. tenant identification) • NSI ID
Response	• NSI attributes, according to the model specified in section 4.2.3
Response Codes	• 200 – NSI retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.6 Query Network Slice Subnet Instance

Table 58: Slice Orchestrator REST APIs: Get all NSSIs

Endpoint	/ns/basic/nslcm/nssi
HTTP Verb	GET
Description	Retrieve all the NSSIs from the Slice Orchestrator inventory. This operation is not allowed to be invoked by the DSP Slice Orchestrator, while can be called by the One Stop API or other SliceNet NSP cognitive management platform component to fulfil their specific logic.
Caller	One Stop API / SliceNet NSP cognitive management platform components
Request	• None
Response	• List of NSSIs and related information (following the format described in section 4.2.4)

Response Codes	• 200 - NSSIs retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

Table 59: Slice Orchestrator REST APIs: Get individual NSSI

Endpoint	/ns/basic/nslcm/nssi/<nsi_id>
HTTP Verb	GET
Description	Retrieve an individual NSSI (belonging to a given DSP) from the Slice Orchestrator inventory. This operation is not allowed to be invoked by the DSP Slice Orchestrator, while can be called by the One Stop API or other SliceNet NSP cognitive management platform component to fulfil their specific logic.
Caller	One Stop API / SliceNet NSP cognitive management platform components
Request	• NSSI ID
Response	• NSSI attributes, according to the model specified in section 4.2.4
Response Codes	• 200 – NSSI retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.7 Terminate Network Slice Instance

Table 60: Slice Orchestrator REST APIs: Delete NSI

Endpoint	/ns/basic/nslcm/nsi/<nsi_id>
HTTP Verb	DELETE
Description	Terminate an existing NSI, given its NSI ID. The Slice Orchestrator starts the termination procedure and returns an HTTP code. In parallel, NSI termination notification is generated asynchronously towards the DSP Service Orchestrator / One Stop API when the NSI is deleted, in case a suitable notification endpoint was provided at the NSI creation time.
Caller	DSP Service Orchestrator / One Stop API
Request	• DSP ID (i.e. tenant identification) • NSI ID
Response	• None

Response Codes	• 200 – NSI termination procedure started successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.8 Terminate Network Slice Subnet Instance

Table 61: Slice Orchestrator REST APIs: Delete NSSI

Endpoint	/ns/basic/nslcm/nssi/<nssi_id>
HTTP Verb	DELETE
Description	Terminate an existing NSSI, given its NSSI ID. The Slice Orchestrator starts the termination procedure and returns an HTTP code. In parallel, NSSI termination notification is generated asynchronously towards the caller when the NSSI is deleted, in case a suitable notification endpoint was provided at the NSSI creation time. This operation cannot be invoked by the DSP Slice Orchestrator, while can be called by an NSP administrator or other SliceNet NSP cognitive management platform component to fulfil their specific logic
Caller	NSP Administrator
Request	• NSSI ID
Response	• None
Response Codes	• 200 – NSSI termination procedure started successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.9 Multi-domain actuation of Network Slice Instance

Table 62: Slice Orchestrator REST APIs: Multi-domain actuation of NSI

Endpoint	/ns/basic/nslcm/nsi/<nsi_id>/actuate-md
HTTP Verb	POST
Description	This operation is used by the DSP Service Orchestration to request for a multi-domain actuation over an NSI, in accordance with the multi-domain FCAPS principles and actuation workflows defined in D6.7 [27]. The Slice Orchestrator starts the NSI modification procedure for multi-domain actuation and returns an HTTP code. In parallel, the modification notification is generated asynchronously towards the caller when the actuation is applied, in case a suitable notification endpoint was provided in the NSI creation request.

Caller	DSP Service Orchestrator
Request	• DSP ID (i.e. tenant identification) • NSI ID • Actuation name (matching one of the actuations listed in the NST related to the NSI and exposed to the DSP) • Actuation parameters (according to the actuation parameters required as defined in the corresponding NST)
Response	• None
Response Codes	• 200 – NSI modification for multi-domain actuation procedure started successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.10 Single-domain actuation for Network Slice Instance optimization

Table 63: Slice Orchestrator REST APIs: Single-domain actuation for NSI optimization

Endpoint	/ns/basic/nslcm/nsi/<nsi_id>/actuate-sd
HTTP Verb	POST
Description	This operation is used by the TAL Rule Engine (part of the NSP FCAPS framework) to request for a single-domain actuation over an NSI, in response to the outcome of a NSP cognitive optimization process (as defined in D5.5 and D5.6). This is therefore an internal NSP actuation part of the cognitive management logics. The Slice Orchestrator starts the NSI actuation procedure and returns an HTTP code. In parallel, the actuation notification is generated asynchronously towards the caller when the actuation is applied, in case a suitable notification endpoint was provided in the request.
Caller	FCAPS TAL Rule Engine
Request	• NSI ID • Actuation name (matching one of the actuations listed in the NSP actuations catalogue) • Actuation parameters (according to the actuation parameters defined in the NSP actuations catalogue) • Notification endpoint, where the caller can receive notifications about the actuation result
Response	• None
Response Codes	• 200 – NSI actuation procedure started successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.11 Single-domain actuation for Network Slice Subnet Instance optimization

Table 64: Slice Orchestrator REST APIs: Single-domain actuation for NSSI actuation

Endpoint	/ns/basic/nslcm/nssi/<nssi_id>/actuate-sd
HTTP Verb	POST
Description	This operation is used by the TAL Rule Engine (part of the NSP FCAPS framework) to request for a single-domain actuation over an NSSI, in response to the outcome of a NSP cognitive optimization process (as defined in D5.5 and D5.6). This is therefore an internal NSP actuation part of the cognitive management logics. The Slice Orchestrator starts the NSSI actuation procedure and returns an HTTP code. In parallel, the actuation notification is generated asynchronously towards the caller when the actuation is applied, in case a suitable notification endpoint was provided in the request.
Caller	FCAPS TAL Rule Engine
Request	• NSSI ID • Actuation name (matching one of the actuations listed in the NSP actuation catalogue) • Actuation parameters (according to the actuation parameters defined in the NSP actuations catalogue) • Notification endpoint, where the caller can receive notifications about the actuation result
Response	• None
Response Codes	• 200 – NSSI actuation procedure started successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.12 Notification of Network Slice Instance lifecycle event

Table 65: Slice Orchestrator REST APIs: Notify NSI lifecycle event

Endpoint	/<endpoint-provided-by-the-dsp-or-onestopapi>
HTTP Verb	POST
Description	This operation is used by the Slice Orchestrator to notify the DSP Service Orchestrator (or the One Stop API) about the results of NSI lifecycle operations that have been executed. The notifications are asynchronous and provide the result of actions that have either triggered by vertical (e.g. instantiation, termination, modification) or by autonomous cognitive driven actuations.

Caller	NSP Slice Orchestrator
Request	• NSI ID • Notification type (e.g. instantiation result, termination result, modification result, actuation result)) • Result (success or failure)
Response	• None
Response Codes	• 200 – Notification received successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.13 Notification of Network Slice Subnet Instance lifecycle event

Table 66: Slice Orchestrator REST APIs: Notify NSSI lifecycle event

Endpoint	/<endpoint-provided-by-the-original-caller>
HTTP Verb	POST
Description	This operation is used by the Slice Orchestrator to notify the original caller about the results of NSSI lifecycle operations that have been executed. The notifications are asynchronous and provide the result of actions that have either triggered by vertical (e.g. instantiation, termination, modification) or by autonomous cognitive driven actuations.
Caller	NSP Slice Orchestrator
Request	• NSSI ID • Notification type (e.g. instantiation result, termination result, modification result, actuation result) • Result (success or failure)
Response	• None
Response Codes	• 200 – Notification received successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.14 Subscription to Network Slice Template related events

Table 67: Slice Orchestrator REST APIs: Subscription to NST related events

Endpoint	/ns/catalogue/subscription

HTTP Verb	POST
Description	This operation is used by the DSP Service Orchestrator or the One Stop API to subscribe to notifications for NST related events, including onboarding of new NSTs, modification of existing NSTs and deletion of NSTs. This allows the DSP and NSP catalogues to be synchronized.
Caller	DSP Slice Orchestrator / One Stop API
Request	• DSP ID • Notification endpoint, where the caller can to receive notifications about the lifecycle or failure events of the NSSI
Response	• None
Response Codes	• 201 – Subscription created successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.1.15 Notification of Network Slice Template event

Table 68: Slice Orchestrator REST APIs: Notification of NST event

Endpoint	/<endpoint-provided-by-the-dsp-or-onestopapi>
HTTP Verb	POST
Description	This operation is used by the Slice Orchestrator to notify the DSP Service Orchestrator (or the One Stop API) about the events occurred to NSTs. These notifications are sent under the condition that a subscription for this type of events has been issued.
Caller	NSP Slice Orchestrator
Request	• NST ID, being either the identifier of an existing NST (modified or deleted) or the identifier of a newly created NST. The caller can use this identifier to issue a query to retrieve full details about the related NST • Notification type (e.g. NEW, UPDATE, DELETION)
Response	• None
Response Codes	• 200 – Notification received successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.2 Management and administrative APIs

The Management and administrative APIs exposed by the Slice Orchestrator implement the following REST operations:

- Create, query and delete Tenants
- Create and delete NSTs and NSSTs

The following subsections detail how these operations translate into REST resources and methods in the form of tables.

7.2.2.1 Create Tenant

Table 69: Slice Orchestrator REST APIs: Create Tenant

Endpoint	/ns/admin/tenant/
HTTP Verb	POST
Description	Create a new tenant in the Slice Orchestrator. A tenant corresponds to a DSP who has established a business relationship with the NSP. Each tenant is able to request NSIs.
Caller	One Stop API
Request	• Tenant ID (which will be used as DSP ID by DSPs) • Tenant name • Tenant credential
Response	• None
Response Codes	• 201 – Tenant created successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.2.2 Query Tenant

Table 70: Slice Orchestrator REST APIs: Query Tenant

Endpoint	/ns/admin/tenant/<tenant_id>
HTTP Verb	GET
Description	Retrieve all the information associated with an existing tenant, including the IDs of the NSTs created by the DSP and the NSIs instantiated for it. Detailed information about each NST and NSIs can then be obtained through the related query operations.
Caller	One Stop API

Request	• Tenant ID
Response	• Tenant ID • Tenant name • Tenant credential • List of NST IDs • List of NSI IDs
Response Codes	• 200 – Tenant retrieved successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.2.3 Delete Tenant

Table 71: Service Orchestrator REST APIs: Delete Tenant

Endpoint	/ns/admin/tenant/<tenant_id>
HTTP Verb	DELETE
Description	Delete an existing tenant in the Slice Orchestrator. A tenant can be removed only when all its NSIs have been terminated.
Caller	One Stop API
Request	• Tenant ID
Response	• None
Response Codes	• 204 – Tenant deleted successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.2.4 Create Network Slice Template

Table 72: Slice Orchestrator REST APIs: Create NST

Endpoint	/ns/catalogue/nstemplate/
HTTP Verb	POST
Description	On-board a new NST in the Slice Orchestrator catalogue. The NST is typically designed off-line and include the list of constituent NSSTs.
Caller	NSP Administrator

Request	• NST, following the model specified in section 4.2.1
Response	• NST Identifier
Response Codes	• 201 – NST created successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.2.5 Create Network Slice Subnet Template

Table 73: Slice Orchestrator REST APIs: Create NSST

Endpoint	/ns/catalogue/nsstemplate/
HTTP Verb	POST
Description	On-board a new NSST in the Slice Orchestrator catalogue, so that it can be used as a baseline by the NSP administrator to build the slice offers in the form of NSTs. The NSST is typically designed off-line.
Caller	NSP Administrator
Request	• NSST, following the model specified in section 4.2.2
Response	• NSST Identifier
Response Codes	• 201 – NSST created successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.2.6 Delete Network Slice Template

Table 74: Slice Orchestrator REST APIs: Delete NST

Endpoint	/ns/catalogue/nstemplate/<nst_id>
HTTP Verb	DELETE
Description	Delete an existing NST from the Slice Orchestrator catalogue
Caller	NSP Administrator
Request	• NST ID
Response	• None

Response Codes	• 204 – NST deleted successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.2.2.7 Delete Network Slice Subnet Template

Table 75: Slice Orchestrator REST APIs: Delete NSST

Endpoint	/ns/catalogue/nsstemplate/<nsst_id>
HTTP Verb	DELETE
Description	Delete and existing NSST from the Slice Orchestrator catalogue
Caller	NSP Administrator
Request	• NSST ID
Response	• None
Response Codes	• 204 – NSST deleted successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.3 Resource-level APIs

The NBI module of the Resource Orchestrator implements a REST API that enables the configuration and management of the offered network services. This API is mainly used for the Slice Orchestrator at the NSP to manage the low level components of the slice. The operations exposed by this API are aimed to fulfill the requirements that have been detailed in section 3.2.3 and supported over the information model described in section 4.3. In addition, following the approach of the Service and the Slice Orchestrators, a second set of API operations has been defined for the management and administration of the Resource Orchestrator. These operations are aimed to provide the network service on-boarding (through the creation of the corresponding NSDs and VNFDs) and the internal administration of the NSP infrastructure.

7.3.1 Network service management APIs

7.3.1.1 List Available Network Services

Table 76: Network service management: List available network services

Endpoint	/nmro/catalogue/ns/
HTTP Verb	GET

Description	Retrieve the list of network services available to be instantiated over the infrastructure
Caller	SS-O at NSP
Request	• None
Response	• List of NSD IDs
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.1.2 Get network service information

Table 77: Network service management: Get network service information

Endpoint	/nmro/catalogue/ns/<nsd_id>
HTTP Verb	GET
Description	Retrieve all the information associated with an existing network service
Caller	SS-O at NSP
Request	• NSD ID
Response	• NSD information
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.1.3 List network service instances

Table 78: Network service management: List network service instances

Endpoint	/nmro/inventory/ns/
HTTP Verb	GET
Description	Retrieve the list of network services currently instantiated over the infrastructure
Caller	SS-O at NSP
Request	• None

Response	• List of network service instances
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.1.4 Get network service instance information

Table 79: Network service management: Get network service instance information

Endpoint	/nmro/inventory/ns/<nsi_id>
HTTP Verb	GET
Description	Retrieve all the information associated with an existing network service instance
Caller	SS-O at NSP
Request	• Network Service Instance ID
Response	• Network Service Instance information
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.1.5 Network Service instantiation

Table 80: Network service management: Network service instantiation

Endpoint	/nmro/ns/lcm
HTTP Verb	POST
Description	Create a new network service instance from the provided network service descriptor reference
Caller	SS-O at NSP
Request	• Network Service Descriptor ID
Response	• Network Service Instance ID
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.1.6 Modify Network Service instance

Table 81: Network service management: Modify network service instantiation

Endpoint	/nmro/ns/lcm/<nsi_id>
HTTP Verb	PUT
Description	Create a new network service instance from the provided network service descriptor reference
Caller	SS-O at NSP
Request	• Network Service Instance ID • Parameter / value list
Response	• None
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.1.7 Notification of Network Service Instance lifecycle event

Table 82: Network Service Management: Notify Network Service Instance lifecycle event

Endpoint	/<endpoint-provided-by-the-original-caller>
HTTP Verb	POST
Description	This operation is used by the Resource Orchestrator to notify the original caller about the results of Network Service instances lifecycle operations that have been executed. The notifications are asynchronous and provide the result of actions done over the network services instances as part of higher-level actuations that have been triggered by either vertical (e.g. instantiation, termination, modification) or by autonomous cognitive driven actuations.
Caller	NMR-O
Request	• Network Service Instance ID • Notification type (e.g. instantiation result, termination result, modification result, actuation result) • Result (success or failure)
Response	• None
Response Codes	• 200 – Notification received successfully • 4xx – Bad request conditions • 5xx - Internal server problems

7.3.1.8 Terminate network service instance

Table 83: Network service management: Terminate network service instance

Endpoint	/nmro/ns/lcm/<nsi_id>
HTTP Verb	DELETE
Description	Terminate the network service instance with the provided identifier
Caller	SS-O at NSP
Request	• Network Service Instance ID
Response	• None
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.2 Management and administrative APIs

7.3.2.1 Add VIM

Table 84: Management and administrative APIs: Add VIM

Endpoint	/nmro/admin/vims
HTTP Verb	POST
Description	Register a new VIM account to the Resource Orchestrator
Caller	NSP Administrator
Request	• Name • Description • Type • URL
Response	• VIM ID
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.2.2 Delete VIM

Table 85: Management and administrative APIs: Delete VIM

Endpoint	/nmro/admin/vims/<vim_id>
HTTP Verb	DELETE
Description	Unregister an existing VIM account from the Resource Orchestrator
Caller	NSP Administrator
Request	• VIM ID
Response	• None
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.2.3 Add WIM

Table 86: Management and administrative APIs: Add WIM

Endpoint	/nmro/admin/wims
HTTP Verb	POST
Description	Register a new WIM account to the Resource Orchestrator
Caller	NSP Administrator
Request	• Name • Description • Type • URL
Response	• WIM ID
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.2.4 Delete WIM

Table 87: Management and administrative APIs: Delete WIM

Endpoint	/nmro/admin/wims/<wim_id>

HTTP Verb	DELETE
Description	Unregister an existing WIM account from the Resource Orchestrator
Caller	NSP Administrator
Request	• WIM ID
Response	• None
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.2.5 Create VNF

Table 88: Management and administrative APIs: Create VNF

Endpoint	/nmro/admin/vnfs
HTTP Verb	POST
Description	Create a new VNF that can be used by a Network Service by uploading the VNFD
Caller	NSP Administrator
Request	• VNFD
Response	• VNFD ID
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.2.6 Delete VNF

Table 89: Management and administrative APIs: Delete VNF

Endpoint	/nmro/admin/vnfs/<vnfd_id>
HTTP Verb	DELETE
Description	Delete an existing VNFD
Caller	NSP Administrator
Request	• VNFD ID

Response	• None
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.2.7 Create Network Service

Table 90: Management and administrative APIs: Create Network Service

Endpoint	/nmro/admin/nss
HTTP Verb	POST
Description	Create a new Network Service that can be instantiated over the virtual infrastructure of the NSP.
Caller	NSP Administrator
Request	• NSD
Response	• NSD ID
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

7.3.2.8 Delete Network Service

Table 91: Management and administrative APIs: Delete Network Service

Endpoint	/nmro/admin/nss/<nsd_id>
HTTP Verb	DELETE
Description	Delete an existing NSD
Caller	NSP Administrator
Request	• NSD ID
Response	• None
Response Codes	• 200 – Success • 4xx – Bad request conditions • 5xx – Internal server problems

8 Software Prototype

8.1 Service and Slice Orchestrators Prototypes

The SliceNet SS-O is composed by the combination of the Service Orchestrator at the DSP level and the Slice Orchestrator at the NSP level, as detailed and specified in section 5.

From a software perspective, the SliceNet SS-O prototype is an enhanced and extended version of the Vertical Slicer (5GT-VS) [41] developed in the 5G-Transformer project.

This means that original 5GT-VS software has been enhanced to split the service and slice orchestration logic according to the SliceNet principles. In practice, two distinct orchestration applications are now embedded in the 5GT-VS. This refactoring exercise has been carried out considering backward compatibility with the original 5GT-VS approach as key, with the aim of not disrupting the 5G-Transformer single domain vertical slice paradigm.

The new enhanced version of the 5GT-VS, i.e. the SliceNet SS-O, which is now split into Service Orchestrator and Slice Orchestrator applications, reuses part of the vertical service management logic and source code, as described in section 5.2, and introduces a standalone network slice management application which interfaces with the vertical service management through a well-defined set of REST APIs (see section 7.2).

8.1.1 Implementation details

The SliceNet SS-O prototype is developed in Java and is based on the Spring framework [42], adopting Apache Maven [43] as build automation tool. The internal software design is based on a modular approach that reflects the Service and Slice Orchestrators architectures depicted in section 5.2 and 5.3. The different internal SS-O modules implement java interfaces to provide their functions towards the other entities, and asynchronous interactions between the software modules, where needed, are handled through messages exchanged in JSON format over a message bus based on RabbitMQ [44]. Inventory persistency as well as the Service and Slice Orchestrator catalogues are managed through PostgreSQL databases [45].

The REST APIs exposed by both the Service and Slice Orchestrators (defined in section 7) are developed through a number of REST controllers which implement the server side of the different interfaces. The requests received at the REST APIs are internally processed through the corresponding services.

The REST requests related management actions at both Service and Slice Orchestrator levels (e.g. tenants, descriptors and templates onboarding) are processed through services which interact with the associated databases in read/write mode. On the other hand, the REST requests related to operational and lifecycle management actions (e.g. creation of a new VSIs and NSIs or an associated runtime lifecycle command) are handled through the dedicated VSI and NSI operation Java services. From here, the processing of each request is handled following an asynchronous approach, coordinating the exchange of ordered messages between the involved software modules cooperating for the execution of the given lifecycle command.

The Service Orchestrator interacts with the Slice Orchestrator through a specific driver which embeds a REST-based implementation of the consumer side of the APIs described in section 7.2. On the other hand, the interaction of the Slice Orchestrator with NMR-O is handled through a specific driver which

embeds a REST-based implementation of the consumer side of the related APIs specified in section 7.3. However, the SS-O software architecture follows a modular approach, in order to support different drivers and, thus, enable the possible interaction with other Slice Orchestrators and NFVOs.

8.1.2 Source code structure

The source code of the core SS-O components is available as open source, under the Apache v2.0 license [46], on the Nextworks public github at:

https://github.com/nextworks-it/slicer

Installation and usage instructions are described in the README.md file. The SEBASTIAN folder contains all of the source code of the core engines of the Service and Slice Orchestrators prototypes, with packages arranged in a standard maven style.

Table 92 shows the source code structure and map it to the Service and Slice Orchestrator architectures depicted in section 5.2 and 5.3.

Table 92: Source code structure

VS_MANAGEMENT/	
VS_MANAGEMENT_APP/	Main application of the Service Orchestrator
VSMF_INTERFACES/ VSMF_SERVICE/	Vertical Service Manager end-to-end Network Slice Manager P&P Manager driver
ARBITRATOR_INTERFACES/ ARBITRATOR_SERVICE/	Arbitrator
VS_RECORD_SERVICE/ VS_RECORD_SERVICE_IM/	Vertical Service Instance information model
NS_MANAGEMENT/	
NS_MANAGEMENT_APP/	Main application of the Slice Orchestrator
NSMF_INTERFACES/ NSMF_SERVICE/	Network Slice Front-End Network Slice LCM P&P Manager driver RAN Control Driver QoS Control Driver
ARBITRATOR_INTERFACES/ ARBITRATOR_SERVICE/	Arbitrator

NS_RECORD_SERVICE/ NS_RECORD_SERVICE_IM/	Network slice and slice subnet information model
VNComService/	
VNComService/	Communication Service at both Service and Slice Orchestrators
SEBASTIAN_COMMON/	
CommonElements/	Common types and objects

For further details, documentation is available in the source code in the form of Javadoc comments. Therefore, HTML files can be generated from those comments by running the following command in the root folder:

```
mvn javadoc:Javadoc
```

Besides the core engines of the Service and Slice Orchestrators, the following additional code repositories provide key functionalities and are part of the whole SliceNet SS-O prototype (as Maven dependencies):

- slicer-catalogue: https://github.com/nextworks-it/slicer-catalogue (Catalogue of blueprints and descriptors for vertical services, including translation features)
- nfvo-drivers: https://github.com/nextworks-it/nfvo-drivers (Drivers to interact with NFV Orchestrators (and NMR-O in SliceNet))
- slicer-identity-mgmt: https://github.com/nextworks-it/slicer-identity-mgmt (Service for tenant management)

8.1.3 Software license and dependencies

The SliceNet SS-O prototype, in all its components, is released as open source software, under the license Apache 2.0 [46]. It includes software developed in SliceNet, enhancing and reusing part of the 5GT-VS software developed by the 5G-Transformer project, and it uses external open source tools and libraries to implement specific functionalities.

Table 93 table lists the SliceNet SS-O prototype software dependencies and related licenses.

Table 93: SS-O prototype dependencies

Component	Description	Software License
Spring framework	Application framework used for the development of the Service Orchestrator and Slice Orchestrator cores	Apache-2.0 https://github.com/spring-projects/spring-boot/blob/master/LICENSE.txt

NFV IFA libraries	Java libraries implementing information models and interfaces compliant with ETSI NFV IFA specifications.	Apache-2.0 https://github.com/nextworks-it/nfv-ifa-libs/blob/master/LICENSE
PostgreSQL	Backend SQL database used for the persistency of catalogues and inventories data	PostgreSQL licence https://opensource.org/licenses/postgresql
RabbitMQ	Message broker used for dispatching messages among the internal modules of the Service Orchestrator and Slice Orchestrator	Mozilla Public License https://www.mozilla.org/en-US/MPL/2.0/

8.2 Resources Orchestrator Prototype

As described in section 5, the core part of the Resource Orchestrator (NMR-O) is implemented by the OSM. OSM is responsible for the configuration of the network services, which are associated with the Network Slice. Such network services are implemented by a set of interconnected VNFs that are deployed over the infrastructure of the NSP. In addition, the OSM is wrapped by a set of modules aimed to interface with the other components of the SliceNet architecture, orchestration, monitoring or control. In this section, a brief insight of the software, prototyping and related developments associated to the Resource Orchestrator is given.

8.2.1 Implementation details

The functional modules of Open Source MANO (i.e. LCM, RO, VCA, NBI, etc.) are developed in Python and containerized in linux-based Docker containers. Inter-module information exchange is held by a unified message bus based on Kafka. The different modules of the architecture also rely on specific database technologies for persistence (i.e. inventories and catalogues), logs, alarms and metrics. In this regard, the LCM relies on MongoDB, while the RO uses MySQL.
The NMR-O software modules that surround the OSM are developed in Java and built by means of Apache Maven. The internals of this part of the NMR-O are completely aligned with the functional structure that has been explained in section 5.4. The Spring framework has been used to define and implement the REST interfaces exposed by the NBI component. To enable the interaction of with the OSM the QoE Rest Client Library has been extended. This library already implemented interfaces to different technologies that are being used within the SliceNet project (such as OpenStack or InfluxDB), as well as the interfaces to some SliceNet components (such as the SliceNet Control Plane and the Policy Framework). In particular, this library is extended in the framework of the orchestration to enable the configuration of OSM from the OSM Driver module of the NMR-O.

The list below shows the source code repositories for the different parts of the NMR-O:

- Open Source MANO: https://osm.etsi.org/gitweb

- NMR-O wrapping modules: https://gitlab.com/SliceNet/nmro
- QoE REST Client Library: https://gitlab.com/SliceNet/qoe-rest-client

8.2.2 On-boarding phase prototyping

The onboarding phase at the Resource Orchestrator level consists on the design and uploading of the NSDs and VNFDs into the catalogue. As explained throughout this document, this is a manual process that is carried out by the NSP to build the offers afterwards exposed to the DSP. In this context, some work has been realized in the framework of task 7.1 mainly related to the design and configuration of the virtual network infrastructure associated to a network slice. More specifically, the virtual Evolved Packet Core (vEPC) virtual infrastructure has been packaged into a network service that can be instantiated at the network slice deployment time.

To do this, a disk image based on Ubuntu 16.04 containing the vEPC software was created. In this case the vEPC software provided by the Mosaic5G project was used [47]. A VNFD was then created to package the vEPC software into a VNF that could be instantiated by means of OSM over the virtual infrastructure manager, which in this case is OpenStack.

Table 94 depicts the YAML file that describes the VNFD packaging. In addition, a cloud-init file was created and included in the VNFD packaging to parameterize the vEPC deployment with the specific attributes of the infrastructure (e.g. IP addresses, host name, etc.). In the last step, the NSD that packages the VNFD into an instantiable network service was created. Table 95 depicts the YAML file that describes the NSD packaging. A preliminary integration of this onboarding in the SliceNet testbeds that implement the three use cases tackled by the project (i.e. SmartGrid, eHealth and SmartCity) was reported in [48].

Table 94: VNFD YAML file

```
vnfd:vnfd-catalog:
  vnfd:
  - connection-point:
    - name: vnf-cp0
    description: Generated by UPC
    id: mosaic5g-allinone_vnfd
    mgmt-interface:
      cp: vnf-cp0
    name: mosaic5g-allinone_vnfd
    short-name: mosaic5g-allinone_vnfd
    vdu:
    - cloud-init-file: cloud-config-mosaic5g.txt
      count: 1
      description: mosaic5g-allinone_vnfd-VM
      id: mosaic5g-allinone_vnfd-VM
      image: SliceNet-oai-epc-NMRO
      interface:
      - external-connection-point-ref: vnf-cp0
        name: ens3
```

```
        type: EXTERNAL
        virtual-interface:
          type: PARAVIRT
      name: mosaic5g-allinone_vnfd-VM
      vm-flavor:
        memory-mb: 6144
        storage-gb: 6
        vcpu-count: 2
    vendor: OSM-UPC
    version: '1.0'
```

Table 95: NSD YAML file

```
nsd:nsd-catalog:
  nsd:
  - constituent-vnfd:
    - member-vnf-index: 1
      vnfd-id-ref: mosaic5g-allinone_vnfd
    description: Generated by UPC
    id: mosaic5g-allinone_nsd
    name: mosaic5g-allinone_nsd
    short-name: mosaic5g-allinone_nsd
    vendor: OSM-UPC
    version: '1.0'
    vld:
    - id: mosaic5g-allinone_nsd_vld0
      name: int-dpath-ovs-epc
      short-name: int-dpath-ovs-epc
      type: ELAN
      vim-network-name: eHealth_net
      vnfd-connection-point-ref:
      - member-vnf-index-ref: 1
        vnfd-connection-point-ref: vnf-cp0
        vnfd-id-ref: mosaic5g-allinone_vnfd
```

8.2.3 Software license and dependencies

Table 96 lists the SliceNet NMR-O prototype software dependencies and related licenses.

Table 96: NMR-O prototype dependencies

Component	Description	Software License
Spring framework	Application framework used for the development of the Service Orchestrator and Slice Orchestrator cores	Apache-2.0 https://github.com/spring-projects/spring-boot/blob/master/LICENSE.txt

QoE Rest Client Library	Java library implementing several communication interfaces for different SliceNet software components.	-
Open Source MANO	NFV Management and Orchestration framework	Apache Public License 2.0

9 Conclusions

With the ongoing transformations in the service providers operational management systems, such as coping with a totally new 5G network domain, managing and delivering Network Slices to vertical industry players across single and multi-provider environments and integrating cognition-based mechanisms to proactively improve the overall network efficiency, it's imperative that actuation mechanisms also evolve and follow this new trend. Actuation mechanisms are highly dependent on orchestration procedures and their relationship with the architecture components handling the services, slices and resources information. The main objective of this deliverable is to thoroughly describe the SliceNet orchestration approach and its implementation.

The document starts with an analysis and description of the most relevant SDOs, OSCs, industry and R&D projects that are working on the slice orchestration domain. As described in section 2, a significant number of 5G orchestration-related activities are ongoing and therefore SliceNet decided to adopt and create additional value on top of already existing solutions. SliceNet orchestration solution is based on 5GPPP 5G-Transformer R&D project, as well as on work being done at ETSI NFV and 3GPP standardization bodies.

The SliceNet orchestration vision is also highlighted in this document, and in particular in section 3. Before designing the architecture and the associated functional workflows, fundamental aspects that impact the orchestration vertical were addressed: support for a new managed network domain (5G), which represents a strong evolution from its predecessor (4G), support for network slices in single and in multi-administrative domains, as well as integration of real time closed-loops based on cognition produced AI models. These type of requirements impose orchestration to evolve and be able to orchestrate "vertically" across different information elements (resources, slices and services), but also to orchestrate "horizontally" across multiple administrative domains.

Based on the aforementioned high-level evolutions, an exhaustive list of requirements was described to guide the definition of the orchestration procedures at the resource, slice and service level. Additionally, the information model associated with each one of these entities was also described in section 4: the vertical service information model, which is the basis of the vertical service lifecycle management implemented by the SS-O at the DSP level; the slice information model, based on the slice part of the 3GPP Network Resource Model [31], which is the basis of the NSP slice orchestrator lifecycle management; finally, the resource orchestration information model based on the ETSI NFV specifications to provide the network services that will compose the upper level network slices.

Following the requirements and information model, section 5 describes the orchestration logical architecture. Herein is specified the service orchestration at the DSP-level and the slice and resource orchestration at the NSP side. Existing work from 5G-Transformer project [17] was adopted, in particular the Vertical Slicer software component, evolving it to deal with the SliceNet information model, as well as with the multi-domain approach. Along with the logical architecture definition, a functional description of the architecture is described in section 6, describing the most relevant orchestration workflows: i) design, onboard and offer, ii) instantiation, iii) NS optimization, iv) E2E NS optimization, v) vertical service reconfiguration and vi) vertical service decommission. Finally, a detailed description of the orchestration APIs at the service, slice and resource level are given in section 7.

The document finishes with a set of considerations in section 8 related with the software implementation of the SliceNet orchestration solution.

References

[1] https://SliceNet.eu/ © SLICENET consortium 2017

[2] https://www.3gpp.org/news-events/1951-sa5_5g

[3] https://www.3gpp.org/DynaReport/28621.htm TS 28.621 - Generic NRM Integration Reference Point (IRP); Requirements

[4] https://www.3gpp.org/DynaReport/28622.htm TS 28.622 - Generic NRM Integration Reference Point (IRP); Information Service (IS)

[5] https://www.3gpp.org/DynaReport/28623.htm TS 28.623 - Generic NRM Integration Reference Point (IRP); Solution Set (SS) definitions

[6] https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/02.01.01_60/gs_MEC003v020101p.pdf Multi-access Edge Computing (MEC); Framework and Reference Architecture

[7] https://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/004/02.06.01_60/gs_NFV-SOL004v020601p.pdf

[8] https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/001/01.01.01_60/gs_ZSM001v010101p.pdf ZSM; Requirements based on documented Scenarios

[9] https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/002/01.01.01_60/gs_ZSM002v010101p.pdf

[10] https://www.etsi.org/deliver/etsi_gs/ZSM/001_099/007/01.01.01_60/gs_ZSM007v010101p.pdf

[11] https://www.etsi.org/technologies/nfv/nfv-plugtests-programme NFV Plugtests Programme

[12] https://www.ericsson.com/en/portfolio/digital-services/automated-network-operations/orchestration/ericsson-orchestrator Ericsson Orchestrator

[13] https://www.nokia.com/networks/solutions/cloudband/#solution-elements NOKIA CloudBand Solution Elements

[14] https://www.nokia.com/blog/dynamic-network-slicing-wavefabrictm/ Dynamic network slicing with WaveFabric

[15] https://osm.etsi.org/ Open Source MANO

[16] https://osm.etsi.org/images/OSM_EUAG_White_Paper_OSM_Scope_and_Functionality.pdf OSM Scope, Functionality, Operation and Integration Guidelines, February 2019

[17] http://5g-transformer.eu/

[18] https://www.etsi.org/deliver/etsi_gr/NFV-EVE/001_099/012/03.01.01_60/gr_NFV-EVE012v030101p.pdf NFV Release 3; Evolution and Ecosystem; Report on Network Slicing Support with ETSI NFV Architecture Framework

[19] http://www.3gpp.org/DynaReport/28801.htm TR 28.801 - Study on management and orchestration of network slicing for next generation network

[20] https://www.etsi.org/deliver/etsi_gs/NFV/001_099/002/01.01.01_60/gs_NFV002v010101p.pdf NFV: Architectural Framework

[21] https://www.onap.org/ Open Network Automation Platform

[22] 5G-TRANSFORMER Deliverable D3.1 “Definition of vertical service descriptors and SO NBI” – 5G-TRANSFORMER Project – March 2018

[23] 5G-TRANSFORMER Deliverable D4.1 “Definition of service orchestration and federation algorithms, service monitoring algorithms” – 5G-TRANSFORMER Project – March 2018

[24] 5G-TRANSFORMER Deliverable D2.1 “Definition of the Mobile Transport and Computing Platform” – 5G-TRANSFORMER Project – March 2018

[25] SliceNet Deliverable D2.4: Management Plane System Definition, APIs and Interfaces

[26] SliceNet Deliverable D6.6: Single-Domain Slice FCAPS management

[27] SliceNet Deliverable D6.7: Multi-Domain Slice FCAPS management

[28] SliceNet Deliverable D6.3: Management for the Plug & Play Control Plane

[29] SliceNet Deliverable D5.5: Modelling, Design and Implementation of QoE Monitoring, Analytics and Vertical-Informed QoE Actuators; Iteration I

[30] SliceNet Deliverable D5.6: Modelling, Design and Implementation of QoE Monitoring, Analytics and Vertical-Informed QoE Actuators; Iteration II

[31] 3GPP TS 28.541, V16.2.0, Management and Orchestration; 5G Network Resource Model (NRM); Stage 2 and stage 3 (Release 16), September 2019

[32] 3GPP TS 28.531 V16.3.0 , 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Management and orchestration; Provisioning; (Release 16)

[33] ETSI GR NFV-IFA-015, v3.1.1, Management and Orchestration; Report on NFV Information Model, September 2018

[34] https://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/013/02.01.01_60/gs_NFV-IFA013v020101p.pdf ETSI GR NFV-IFA-013, v3.3.1, Management and Orchestration; Os-Ma-Nfvo Reference Point - Interface and Information Model Specification, September 2019

[35] https://osm.etsi.org/wikipub/index.php/OSM_Information_Model

[36] ETSI GR NFV-IFA-024, v3.2.1, Network Functions Virtualisation (NFV) Release 3; Information Modelling; Report on External Touchpoints related to NFV Information Model , April 2019

[37] C. Badulescu et. al., “ETSI NFV, the Pillar for Cloud Ready ICT Deployments”, Journal of ICT Standardization, Vol 7, Issue 2, May 2019

[38] ETSI TS 128 622, V15.3.1, Universal Mobile Telecommunication System; LTE; Telecommunication Management; Generic Network Resource Model (NRM); Integration Reference Point (IRP); Information Service (IS) (3GPP TS 28.622 version 15.3.1 Release 15), October 2019

[39] ETSI GS NFV 002 V1.1.1 NFV: Architectural Framework, October 2013

[40] 3GPP TS 28.531 V16.3.0 , 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Management and orchestration; Provisioning; (Release 16)

[41] https://github.com/5g-transformer/5gt-vs

[42] https://spring.io/

[43] https://maven.apache.org/

[44] https://www.rabbitmq.com/

[45] https://www.postgresql.org

[46] https://www.apache.org/licenses/LICENSE-2.0

[47] http://mosaic-5g.io/

[48] SliceNet Deliverable D8.4: SliceNet System Integration and Testing (Iteration II)

[end of document]